Karlsruhe Institute of Technology

Reliable Analog Circuit Design and Computing for Ultra-Resource-Constrained Edge Intelligence

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)
genehmigte

Dissertation
von
Priyanjana Pal
aus Agartala, Tripura, India

Tag der mündlichen Prüfung: 04. May 2026

1. Referent: Prof. Dr. Mehdi B. Tahoori
Karlsruhe Institute für Technology (KIT)

2. Referent: Prof. Dr. Haralampos-G. Stratigopoulos
Sorbonne Université - LIP6, France

Acknowledgement

Reflecting on this journey, I feel incredibly fortunate to have been surrounded by extraordinary individuals whose support, kindness, and patience made this dissertation possible.

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Dr. Mehdi B. Tahoori. I could not have asked for a better mentor throughout my PhD. His scientific insight, thoughtful feedback, and unwavering support, especially during the more challenging phases of this work, have profoundly shaped both this dissertation and my development as a researcher.

My sincere thanks go to all my colleagues at the Chair of Dependable Nano Computing (CDNC). I am especially grateful to our secretary, Ms. Iris Schröder-Piepka, for her tireless efforts in keeping everything running smoothly in the background. I would like to thank my co-authors Dr.-Ing. Haibin Zhao, Dr.-Ing. Michael Hefenbrock, and Dr.-Ing. Dennis D. Weller for their generous help, insightful discussions, and willingness to answer my questions, even after they transitioned to positions outside KIT.

I would also like to sincerely thank Prof. Dr. Sule Ozev from Arizona State University, USA and Dr. Sani Nassif from Radyalis LLC, USA for their valuable insights, constructive feedback, and inspiring discussions on reliability that significantly enriched the scientific depth of this work. My gratitude extends to Dr. Georgios Zervakis from the University of Patras for his thoughtful comments on emerging computing paradigms. I am equally grateful to Dr. Emre Ozer and Suman Balaji from Pragmatic Semiconductors, UK for their support and for sharing their expertise on flexible electronics technologies, which provided an important industrial perspective to this research.

Most importantly, I would like to express my heartfelt gratitude to my family. I am profoundly thankful to my parents, Mr. Parimal Pal and Mrs. Ratna Pal, whose unconditional love, sacrifices, and belief in me have been a constant source of strength. I am equally grateful to my elder sister, Dr. Rupanjana Pal, my brother-in-law Dr. Praloyraj Modak and my dear little niece, Ms. Pragnika Modak, whose affection and encouragement brought warmth and joy across the distance. My heartfelt thanks also go to my childhood and bachelor friends, whose unwavering support and shared memories reminded me that life extends beyond research. I also owe special thanks to my dear friends Dr.-Ing. Brojogopal Sapui and Dr.-Ing. Surendra Hemaram, who were like family to me during this journey and whose friendship, conversations, and humour helped me through many difficult days. I am likewise grateful for the friendship and support of my wonderful colleagues and friends already mentioned above, including Atousa Jafari, Sina Bakhtavari Mamaghani, Seyedehmaryam Ghasemi, Haneen Seyam, Dina Moussa, Shanmukha Mangadahalli Siddaramu, Ali Nezhadi Khelejani, Gürol Saglam, Maha Shatta, Zhe Zhang, Tara Gheshlaghi, Florentia Afentaki, Aradhana Dube, Mahboobe Sadeghipourrudsari, Mahta Mayahinia and Vincent Meyers, who each, in their own way, made this path lighter and more enjoyable.

Finally, my thanks goes to the European Research Council (ERC) which supported me financially in the form of the PRICOM project throughout my doctoral program.

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben haben und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen – die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Karlsruhe, 28. February 2026
Priyanjana Pal

Abstract

In the rapidly evolving era of Internet of Things (IoT) ecosystem, edge intelligence is moving from conventional embedded SoC platforms to ultra-resource-constrained and low-cost, lightweight application-specific systems such as smart packaging, on-skin patches, and batteryless sensors. These applications require extremely low power and low cost, and often demand implementation on flexible or unconventional substrates. While silicon System-on-Chips (SoCs) deliver high performance, they are frequently unable to satisfy the strict form-factor, cost, and energy constraints of such emerging domains. Large-area electronics and analog computing offer a compelling alternative by enabling bespoke local intelligence directly on non-traditional substrates, thereby reducing the hardware overhead of high-precision digital processing.

However, moving to these emerging large-area substrates introduces significant reliability and security challenges. Unlike standard silicon CMOS, large-area electronics suffer from process variations, environmental drift, and aging, which severely degrade classification accuracy and temporal stability. Also, these analog computing designs are physically exposed at the edge, due to lack of rigid packaging, making them vulnerable to physical attacks such as side-channel analysis. Ensuring trustworthiness is uniquely difficult because traditional heavyweight protection mechanisms cannot be applied under such extreme area and power budgets. So, a new class of design methodologies is required that co-optimizes accuracy, power and area.

This dissertation addresses these requirements by developing a cross-layer co-design framework for the reliable and secure design of emerging analog computing paradigms including spiking neuromorphic circuits. It systematically tackles the challenges of variability and fault tolerance through Neural Architecture Search (NAS) and evolutionary methods; introduces power-aware/power-constrained training to ensure energy efficiency and fault-endurance; and also establishes the comprehensive security and diagnostic methodologies for large-area analog computing circuits. The outcome is a unified methodology that enables robust, energy-efficient, and attack-resilient edge intelligence, even on the most resource-constrained and unreliable substrates.

Zusammenfassung

Im sich rasant entwickelnden Ökosystem des Internet of Things (IoT) verlagert sich Edge-Intelligenz von konventionellen eingebetteten SoC-Plattformen hin zu ultra-ressourcenbeschränkten, kostengünstigen und leichten, anwendungsspezifischen Systemen wie Smart Packaging, On-Skin-Patches und batterielosen Sensoren. Diese Anwendungen erfordern extrem niedrige Leistungsaufnahme und geringe Kosten und müssen häufig auf flexiblen oder unkonventionellen Substraten realisiert werden. Obwohl siliziumbasierte System-on-Chips (SoCs) eine hohe Leistung bieten, können sie die strengen Anforderungen an Formfaktor, Kosten und Energieverbrauch in diesen neuen Einsatzfeldern oft nicht erfüllen. Großflächige Elektronik (Large-Area Electronics) und analoge Rechenparadigmen stellen eine attraktive Alternative dar, da sie maßgeschneiderte lokale Intelligenz direkt auf nicht-traditionellen Substraten ermöglichen und dadurch den Hardware-Overhead hochpräziser digitaler Verarbeitung reduzieren.

Der Wechsel zu solchen großflächigen Substraten bringt jedoch erhebliche Herausforderungen hinsichtlich Zuverlässigkeit und Sicherheit mit sich. Im Gegensatz zu standardmäßiger Silizium-CMOS-Technologie leidet großflächige Elektronik unter ausgeprägten Prozessvariationen, umweltbedingter Drift und Alterung, was die Klassifikationsgenauigkeit und die zeitliche Stabilität deutlich verschlechtert. Zudem sind diese analogen Rechendesigns am Edge physisch exponiert, unter anderem aufgrund fehlender starrer Verpackung, und dadurch anfällig für physische Angriffe wie Side-Channel-Analysen. Die Gewährleistung von Vertrauenswürdigkeit ist besonders schwierig, da klassische, rechen- und flächenintensive Schutzmechanismen unter den extremen Energie- und Flächenbudgets nicht einsetzbar sind. Daher wird eine neue Klasse von Entwurfsmethoden benötigt, die Genauigkeit, Leistungsaufnahme und Fläche gemeinsam optimiert.

Diese Dissertation adressiert diese Anforderungen durch die Entwicklung eines schichtenübergreifenden Co-Design-Frameworks für den zuverlässigen und sicheren Entwurf neuartiger analoger Rechenparadigmen einschließlich spikender neuromorpher Schaltungen. Sie begegnet den Herausforderungen von Variabilität und Fehlertoleranz systematisch mittels Neural Architecture Search (NAS) und evolutionären Methoden, führt power-aware bzw. power-constrained Training zur Sicherstellung von Energieeffizienz und Fehler-Endurance ein und etabliert umfassende Sicherheits- sowie Diagnosemethoden für großflächige analoge Rechenschaltungen. Das Ergebnis ist eine einheitliche Methodik, die robuste, energieeffiziente und gegen Angriffe widerstandsfähige Edge-Intelligenz selbst auf stark ressourcenbeschränkten und unzuverlässigen Substraten ermöglicht.

Contents

List of Figures

List of Tables

Acronyms

EA evolutionary algorithm.

FT fault-tolerant.

LDO low-dropout regulator.

NAS neural architecture search.

***P*-AT** power-aware training.

***P*-CT** power-constrained training.

TR thermal-resilient.

TU thermal-unresilient.

***f*-NN** flexible neural network.

pTPB printed temporal processing block.

pTPNN printed temporal processing neural network.

***p*-NN** printed neural network.

a-IGZO amorphous Indium-Gallium-Zinc Oxide.

ADC Analog-to-Digital Converter.

AEC Analog Error Correction.

AF Activation Function.

AI Artificial Intelligence.

ANN Artificial Neural Network.

CPA Correlation Power Analysis.

DC Direct Current.

EGT Electrolyte-gated transistor.

FAT fault-aware training.

FE Flexible Electronics.

FPGA Field-Programmable Gate Array.

ISI Inter-Spike Interval.

LDA Linear Discriminant Analysis.

MI Mutual Information.

MIM Metal–Insulator–Metal.

ML Machine Learning.

MLP Multi-Layer Perceptron.

NC Neuromorphic Computing.

NN Neural Network.

PDK Process Design Kit.

PDN Power Delivery Network.

PE Printed Electronics.

pPDK Printed Process Design Kit.

PUF Physical Unclonable Function.

PVT Process, Voltage, and Temperature.

QDA Quadratic Discriminant Analysis.

RNN Recurrent Neural Network.

SCA Side-Channel Analysis.

SLI Spike-Leakage Index.

SNN Spiking Neural Network.

TFT Thin-Film Transistor.

VDD Supply Voltage.

Part I.

Preliminaries

1. Introduction and Motivation

1.1. Motivation

The rapid evolution of the Internet of Things (IoT), wearables, and consumer-edge devices has shifted computation from centralized cloud infrastructures toward distributed intelligence at the extreme edge. Emerging platforms such as smart packaging, smart bandages, on-skin patches, and battery-less sensing labels must sense, process, and sometimes act on data locally, while operating under severe constraints in power, cost, and form factor. These nodes are often disposable, ultra-thin, and mechanically flexible, targeting applications ranging from food quality monitoring and drug delivery to personalized healthcare and environmental sensing.

Traditional silicon System-on-Chip (SoC) platforms offer excellent performance and benefit from highly mature design flows. However, they are fundamentally limited by rigid substrates, complex and expensive fabrication steps, and non-negligible per-die cost. Their packaging and thermal budgets are tailored for long-lifetime electronics, not for ultra-low-cost, mechanically compliant, or biodegradable systems. Furthermore, conventional digital systems incur significant overhead to perform even simple machine learning (ML) tasks close to the sensor, due to the need for precise data conversion, clocked logic, and general-purpose programmability. In many edge scenarios, transmitting raw sensor data wirelessly to a gateway or cloud consumes more energy than performing the desired local computation, making purely cloud-centric approaches unsustainable. This shift enables bespoke, application-specific intelligence at the sensor, making it possible to deploy large numbers of ultra-low-cost intelligent tags and patches.

1.1.1. Characteristics of Ultra-Resource-Constrained Edge Nodes

In this thesis, we focus on a restrictive design point, which we refer to as *ultra-resource-constrained edge intelligence*. Compared to typical embedded systems, these platforms are characterized by:

- **Tight power and energy budgets**: Nodes may be powered by printed batteries, thin-film photovoltaics, or energy harvesters, providing average power in the μW–low mW range. Operation must often be intermittent, with long sleep phases and short active bursts.
- **Severe area and device-count limits**: Large feature sizes and low integration densities in thin-film technologies constrain the number of transistors or passive components that can be afforded on a single tag, patch, or label.
- **Rigid cost constraints**: Many use cases (e.g., food packaging, logistics labels, disposable medical patches) require total electronics cost on the order of fractions of a cent per unit, ruling out complex SoCs, large memories, and high-precision ADCs.
- **Mechanical and material requirements**: Electronics must be flexible, bendable or even stretchable, sometimes breathable or conformal to skin, and increasingly also biocompatible or biodegradable. This favors thin-film and printed technologies over bulk silicon.
- **Application-specific workloads**: Unlike general-purpose edge devices, these systems typically implement a single ML task (e.g., stress detection, spoilage detection, or event classification) with a small input dimensionality and modest accuracy requirements, but extremely tight constraints on footprint and power.

These characteristics fundamentally change the design problem. The goal is not to maximize raw computational throughput, but to realize the *minimum viable intelligence*—the smallest and most energy-efficient model that satisfies application requirements—within a severely constrained technological envelope.

1.1.2. Limitations of Conventional Digital Edge Architectures

A conventional IoT node usually consists of a sensor front-end, an analog-to-digital converter (ADC), a microcontroller or digital ML accelerator, and a radio. From a system perspective, this architecture exhibits several inefficiencies in the targeted regime:

- **Data conversion overhead**: High-resolution ADCs dominate both area and power when implemented in large-area or printed technologies, and their precision is often unnecessary for simple classification tasks. Even in silicon, data conversion can consume a significant share of system energy.
- **Digital control and memory cost**: Microcontrollers require non-trivial amounts of digital logic and embedded memory. Mapping even a small artificial NN (ANN) or spiking NN (SNN) to such a platform introduces overheads in instruction decoding, control, and data movement that are disproportionate to the actual computation.
- **Form factor mismatch**: Rigid dies and packages must be mounted on PCBs or interposer foils, which increases thickness and reduces flexibility. This is incompatible with applications where electronics must be imperceptible on skin, integrated directly into packaging, or laminated into textiles.
- **Energy for communication**: In battery-less or energy-harvested systems, wireless transmission of raw or lightly processed sensor streams is often infeasible. It is more energy-efficient to perform aggressive local processing and communicate only decisions or compressed features.

As a consequence, simply porting standard digital edge architectures to flexible or printed substrates is neither economical nor technically attractive. Instead, radically different computing paradigms are needed—paradigms that co-design algorithms, circuits, and fabrication technology for this ultra-constrained envelope.

1.1.3. Emerging Analog Computing Paradigms on Large-Area Electronics

In response, a new class of *emerging analog computing paradigms* implemented on *large-area electronics* has gained traction. Instead of relying on dense CMOS logic, these paradigms exploit thin-film device technologies, both printed and flexible, to realize bespoke, task-specific analog processing directly at the sensor. In particular:

- **Printed Electronics (PE)** use additive manufacturing techniques such as gravure or inkjet printing to deposit conductive, semiconductive, and dielectric inks on substrates like plastic, paper, or textiles. Typical device examples in this thesis are printed inorganic n-type EGTs, which can operate at sub-1 V supply voltage and are therefore well-suited to energy-harvested nodes.
- **Flexible Electronics** (**FE**)based on amorphous oxide semiconductors, e.g., n-type a-IGZO TFTs, are fabricated using low-temperature processes on polymer substrates. While not necessarily purely additive, they provide mechanically flexible, large-area transistor arrays with better device uniformity and higher switching speed than many printed devices.

Both technology classes share key system-level advantages: ultra-low-cost fabrication for modest integration scales, mechanical flexibility and conformability, and the ability to print or deposit devices side-by-side with sensors, energy harvesters, and interconnects on the same substrate. At the same

time, they suffer from pronounced process variations, limited device counts, and parasitics that make conventional digital design unattractive. This motivates a shift toward *analog* computing paradigms that minimize the number of required devices and avoid expensive digital blocks such as high-resolution ADC and large memories.

1.1.4. Printed (PE) and Flexible Electronics (FE) as an Enabler

PE and FE provide fabrication platforms that differ fundamentally from wafer-scale CMOS. Instead of deep-submicron lithography on rigid silicon, these technologies realize thin-film devices on mechanically compliant substrates such as plastics, metal foils, or polymer laminates. In this thesis we exploit two complementary device platforms:

- *Printed inorganic n-EGTs* fabricated additively on polymer substrates, as used in our printed NNs and spiking networks.
- *Flexible a-IGZO TFTs* fabricated on plastic foils using low-temperature thin-film deposition and subtractive patterning, as used in our Thermo-NAS and ReFLEX-*LDO* studies.

Printed solution-processed n-EGTs. PE rely on maskless, additive processes such as gravure and inkjet printing to deposit functional inks (conductive, semiconductive, dielectric) layer-by-layer onto flexible substrates.[1] The printed devices used in this thesis are inorganic n-type EGTs based on oxide semiconductors. A solid polymer electrolyte replaces the conventional gate dielectric, yielding very high gate capacitance and enabling operation at sub-1 V supply voltages. This is crucial for scenarios where power must be harvested from thin-film batteries, printed photovoltaics, or other low-capacity energy sources.

The additive nature of PE provides several key advantages:

- **Ultra-low-cost, bespoke fabrication**: Roll-to-roll and sheet-based printing minimize non-recurring engineering cost. Even for small volumes, per-unit cost can reach the sub-cent range, making it economically feasible to fabricate bespoke circuits whose parameters are fully hardwired after off-line training.
- **Mechanical flexibility and conformability**: Printed films on thin plastic or polymer foils can bend, twist, or wrap around curved surfaces, enabling integration in smart bandages, packaging, or textiles.
- **Material and substrate diversity**: By choosing suitable inks and substrates, printed devices can be made soft, porous, biocompatible, or even biodegradable, properties that are difficult to achieve in conventional silicon flows.

However, these benefits come at a price: printed transistors typically feature large geometries, low integration densities, and substantial process variation originating from ink dispersion, droplet statistics, wetting, and printer inaccuracies. Device characteristics drift with humidity, aging, and environmental stress, and only n-type devices are available in our technology node. As a result, printed circuits must be designed to tolerate wide parametric spreads, reduced matching, and limited device counts—constraints that strongly motivate analog, neuromorphic, NN and bespoke architectures rather than large, general-purpose digital systems.

Flexible a-IGZO TFT electronics. FE, in contrast, are thin-film circuits fabricated on bendable substrates using more conventional semiconductor processing, but at reduced temperatures compatible with polymers. In this thesis we consider a flexible oxide-TFT platform based on a-IGZO transistors patterned on polymer

[1] Typical printing processes and device cross-sections for printed electrolyte-gated transistors (EGTs) are discussed in Chapter 2.

or foil substrates using physical-vapor or atomic-layer deposition, followed by photolithographic patterning and etching. These devices provide higher mobility and better uniformity than fully printed organic TFTs, and can operate from sub-1 V to a few volts, making them attractive for low-power neuromorphic, NN and power-management blocks on flexible foils.

Structurally, the a-IGZO TFT technology used in this thesis is:

- **Unipolar and n-type only**: Only enhancement-mode n-TFTs are available, forcing resistor–transistor logic and precluding complementary pull-up/pull-down structures. This significantly constrains achievable gain, noise margins, and circuit topologies.
- **Thin-film and large-geometry**: Feature sizes remain in the micrometer range, and practical integration densities are limited to a few thousand devices per die. As in PE, this naturally favors compact, application-specific neuromorphic, NN accelerators and power-management blocks rather than large digital processors.
- **Mechanically and thermally sensitive**: The absence of rigid packaging and heat sinks means that a-IGZO TFTs are highly susceptible to bending-induced strain and temperature variations. Threshold-voltage shifts, mobility degradation, and bias-stress effects can lead to significant drift in analog behavior and even functional failures under thermal or mechanical stress.

Despite these challenges, flexible a-IGZO TFTs enable neuromorphic, NN and mixed-signal hardware that can be directly laminated onto curved surfaces, integrated into wearable or epidermal devices, or embedded into packaging. In this thesis we exploit this platform to realize flexible NNs and support blocks such as low-dropout regulators, and to study thermal-aware training and reliability-driven design methodologies that explicitly account for the thermal and mechanical fragility of a-IGZO devices.

Common design implications. Both printed n-EGT and flexible a-IGZO TFT technologies share several important traits: large device variations, limited device types (n-only), moderate integration density, and strong sensitivity to the environment (temperature, bending, humidity, and aging). Consequently, algorithm–hardware co-design is not a luxury but a necessity. Throughout this thesis, these constraints motivate:

- analog NNs that avoid costly ADCs and exploit inherent error tolerance,
- bespoke, application-specific architectures where all parameters are hardwired after off-line optimization, and
- explicit consideration of power, variability, thermal resilience, reliability, testing, and security at the algorithmic level.

Together, PE and FE thus act as the enabling substrate for the research in this thesis: they provide exactly the low-cost, mechanically compliant, but variation- and reliability-limited hardware context in which the proposed neuromorphic, NN, optimization, testing, and security techniques are developed and evaluated.

1.1.5. Large-Area Bespoke Analog NN Hardware

To endow printed and flexible platforms with local intelligence, one natural direction is to implement *analog NNs* directly in these technologies. Such circuits mirror the computational primitives of artificial NNs (ANNs), weighted sums followed by nonlinear activation, using resistor crossbar arrays and inverter-based nonlinear circuits. In printed NNs, weights are encoded as conductances in resistive crossbars, while the neuron nonlinearity is implemented using transistor-level activation circuits (e.g., p-tanh, p-ReLU, or other bespoke analog transfer functions). In flexible thin-film implementations based on a-IGZO TFTs, similar resistor–transistor primitives can be realized on bendable substrates, enabling neuromorphic classifiers that are both mechanically compliant and ultra-low-cost.

From a technology perspective, this thesis targets *two complementary large-area platforms*:

- **Printed n-EGTs** on paper or plastic substrates, where channel conductance is modulated by ionic gating. These devices operate at sub-1 V supply voltages and are well-suited for ultra-low-power analog computation and unary or multilevel signal processing directly at the sensor.
- **Flexible a-IGZO TFTs** on polymer substrates, fabricated via low-temperature deposition and patterning. Here, unipolar n-type devices and resistive loads impose strict constraints on gain and integration density, but offer mechanically robust neuromorphic building blocks that can be wrapped onto curved surfaces or integrated into flexible patches.

Despite differences in materials and fabrication flows, both platforms share key characteristics: unipolar device libraries, relatively low integration densities (up to a few thousand devices), and pronounced process and environmental variations. These constraints strongly favor compact, *bespoke* analog NNs rather than large programmable digital accelerators.

Analog neuron primitives on large-area substrates. At the circuit level, large-area analog neuromorphic hardware implements neural computation using a small set of reusable primitives:

- **Resistor crossbars** that realize weighted sums by exploiting Ohm's and Kirchhoff's laws. Input voltages are applied to crossbar rows, and column currents or voltages encode the weighted combination of inputs; individual conductances encode synaptic weights.
- **Nonlinear activation circuits** implemented with n-EGTs or a-IGZO TFTs, such as analog tanh-like, ReLU-like, or saturating transfer functions. Their shapes are controlled by device dimensions (e.g., W/L), biasing networks, and passive components, and can be tuned to match the desired neural nonlinearity.
- **Support blocks** such as bias generation, level shifting, and simple PDNs, which provide stable operating points despite low supply voltages and large variations.

In these architectures, circuit parameters—resistor values, transistor widths and lengths, bias voltages, and in some cases even device layout options—become the analog equivalents of learnable weights. This perspective enables a *machine-learning-based design* flow: a differentiable behavioral model of the *p*-NN or flexible neuromorphic circuit is trained in software, and the resulting parameters are subsequently mapped to physical circuit quantities.

Temporal and event-driven analog computation. Beyond static classification, many edge applications involve time-varying signals (e.g., biomedical, inertial, or environmental data). Large-area analog NNs can be extended to *temporal processing* in several ways:

- **Learnable analog filters and recurrent connections**, implemented as RC networks and feedback paths around crossbars and activation circuits, provide finite impulse response (FIR)-like or infinite impulse response (IIR)-like dynamics. Their component values are trained jointly with the neuromorphic weights to optimally process temporal patterns.
- **Spiking NNs** use analog membrane voltages and thresholding elements to generate spikes. Printed implementations can encode information in spike timing and rate, enabling event-driven operation where computation and power consumption scale with activity rather than clock frequency.
- **Mixed-signal interfaces**- for example, bespoke ADCs co-designed with simple classifiers—enable hybrid analog/digital decision-making where the analog front-end is tuned to the statistics of the data and the classifier, reducing resolution and conversion cost while preserving task accuracy.

These temporal and event-driven extensions form a continuum of analog and mixed-signal neuromorphic solutions, ranging from purely analog *p*-NNs and flexible analog NNs to hybrid systems that combine analog preprocessing with lightweight digital logic.

Design automation and hardware-aware training. A major challenge is that the design space of bespoke NNs on large-area technologies is highly constrained and strongly affected by device nonidealities. Manually exploring trade-offs between accuracy, energy, and robustness is infeasible. In response, this thesis adopts and extends *hardware-aware training and architecture search*:

- **Power- and area-aware training** treats static and dynamic power, device count, and even current densities as differentiable cost terms during training, yielding *p*-NNs that respect tight energy budgets and device limits without post-hoc pruning.
- ***NAS* and evolutionary design** are leveraged to automatically explore the discrete design space of layer sizes, connectivity patterns, and analog circuit primitives under process and thermal variations. This includes printed *p*-NN architectures as well as flexible a-IGZO-based networks, where thermal gradients on the flexible substrate can significantly perturb device characteristics.
- **Thermal-aware co-optimization** extends these techniques to explicitly account for spatially varying temperature on flexible a-IGZO substrates, ensuring that the neuromorphic classifier remains accurate even in the presence of non-uniform self-heating and ambient temperature changes.

Through these methods, analog NNs are not only trained for accuracy on target datasets, but also co-optimized for realistic device behavior, process variation, and physical operating conditions of PE and FE.

Beyond neuromorphic: other analog computing primitives. While analog NNs form the core of this thesis, large-area analog computing is not limited to ANNs and SNNs. Two important complementary directions are:

- **Unary-encoded analog computing with error correction**, where information is represented as unary pulse streams or resistor networks, and robust analog error-correction blocks are used to mitigate noise and variability in printed hardware. These architectures target, among others, printed physically unclonable functions (PUFs) and simple decision-making logic, further extending the design space of ultra-low-cost analog computing.
- **Analog-aware mixed-signal ML pipelines**, such as on-sensor classifiers using bespoke ADCs co-designed with decision trees. Here, analog front-ends and quantization structures are optimized jointly with simple digital classifiers, showing that the cross-layer ideas developed for *p*-NNs also generalize to non-neuromorphic ML models.

Together, these efforts position large-area analog neuromorphic hardware as part of a broader family of *analog and mixed-signal computing architectures* tailored for ultra-resource-constrained edge intelligence. Later chapters of this thesis will detail how these concepts are instantiated on both printed n-EGT-based *p*-NNs and flexible a-IGZO-based neuromorphic and analog computing blocks.

To visually support this discussion, an illustrative figure is included, e.g., as Fig. 1.1 to give a brief overview.

1.2. Problem Statement

The overarching problem addressed in this thesis can be stated as follows:

> *How can we design reliable and secure analog NNs on emerging large-area substrates that operate under extreme power, area, and cost constraints, while maintaining meaningful machine learning performance on edge tasks?*

This problem spans multiple layers—from neural architecture and training schemes to fault modeling, endurance management, testing, and security analysis. It is shaped both by the opportunities of PE and FE and by their pronounced technological limitations.

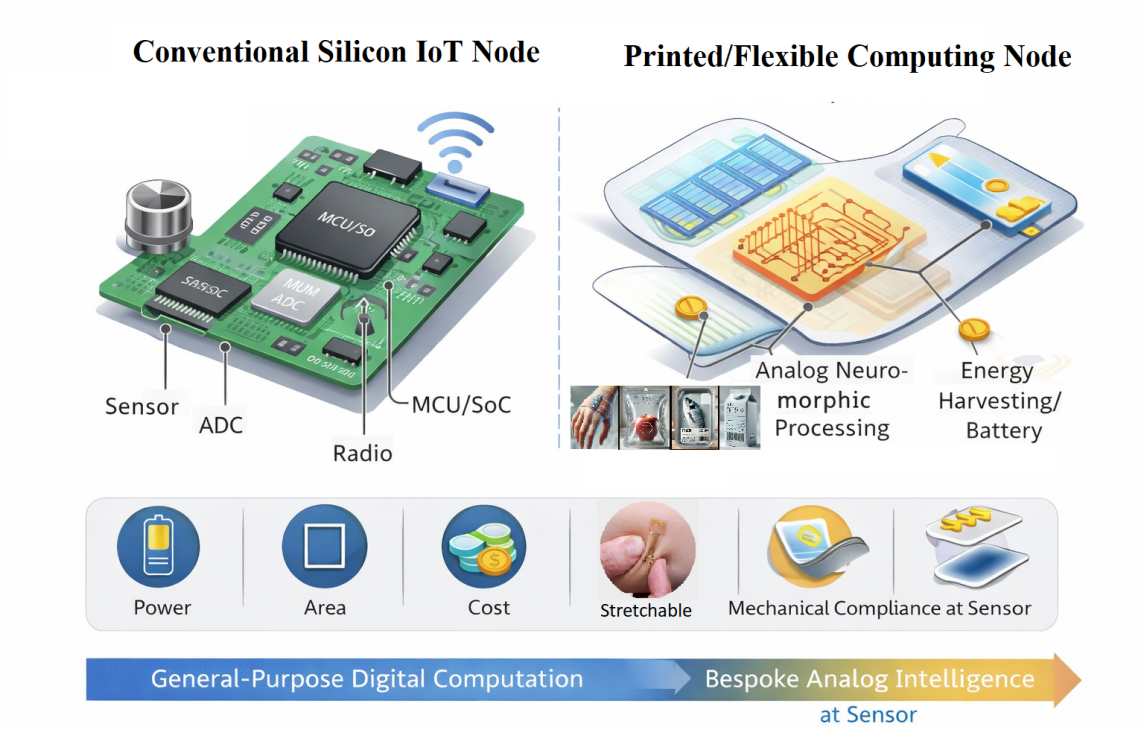


Figure 1.1.: Comparison between conventional silicon-based edge nodes and ultra-resource-constrained nodes based on printed and flexible analog AI hardware. Left: a traditional IoT node with sensor, ADC, MCU/SoC, and radio on a rigid PCB. Right: a flexible label or patch with co-integrated printed or a-IGZO TFT sensors, analog neuromorphic processing, energy harvester/printed battery, and a minimal communication interface. The bottom row highlights the dominant constraints (power, area, cost, mechanical compliance) and the shift from general-purpose digital computation to bespoke analog intelligence at the sensor.

1.2.1. Research Gaps

Existing work on printed and flexible neuromorphic and NN computing circuits and related analog computing approaches falls short in several ways. We summarize the most critical gaps that this thesis aims to address.

Gap 1: Power-Energy Efficiency and Temporal Processing

Analog neuromorphic circuits are often claimed to be power and energy efficient, yet most training methods ignore detailed power models. Without accurate and differentiable power estimation, it is difficult to enforce strict power budgets or to explore trade-offs between accuracy and power.

In addition, many edge applications involve temporal signals (e.g., physiological waveforms, motion traces, or environmental time series). There is a lack of designs that combine energy-aware training with temporal processing primitives, such as learnable filters or spiking neuron dynamics, in a way that remains compatible with printed or flexible fabrication.

Gap 2: Architecture-Aware Robust Design

Most neuromorphic hardware design flows either assume a fixed network topology or tune architectures manually using heuristic rules. In printed and flexible technologies, however, process-induced variability interacts strongly with architecture choices: crossbar size, depth, activation function circuit family, and analog operating range all influence variation tolerance.

Current approaches do not provide an automated framework that jointly explores network topology, activation circuit design, and analog parameters under realistic process variation models. Moreover,

environmental effects such as self-heating and ambient temperature are rarely accounted for during architecture search, despite their strong impact on device characteristics.

Gap 3: Reliability, Endurance and Test

Printed NNs suffer from a mix of permanent and intermittent faults arising from manufacturing defects, wear-out, and environmental stress. While some work examines basic reliability aspects, there is limited understanding of fault sensitivity in bespoke printed MLP classifiers across different analog and digital realizations.

At the same time, traditional digital error-correcting codes are too heavy for large-area analog circuits. Lightweight Analog Error Correction (AEC) schemes that operate directly on unary or stochastic analog encodings are underexplored, as are adaptive endurance mechanisms that can extend overall lifetime by reconfiguring or derating hardware.

Gap 4: Security Analysis and Trust

Security research on flexible and PE is still in its infancy. Bespoke neuromorphic classifiers implemented on TFT technologies exhibit unique leakage characteristics and attack surfaces. Existing work on digital side-channel analysis cannot be directly transferred to analog flexible NNs, which exhibit continuous, noisy leakage patterns.

Furthermore, manufacturing test and diagnostic frameworks for large-area NNs are largely missing. Classical digital test pattern generation methods do not address analog behavior. There is a need for test, diagnosis, and security frameworks that are aware of analog computing behavior and are compatible with ultra-low-cost manufacturing.

1.3. Research Objectives and Contributions

To address the challenges and gaps outlined above, this thesis is organized around a set of research objectives that are made more precise through four central research questions. Each research question corresponds to one major contribution axis of the thesis and maps directly to one or more contribution chapters.

1.3.1. Research Questions

Based on the above gaps, this thesis is structured around four central research questions:

RQ1: Energy-efficient bespoke robust processing: *How can we incorporate accurate, differentiable power models into the training of analog and spiking NNs to enforce strict power envelopes, particularly for temporal processing tasks on ultra-low-power platforms?*

RQ2: Architecture-aware design: *How can we automatically synthesize analog neuromorphic architectures on large-area substrates that remain robust under process variations, environmental drift, and sensor noise, while respecting tight area and device-count budgets?*

RQ3: Fault sensitivity, error correction, and endurance: *How can we characterize and mitigate the impact of manufacturing defects and wear-out in printed and flexible NNs using lightweight AEC and adaptive fault tolerance, thereby extending effective lifetime?*

RQ4: Security, analysis, and trust: *How can we analyze and harden the security of flexible NNs and related emerging-memory-based systems against side-channel and fault attacks, and how can we integrate test and diagnostic capabilities that are compatible with large-area, analog-centric designs?*

These research questions are tightly coupled: architecture and training choices affect both reliability and security, and diagnostic capabilities feed back into endurance and adaptive reconfiguration strategies. The thesis therefore adopts a holistic, cross-layer approach.

1.3.2. Thesis Contributions and Structure

To answer these research questions, the thesis makes a series of concrete contributions, which are organized into thematic contribution chapters:

- **Chapter 2** introduces the necessary background on flexible and PE, NN models (ANNs, RNNs, SNNs), analog NNs, error correction, reliability, test, and security. It positions the thesis relative to the state of the art.
- **Chapter 3** (*Power and Energy-Efficient Robust NN Design*) tackles **RQ1** by integrating (power-aware training (*P*-AT)) and (power-constrained training (*P*-CT)) with event-driven and temporal processing primitives, including spiking NNs and on-sensor mixed-signal designs with bespoke ADCs and decision trees.
- **Chapter 4** (*Architecture-Aware Robust NN Design*) addresses **RQ2** through *NAS*, evolutionary design, and thermally resilient architecture search for large-area NNs on both printed n-EGT and flexible a-IGZO TFT technologies.
- **Chapter 5** (*Reliability, Endurance, and Test*) focuses on **RQ3**, covering fault sensitivity analysis of printed analog and digital NN classifiers, AEC for unary-encoded computing, adaptive endurance frameworks such as PRINT-SAFE, and manufacturing test for large-area NNs and related analog blocks (e.g., flexible *LDO*s).
- **Chapter 6** (*Security Analysis and Trust*) addresses **RQ4**, providing side-channel vulnerability analysis of flexible NN architecture, security frameworks for printed and flexible neuromorphic hardware.
- **Chapter 7** concludes the thesis, summarizes the main findings, and outlines future research directions, including deeper neuromorphic architectures, online learning, and tighter integration of diagnostics and security with neuromorphic design.

This mapping from research questions to chapters ensures that each objective is supported by a coherent set of methods, experiments, and hardware-aware analyses.

1.4. Methodology: Cross-Layer Co-Optimization

To address the research questions, this thesis develops a *cross-layer design and optimization framework* for large-area analog neuromorphic hardware. The core methodological elements span from device and circuit modeling to architecture search, training, test, and security evaluation as shown in Figure 1.2.

1.4.1. Power and Energy-Efficient Robust NN Design (Ch. 3)

For **RQ1**, we derive accurate power models for crossbar arrays and nonlinear activation circuits, including spiking elements and event-driven primitives. These models are differentiable and are integrated into the training objectives of analog and spiking networks:

- **Power-aware training**: Joint optimization of accuracy and power, resulting in Pareto fronts of energy-accuracy trade-offs for large-area NNs.
- **Power-constrained training**: Augmented Lagrangian methods that enforce hard power budgets during training, ensuring operational compliance with strict energy envelopes.

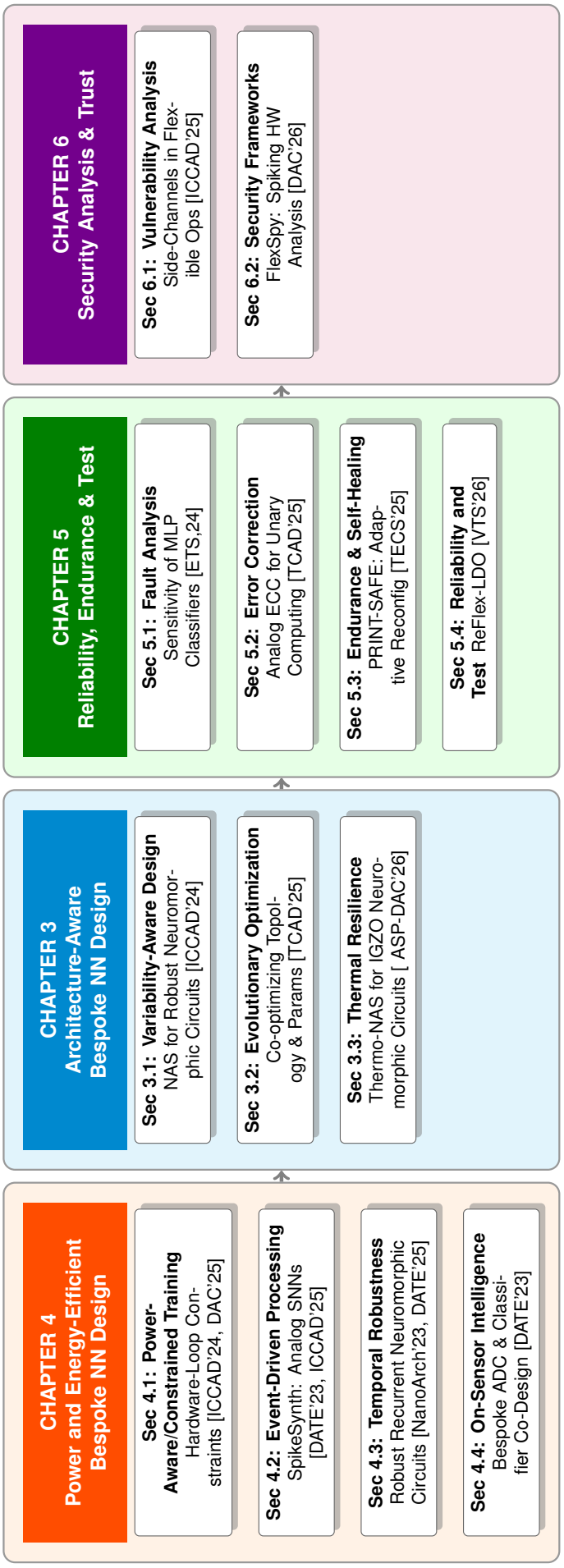


Figure 1.2: Structure of the dissertation, highlighting the four core contribution pillars mapping to Chapters 3–6.

- **Temporal and event-driven processing**: Development of large-area analog spiking circuits with learnable spike generators and temporal filters that enable sparse, event-driven computation under stringent power limits.
- **On-sensor mixed-signal intelligence**: Co-design of bespoke ADCs and classifiers, reducing interface overhead and allowing analog-domain pre-processing before digitization.

1.4.2. Architecture-Aware NN Design (Ch. 4)

For **RQ2**, we model large-area NNs using differentiable surrogate models that capture the mapping from physical circuit parameters (e.g., crossbar conductances, transistor sizes) to functional behavior. On top of this, we perform:

- ***NAS***: An automated search over network topologies, activation circuit families, and analog operating ranges, guided by variation-aware objectives that account for process-induced perturbations.
- **Evolutionary co-optimization**: Evolutionary algorithms that jointly optimize architecture and analog parameters for compactness and robustness, refining Pareto trade-offs between area, accuracy, and variation tolerance.
- **Thermal and temporal resilience**: Augmented search procedures that incorporate self-heating and ambient temperature models, as well as learnable temporal filters to mitigate sensor noise and device variability.

1.4.3. Reliability, Endurance, and Test (Ch. 5)

For **RQ3**, we build fault and defect models at the device and circuit levels for large-area analog and digital NN classifiers:

- **Fault sensitivity analysis**: Monte Carlo based evaluation of stuck-open/stuck-short faults in analog NNs and transistor- or resistor-level faults in digital implementations, across multiple datasets.
- **Analog error correction**: Efficient AEC schemes targeting unary-encoded computations, designed to significantly reduce error rates with minimal digital overhead.
- **Adaptive endurance (PRINT-SAFE)**: A scalable framework that monitors fault patterns and dynamically reconfigures or derates hardware to extend operational lifetime, enabling self-healing behavior.
- **Manufacturing test**: Delay-based test strategies for flexible power management blocks, and extensions toward test of analog front-ends of emerging technologies.

1.4.4. Security Analysis and Trust (Ch. 6)

For **RQ4**, we conduct the first systematic side-channel and fault analysis of flexible NNs:

- **Side-channel vulnerability analysis of flexible NNs**: Modeling and evaluation of power-based leakage in both analog and digital TFT-based NNs, including CNN-based regression attacks and correlation power analysis.
- **Security frameworks for spiking neuromorphics**: FlexSpy-style frameworks tailored to flexible spiking hardware, capturing continuous-time leakage behavior and attack surfaces.

Throughout the thesis, physics-informed simulation and, where available, experimental data from *Institute of Nanotechnology (INT)* are used to validate the proposed techniques on a diverse set of benchmark tasks and realistic hardware models. The co-optimization across algorithms, circuits, and technologies is a central theme: decisions at one layer are always evaluated in terms of their impact on energy, robustness, testability, and security at the others.

1.5. Chapter Summary

This chapter motivated the move from traditional silicon-based SoCs to large-area analog computing platforms for ultra-resource-constrained edge intelligence. It outlined the unique opportunities offered by PE and FE, including ultra-low-cost, mechanically compliant integration with sensors and energy harvesters, and highlighted the coupled challenges of variability, energy efficiency, fault tolerance, test, and security.

Based on these challenges, four central research questions were formulated, spanning architecture and variability-aware design, energy-efficient temporal processing, fault sensitivity and endurance, and security, test, and diagnostics. To address these questions, the chapter introduced a cross-layer methodology that unifies *NAS*, analog-aware training, reliability frameworks, and security analysis within a single design flow, and mapped these methods and objectives to the contribution chapters of the thesis.

The next chapter provides the background and related work needed to understand the technological and methodological context of this thesis.

2. Background and Related Work

This chapter provides the technological and methodological background of the thesis. Section 2.1 - Section 2.2 introduces the large-area and flexible device platforms that enable ultra-low-cost, mechanically compliant edge hardware. Section 2.3 revisits the NN models that use edge intelligence, including feed-forward ANNs, temporal RNNs models, and SNNs. Section 2.3.1 then discusses analog NN circuit primitives and their realization on printed and flexible substrates, followed by other analog and mixed-signal computing paradigms in Section 2.4. Sections 2.5–2.6 survey prior work on reliability, test, and security of printed and flexible NN systems. Finally, Section 2.7 summarizes the main trends and outlines how the thesis builds on and extends the state of the art.

Large-area electronics refers to technologies in which devices are fabricated on extended, often flexible substrates such as polymer foils, paper, or thin metal films. Rather than maximizing integration density, these platforms emphasize low cost, mechanical compliance, and compatibility with unconventional form factors. Two families of technologies are particularly relevant for this thesis:

- *Printed solution-processed electronics*, where devices and interconnects are formed by additively depositing functional inks.
- *Flexible thin-film electronics* based on amorphous oxide TFTs fabricated at low temperature on polymer substrates.

Both technologies trade off device performance and integration density against cost and mechanical properties, and have been widely explored for applications such as smart labels, wearables, and soft robotics.[86], [103]

2.1. Printed Electronics (PE)

PE refers to the fabrication of electronic devices and circuits by patterning *functional inks* (conductors, semiconductors, and dielectrics) directly onto a substrate. In contrast to conventional silicon microelectronics, which relies on wafer-based lithography, rigid substrates, and high-cost infrastructure—PE emphasizes simplified manufacturing, broader substrate compatibility, and scalable large-area fabrication. This makes PE particularly attractive for applications where flexibility, lightweight form factors, and low cost-per-area dominate over maximum switching speed and integration density (e.g., flexible systems, large-area sensor interfaces, and ultra-low-cost electronics)[131], [134].

2.1.1. Additive and subtractive fabrication in PE

PE process flows can be grouped into *subtractive* and *fully additive* strategies. In subtractive approaches, layers are first deposited as continuous films and then patterned using steps analogous to microfabrication (e.g., resist processing and etching). While subtractive patterning can provide good feature definition, it typically increases process complexity and infrastructure requirements [134], [269]. Fully additive printing instead constructs device stacks through sequential deposition of inks only where material is needed, enabling simplified workflows and reduced non-recurring engineering effort, especially for rapid prototyping and low-to-medium volume fabrication [131], [134].

2.1.1.1. Cost–performance positioning and hybrid integration

A key limitation of PE is its comparatively large feature size, typically in the micrometer range. Consequently, printed circuits generally offer lower switching performance and lower functional density

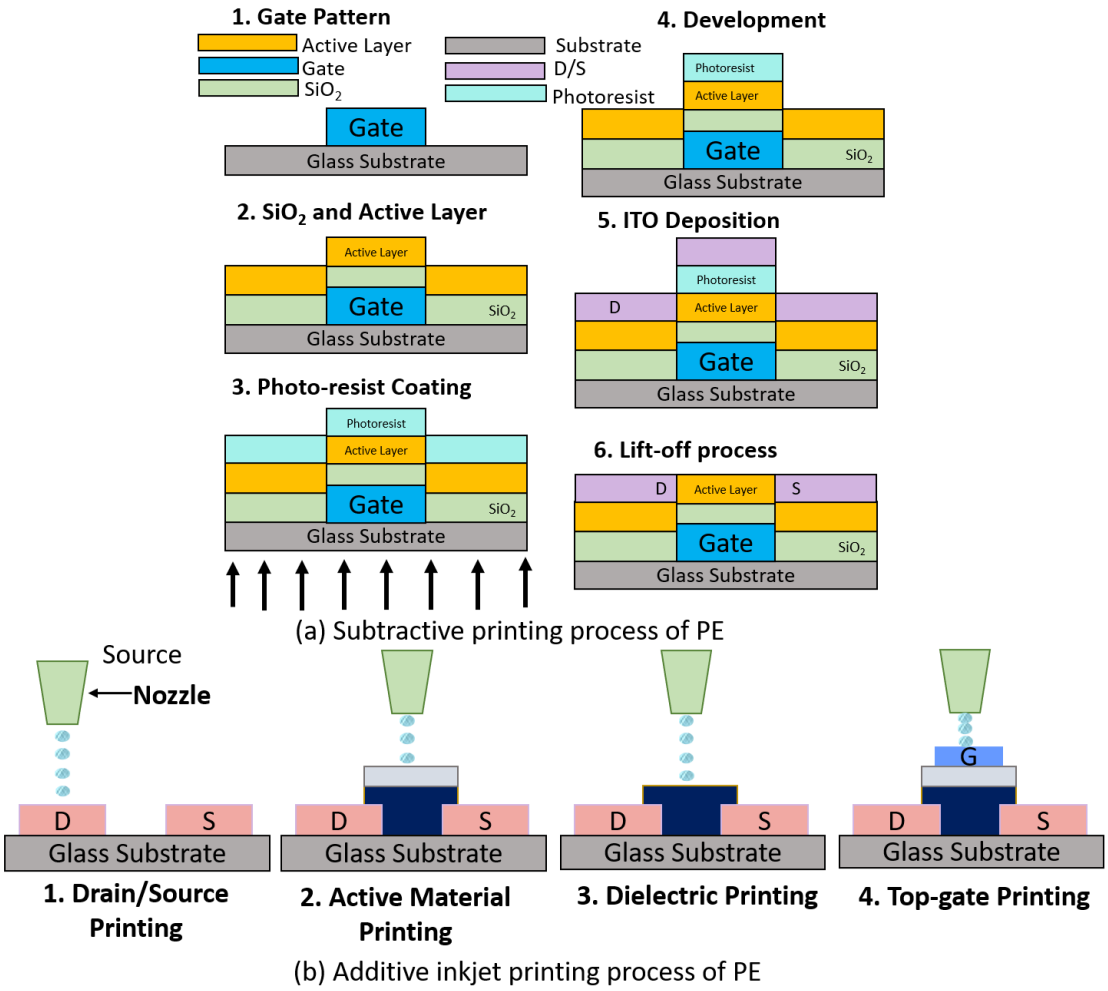


Figure 2.1.: Comparison between subtractive-based patterning and fully additive printed electronics fabrication. Subtractive approaches introduce additional steps such as resist processing and etching, whereas additive approaches build devices by layer-by-layer ink deposition.

than nanometer-scale silicon technology. However, PE can provide substantial advantages in manufacturing simplicity, substrate choice, and cost-per-area. Because neither PE nor silicon is universally optimal across all metrics, PE is commonly positioned as a *complementary* technology. This motivates hybrid systems that combine printed circuits (for low-cost, flexible, or large-area functions) with silicon components (for high-performance computation and dense integration) [130], [200].

2.1.2. Printing Methods for PE

PE fabrication is implemented using a variety of printing processes. Commonly used approaches include inkjet printing (jet printing), screen printing, and gravure printing. These processes differ in whether patterns are generated digitally or via physical masters, and they impose different constraints on ink viscosity, resolution, and throughput [151], [269].

2.1.2.1. Inkjet printing

Inkjet printing is a digital, maskless deposition technique in which droplets are ejected from a nozzle and placed on the substrate under software control. Droplet generation is typically actuated (e.g., via piezoelectric elements), enabling patterns to be printed by coordinating ejection timing with printhead motion. Since printing is contactless, inkjet printing supports a wide range of substrates (including

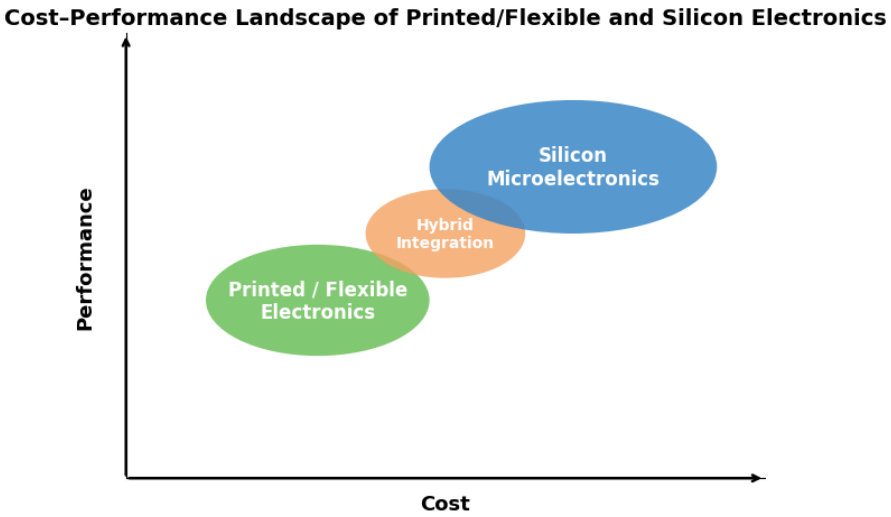


Figure 2.2.: Qualitative cost–performance landscape highlighting PE/FE as a complementary technology to silicon microelectronics and motivating hybrid integration.

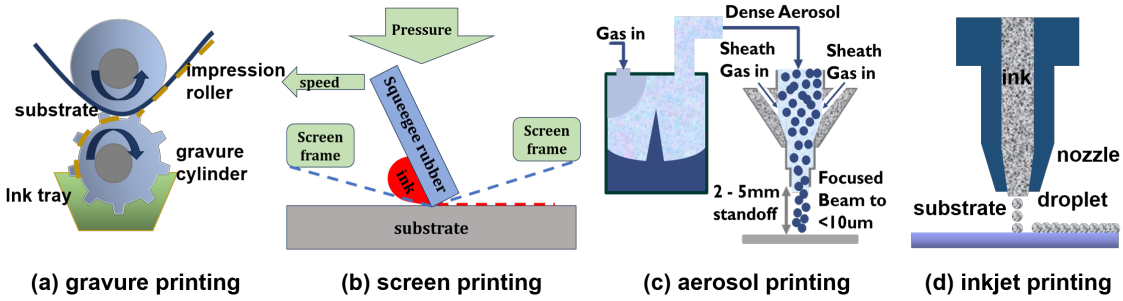


Figure 2.3.: Schematics of common PE processes: gravure printing (engraved cylinder transfer), screen printing (stencil-based transfer), aerosol printing and inkjet printing (digital droplet deposition).

flexible and rough surfaces) and enables rapid design iteration without manufacturing physical masters, reducing setup cost and facilitating application-specific customization [151].

2.1.2.2. Screen printing

Screen printing is an analog process that transfers ink through a patterned mesh (stencil) onto the substrate. A blade forces ink through open regions of the screen to form the printed pattern. Screen printing is widely used industrially due to its relatively simple equipment and low process cost; however, it typically provides lower resolution than other methods and often requires highly viscous inks, which can degrade electrical performance depending on the material system [269].

2.1.2.3. Gravure printing

Gravure printing is an analog process in which patterns are defined by an engraved cylinder. Ink fills the engraved features and excess ink is removed using a doctor blade; during rolling contact, ink transfers to the substrate. Gravure printing supports low-viscosity inks and provides strong control over ink usage and repeatability across very large print runs. Its main limitation is the cost of master/cylinder fabrication, making it most economical for large batch sizes [269].

2.1.2.4. Resolution–throughput trade-offs

Printing methods trade off achievable resolution against manufacturing throughput. Gravure printing is typically associated with fine resolution and high throughput (often compatible with roll-to-roll production), whereas inkjet and screen printing are commonly used for low-to-medium volume fabrication and larger feature sizes. Substrate compatibility also varies; multiple methods support plastic and paper, while inkjet and screen printing can additionally be applied to rigid substrates such as glass depending on materials and system configuration [134], [215].

Table 2.1.: Key properties and parameters of different printing methods

Property/Parameter	Gravure	Screen	Aerosol	Inkjet
Throughput (m^2/s)	3–60	2–3	0.01–0.1	0.01–0.5
Resolution (lines/cm)	20–400	50	10–100	60–250
Printing Speed (m/min)	100–1000	10–15	5–50	15–500

2.1.3. Inkjet Printing for On-Demand, Maskless Fabrication

Among PE processes, inkjet printing is particularly attractive for research and application-specific manufacturing because it enables on-demand fabrication without physical masks or masters. This drastically reduces non-recurring engineering cost and allows circuit layouts to be customized and produced in low batch sizes. Desktop materials printers enable "fab-in-the-box" style workflows, where rapid iteration and bespoke layouts can improve footprint and system-level integration [199]. A broad range of printed building blocks has been demonstrated using inkjet-printed materials (e.g., transistors, logic, and mixed-signal blocks), but supply voltage requirements in many printed technologies can still span from a few volts to significantly higher values, which is problematic for systems powered by printed batteries or energy harvesters [182], [235].

2.1.4. Electrolyte-Gated Transistors (EGTs)

EGTs are a promising device class for low-voltage printed electronics. Instead of relying on a conventional solid dielectric, EGTs use an electrolyte as a dielectric substitute. The electrolyte/semiconductor interface can provide very high effective gate capacitance, enabling low threshold voltage operation. As a result, EGTs can be operated at sub-volt supply levels, which is attractive for portable printed systems and for integration with low-voltage energy sources [197].

2.1.5. Device Fabrication Flow

EGTs can be fabricated using a small number of sequential, inkjet-printable process steps. In the metal-oxide EGFET flow considered in this work, the device stack is formed by (i) patterned electrodes and interconnects, (ii) an inkjet-printed indium-oxide semiconductor channel, (iii) an inkjet-printed composite solid polymer electrolyte (CSPE), and (iv) an inkjet-printed conductive polymer gate.

Substrate preparation and patterning. A common starting point is an ITO-coated glass substrate, where the drain, source, and gate electrodes as well as passive structures (e.g., resistors and interconnects) are defined by patterning the ITO layer [197]. After structuring, the substrate is cleaned to remove residues prior to printing the functional layers.

Semiconductor channel printing and conversion. The transistor channel is formed by inkjet printing an indium-nitrate precursor ink into the gap between drain and source electrodes while applying mild stage heating (e.g., around 60 °C) [197]. A representative formulation uses 0.05 M indium nitrate hydrate dissolved in double-deionized water and glycerol (4:1), followed by filtration prior to printing [173], [197].

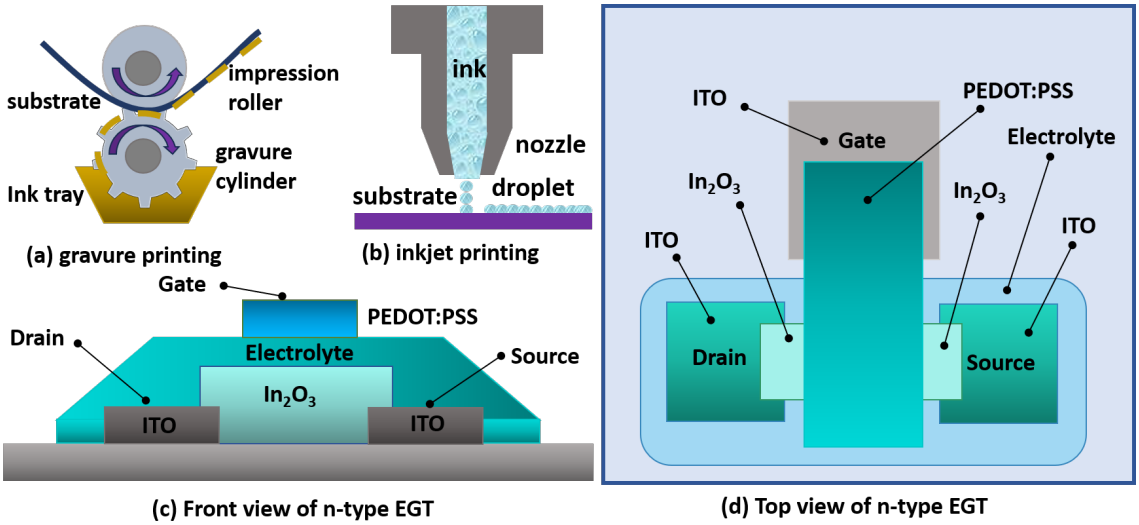


Figure 2.4.: Schematic of a printed n-EGT: (a) top view and (b) front view.

The printed precursor is converted into In_2O_3 through thermal annealing (e.g., 400 °C for 2 h), where controlled ramp-up and cool-down can be used to mitigate crack formation and thermal stress [197]. While the high temperature supports improved oxide quality, it restricts substrate selection; lower-temperature chemical or photonic curing routes can relax substrate constraints at the expense of reduced mobility [197].

Electrolyte (CSPE) printing. Subsequently, a CSPE is inkjet-printed on top of the semiconductor channel to serve as the gate dielectric [197]. A typical preparation dissolves PVA in DMSO at elevated temperature and dissolves $LiClO_4$ in propylene carbonate at room temperature; the solutions are mixed until homogeneous, filtered, and printed with moderate stage heating (e.g., around 40 °C) [173], [197].

Gate formation: in-plane vs. top gate. The gate electrode can be realized as an in-plane gate (laterally displaced from the channel) or as a printed top gate above the CSPE. For in-plane gates, the electrolyte-covered gate area should be significantly larger than the channel area to maintain efficient gating [197]. In many cases, the gating efficiency is maximized by printing a top gate on the CSPE using a PEDOT:PSS-based ink, which avoids high-temperature post-treatment that could degrade the electrolyte [197]. Foundational work on printed metal-halide oxide EGTs further demonstrates that In_2O_3/CSPE devices can achieve low-voltage operation with strong electrostatic control [55].

Applications and system-level characteristics PE have been explored for RFID tags, smart packaging and labels, printed sensors, and disposable medical devices.[60], [64], [82], [100], [122], [138], [143], [183], [192], [215], [219] Cui's book surveys materials and devices for applications such as energy harvesters, displays, and large-area sensor arrays.[86] Typical device counts range from a few tens to a few thousand transistors per circuit, making PE attractive for application-specific, small-footprint analog blocks rather than complex programmable systems.

Relation to NN and ML hardware

More recently, PE have been proposed as a platform for neuromorphic computing and ML accelerators. Nawrocki *et al.* demonstrated polymer-based neural units using memristive devices and polymer transistors,[69] and Weller *et al.* introduced programmable neuromorphic circuits based on inorganic printed n-EGTs.[175], [202] These works paved the way for the printed NN hardware and spiking NNs that later chapters of this thesis will build upon.

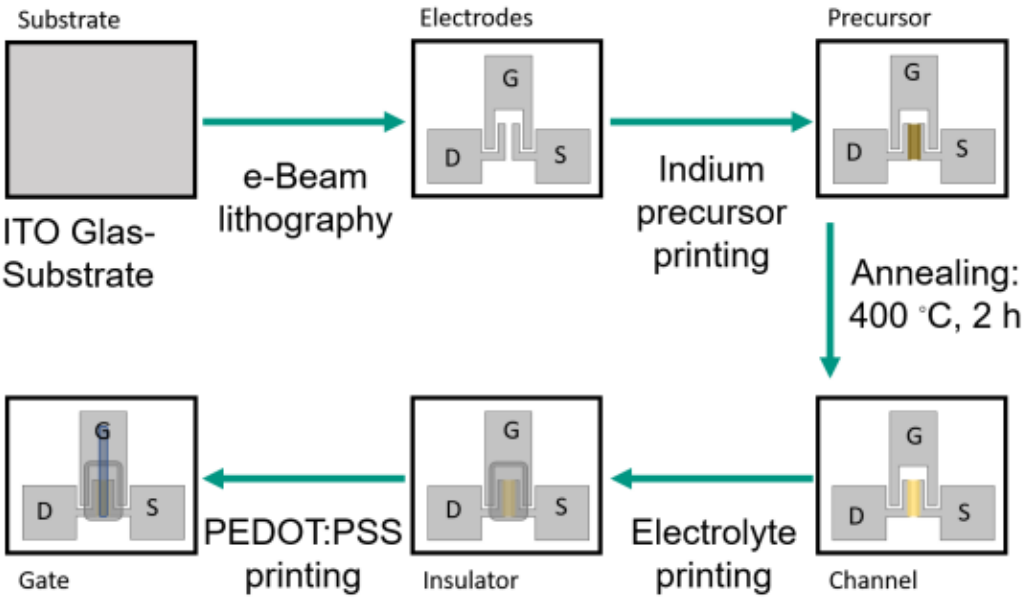


Figure 2.5.: Fabrication methodolgoy for printed n-EGTs with In_2O_3 channel. Adapted from [197].

2.2. Flexible Electronics (FE)

Flexible TFTs form the technological backbone of large-area electronics, where the primary requirement is not aggressive feature-size scaling but rather the ability to distribute electronics across very large surfaces and, increasingly, across mechanically flexible substrates. Historically, TFT development was strongly driven by display backplanes, where manufacturing area and cost-per-area dominate over minimum transistor dimensions. This "large-area" design space has expanded substantially beyond displays to include flexible sensor interfaces, wearable health patches, identification tags, and other systems where the circuit footprint is inherently set by the application geometry rather than by integration density.

A key advantage of TFT platforms is that the active semiconductor is deposited as a thin film onto a carrier (e.g., glass or polymer), enabling electronics on substrates that are incompatible with conventional bulk-silicon processing. In practice, this shifts the design focus from maximum speed and density towards manufacturability, mechanical robustness, and achieving sufficient electrical performance under tight processing temperature constraints [127], [190], [216].

In contrast to bulk silicon MOSFET technologies—where the transistor is formed by patterning and doping a crystalline wafer—TFTs are realized by depositing the semiconductor film on top of a carrier. This allows transistor fabrication on a broad set of substrates, including rigid (silicon, glass) and flexible (polyimide, PET) carriers, subject to the thermal budget and process compatibility [190], [216].

A generic TFT can be viewed as a gated conduction channel formed in a deposited semiconductor film contacted by source/drain electrodes and insulated from the gate by a dielectric layer. Since the active semiconductor is not part of a bulk body, TFTs are often treated as more "geometrically symmetric" devices from a high-voltage standpoint, which can be beneficial when large voltage swings are required.

2.2.1. Design implications for flexible TFT circuits

Two practical aspects strongly shape circuit design in flexible TFT technologies.

Interconnect limitations. Compared to advanced CMOS, TFT processes frequently provide fewer metal layers, and the back-end stack is typically less optimized due to cost and process constraints. The limited routing resources can become an area bottleneck, especially for circuits requiring dense interconnect or strict pitch matching to sensors/actuators.

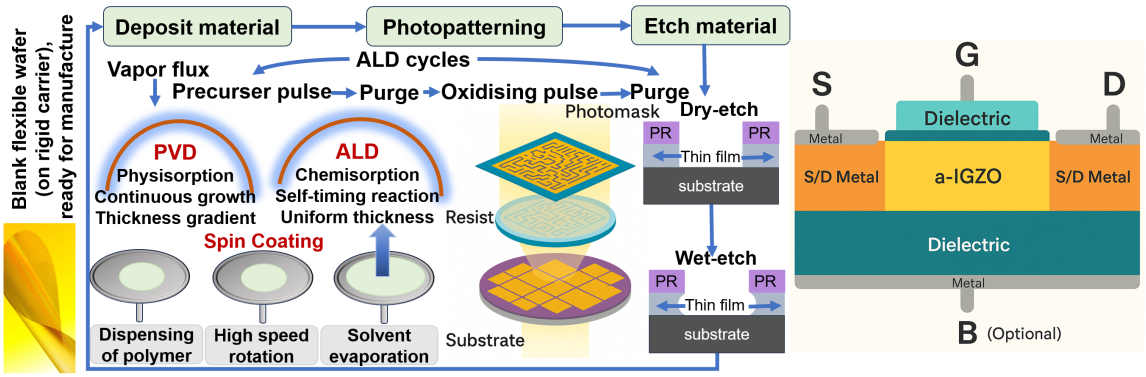


Figure 2.6.: Fabrication flow of a-IGZOs TFT, including material deposition (PVD, ALD, spin coating), photo-patterning, and etching. Insets show flexible substrate, and device structure [272].

Variability, and model maturity. While CMOS design flows benefit from mature PDKs and statistical compact models, thin-film technologies often provide less mature modelling infrastructure. Consequently, robust circuit techniques (guard-banding, calibration, variation-tolerant topologies) become important, but typically trade off area, speed, and/or power [127], [190], [216].

2.2.2. Semiconductor options for flexible TFT technologies

The semiconductor material largely determines achievable mobility, stability under bias stress, polarity (unipolar vs. complementary), and process temperature—all of which are central to FE.

2.2.2.1. Organic semiconductors

Organic TFTs are attractive for flexible systems due to low processing temperatures and the potential for low-cost manufacturing. However, many organic technologies are predominantly unipolar, and their electrical characteristics can be sensitive to the environment, requiring encapsulation for long-term stability. Vertical organic device concepts have also been reported to enable complementary circuit blocks, which is appealing for low static power digital logic [167], [189].

2.2.2.2. Amorphous silicon (a-Si)

Hydrogenated amorphous silicon has historically been a major workhorse for display backplanes. Its key limitations for flexible/high-performance circuits are relatively low carrier mobility and pronounced stability effects under prolonged bias, which can translate into drift in circuit operating points [45].

2.2.2.3. Metal-oxide TFTs (e.g., a-IGZO)

Metal-oxide TFTs, and a-IGZO in particular, have become central in flat-panel manufacturing due to their favourable balance of mobility, stability, and compatibility with large-area fabrication. These devices also support transparent electronics and can be processed within temperature windows suitable for various flexible substrates, while offering strong resilience to mechanical strain in relevant process stacks [46], [104], [128], [214].

A commonly cited limitation is that many oxide TFT platforms are predominantly n-type, which complicates fully complementary logic. Although complementary oxide approaches exist, they often introduce either reduced mobility or performance mismatch between n- and p-type devices [46].

2.2.2.4. Carbon nanotube (CNT) TFTs

CNT TFTs are promising for FE because they can be processed at low temperatures and are compatible with printed/direct-write approaches. Importantly, CNT technologies can offer n-type, p-type, and even

Table 2.2.: Qualitative comparison of TFT semiconductor options for flexible/large-area electronics.

Platform	Typical strengths	Typical limitations
Organic	Low-temperature processing; mechanically compliant; potentially low-cost	Often unipolar; environmental sensitivity; stability/aging concerns [167], [189]
a-Si	Mature in display manufacturing; low-cost on glass	Low mobility; bias-stress related drift; limited suitability for high-performance flexible circuits [45]
Metal-oxide (a-IGZO)	Good mobility/stability balance; large-area compatible; good off-current; flexible compatibility	Frequently n-type dominant; complementary integration remains challenging in many stacks [46], [214]
CNT	Low-temperature / printable; ambipolar possible; promising for flexible logic	Variability, hysteresis, and long-term bias stress challenges [85], [145]
LTPS	High mobility; complementary feasible; higher circuit performance	Higher process complexity/cost; uniformity constraints for very large panels [37], [70]

ambipolar behaviour, enabling compact logic styles. Key challenges remain in controlling device-to-device variability and ensuring long-term stability under bias stress [85], [118], [145], [180].

2.2.2.5. Low-temperature poly-silicon (LTPS)

LTPS TFTs offer among the highest mobilities in flat-panel compatible technologies and can support complementary device integration, which is valuable for dense and power-efficient digital circuitry. LTPS has therefore been adopted in high-performance display backplanes and is also demonstrated on plastic substrates. These advantages come with higher process complexity and manufacturing cost, and practical uniformity constraints that can limit scalability to very large panel sizes [20], [37], [70].

Table 2.2 summarizes the key trade-offs of the most common TFT material platforms used in flexible and large-area electronics. The "best" choice is application-driven: performance-demanding large-area systems may favour LTPS or oxide TFTs, while ultra-low-cost and mechanically compliant systems may leverage organics or CNT-based approaches when their variability and stability constraints can be managed.

From a circuit designer's perspective, flexible TFT technologies require a careful balance between (i) performance targets (speed, voltage swing, noise, leakage), (ii) manufacturability and cost-per-area, (iii) mechanical reliability, and (iv) the maturity of compact models and design kits. As a result, robust circuit techniques and architecture choices are not an optional "nice-to-have" but a core part of successful thin-film system design [127], [190].

Relation to this thesis

Recent works have begun to exploit a-IGZO TFTs for neuromorphic, NN hardware and ML tasks. Ozer *et al.* presented a bespoke ML processor on flexible substrates [146] and a hardwired ML engine fabricated with submicron metal-oxide TFTs on flexible foils, [171] demonstrating that modest-size classifiers can be implemented with limited device resources. Lebanov *et al.* proposed flexible unipolar a-IGZO integrate-and-fire neurons for spiking neuromorphic applications, [250] and Giustolisi *et al.* designed

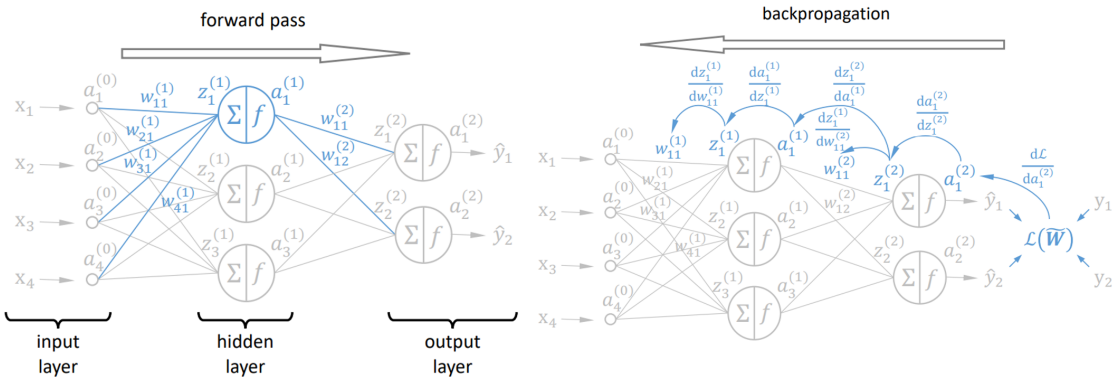


Figure 2.7.: (Left) Forward pass of a 4–3–2 multilayer perceptron, mapping four inputs $(x_1, \ldots, x_4)$ to two outputs $(\hat{y}_1, \hat{y}_2)$. Circles denote neurons performing weighted-sum and nonlinear activation, edges represent weights, and the blue path highlights the forward computation of a single neuron. (Right) Backpropagation in a multilayer perceptron, illustrating gradient propagation from the loss $\mathcal{L}$ to network weights (e.g., $w_{11}^{(1)}$ and $w_{11}^{(2)}$). The blue path highlights one backpropagation chain.

a three-stage operational transconductance amplifier in a TFT flexible-substrate process, highlighting analog building blocks for future systems[246]. These studies show that flexible oxide TFTs can host both digital and analog neuromorphic circuits, but they also underline the strong impact of temperature and bias-stress effects on circuit behavior, motivating the thermal-aware design methodologies addressed later in the thesis.

The prior literature establishes printed n-EGT and flexible a-IGZOs TFT platforms as promising but variability-prone substrates for neuromorphic circuits and ML accelerators. The thesis will build on these foundations by developing architecture, training, reliability, and security frameworks that specifically target these technologies under the extreme power, area, and cost constraints outlined in Chapter 1.

2.3. Neural Network (NNs) Computing Models for Edge Intelligence

Brain-inspired computation as shown in Figure 2.8 has been discussed since the early foundations of computer science and digital architectures, including the era of Alan Turing and John von Neumann [4], [8]. In modern terminology, *NN computing models* refer to computing paradigms that draw inspiration from information processing in biological nervous systems [112]. A central architectural distinction from the classical von Neumann design is the *tight coupling of storage and computation*, often described as processing-in-memory (PIM) [226]. By reducing the repeated data transfers between separated processing and memory units, such architectures mitigate bandwidth limitations and data-movement overheads that commonly limit performance and energy efficiency in conventional systems [68]. This approach naturally supports massive parallelism and can be advantageous for workloads that resemble neural processing, including applications in neuroscience and ML [42], [95].

Multiple neural-inspired models exist, spanning a spectrum of biological fidelity. Commonly referenced examples include the Hodgkin–Huxley model, the FitzHugh–Nagumo model, membrane-dynamics-based formulations, integrate-and-fire (I&F) neurons, and the McCulloch–Pitts abstraction [2], [5], [9], [23], [43]. While the more detailed models can closely represent biophysical processes, the simpler abstractions are often preferred when computational cost and scalable training are primary concerns.

Two simplified model families are particularly relevant to contemporary learning systems. The I&F model is widely used as a basis for SNNs, which are commonly regarded as *Neuromorphic* neural models [24]. In contrast, the McCulloch–Pitts formulation underpins many standard ANNs designs, including feed-forward MLPs [35].

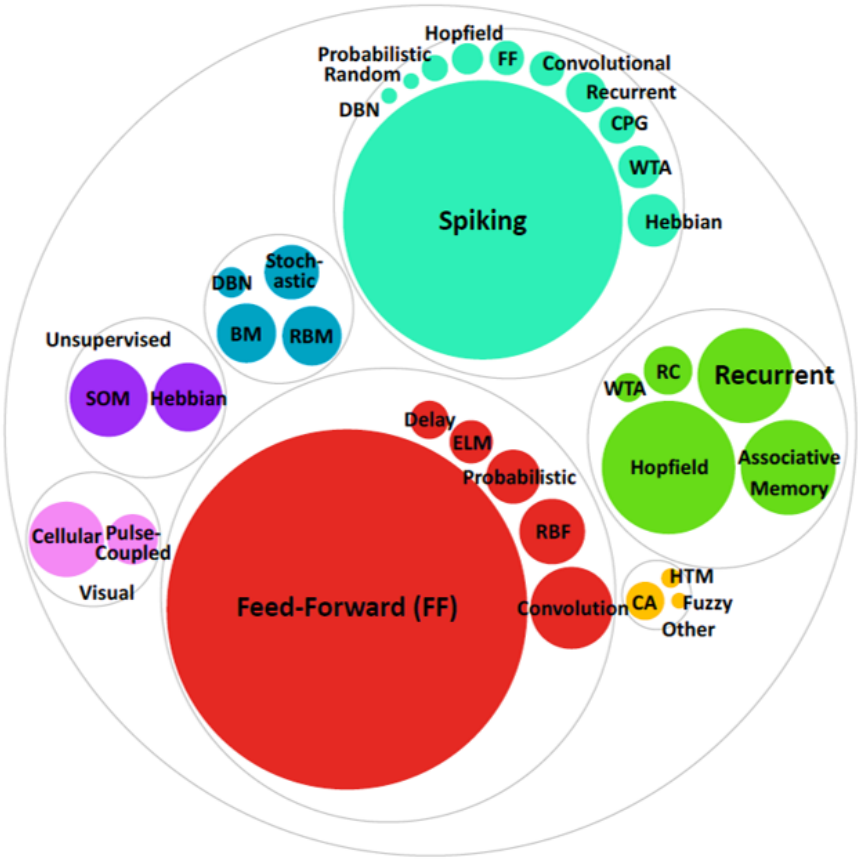


Figure 2.8.: A breakdown of network models in neuromorphic implementations, grouped by overall type and sized to reflect the number of associates papers. Sourced from [112] (2017).

2.3.1. Feed-forward Multilayer Perceptron-Style Computation

Among Neuromorphic Computing (NC)[1] models, feed-forward MLP-style computation has received extensive attention due to its operational simplicity, strong empirical performance, and compatibility with efficient implementations [35]. In hardware contexts, specialized devices and circuits can serve as accelerators for such networks [137]. Accordingly, systems that implement MLP-like computation directly in hardware are sometimes described as *hardware MLPs* [30]. The following sections summarize the MLP formalism and then discuss how these computations can be adapted to printed circuit platforms.

A typical MLP is composed of an input layer, one or more hidden layers, and an output layer. Each neuron forms a weighted combination of inputs from the previous layer and then applies a nonlinear activation function. For the l-th layer, a common mathematical description is

$$\mathbf{z}^{(l)} = \mathbf{a}^{(l-1)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}, \tag{2.1}$$

$$\mathbf{a}^{(l)} = f\left(\mathbf{z}^{(l)}\right), \tag{2.2}$$

where $\mathbf{a}^{(l-1)} \in \mathbb{R}^{1\times N_{l-1}}$ is the previous-layer output (with N_{l-1} neurons), $\mathbf{W}^{(l)} \in \mathbb{R}^{N_{l-1}\times N_l}$ is the weight matrix, $\mathbf{b}^{(l)} \in \mathbb{R}^{1\times N_l}$ is the bias vector, and $f(\cdot)$ is a nonlinear activation (e.g., ReLU, sigmoid, or tanh) [113].

[1] Here, in these works, the term "neuromorphic" is used broadly to indicate analog computational circuits inspired by neural network architectures, using nonlinear AFs and resistor crossbar arrays for parallel computation[202].

Bias absorption into an augmented weight matrix. For notational convenience, biases are often folded into an extended weight matrix by augmenting the input with a constant 1:

$$\mathbf{z}^{(l)} = \underbrace{\begin{bmatrix}\mathbf{a}^{(l-1)} & 1\end{bmatrix}}_{:=\tilde{\mathbf{a}}^{(l-1)}} \underbrace{\begin{bmatrix}\mathbf{W}^{(l)} \\ \mathbf{b}^{(l)}\end{bmatrix}}_{:=\tilde{\mathbf{W}}^{(l)}}. \tag{2.3}$$

This representation consolidates all trainable parameters of a layer into a single matrix $\tilde{\mathbf{W}}^{(l)} \in \mathbb{R}^{(N_{l-1}+1)\times N_l}$.

Mini-batch processing. When processing a mini-batch of size B, inputs can be stacked into a matrix $\tilde{\mathbf{A}}^{(l-1)} \in \mathbb{R}^{B\times(N_{l-1}+1)}$, producing $\mathbf{Z}^{(l)} \in \mathbb{R}^{B\times N_l}$. Over L layers, the network mapping can be expressed compactly as

$$\hat{\mathbf{Y}}(\tilde{\mathbf{W}}) = f\Big(\cdots f\Big(f(\tilde{\mathbf{X}}\tilde{\mathbf{W}}^{(1)})\tilde{\mathbf{W}}^{(2)}\Big)\cdots\tilde{\mathbf{W}}^{(L)}\Big), \tag{2.4}$$

where $\tilde{\mathbf{X}}$ is the augmented input matrix and $\tilde{\mathbf{W}} = \bigcup_{l=1}^{L}\tilde{\mathbf{W}}^{(l)}$ denotes the full set of trainable parameters.

2.3.2. Training of MLPs

Historically, early neural-inspired circuits often used fixed or randomly chosen parameters [3], [7]. In contrast, modern learning systems typically train $\tilde{\mathbf{W}}$ using gradient-based optimization, most notably backpropagation [13]. Training can be formulated as minimizing a loss function that quantifies mismatch between the network output $\hat{\mathbf{Y}}$ and the target labels $\mathbf{Y}$ in a dataset.

Regression losses. For regression, a standard choice is the mean squared error (MSE):

$$\mathcal{L}(\tilde{\mathbf{W}}) = \frac{1}{B\,N_L}\left\|\hat{\mathbf{Y}}(\tilde{\mathbf{W}}) - \mathbf{Y}\right\|_F^2, \tag{2.5}$$

where $\|\cdot\|_F$ is the Frobenius norm [49].

Classification losses. For classification, accuracy is discrete and not directly suitable for gradient descent. A widely used differentiable surrogate is the cross-entropy loss [233]. Using a one-hot encoding $\mathbf{Y}_{\mathrm{OH}}$ of the labels, one common form is

$$\mathcal{L}(\tilde{\mathbf{W}}) = -\frac{1}{B}\Big(\mathbf{Y}_{\mathrm{OH}} \odot \log\hat{\mathbf{Y}}(\tilde{\mathbf{W}}) + (\mathbf{1} - \mathbf{Y}_{\mathrm{OH}}) \odot \log\left(\mathbf{1} - \hat{\mathbf{Y}}(\tilde{\mathbf{W}})\right)\Big), \tag{2.6}$$

where $\odot$ denotes elementwise multiplication and $\mathbf{1}$ is an all-ones matrix of matching size. In typical classifiers, the number of output neurons equals the number of classes, and prediction is obtained by selecting the index of the largest output activation.

Gradient descent and backpropagation. Optimization is performed iteratively, e.g.,

$$\tilde{\mathbf{W}} \leftarrow \tilde{\mathbf{W}} - \alpha\nabla_{\tilde{\mathbf{W}}}\mathcal{L}(\tilde{\mathbf{W}}), \tag{2.7}$$

with learning rate $\alpha > 0$. Backpropagation computes $\nabla_{\tilde{\mathbf{W}}}\mathcal{L}$ efficiently by applying the chain rule through the network layers, as conceptually depicted in Fig. 2.7 [13]. Modern frameworks such as PyTorch and TensorFlow provide automatic differentiation and practical optimizers (e.g., RMSprop and Adam) [51], [67], [71], [147].

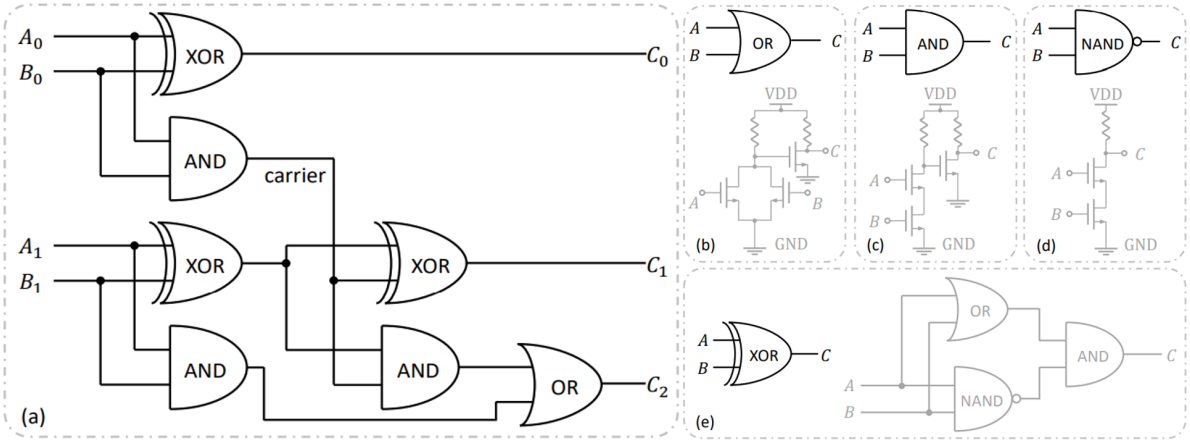


Figure 2.9.: Schematic of a printed 2-bit digital adder. (a) Gate-level implementation of the adder. (b)–(d) Transistor-level realizations of OR, AND, and NAND gates, respectively, where logic gates are shown in blue and transistors in gray. (e) XOR gate constructed using OR, AND, and NAND primitives.

2.3.3. Adapting MLPs to printed NN Hardware

Common routes for deploying an MLP on specialized hardware include: (i) *hardware synthesis*, which translates an MLP into circuit-level descriptions for application-specific implementations [19]; (ii) *mapping*, where a trained network is placed onto a given hardware architecture while accounting for resource constraints [33]; and (iii) *programmable implementations*, which leverage reconfigurable platforms (e.g., Field-Programmable Gate Arrays (FPGAs)) and accompanying toolchains [80], [148].

While these approaches are well established for silicon-based digital systems, their direct application to printed circuit platforms can be less effective. Printed components often provide only approximate analog behaviors for nonlinearities (e.g., printed tanh-like or ReLU-like responses), and practical device/circuit constraints can require redesign or co-optimization of network structure and circuit parameters [177], [199], [202].

2.3.4. Printed Neural Network Computing Circuits

Printed NN circuits merge (i) PE manufacturing advantages (low-cost, mechanically compliant/soft substrates, and flexible form factors) with (ii) NN computing models that can operate in highly parallel, low-power regimes [87], [202]. This combination is promising for scenarios where lightweight, low-cost sensing and local decision-making are required near the edge.

2.3.4.1. Analog versus digital realizations

A key constraint in PE is the comparatively large feature size (typically micrometer to millimeter scale), which makes large Boolean digital systems expensive in area and device count relative to silicon technologies [36]. Even modest digital blocks may require many gates and transistors.

In addition, additive fabrication can introduce both parametric variation and catastrophic defects, strengthening the motivation to minimize device count and simplify testability [199]. For these reasons, analog implementations are often preferred in printed NN circuits.

2.3.4.2. Circuit primitives for printed NN computation

Foundational building blocks for printed NN circuits typically include (i) resistor crossbars for weighted summation, (ii) printed nonlinear circuits that approximate activation functions, and (iii) mechanisms to express negative weights in architectures where conductances are intrinsically positive [176], [202], [213].

Table 2.3.: Comparison of the hardware cost between analog and digital (4-bit and 8-bit) approaches for a 3-input neuron. (ADC: analog-to-digital converter, ReLU: rectified linear unit, #T: number of transistors). Sourced from [202].

Approach	Components	Delay (ms)	Area (mm^2)	Power (μW)	#T
4-bit	ADC	13.8	25.4	328	185
	Adder	13	7.9	289	59
	Multiplier	13.6	15	550	103
	ReLU	2.5	1.7	80	10
	Neuron	**69**	**48**	**1250**	**357**
8-bit	ADC	154	957	37180	5938
	Adder	29	22	793	144
	Multiplier	28	85	3100	583
	ReLU	2.55	3.7	210	22
	Neuron	**522**	**1068**	**41250**	**6602**
Analog	**Neuron**	**27**	**0.49**	**859**	**4**

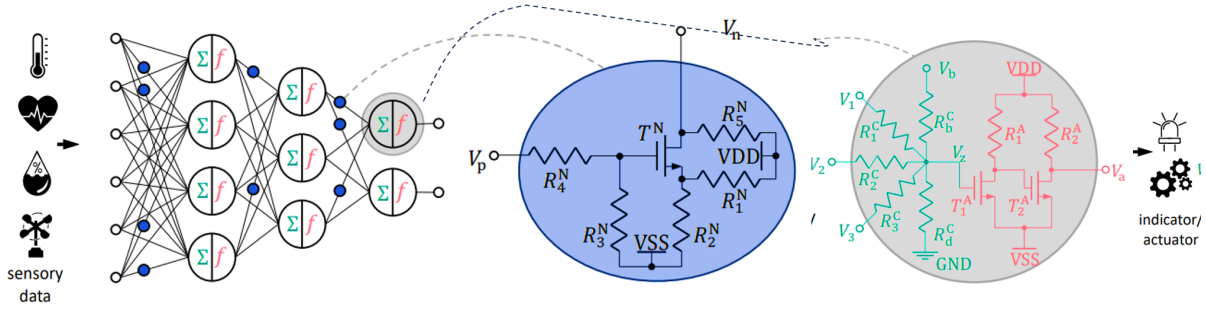


Figure 2.10.: Schematic of a printed NN circuit receiving analog sensory signals and producing analog outputs. A resistor crossbar performs weighted summation, followed by a printed negative weight circuit and a printed activation block[202].

Resistor crossbar as a weighted-sum operator. A resistor crossbar can implement an analog weighted combination of input voltages. Applying Kirchhoff's current law at the summation node yields

$$\sum_j \frac{V_j - V_z}{R_j^C} + \frac{V_b - V_z}{R_b^C} - \frac{V_z}{R_d^C} = 0, \tag{2.8}$$

where R_j^C represent crossbar resistances, V_b is a bias voltage, and V_z is the node voltage. Writing conductance as $g = 1/R$ and fixing $V_b = 1\,\mathrm{V}$, the output can be rearranged as

$$V_z = \sum_j \frac{g_j^C}{G} V_j + \frac{g_b^C}{G}, \qquad G = \sum_i g_i^C + g_b^C + g_d^C, \tag{2.9}$$

which mirrors the affine operation in Eq. (2.1): the effective weights and bias are realized through conductance ratios. Therefore, selecting and printing appropriate conductances provides a direct physical route to implement desired network parameters [1], [137], [226].

Printed tanh-like activation circuit. After weighted summation, an inverter-based nonlinear block can approximate a tanh-shaped activation. One convenient parametric representation is

$$V_a = \mathrm{ptanh}(V_z) = \eta_1^A + \eta_2^A \tanh\Big((V_z - \eta_3^A)\,\eta_4^A\Big), \tag{2.10}$$

where $\boldsymbol{\eta}^A = [\eta_1^A, \eta_2^A, \eta_3^A, \eta_4^A]$ controls scaling and shifting of the nonlinearity. These parameters are ultimately determined by physical circuit quantities (e.g., resistances and transistor geometries) [201]. Prior studies often fixed these circuit choices [202]. In later chapters of this thesis, a parametric model is introduced so that such physical quantities can be treated as optimizable design parameters during training.

Negation circuit for representing negative weights. Because conductances are nonnegative, a pure crossbar naturally realizes only nonnegative weights. However, negative weights are essential for representing inhibitory relationships in neural computations. A practical approach is to insert a printed inverter-based negation block on selected inputs [202]. A typical transfer approximation can be written as

$$\mathrm{neg}(V_{\mathrm{in}}) = -\Big(\eta_1^N + \eta_2^N \tanh((V_{\mathrm{in}} - \eta_3^N)\,\eta_4^N)\Big), \tag{2.11}$$

with $\boldsymbol{\eta}^N = [\eta_1^N, \eta_2^N, \eta_3^N, \eta_4^N]$ set by circuit-level physical parameters [202]. This allows a negative weight $-|w|$ to be emulated by negating the input and using a positive magnitude, i.e.,

$$(-|w|)\,V_{\mathrm{in}} \;\approx\; |w|\,(-V_{\mathrm{in}}) \;\leftarrow\; |w|\,\mathrm{neg}(V_{\mathrm{in}}). \tag{2.12}$$

Printed neuron model. Combining the crossbar weighted sum (Eq. (2.9)), the printed activation (Eq. (5.8)), and optional input negation (Eq. (2.11)), the effective input–output behavior of a printed neuron can be summarized as

$$V_a = \mathrm{ptanh}\left(\sum_j \frac{g_j^C}{G} V_j' + \frac{g_b^C}{G}\right), \tag{2.13}$$

where the modified input voltage V_j' depends on whether a negation block is present:

$$V_j' = \begin{cases} \mathrm{neg}(V_j), & \text{if a negation circuit is inserted on input } j, \\ V_j, & \text{otherwise.} \end{cases} \tag{2.14}$$

Using an augmented input vector (to absorb bias terms), Eq. (2.13) can also be written in a compact matrix form:

$$V_a = \mathrm{ptanh}\Big(\tilde{\mathbf{V}}'\,\tilde{\mathbf{W}}\Big), \tag{2.15}$$

where $\tilde{\mathbf{V}}'$ stacks the modified inputs together with constant entries (e.g., a bias reference voltage). One convenient expression for the effective weight matrix derived from conductances is

$$\tilde{\mathbf{W}} = \mathrm{diag}(\mathbf{g}\,\mathbf{1})^{-1}\mathbf{g}^\top, \qquad \mathbf{g} = [g_1^C, g_2^C, \ldots, g_b^C, g_d^C], \tag{2.16}$$

where $\mathbf{1}$ is an all-ones vector with compatible dimension.

2.3.5. Recurrent and Spiking Neural Networks (RNNs and SNNs)

Many edge sensing tasks involve time-varying signals such as biomedical waveforms, motion traces, or environmental time series. Recurrent NNs (RNNs) extend feed-forward ANNs by maintaining an internal state, enabling them to process sequences of arbitrary length. Classical Elman RNNs [12], Long Short-Term Memory (LSTM) networks [17] and Gated Recurrent Units (GRUs) [58] are widely used temporal models in software.

Hardware implementations of full LSTM or GRU cells are resource-intensive due to the number of parameters and gating operations involved. In the context of large-area electronics, more hardware-friendly approaches have been proposed in this thesis, in which temporal behavior is realized by simple analog filters or delay elements instead of full RNN cells. Printed temporal processing blocks (pTPBs) demonstrate this idea: learnable first- or second-order RC filters are inserted between crossbar outputs and activation circuits, and their parameters are trained alongside network weights [240]. These blocks emulate low-order recurrent dynamics while reusing existing circuit primitives, making them suitable for temporal processing on printed n-EGT platforms.

Recent SNN research has also investigated temporal information dynamics and how spike timing encodes information in deep spiking networks [230]. Such work motivates the printed spiking neuromorphic architectures as shown in Section 3.4.

SNNs model neural activity using discrete spikes, more closely mimicking biological neurons than continuous-valued ANNs. A typical leaky integrate-and-fire (LIF) neuron integrates synaptic inputs on a membrane capacitor and emits a spike when a threshold is crossed; after firing, the membrane potential is reset. Information can be encoded in spike rates, temporal spike patterns, or precise spike timing. Schuman *et al.* review neuromorphic computing and SNN hardware across digital, analog, and mixed-signal platforms, emphasizing energy-efficiency and event-driven operation [110]. Digital SNN chips exploit low-leakage CMOS and on-chip memories, while analog SNN circuits use subthreshold or weak-inversion operation of transistors to emulate biophysical dynamics.

On printed and flexible substrates, several works have demonstrated spiking primitives. Hosseini *et al.* realized an axon-hillock neuromorphic circuit using printed organic electronics, showing biologically compatible and physically flexible integrate-and-fire behavior [163]. Tischler *et al.* fabricated an integrate-and-fire neuron based on printed organic FETs, demonstrating neuron-like spiking on fully printed devices [238]. Lebanov *et al.* recently proposed flexible unipolar a-IGZO integrate-and-fire neurons, enabling spiking neuromorphic functions in oxide TFT technology [250]. These developments show that spiking primitives can be implemented in both organic and oxide-based large-area technologies, although circuit complexity and device counts remain limited.

Analog printed spiking neuromorphic circuits based on inorganic n-EGTs extend these ideas by leveraging printed resistor crossbars and inverter-based spike generators, and represent a first step toward programmable SNNs in PE [253].

Relation to this thesis

The literature on ANNs, temporal RNN models, and SNNs provides a rich algorithmic toolbox for edge intelligence. However, most hardware work targets CMOS or, in the case of large-area technologies, focuses on isolated building blocks or small fixed architectures. The thesis builds on these models but couples them tightly with the constraints and device characteristics of printed and flexible substrates, enabling hardware-aware training, architecture search, and robustness optimization in later chapters.

2.4. Other Analog and Mixed-Signal Computing Paradigms

In addition to NN hardware, several analog and mixed-signal paradigms have been explored for low-cost edge intelligence on large-area substrates. This section highlights unary computing with analog error correction and on-sensor mixed-signal classification with bespoke ADCs.

2.4.1. Analog Error Correction and Unary Computing

Unary encoding represents a value by the number of active elements (e.g., lines in a resistor array, unit current sources, or pulses in a bitstream) rather than by a binary-weighted combination. Unary representations are attractive in PE for several reasons:

- They map naturally to arrays of identical printed devices, where each element contributes a single quantum of conductance or current.
- Device failures affect only a small fraction of the total code, often resulting in small magnitude errors rather than catastrophic failures.
- Simple analog operations, such as averaging or majority voting, can implement powerful error-mitigation functions without full digitization [39].

In the security domain, unary or near-unary structures are widely used in physically unclonable functions (PUFs). Error-correction schemes for PUFs often employ helper data and strong digital error-correcting codes such as BCH or LDPC codes to recover stable keys in the presence of noisy responses [47], [48], [62]. However, these techniques typically assume CMOS with relatively abundant digital resources.

Motivated by printed unary computing and PUFs, recent work has proposed *analog error correction* blocks that operate directly on unary analog signals. Instead of fully converting unary codes to digital words, these blocks reshape analog amplitudes or currents to the nearest plausible unary value by simple majority-like operations, trading fine-grained precision for robustness and ultra-low overhead [266].

Such analog error correction is particularly appealing on printed substrates where ADCs and complex digital decoders are prohibitively expensive. It also enables robust unary-based decision logic and security primitives co-integrated with neuromorphic circuits.

2.4.2. On-Sensor Mixed-Signal Classification with Bespoke ADCs

Many edge applications require a small digital classifier operating directly on sensor data. Decision trees (DTs) are an attractive model class for such tasks because they can be implemented in hardware with simple comparators and logic.[248] On printed substrates, however, the analog front-end, particularly the ADC, dominates area and power.

Conventional N-bit flash ADCs require $2^N - 1$ comparators and an encoder, making them ill-suited for resource-limited printed tags. Yet, the thresholds used in a trained decision tree are fixed and sparse. Recent work exploits this by co-designing the ADC and DT: thresholds are mapped to unary comparator outputs, and only those comparators corresponding to used unary digits are retained [244]. The resulting bespoke ADC–DT systems significantly reduce the number of comparators and logic gates, while still achieving task-level accuracy [207], [244].

These ideas have been demonstrated both in PE and in flexible oxide TFT platforms, and are conceptually related to the tiny classifier circuits proposed by Iordanou *et al.* for CMOS and TFT technologies [248].

Relation to this thesis

Unary computing with analog error correction and bespoke ADC–DT co-design illustrate that algorithm–hardware co-optimization is not limited to NN architectures. The thesis builds on these paradigms to argue for a broader family of analog and mixed-signal computing blocks on large-area substrates and later integrates some of these ideas into reliability and security analyses.

2.5. Reliability, Fault Endurance, and Test

Reliability is a central concern for PE and FE. Compared to CMOS, these technologies typically exhibit higher rates of both parametric variations and catastrophic faults due to fabrication imperfections, material non-idealities, and environmental stress.

2.5.1. Sources of Faults in Printed and Flexible NN Classifier Circuits

Fault mechanisms in printed and flexible circuits include:

- **Manufacturing defects**, such as opens and shorts in printed lines caused by misalignment, nozzle clogging, coffee-ring effects, or incomplete curing [92], [97], [187].
- **Parametric variations**, including spreads in threshold voltage, mobility, and sheet resistance due to variations in film thickness, channel geometry, and electrolyte/material composition [151].
- **Aging and wear-out**, arising from repeated bending, humidity exposure, temperature cycling, and electrochemical degradation. In oxide TFT technologies (e.g., a-IGZO), bias-stress and temperature-dependent instabilities have been studied extensively [76], [78], [108], [258].

These non-idealities can be especially impactful in mixed-signal and neuromorphic circuits, where device-level shifts translate into gain errors, delay variations, and drift in time constants.

Fault Sensitivity Analysis of NN Classifiers To quantify accuracy degradation under non-ideal hardware behavior, several works perform fault sensitivity analysis by injecting faults (or parameter perturbations) into NN hardware models and evaluating the resulting inference performance. For example, Li *et al.* studied reliability–efficiency trade-offs in memristor-based NN designs and highlighted the need for cross-layer mitigation techniques [140]. Vatajelu *et al.* reviewed reliability challenges for hardware-implemented SNNs, covering fault mechanisms across device, circuit, and algorithm levels [154].

In printed implementations, Hefenbrock *et al.* demonstrated in-situ tuning strategies that compensate process variation and recover accuracy after fabrication [211]. Zhao *et al.* studied highly-dependable printed NN circuits that incorporate timing- and structure-level redundancy to enhance fault tolerance [242]. Overall, this body of work motivates fault models and evaluation methodologies that connect device/circuit-level non-idealities to end-to-end ML metrics.

Endurance and Adaptive Fault Tolerance Endurance refers to the ability of a system to maintain acceptable performance over time despite the accumulation of faults and drift. In large-area electronics, faults may be spatially correlated (e.g., due to cracks, delamination, or local process excursions), which can simultaneously affect multiple neighboring devices and interconnects.

Adaptive fault-tolerant strategies explored in NN hardware include reweighting or disabling faulty synapses/neurons, training with fault injection, and leveraging redundancy or over-parameterization [140], [154]. In PE, post-fabrication calibration/tuning and redundancy have been proposed as practical approaches to mitigate high variability under tight cost constraints [211], [242]. Finally, because yield and gross defects are recurring concerns in printed platforms, low-cost screening techniques are also relevant. For transparent PE with large feature sizes, learning-based optical inspection has been proposed to detect defective devices and reduce reliance on exhaustive electrical measurements [187].

Test of Power Regulation in FE Flexible power-management circuits such as low-dropout regulators (LDOs) are key building blocks for flexible NN hardware. *ReFlex-LDO*-type designs use a-IGZOs TFTs to implement regulators and employ delay-based testing schemes to detect parametric variations and subtle faults[275]. Instead of probing analog voltages directly, they monitor timing behavior in replica paths or oscillators whose delay is sensitive to regulator parameters, enabling test with simple digital measurement structures on flexible foils. Such delay-based/test-time reuse strategies are attractive for large-area electronics, where adding precision measurement circuits is expensive.

Relation to this thesis

The existing literature on reliability and endurance illustrates both the magnitude of the problem in large-area electronics and the promise of cross-layer solutions. This thesis builds on these insights by combining variation-aware training, analog error-correction, adaptive filtering, and tailored test and security analyses, with a focus on printed and flexible NN circuits [243], [262], [264], [267].Chapters on reliability in this thesis will build upon self endurance and delay-based test to propose integrated frameworks for diagnosing NN and support circuits[263], [273].

2.6. Security of Flexible NN Hardware

Large-area edge devices are often deployed in physically exposed environment and may process sensitive data (e.g., biomedical signals, activity patterns, or secrets derived from PUFs). As a result, security concerns such as side-channel leakage are relevant even for low-cost printed and flexible platforms.

Basics of Side-Channel Attacks Side-channel attacks (SCAs) exploit statistical dependence between secret-dependent internal states and physical leakage such as power consumption, electromagnetic emissions, and timing. A common non-profiled method is correlation power analysis (CPA), which assumes a leakage model and correlates measured traces with hypothetical intermediate values to infer

secrets. Profiled attacks (e.g., template attacks) learn device-specific leakage characteristics using a training phase, often reducing the number of traces required for key recovery.

In mixed-signal and analog circuits, leakage is continuous-valued and may be nonlinear and sensitive to operating conditions, which complicates modeling. However, these properties do not imply inherent security: data-driven profiled attacks can still extract information from traces when sufficient training data and measurement stability are available.

2.6.1. Side-Channel Leakage in ML and Flexible NN Hardware

In hardware ML systems, side-channel leakage has been demonstrated in various digital implementations, where power/timing traces may reveal internal activations, inputs, or model parameters under certain threat models. Similar concerns extend to flexible and printed realizations: digital designs leak through switching activity in logic, while analog implementations can leak through bias currents, charge/discharge events in capacitors, and transient dynamics during computation.

For printed and flexible security primitives, prior work has explored PUF primitves and majority-vote stabilization techniques to improve reproducibility under noise and variation [57], [105]. Surveys also emphasize that the reliability and bias of underlying devices can materially affect key stability, uniqueness, and overall authentication strength [194].

Relation to this thesis

The security literature for printed and flexible NN circuits is still in research. The works mentioned above highlight that both analog and digital realizations can leak sensitive information and that ML-based side-channel analysis is a powerful tool. This thesis will extend such analyses, focusing on flexible NN classifiers and printed analog blocks, and will explore lightweight, analog-friendly mitigation strategies within tight area and power budgets[267], [268].

2.7. Positioning of This Thesis

The literature surveyed in this chapter demonstrates significant progress across multiple domains:

- PE and FE provide ultra-low-cost, mechanically compliant platforms for sensing and computation, but suffer from strong variability, unipolar device libraries, and limited integration density[64], [86], [103], [151], [157].
- NN circuits, temporal processing blocks, and spiking networks have been realized in both silicon and large-area technologies, but often as fixed architectures with limited hardware-aware training[40], [63], [69], [110], [185], [202], [210], [238], [250], [253].
- Power-aware and robustness-aware training methods have emerged for printed NN circuits,[159], [168], [205], [211], [222], [224], [225], [231], [240], [242], [243], [262], [264] but a unified framework that jointly considers power, temporal dynamics, and variability across printed and flexible platforms is still lacking.
- Unary computing with analog error correction and bespoke ADC–DT co-design demonstrate powerful analog and mixed-signal alternatives to conventional digital processing, yet their integration with NN architectures and security primitives is still at an early stage[57], [96], [105], [109], [153], [178], [194], [207], [244], [266].
- Reliability, test, and security studies have addressed individual building blocks, printed NN circuits, flexible power regulators, unary PUFs, and flexible ML engines—but comprehensive cross-layer frameworks that span architecture, training, test, endurance, and security for ultra-resource-constrained large-area hardware remain scarce[10], [154], [236], [263], [267], [268], [273].

Against this backdrop, the thesis aims to provide a unified cross-layer methodology for reliable, energy-efficient, and secure analog NN computing on printed and flexible substrates. Chapters 3–6 will build on the background presented here to introduce new architecture and variability-aware design flows, power-aware temporal and spiking networks, reliability and endurance frameworks, and security and test methodologies tailored to ultra-resource-constrained large-area hardware[252], [259], [274].

2.8. Chapter Summary

This chapter has introduced the technological and methodological foundations of the thesis. It reviewed printed and flexible large-area electronics, highlighting printed n-EGT and flexible a-IGZOs TFT platforms as key enablers for low-cost, mechanically compliant edge devices. The chapter then summarized the NN models of interest (feed-forward ANNs, temporal RNN models, and SNNs) and described how their computational primitives map to analog NN circuits using resistor crossbars, activation circuits, temporal filters, and spiking elements.

Complementary analog and mixed-signal paradigms were discussed, including unary computing with analog error correction and on-sensor mixed-signal classification via bespoke ADC–Decision Tree co-design. The chapter further examined reliability challenges, fault mechanisms, endurance strategies, and testing methodologies for printed and flexible analog hardware, as well as side-channel security considerations for flexible NN systems and unary-based security primitives.

Part II.

Contributions

3. Power and Energy-Efficient Bespoke NN Design

This chapter focuses on design methodologies that simultaneously target energy efficiency and robustness for *p*-NNs and related on-sensor computing fabrics. It builds on the printed analog neuromorphic primitives introduced in the previous chapters and shows how algorithmic training, circuit-level design and temporal/on-sensor intelligence can be co-optimized under tight power and variability constraints.

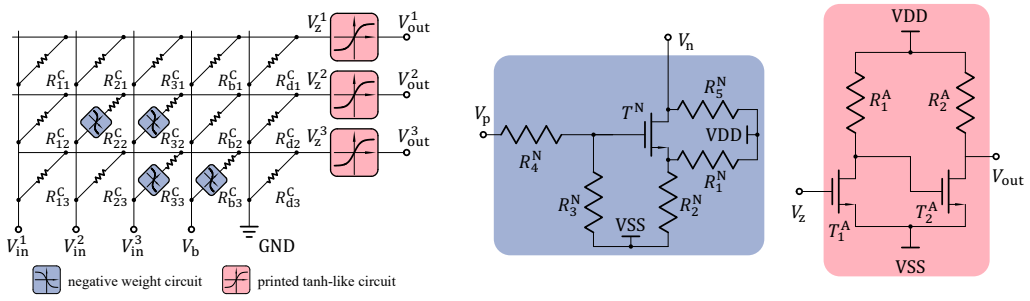
3.1. Power-Aware and Power-Constrained Training (*P*-AT and *P*-CT)

p-NNs are excellent candidates for ultra-low-cost classification at the extreme edge, but their deployment is fundamentally limited by tight power budgets imposed by printed batteries and energy harvesters. At the same time, the large device dimensions and resistive crossbars of printed technologies lead to non-negligible static and dynamic power consumption, especially in the nonlinear blocks and negative-weight circuits.

This section discusses two complementary algorithmic design flows:

1. *Power-aware training (P-AT)* that treats power as a soft objective and explores Pareto trade-offs between accuracy and power.
2. *Power-constrained training (P-CT)* that enforces hard power limits using an augmented Lagrangian optimization scheme.

Both flows rely on differentiable power models for the underlying printed circuits and are evaluated on the same family of benchmark datasets.



(a) Exemplary printed neuron (b) Negative weight circuit (c) Printed tanh-like circuit

Figure 3.1.: Schematic of *p*-NNs. (a) Example of a 3-input and 3-output printed neuron based on crossbar array. (b) Schematic of inverter-based negative weight circuit. (c) Schematic of inverter-based printed tanh-like circuit.

In many target applications of PE such as smart packaging, the printed devices are possibly disposable and consequently may not be accessible for recharging. Therefore, they are generally powered by their initial printed batteries [54] or printed energy harvesters [106]. In this case, the low power consumption of the circuit becomes particularly crucial. Moreover, due to resistive nature of weighted sum crossbar and lack of P-type transistors in this printed technology, the need for low-power design is even further justified.

In this work, we first modify the existing circuit structure in a more power-efficient way, and then propose power-aware training for *p*-NN by explicitly integrating power models into the objective function. Specifically, we derive the accurate power models for the circuit primitives in the *p*-NNs. Afterwards, by integrating these models into the *p*-NN framework, the power of the circuits can be estimated during the training process. Finally, by combining the original loss function (for classification accuracy) with the estimated power, a Pareto front of power-accuracy trade-offs can be established.

3.1.1. Modified Power-Efficient Circuit Structure

In Figure 3.1(a), negative weight circuits are prepended to the respective resistors whenever negative weights are necessitated. However, this approach is suboptimal regarding power conservation, as some inputs are repetitively converted to their corresponding negatives (e.g., V_{in}^3 at R_{32}^{C} and R_{33}^{C}). To eliminate this redundancy and thus reduce the power, we modified the circuit design, as shown in Figure 3.2. With this modified structure, only a single negative weight circuit is required for each input. Subsequently, resistors may be connected to either V_{in}^i or $\text{neg}(V_{\text{in}}^i)$, depending on the sign of the corresponding weights.

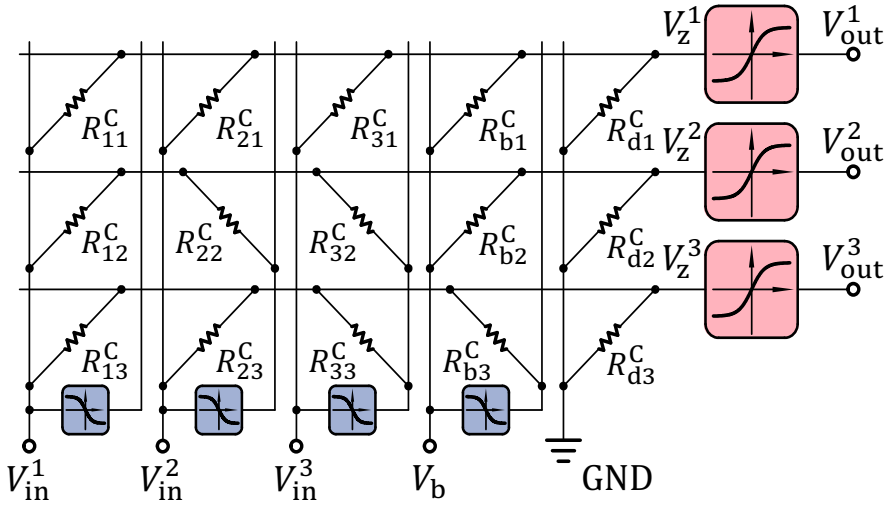


Figure 3.2.: Modified power-efficient design of a printed neuron.

To validate the new circuit design and assess other characteristics such as the latency of both circuits (Figure 3.1 and Figure 3.2), we performed SPICE simulations with the pPDK [129]. The new circuit structure yields the same output and a similar latency as the previous design; however, the power consumption with respect to V_{in}^3 decreases due to the reduced number of negative weight circuits.

3.1.2. Power Consumption Model

Due to the structural simplicity of resistor crossbar arrays, we directly employ fundamental circuit formulae to derive analytical solutions for the power consumption. In contrast, due to the complexity of the nonlinear circuits, we obtain the power models by approximating data from SPICE simulations based on the pPDK [129]. Specifically, after SPICE simulations with various circuit configurations, ANNs are employed to approximate the transformation from the physical quantities q^{N} and q^{A} to the circuit power P^{N} and P^{A}. We refer to this ANN-based transformation as the *surrogate power consumption models* for the nonlinear circuits.

3.1.2.1. Power Consumption Model for the Crossbar

Due to the pure resistivity of the resistor crossbar array (excluding the negative weight circuits), the analytical power model can be directly obtained from the formula of electronic power. For each individual resistor, the power can be calculated by

$$P = \frac{\Delta V^2}{R} = \Delta V^2 \cdot g,$$

wherein ΔV refers to the potential difference between the two ends of the resistor. Therefore, the power consumption for the crossbar excluding negative weight circuits can be modeled as

$$\text{P}^{\text{C}} = ((\tilde{\text{V}}_{\text{in}} \odot \mathbb{1}_{\{\theta \geq \mathbf{0}\}} + \text{neg}(\tilde{\text{V}}_{\text{in}}) \odot \mathbb{1}_{\{\theta < \mathbf{0}\}}) - \tilde{\mathbf{V}}_{\text{z}})^2 \odot |\theta|,$$

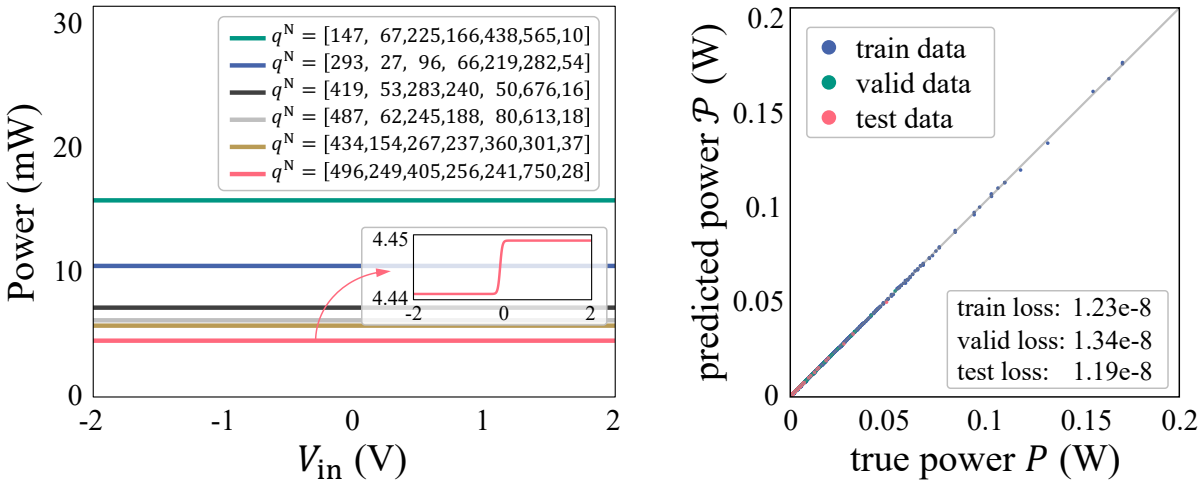


Figure 3.3.: Left: Power of some negative weight circuits with input voltages V_{in} ranging from −2V to 2V, the legend shows the configuration of the circuit components q^N, the right bottom box shows the shape of the pink curve. Right: visualization of the results from the surrogate power consumption model. The x-axis and the y-axis refer to the true power P and predicted value $\mathcal{P}$. Blue, green, and red colors denotes the data from training, validation, and test sets.

where $(\cdot)^2$ denotes an element-wise square operation, moreover,

$$\tilde{V}_{in} = \left[V_{in}^\top, \cdots, V_{in}^\top\right] \in \mathbb{R}^{(M+2)\times N},$$

and

$$\tilde{V}_z = \begin{bmatrix} V_z \\ \vdots \\ V_z \end{bmatrix} \in \mathbb{R}^{(M+2)\times N}.$$

In this way, each element in the matrix P^C represents the power of the corresponding resistor. By summing all elements in P^C, the over all power consumption of the crossbar can be obtained by

$$\mathcal{P}^C = \mathbf{1}_{M+2}^\top \cdot P^C \cdot \mathbf{1}_N, \tag{3.1}$$

where $\mathbf{1}_{M+2} \in \mathbb{R}^{M+2}$ and $\mathbf{1}_N \in \mathbb{R}^N$ are a vector with all the elements being 1.

3.1.2.2. Surrogate Power Consumption Models for Nonlinear Circuits

For the nonlinear circuits, i.e., negative weight circuits and printed tanh-like circuits, estimating the power consumption based on the physical quantities q^N and q^A is challenging. We therefore train ANNs to approximate the power consumption of these circuits based on SPICE simulations. Note that since the operations in ANNs are fully differentiable, the physical quantities can be optimized for reducing the circuit power through gradient-based algorithms.

Since the methodologies for both negative weight circuits and the printed tanh-like circuits are identical, we only describe our approach for the negative weight circuit as an example.

We firstly define the feasible design space $\mathbb{Q}^N$ to guarantee the desired negative tanh shapes of the characteristic curves. $\mathbb{Q}^N$ consists of MIN-MAX constraint on each physical quantity, i.e., $q^N \in [\boldsymbol{q}_{min}^N, \boldsymbol{q}_{max}^N]$ and inequality constraints among individual values, i.e., $R_1^N > R_2^N$, $R_3^N > R_4^N$, and $W^N > L^N$. Subsequently, we employ a Quasi Monte-Carlo method with Sobol sequence to sample 10 000 points within the feasible space, with each point referring to a unique circuit configuration. Afterwards, we performed SPICE simulations using the pPDK [129] to gain the power consumption of each sampled circuit.

The left side in Figure 3.3 exemplifies the power of multiple negative weight circuits, with input voltage V_{in} ranging from −2V to 2V. The legend denotes the corresponding circuit configuration q^N. It is notable that, although the power varies with changing input voltage, as shown by the pink curve in the right bottom

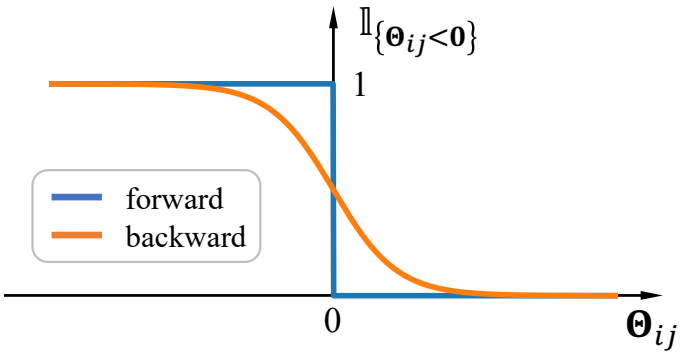


Figure 3.4.: Straight through gradient estimator for the soft-count of the negative weight circuits.

box, the variation is so small that the power consumption can be regarded as a constant w.r.t. the DC input voltage V_{in}. Moreover, due to the absence of a priori knowledge for the magnitude of input voltages, the distribution of the input voltages should be assumed as a uniform distribution ranging between −2V and 2V according to the principle of maximum entropy [6]. Consequently, the expected power consumption P^{N} is represented by the mean value w.r.t. input voltages.

After obtaining the sampled value $\text{q}_i^{\text{N}}, i = 1, ..., 10\,000$ and the corresponding power consumption P_i^{N}, we can train an ANNs as the surrogate power consumption model, which is denoted by $\mathcal{P}^{\text{N}}(\text{q}^{\text{N}})$.

To train the ANN, we randomly split the dataset $\{\text{q}_i^{\text{N}}, P_i^{\text{N}}\}_{i=1}^{10\,000}$ into training set (70%), validation set (20%), and test set (10%). The training set serves to guide the training process, the validation set stops the training to prevent overfitting, and the test set is used to evaluate the trained ANN.

To enhance performance of the ANN, we employ some typical techniques, such as data normalization, *NAS*, and hyperparameter tuning on the learning rate. Finally, a 15-layer ANN is chosen as the surrogate power model.

The performance of the surrogate model is demonstrated on the right side of Figure 3.3, where the horizontal axis denotes the true power consumption from SPICE simulation and the vertical axis refers to the predicted power from the surrogate model. We can qualitatively conclude that, the surrogate model generates acceptable power estimations. Moreover, the losses on training and test sets indicate that the model generalizes well.

Power Estimation for a Printed Neuron Building upon the developed power consumption models, we are able to estimate the power of each printed neuron by accumulating the power of each circuit primitive, namely:

$$\mathcal{P} = \mathcal{P}^{\text{C}} + N^{\text{N}} \cdot \mathcal{P}^{\text{N}} + N^{\text{A}} \cdot \mathcal{P}^{\text{A}}, \tag{3.2}$$

where N^{N} and N^{A} denote the number of negative weight circuits and printed tanh-like circuits. Moreover, $\mathcal{P}^{\text{N}}$ and $\mathcal{P}^{\text{A}}$ are the estimated power consumption from the surrogate power models.

It is notable that, according to the SPICE simulation, despite the similar inverter-based structures between negative weight circuits and printed tanh-like circuits, their power consumptions differ by orders of magnitude. Specifically, the power of the inverter circuits is at the mW level, whereas for the activation function is at the µW level. This difference can be attributed primarily to the feasible range of resistor values. Consequently, reducing the power consumption of negative weight circuits becomes even more significant. However, since we want to leverage gradient-based optimization to reduce the power consumption, we require useful gradient information of the power with respect to all our design parameters. Unfortunately, N^{N} in Equation 3.2, representing the number of negative weight circuits, depends on θ but represents a piece-wise constant function. Specifically, N^{N} is expressed by

$$\text{P}^{\text{C}} = \left((\tilde{\text{V}}_{\text{in}} \odot \mathbb{1}_{\{\theta \geq \mathbf{0}\}}) + (\text{neg}(\tilde{\text{V}}_{\text{in}}) \odot \mathbb{1}_{\{\theta < \mathbf{0}\}}) - \tilde{\text{V}}_{\text{z}}\right)^2 \odot |\theta|.$$

Table 3.1.: Result of the Experiment on 13 Benchmark Datasets

Dataset	α = 0		α = 0.25		α = 0.5		α = 0.75		α = 1	
	Accuracy	Power (mW)	Accuracy	Power (mW)	Accuracy	Power (mW)	Accuracy	Power (mW)	Accuracy	Power (mW)
AcuteInf	1.000 ± 0.000	52.0 ± 2.5	1.000 ± 0.000	35.1 ± 6.7	1.000 ± 0.000	31.0 ± 3.3	0.999 ± 0.002	28.8 ± 2.0	0.461 ± 0.018	8.3 ± 1.3
Bal.Sc	0.901 ± 0.017	47.3 ± 4.3	0.901 ± 0.028	43.4 ± 4.3	0.895 ± 0.016	39.5 ± 2.7	0.893 ± 0.020	33.4 ± 4.0	0.435 ± 0.018	10.0 ± 1.2
BreastCanc	0.971 ± 0.001	114.2 ± 9.9	0.969 ± 0.002	82.0 ± 4.0	0.966 ± 0.001	57.9 ± 4.8	0.915 ± 0.020	45.8 ± 2.3	0.740 ± 0.021	9.7 ± 3.4
Cardio	0.880 ± 0.007	196.6 ± 45.0	0.863 ± 0.019	125.2 ± 11.4	0.844 ± 0.011	97.0 ± 12.6	0.824 ± 0.014	71.7 ± 6.9	0.770 ± 0.003	16.1 ± 7.7
En-(y_1)	0.911 ± 0.019	76.7 ± 6.3	0.914 ± 0.013	63.6 ± 4.8	0.919 ± 0.013	52.3 ± 4.4	0.918 ± 0.014	46.6 ± 4.0	0.657 ± 0.010	11.8 ± 1.9
En-(y_2)	0.895 ± 0.016	80.9 ± 4.7	0.897 ± 0.008	59.6 ± 5.2	0.899 ± 0.006	52.7 ± 3.0	0.892 ± 0.010	48.4 ± 2.8	0.656 ± 0.009	11.5 ± 1.3
Iris	0.964 ± 0.005	52.1 ± 3.5	0.964 ± 0.003	42.4 ± 4.4	0.962 ± 0.004	40.6 ± 2.8	0.958 ± 0.009	32.2 ± 1.6	0.539 ± 0.011	9.2 ± 1.0
Mammo	0.791 ± 0.003	63.8 ± 6.8	0.789 ± 0.003	55.2 ± 3.0	0.792 ± 0.003	42.1 ± 2.9	0.789 ± 0.006	32.2 ± 3.0	0.635 ± 0.031	10.6 ± 0.7
PenD	0.617 ± 0.054	160.1 ± 19.5	0.536 ± 0.041	116.8 ± 7.4	0.479 ± 0.041	91.5 ± 5.5	0.371 ± 0.034	65.9 ± 12.3	0.068 ± 0.009	9.7 ± 11.9
Seeds	0.903 ± 0.031	80.4 ± 6.6	0.900 ± 0.015	65.0 ± 9.1	0.895 ± 0.017	56.2 ± 5.7	0.900 ± 0.015	50.4 ± 2.9	0.476 ± 0.035	12.8 ± 0.8
TicTac	0.999 ± 0.001	115.3 ± 6.1	0.998 ± 0.001	92.9 ± 7.5	0.926 ± 0.047	59.7 ± 6.5	0.818 ± 0.004	40.6 ± 1.2	0.594 ± 0.052	14.4 ± 4.1
Vert-2C	0.829 ± 0.007	62.7 ± 4.0	0.827 ± 0.004	55.6 ± 3.0	0.824 ± 0.009	51.6 ± 2.9	0.768 ± 0.038	35.2 ± 4.1	0.664 ± 0.017	5.5 ± 1.4
Vert-3C	0.808 ± 0.010	68.2 ± 7.1	0.817 ± 0.006	57.7 ± 8.3	0.817 ± 0.007	50.3 ± 7.8	0.819 ± 0.004	42.3 ± 3.7	0.445 ± 0.007	10.8 ± 1.3
Average	**0.882 ± 0.013**	**90.0 ± 9.7**	**0.875 ± 0.011**	**68.8 ± 6.1**	**0.863 ± 0.013**	**55.6 ± 5.0**	**0.836 ± 0.015**	**44.1 ± 3.9**	**0.549 ± 0.018**	**10.8 ± 2.9**

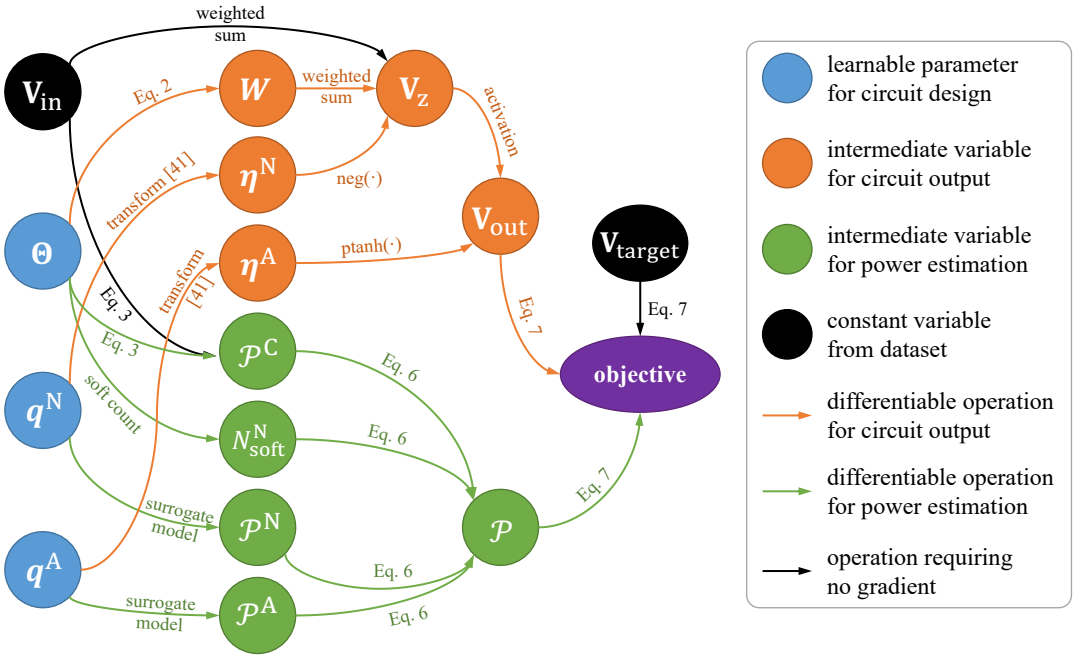


Figure 3.5.: Computation graph of the power-aware training of the printed neural networks with one neuron. The orange part refers to previous work, while the green part denotes the contribution of this work.

$$N^N = \mathbf{1}^\top_{M+2} \cdot row\ max\left\{\mathbb{1}_{\{\Theta<0\}}\right\}, \tag{3.3}$$

where row max(·) returns the row-wise maximum values. The blue curve in Figure 3.4 visualizes the indicator function $\mathbb{1}_{\{\theta<0\}}$. It is evident that, except for $\theta_{ij} = 0$, all gradients are 0, meaning that, within the context of gradient-based optimization, θ will not be modified for the purpose of reducing N^N. To address this issue and enable the optimization of N^N through θ, we introduce the *soft count* of negative weight circuits, denoted by N^N_{soft}. In the forward pass of the soft count, N^N_{soft} is still calculated by Equation 3.3, however, in the backpropagation, a relaxed function,

$$\mathbf{1}^\top_{M+2} \cdot \text{row max}\left\{1 - \text{sigmoid}(\theta)\right\},$$

is employed to generate the gradient for updating θ. Compared to Equation 3.3, the indicator function is relaxed as a sigmoid function, as shown by the orange curve in Figure 3.4 . This kind of separate treatment for the forward and backward pass is also referred to as the straight-through gradient estimator [52].

By replacing N^N in Equation 3.2 with soft count of the negative weight circuits, the resulting power estimation of the printed neuron can be formulated as

$$\mathcal{P} = \mathcal{P}^C + N^N_{soft} \cdot \mathcal{P}^N + N^A \cdot \mathcal{P}^A. \tag{3.4}$$

The computation graph for the complete power estimation is shown by the green part in Figure 3.5. Note that this figure only represents the computation graph for one neuron. In case multiple neurons are adopted, the V_{out} of one neuron will be passed to the next neuron as the input voltages. Consequently, the output of the last neuron will be regarded as the actual output. Moreover, the power consumption of all neurons will be summed up, serving as the final estimate for the power consumption.

3.1.3. Power-Aware Training (*P*-AT)

For training ANNs, loss functions are generally utilized to guide the optimization process and reflect the performance of the ANNs. A typical loss function for classification tasks is cross-entropy. However, to

account for hardware limitations, such as minimal distinguishable voltages, a modified multi-class hinge loss [202] is employed to guide the training of p-NNs. It can be expressed as

$$L(\theta, \mathrm{q}^{\mathrm{N}}, \mathrm{q}^{\mathrm{A}}) = \left((m + T - \mathrm{V}_{\mathrm{out}}) \odot \mathrm{V}_{\mathrm{target}} \cdot \mathbf{1}_N\right)^+ + \left(\max\{(m + \mathrm{V}_{\mathrm{out}}) \odot (1 - \mathrm{V}_{\mathrm{target}})\}\right)^+,$$

where T represents the measuring threshold, m denotes the sensing margin, $(\cdot)^+ = \max\{0, \cdot\}$, and $\mathrm{V}_{\mathrm{target}}$ refers to the target class after one-hot encoding. This loss function encourages the voltage for the correct class to exceed $m + T$, while suppressing outputs corresponding to incorrect classes.

Consequently, to jointly optimize both classification accuracy and power consumption, the power-aware training objective of the p-NN is given by

$$\mathcal{L}(\theta, \mathrm{q}^{\mathrm{N}}, \mathrm{q}^{\mathrm{A}}) = (1 - \alpha) \cdot L(\theta, \mathrm{q}^{\mathrm{N}}, \mathrm{q}^{\mathrm{A}}) + \alpha \cdot \mathcal{P}, \tag{3.5}$$

where $\alpha \in \mathbb{R}^+$ denotes a scaling factor to express the trade-off between loss and power consumption. If $\alpha = 0$, the training objective entirely corresponds to the accuracy of the classification tasks. In this case, the trained p-NN should achieve the highest accuracy, which can be regarded as the upper bound. However, since power consumption is totally ignored, the corresponding power should also be regarded as an upper bound. Conversely, if $\alpha = 1$, power $\mathcal{P}$ dominates the training objective whereas the accuracy is disregarded. Therefore, the trained p-NNs may exhibit the lowest power consumption but, at the same time, also the poorest accuracy. Since the trade-off between power and accuracy is only implicitly influenced by α, and, considering that a specific trade-off will be chosen based on different application scenarios, we decide to train p-NNs with different $\alpha \in [0, 1]$ and construct a Pareto front to facilitate the selection of various trade-offs with Pareto optimality.

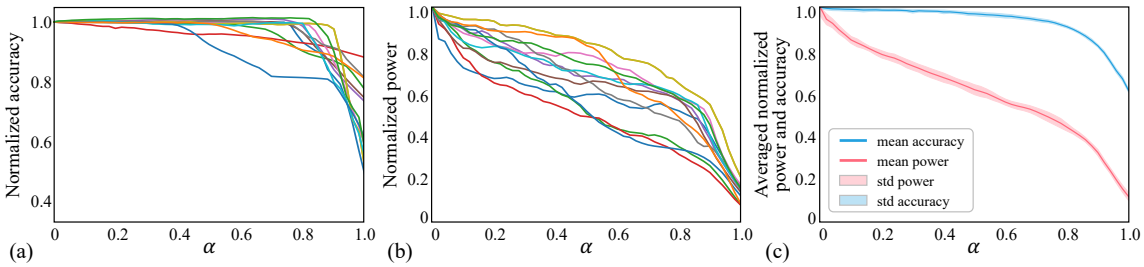


Figure 3.6.: Results of experiment with 50 different α values. (a) normalized accuracies of 13 tasks. Each task is indicated by a different color. (b) normalized power consumption of 13 circuits for the corresponding tasks. (c) averaged normalized accuracy and power, the curves and area denote the mean and standard deviation w.r.t. random seeds.

3.1.4. Evaluation

To evaluate the effectiveness of the power-aware training of p-NNs, we implemented the proposed approach[1] with PyTorch [149] and conduct experiments on the 13 benchmark datasets, which were also used in the related works, such as [220] and [241]. Moreover, these benchmark datasets exhibit a complexity and scenario that matches the target application domains of PE. The experiment is conducted at simulation level based on the pPDK [129]. The functionality of the printed neuromorphic hardware has been experimentally validated in [202] and [218].

3.1.5. Experiment

We first split the datasets into training (60%), validation (20%), and test (20%) sets. Subsequently, we use a consistent topology (#*inputs*-3-#*outputs*) for all p-NNs on each dataset. The learnable

[1] `https://github.com/Neuromophic/Power-Aware-Training`

parameter Θ is randomly initialized, while for the nonlinear circuits, η^N and η^A are initialized as [463Ω,109Ω, 10kΩ, 9kΩ, 24kΩ, 283µm, 69µm] and [205MΩ, 7kΩ, 80µm, 80µm, 480µm, 40µm]. The corresponding auxiliary parameters are $\jmath^N = [-0.006, 1.024, 0.016, 1.006]$ and $\jmath^A = [0.290, 0.710, -0.017, 20]$, respectively. Regarding the training, we employ full-batch training with the Adam [66] optimizer in default parameterization to update parameters in p-NNs. To prevent overfitting, we calculated the loss on validation set for early-stopping [18] after each parameter update. We start with an initial learning rate of 0.1 and halve it after a patience (updates without improvement on objective function) of 100-epochs on the validation set. Additionally, the training process is stopped, when the learning rate was halved 10 times. In investigate the trade-off between accuracy and power, we uniformly select 50 values in $\alpha \in [0, 1]$.

The training is repeated 10 times (with seeds varying from 1 to 10) for different initialization for each value of α to make sure to achieve a sufficiently good solution for each value of α. Finally, the hardware-related hyperparameters in the loss function, i.e., measuring threshold and margin, are chosen to be $T = 0.1$ and $m = 0.3$ to keep in line with other works [202].

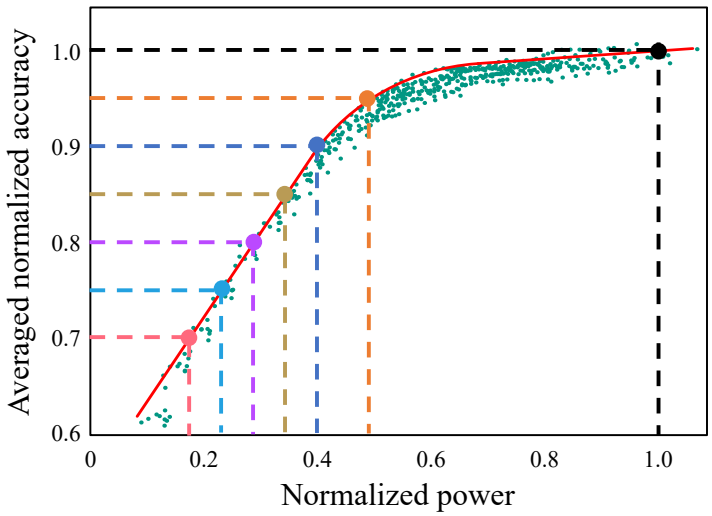


Figure 3.7.: Scatter plot of normalized accuracy versus power for all runs. The red curve displays the Pareto front, and the bold points denote different possible trade-offs on with Pareto optimality.

3.1.6. Result

After training, we evaluate the trained p-NNs on the test sets. Table 3.8 reports the accuracies and powers with $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$.

To analyze the impact of α more clearly and to eliminate the disparate difficulties among different tasks, we normalize the accuracy by the baseline ($\alpha = 0$), which refers to the power-unaware training, and should theoretically achieve the best accuracy. Note that this is not always true in practice due to the complex nature of the non-convex optimization problem that neural network training resembles. The resulting curves are displayed in Figure 3.6(a). Analogously, the power consumption is also normalized by the baseline power consumption. Because compared to the exact values, the relative power reduction serves as a more informative metric. The normalized powers are visualized in Figure 3.6(b).

To investigate the effectiveness of the power-aware training within a comprehensive and generic scenarios, we calculate the averaged normalized accuracy across all tasks, which is intended to exemplify the expected performance of the p-NNs on multiple datasets. The statistical result (w.r.t. 10 random seeds) of the averaged normalized accuracy (blue curve) and power (red curve) are summarized in Figure 3.6(c).

In order to obtain the Pareto front, we plot the entirety of normalized powers versus their respective normalized accuracies for all runs (random seeds) and all values of α by the green points in Figure 3.7. Subsequently, we can delineate the Pareto front by the red curve.

Discussion As expected, for $\alpha = 0$ (no consideration of the power consumption) *p*-NNs yield the highest accuracies and power consumption (Figure 3.6). As α progressively increases to 1, both accuracy and power decline. However, the reduction in accuracy is less significant than that in power. This phenomenon enables the power conservation without a substantial drop in accuracy.

The Pareto front in Figure 3.7 illustrates the relationship between power and accuracy. In comparison to power-unaware training (black point), if accuracy is allowed to decrease by 10%, 2.5× power reduction can be achieved (blue point). Furthermore, if a 20% accuracy drop is allowed, the power consumption can be reduced to 3.6×. Other examples of trade-offs with Pareto optimality are reported in Table 3.2. Beyond the examples listed, every point on the Pareto front may be chosen in consideration of the specific design requirements and application contexts.

It is notable that, when power decreases from 100% to 50%, the accuracy reduces in a gradual way. Conversely, a more substantial decrease in accuracy can be observed as the power budget continues to diminish from 50%. Thus, within the scope of this experiment, employing 2× power reduction to achieve 95% accuracy may represent a reasonable trade-off.

Table 3.2.: Accuracy-Power Trade-off

Accuracy (%)	100	95	90	85	80	75	70
Power (%)	100	50	40	34	28	23	18

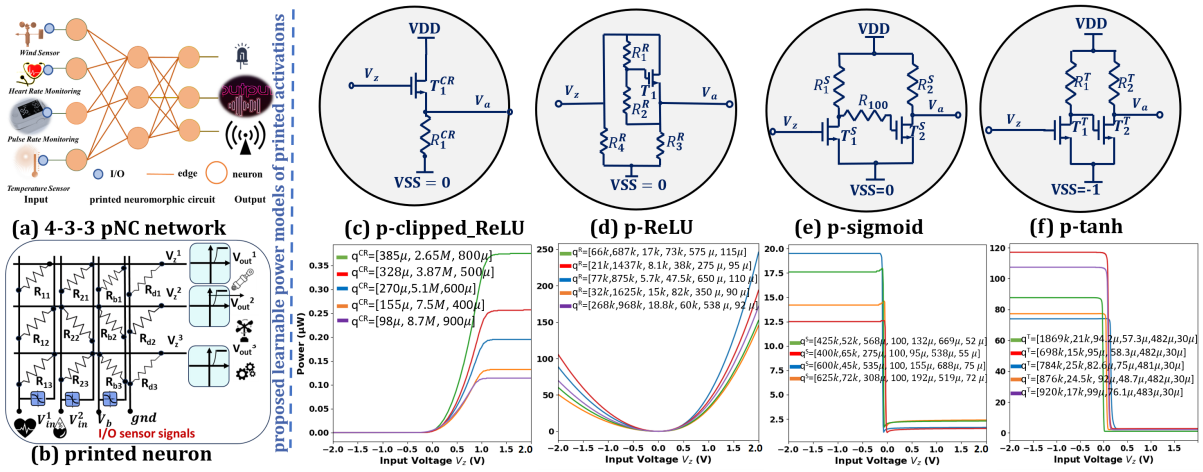


Figure 3.8.: Schematic of *p*-NN. (a) Example of a 4-3-3 *p*-NN network, (b) 3-input-3-output printed neuron based on crossbar array; surrogate power models of: (c) p-clipped_ReLU, (d) p-ReLU, (e) p-sigmoid and (f) p-tanh activation circuit.

***P*-CT with Augmented Lagrangian** While traditional NN computing has made significant strides in low-power design [84], most existing methods rely on best-effort power optimization with soft constraints, where minimizing power is desirable but not mandatory [116], [139]. However, many real-world applications require strict (hard) power constraints to ensure reliable operation within predefined energy budgets. For example, wearable health devices, disposable medical sensors, and IoT smart labels depend on limited power sources like small batteries or energy harvesters, making it critical to enforce precise power budgets.

Our work directly addresses this gap by modeling and optimizing strict power constraints during the training of *p*-NNs. Unlike soft constraints, where achieving optimal power-performance trade-offs is the goal, we enforce hard constraints to achieve the best accuracy and efficiency without exceeding the predefined power budget. This approach integrates AFs as learnable parameters and uses SPICE simulation for accurate power estimation, ensuring a robust and practical methodology for power-constrained electronic design.

To enable training under strict power constraints, we propose a training framework based on the augmented Lagrangian method. The primary objective is to maximize classification accuracy given the circuit parameters θ and q, while ensuring that power consumption below a predefined limit. Formally, this can be described by

$$\underset{\theta,\boldsymbol{q}}{minimize}\ \mathcal{L}(\mathcal{D},\theta,\mathrm{q})\ \text{s.t.}\ c(\theta,\mathrm{q}) = P(\theta,\mathrm{q}) - \overline{P} \leq 0, \tag{3.6}$$

where $\mathcal{L}(\mathcal{D},\theta,\mathrm{q})$ denotes the cross-entropy loss on the training data $\mathcal{D}$, which ensures high classification accuracy, and $c(\theta,\mathrm{q}) = P(\theta,\mathrm{q}) - \overline{P}$ represents the constraint function, ensuring that the estimated power consumption $P(\theta,\mathrm{q})$ remains within the strict task-specific upper limit $\overline{P}$. Evaluating the power contraint $c(\theta,\mathrm{q})$ requires accurate power estimation. Furthermore, since we will require derivatives through the contraint in the training procedure described later, we require a differentiable power model $P(\boldsymbol{\theta},\mathrm{q})$.

3.1.6.1. Surrogate power Modeling for Power Estimation

To simplify power analysis for resistor crossbar arrays, we adopt analytical solutions derived in [243]. However, modeling power consumption for nonlinear circuits, such as printed activation circuits (AFs), is more complex. We address this by approximating power consumption through SPICE simulations using the pPDK [129]. For each AF (p-ReLU, p-clipped ReLU, and p-sigmoid), we run 10,000 SPICE simulations, revealing distinct power behaviors as illustrated in Figure 3.8 (c)-(f) (bottom). For instance, in p-clipped_ReLU, power spikes near a threshold as transistors conduct more current, then stabilizes due to the clipping effect; p-ReLU exhibits a smooth increase in power with input voltage, reflecting its unbounded nature; and p-sigmoid shows asymmetric power patterns due to higher current demands at negative voltages. Thus, the choice of AFs and their learnable parameters $q^{\mathrm{AF}} = [\mathbf{R},\mathbf{W},\mathbf{L}]$ (representing vectors of resistance, transistor widths, and transistor lengths) significantly impacts power consumption in p-NNs.

To model learnable AFs, we define a feasible design space, $\mathbb{Q}^{\mathrm{AF}}$, with specified bounds for each parameter q^{AF}. We sample 10,000 circuit configurations using a Sobol sequence and simulate their power consumption using SPICE.

Based on this data, we train a 15-layer ANN for each AF as a surrogate model to map the corresponding physical variables q^{AF} to power consumption, $\mathcal{P}^{\mathrm{AF}}(\mathrm{q}^{\mathrm{AF}})$. To enhance accuracy, data normalization and hyperparameter tuning are applied. This surrogate model provides efficient power estimation, enabling the total power calculation and analysis of the impact of AF configurations on power.

Power Estimation for a Printed Neuron We estimate the power consumption of each printed neuron using the power of the crossbar ($\mathcal{P}^{C}$) and surrogate power models for its components: $\mathcal{P}^{\mathrm{AF}}$ and $\mathcal{P}^{\mathrm{N}}$, corresponding to the AF and negation circuits, respectively. To calculate the total power consumption, we determine N^{AF} and N^{N}[243], which represent the counts of AFs and negation circuits, respectively.

Counting Activation Circuits. Initially, N^{AF} is calculated using an indicator function:

$$N^{\mathrm{AF}} = \mathbf{1}_N^{\top} \cdot \text{row max}\left\{\mathbb{1}_{\{|\theta|>0\}}\right\}, \tag{3.7}$$

where $\mathbf{1}_N \in \mathbb{R}^N$ is a vector of all ones, $\mathbb{1}_{\{|\theta|>0\}}$ assigns 1 to active circuits and 0 otherwise, and row max$(\cdot)$ returns the row-wise maximum values, as each row in θ corresponds to a single activation circuit. If all elements in a row of surrogate conductances are zero, the corresponding activation circuit does not need to be printed or activated. This calculation ensures that any row with at least one non-zero conductance is counted only once. However, this piecewise constant function is non-differentiable, limiting its utility for gradient-based optimization.

Soft Count for Differentiability For optimization, we replace $\mathbb{1}_{\{|\theta|>0\}}$ in Eq. (4.12) with a sigmoid function $\sigma(|\theta|)$ for gradient computation, defining $N^{\mathrm{AF}}_{\mathrm{soft}}$ as

$$N^{\mathrm{AF}}_{\mathrm{soft}} = \mathbf{1}_N^\top \cdot \text{row max}\left\{\sigma(|\theta|)\right\}.$$

This relaxation allows gradient-based updates for N^{AF} by providing smooth gradients with respect to θ, as was previously done for $N^{\mathrm{N}}_{\mathrm{soft}}$ in [243].

Final Power Estimation. During the backward pass, the soft count facilitates gradient-based optimization. However, the final power estimation for a neuron is determined using the indicator function, as shown below:

$$\mathcal{P} = \mathcal{P}^C + N^{\mathrm{N}} \cdot \mathcal{P}^{\mathrm{N}} + N^{\mathrm{AF}} \cdot \mathcal{P}^{\mathrm{AF}}.$$

Power estimation in p-NNs is influenced directly by the learnable parameters q^{N}, q^{AF}, and θ. Additionally, θ indirectly affects the number of active elements in the circuit, which further impacts the total power consumption $P(\theta, \mathrm{q})$. In multi-neuron configurations, the output voltage of each neuron serves as the input for the subsequent neuron, with the final neuron's output representing the overall p-NN output. The total power consumption is determined by summing the contributions from all neurons.

3.1.7. Power-Constrained Training (*P*-CT)

To respect the constraints in training with Eq. (3.6), we require an adapation of the classic training procedure towards methods that can handle constraints. One such method that integrates particularly well backpropagation-based training schemes is the augmented Lagrangian method, see e.g. [15], [29]. It can be motivated by solving a sequence of unconstrained problems that converge to the solution of the constrained problem. Specifically, the method alternates between solving

$$\underset{\theta,\boldsymbol{q}}{\mathit{minimize}} \quad \max_{\lambda \geq 0} \; \mathcal{L}(\mathcal{D}, \theta, \mathrm{q}) + \lambda \cdot c(\theta, \mathrm{q}) - \frac{1}{2\mu}(\lambda - \lambda')^2, \tag{3.8}$$

and updates to the Lagrange multiplier estimate λ' by

$$\lambda' \leftarrow \max\{0, \lambda' + \mu \cdot c(\theta, \mathrm{q})\}. \tag{3.9}$$

The parameter $\mu \in \mathbb{R}^+$ is a hyperparameter that controls the speed of convergence and influences the stability of the method.

The minimization in Eq. (3.8) may be performed using classic backpropagation based training.

The inner maximization over λ can be solved analytically for a given λ', details see [245]. After solving Eq. (3.8), the update of λ' in Eq. (3.9) adjusts the estimate for the Lagrange multiplier, encouraging the constraints to hold for the next minimization attempt. To save computation time, θ and q should be warmstarted with the last solution obtained. The alternations of these steps can be stopped when a sufficiently good feasible solution is obtained.

3.1.8. Evaluation

To evaluate the effectiveness of the power-aware training of p-NNs, we implemented the proposed approach[2] with PyTorch [149] and conduct experiments on the 13 benchmark datasets, which were also used in the related works, such as [220] and [241]. Moreover, these benchmark datasets exhibit a complexity and scenario that matches the target application domains of PE. The experiment is conducted at simulation level based on the pPDK [129]. The functionality of the printed neuromorphic hardware has been experimentally validated in [202] and [218].

[2] `https://github.com/Neuromophic/Power-Aware-Training`

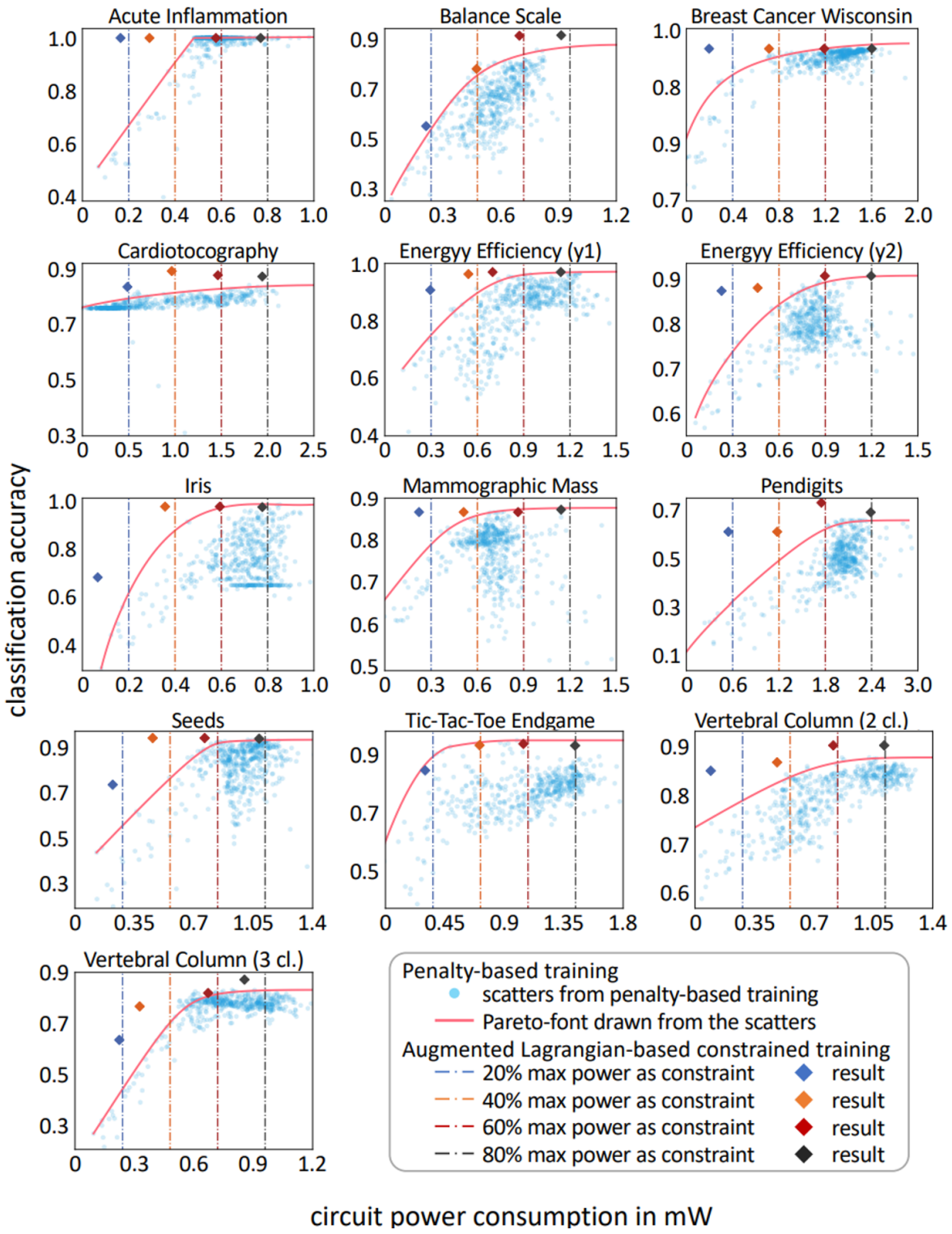


Figure 3.9.: Classification accuracy and power consumption of p-NNs using exemplary p-tanh AF from penalty-based and Augmented Lagrangian-based training approaches. The blue scatters are the results of penalty-based training, while the pink curves are Pareto fronts drawn from the scatters. The vertical lines indicate the power constraints in the augmented Lagrangian approach, whereas the rhombus with the same color refers to the results from the augmented Lagrangian

3.1.9. Experiment

We first split the datasets into training (60%), validation (20%), and test (20%) sets. Subsequently, we use a consistent topology (#*inputs*-3-#*outputs*) for all p-NNs on each dataset. The learnable

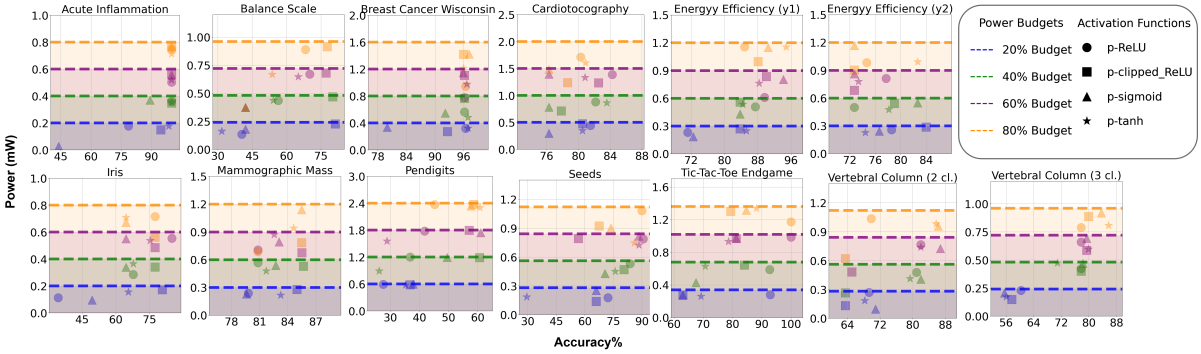


Figure 3.10.: Classification accuracy and power consumption of *p*-NNs using exemplary p-tanh AF from penalty-based and Augmented Lagrangian-based training approaches. The blue scatters are the results of penalty-based training, while the pink curves are Pareto fronts drawn from the scatters. The vertical lines indicate the power constraints in the augmented Lagrangian approach, whereas the rhombus with the same color refers to the results from the augmented Lagrangian

parameter Θ is randomly initialized, while for the nonlinear circuits, q^N and q^A are initialized as [463Ω,109Ω, 10kΩ, 9kΩ, 24kΩ, 283µm, 69µm] and [205MΩ, 7kΩ, 80µm, 80µm, 480µm, 40µm]. The corresponding auxiliary parameters are $\jmath^N = [-0.006, 1.024, 0.016, 1.006]$ and $\jmath^A = [0.290, 0.710, -0.017, 20]$, respectively. Regarding the training, we employ full-batch training with the Adam [66] optimizer in default parameterization to update parameters in *p*-NNs. To prevent overfitting, we calculated the loss on validation set for early-stopping [18] after each parameter update. We start with an initial learning rate of 0.1 and halve it after a patience (updates without improvement on objective function) of 100-epochs on the validation set. Additionally, the training process is stopped, when the learning rate was halved 10 times. In investigate the trade-off between accuracy and power, we uniformly select 50 values in $\alpha \in [0, 1]$.

The training is repeated 10 times (with seeds varying from 1 to 10) for different initialization for each value of α to make sure to achieve a sufficiently good solution for each value of α. Finally, the hardware-related hyperparameters in the loss function, i.e., measuring threshold and margin, are chosen to be $T = 0.1$ and $m = 0.3$ to keep in line with other works [202].

3.1.10. Result

To evaluate the effectiveness of our *P*-CT approach, we compared the results with a penalty-based method [243]. Four neural networks, each using a different AF (p-ReLU, p-Clipped_ReLU, p-sigmoid, and p-tanh), were trained under four power budgets: 20%, 40%, 60%, and 80% of the maximum power consumption derived from unconstrained power training. The top three models per dataset were selected based on test accuracy. As shown in Figure 3.10, each point represents classification accuracy and power consumption for a dataset. Marker shapes indicate AF (circles for p-ReLU, squares for p-Clipped_ReLU, triangles for p-sigmoid, and stars for p-tanh), and marker colors represent power budgets, aligned with horizontal dashed lines for power thresholds. The x-axis shows accuracy (percentage), and the y-axis shows power (milliwatts).

Table 3.3 summarizes the average performance metrics across 13 datasets, comparing power consumption, accuracy, and device count among the proposed AF (p-ReLU, p-Clipped_ReLU, p-sigmoid, and p-tanh). On the right side of the table, the results for the penalty-based baseline method [243] are presented, showing accuracy and power under different scaling factors α (0.25, 0.5, 0.75, and 1), which indicate the trade-off between power and accuracy. Additionally, for the baseline method, we generated a Pareto front (pink curve) using 500 penalty-based runs, varying the scaling factor α over the range $[0, 1]$. Figure 3.9 compares the optimal solutions found by our method in a single training run against this Pareto front. Vertical dashed lines highlight power constraints, demonstrating that our method efficiently achieves power-constrained optima, often matching or surpassing the Pareto front with significantly fewer training runs.

Table 3.3.: Averaged Performance Metrics Across 13 Datasets: Comparison of Metrics (Pow: Power (mW), Acc: Accuracy, Dev: Device Count) Across AF at Different Power Budgets and Penalty-Based Baseline

Metric \ ACT		p-ReLU	p-clipped_ ReLU	p-sigmoid	p-tanh	Baseline	
20%	Pow	0.27	0.26	0.23	0.25	10.8	$\alpha = 1$
	Acc	67.94	**74.56**	62.60	67.29	54.9	
	#Dev	**17**	27	27	19	-	
40%	Pow	0.56	0.56	0.53	0.53	44.1	$\alpha = 0.75$
	Acc	78.11	**80.51**	73.96	74.60	83.6	
	#Dev	**28**	36	41	30	-	
60%	Pow	0.85	0.84	0.88	0.84	55.6	$\alpha = 0.5$
	Acc	**82.76**	78.54	78.89	78.45	86.3	
	#Dev	**34**	47	54	40	-	
80%	Pow	1.10	1.05	1.12	1.09	68.8	$\alpha = 0.25$
	Acc	80.42	78.84	78.05	**81.82**	87.5	
	#Dev	**37**	45	57	58	-	

Discussion Our experimental results demonstrate the effectiveness of the augmented Lagrangian approach in achieving high accuracy while adhering to defined power constraints across various datasets. As shown in Figure 3.10, all results lie below the defined power levels, confirming the method's capability to enforce power limits effectively. Notably, at 20% of the power budget, accuracy decreases compared to higher power budgets, highlighting the trade-off between power and performance. As shown in Table 3.3, the baseline demonstrates high accuracy at $\alpha = 0.25$ but with higher power consumption, while $\alpha = 1$ reduces power at the cost of accuracy. In contrast, the proposed method achieves a balanced trade-off between power and accuracy across power budgets in a single training run. Furthermore, it delivers results comparable to or better than the Pareto front obtained from penalty-based objectives (Figure 3.9) without the need for extensive hyperparameter tuning.

The success of this approach is due to two main factors. First, by directly incorporating power constraints into the training objective, the augmented Lagrangian method avoids the instability often caused by high power penalties in penalty-based methods. Second, this explicit formulation focuses on achieving the best feasible accuracy for each specified power budget without unnecessarily minimizing power further.

Compared to penalty-based methods, which often create ill-conditioned optimization problems and fail to align with optimal power-accuracy trade-offs, the augmented Lagrangian method dynamically enforces constraints while maintaining accuracy. This ensures a balanced solution on the Pareto front with fewer computational resources.

Analyzing individual AF reveals trade-offs between accuracy and device count. For example, p-tanh achieves up to 81.82% accuracy at an 80% power budget, but requires an average of 58 devices, whereas p-ReLU achieves 80.42% accuracy with only 37 devices—a 36% reduction. This makes p-ReLU particularly suitable for hardware-limited applications. These findings demonstrate that such trade-offs persist even under strict power constraints, with reduced device counts contributing to lower power consumption.

3.2. Event-Driven Printed Analog Spiking Neural Networks (p-SNN)

As discussed in section 2.3 and illustrated in Figure 2.8 [112], spiking neural networks (SNNs) [155] constitute one of the earliest and most biologically inspired paradigms in neuromorphic computing. They continue to attract significant research interest due to their event-driven nature, compact circuit realizations,

energy efficiency, and suitability for real-time processing. In fact, some perspectives consider SNNs as the most authentic embodiment of neuromorphic computation. Several prior works have demonstrated analog implementations of printed spiking neurons [163], [238]. However, these realizations typically employ organic thin-film transistors operating at relatively high supply voltages, which conflicts with the low-power and edge-oriented application scenarios targeted in this thesis.

In addition, most existing studies concentrate primarily on reproducing neuron dynamics at the circuit level, for example by emulating integrate-and-fire (I&F) or leaky-integrate-and-fire (LIF) behaviors. While such implementations demonstrate functional feasibility, they generally lack a consistent parametric modeling framework that enables hardware-aware training of networks built from these circuits. A key challenge arises from the mismatch between the abstract neuron model assumed during training and the physical behavior of the implemented hardware.

For example, certain works claim to realize analog printed neurons based on the I&F principle. In practice, however, an ideal I&F model presumes the availability of an ideal comparator and instantaneous switching behavior. This implies abrupt transitions between cut-off and saturation regions of a transistor and instantaneous charging or discharging of capacitors, including immediate reset of membrane and output voltages. Such idealized characteristics are physically unattainable in real analog circuits, where finite gain, non-zero transition times, and parasitic effects inevitably limit performance.

As a result, training based on an ideal I&F abstraction while deploying a circuit with fundamentally different dynamics introduces a modeling inconsistency. This discrepancy can substantially degrade inference accuracy and, in extreme cases, prevent reliable operation of the fabricated system.

To overcome these limitations, this section proposes a hardware–software co-design strategy centered on a practical spike-generation circuit composed of fast charge–discharge paths, referred to here as a *spike generator*. Rather than enforcing strict equivalence to classical neuron models such as I&F or LIF, we adopt an approximation-driven modeling approach that directly captures the input–output characteristics of the implemented circuit. This surrogate model enables hardware-aware training and ensures alignment between algorithmic optimization and circuit-level behavior. By integrating the proposed spike generator into *p*-NNs, we realize printed SNNs that can be trained end-to-end while maintaining consistency between the computational model and its physical implementation.

3.2.0.1. Circuit Design of printed SNNs

Before presenting the detailed circuit implementation of the p-SNNs, we first revisit the underlying neuron models that guide the design.

Spiking Neural Networks. Both artificial neural networks (ANNs) and spiking neural networks (SNNs) rely on weighted summation followed by nonlinear transformation. The essential distinction lies in the form of the nonlinearity. In conventional ANNs, the activation operates in the amplitude domain, mapping one magnitude value to another through a static nonlinear function. In contrast, SNNs encode information temporally: the weighted sum is converted into a sequence of spikes, where information is represented through spike timing or firing rate.

As illustrated in Figure 3.11, the classical integrate-and-fire (I&F) neuron operates according to the following sequence:

1. the input signal is accumulated as a membrane voltage,
2. the membrane potential increases over time,
3. once a predefined threshold is reached,
4. an output spike is emitted, and
5. the membrane voltage is reset.

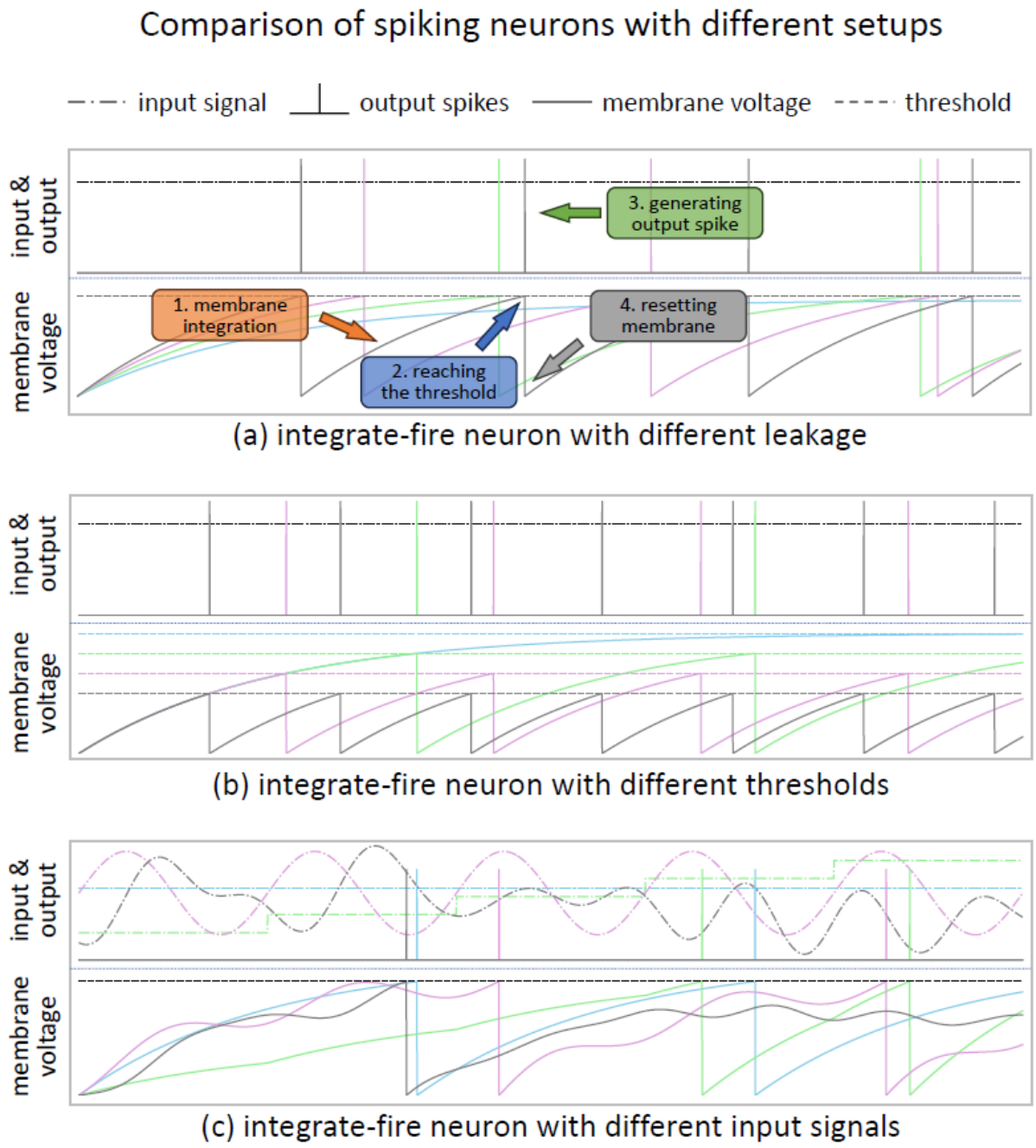


Figure 3.11.: Illustration of integrate-and-fire (I&F) neuron behavior under (a) varying membrane leakage, (b) different firing thresholds, and (c) different input signals. In each subfigure, matching colors denote corresponding inputs, membrane voltages, thresholds, and outputs. The blocks and arrows in (a) summarize the operation: leaky integration, threshold comparison, spike generation, and membrane reset.

From a hardware perspective, the threshold detection and instantaneous reset operations (steps 3 and 5) are particularly challenging in analog implementations. They would require an ideal comparator with abrupt switching behavior and instantaneous discharge of capacitive nodes, assumptions that are physically unrealistic in practical printed electronics.

To overcome these constraints, we adopt an end-to-end hardware-aware design methodology. Rather than strictly enforcing an ideal I&F abstraction, we focus on preserving the functional objective of spike encoding while relaxing the need for ideal switching.

The fundamental insight is that an I&F neuron effectively maps the magnitude of the input signal to a temporal property, specifically, the time required to reach threshold and thus the resulting spike frequency.

In other words, input amplitude is translated into spike occurrence within a given time window. Based on this observation, the design of the proposed p-SNNs follows two principles:

1. The spike count within a predefined temporal interval should increase monotonically with input magnitude.

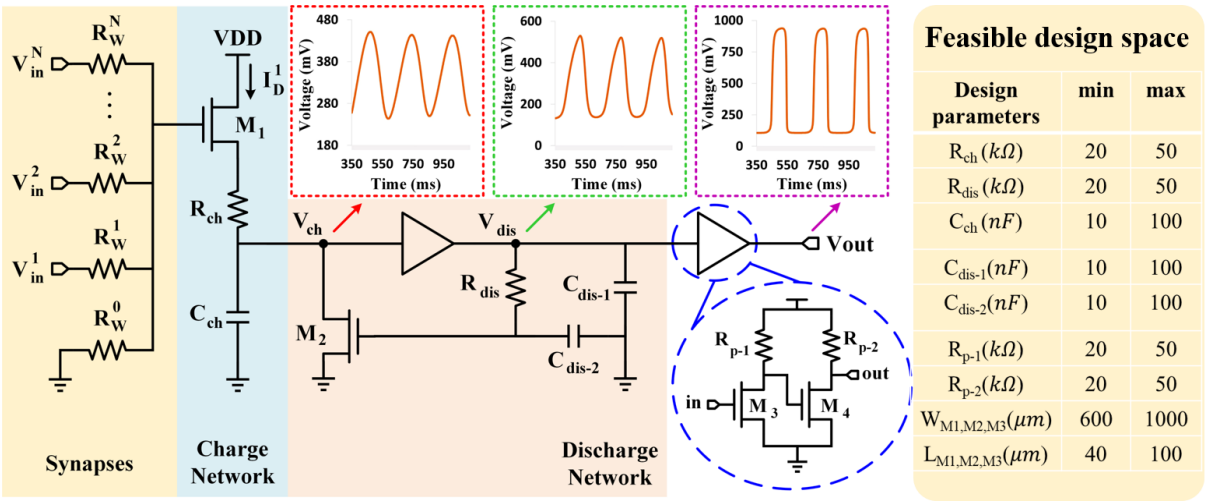
2. Spike generation can be realized using a fast charging mechanism followed by a controlled discharging phase, avoiding ideal instantaneous reset.

Guided by these principles, we develop a printed spike-generator circuit consisting of dedicated charging and discharging stages that convert weighted input voltages into spike-like output pulses.

Synapses Synapses are the part of the neuron that transmit signals from the presynaptic neuron's axon to the postsynaptic neuron's dendrite. For the sake of circuit realization, presynaptic neurons are represented as voltage inputs, and each presynaptic neuron is assigned a weight through a resistor crossbar as shown in Figure 3.12. Using Nodal Analysis, the weighted sum is expressed by

$$\frac{V_g^1}{R_w^0} + \frac{V_g^1 - V_{in}^1}{R_w^1} \cdots + \frac{V_g^1 - V_{in}^N}{R_w^N} = 0, \tag{3.10}$$

where V_g^1 is the gate voltage of M_1, V_{in}^1 is the input voltage to the neuron, R_w^0 works as a voltage divider, R_w^1 and R_w^N are used to resemble the weights of the connected neurons and N is the number of the presynaptic neurons connected as inputs. Additionally, another circuit is required for the negative weights, as proposed in [202]. To determine V_g^1, please refer to subsection 3.2.3 for detailed calculation procedures.



Feasible design space

Design parameters	min	max
R_{ch} $(k\Omega)$	20	50
R_{dis} $(k\Omega)$	20	50
C_{ch} (nF)	10	100
$C_{dis\text{-}1}$ (nF)	10	100
$C_{dis\text{-}2}$ (nF)	10	100
$R_{p\text{-}1}$ $(k\Omega)$	20	50
$R_{p\text{-}2}$ $(k\Omega)$	20	50
$W_{M1,M2,M3}$ (μm)	600	1000
$L_{M1,M2,M3}$ (μm)	40	100

Figure 3.12.: Circuit level implementation of Printed Spiking Neuron which includes three stages: Synapses, Charge Network, and Reset and Discharge Network.

Printed Spike Generator. The schematic of the proposed printed spiking neuron is shown in Figure 3.12. The resistive crossbar used for weighted summation is identical to that employed in the basic p-NNs.

After the weighted summation, the resulting voltage serves as the gate bias of transistor T^{ch}, which regulates the drain–source current responsible for charging the subsequent node. The charging network primarily consists of R^{ch} and C^{ch}, forming an RC stage that generates a voltage V^{ch}. The charging delay is determined by the time constant $R^{ch}C^{ch}$, which directly influences the spike generation frequency.

The voltage V^{ch} is then amplified using a gain stage with an amplifier architecture. The amplifier output is connected to two RC branches. The first branch comprises R_1^{dis} and C_1^{dis}, while the second consists of R_2^{dis} and C_2^{dis}. These RC networks introduce controlled phase shifts due to capacitive charging effects, enabling oscillatory behavior required for spike generation.

The voltage across C_2^{dis} controls transistor T^{dis}, which governs the discharge path of C^{ch}. When the gate voltage of T^{dis} exceeds its threshold, current flows through the transistor and initiates discharge of C^{ch}. Meanwhile, the input from the crossbar continues to supply current to the charging network, causing C^{ch} to recharge. This alternating charging and discharging process produces sustained spike oscillations.

Since the intermediate output voltage at C_1^{dis} remains significantly below the supply voltage V_{DD}, an additional amplification stage is employed to boost the spike amplitude at the final output node V_{out}. This ensures reliable activation of subsequent neurons in the network. The feasible component values used in the spike generator are summarized on right-side of Figure 3.12.

Spiking neural networks (SNNs) are particularly attractive for PE because they process information in an event-driven fashion, potentially reducing switching activity and enabling ultra-low-power operation. This section discusses the progression from an analog printed SNN [253] with a fixed spike generator to a learnable, robustness-aware spike generator as proposed in SpikeSynth [270].

3.2.1. Modeling and training of P-SNN

By connecting multiple printed spiking neurons, the computational paradigm of the SNN can be emulated, thereby offering the potential to achieve the desired functionalities. However, to fully leverage this potential, the component values of the circuits (e.g., the crossbar conductances representing the weights) should be designed and optimized for specific target tasks. For this, it is necessary to establish a p-SNN optimization model.

3.2.1.1. Modeling of P-SNN

An analytical model for the behavior of the resistor crossbar for weighted-sum is given in [220] as

$$V_g^1 = \sum_{n=1}^{N} V_{in}^n \left(w_n \cdot \mathbb{1}_{\{g_n \geq 0\}}\right) + inv(V_{in}^n) \left(w_n \cdot \mathbb{1}_{\{g_n < 0\}}\right), \tag{3.11}$$

where g_n denotes the crossbar conductance by its absolute value and encodes with its sign, if the respective input is inverted (to express negative weight). Furthermore, $inv(\cdot)$ refers to the negative tanh function that describes the transfer characteristic of the negation circuit. Finally, $\mathbb{1}_{\{.\}}$ is an indicator function that returns 1 if the respective condition is true, else 0. Additionally, w_n refers to the weight given by

$$w_n = \frac{|g_n|}{\sum_m |g_m|}. \tag{3.12}$$

To enable gradient-based training via backpropagation [53], a fully differentiable model to describe the transfer characteristic of the printed spike-generator circuit is needed. However, given that the common hardware-agnostic SNN training frameworks are based on (Leaky-) Integration-Fire mechanism [188], [237], they are incompatible with proposed circuits, due to their device and material constraints, and so can not describe their transfer behavior. So, considering the circuit complexity, we utilize a Transformer-based neural network as the *surrogate spike-generator (SG) model* to learn the circuit behavior for mapping the input voltages into the output voltages. A Transformer [114] is a neural network model initially proposed for natural language processing. It is, therefore, aptly suited for processing sequential data. The essential part of the Transformer is the attention mechanism, which enables the model to account for positional and value correlations. The effectiveness of Transformer has been shown by numerous state-of-the-art models like BERT [120] and GPT [217].

To prepare the data required for training the SG model, we conducted 5,000 SPICE simulations for a single spike-generator circuit in Figure 3.12 based on the Printed Process Design Kit (pPDK) [151]. The duration of the input voltage (V_g^1) is 3 s and the temporal step size is 1ms. To ensure that the surrogate model can comprehensively and accurately mimic the behavior of the original spike-generator circuit in any operating scenario, we designed the following patterns of input voltages (V_g^1), namely,

1. constant voltages ranging from 0V to 2V, serving to represent the case of stable inputs;

2. the output voltages obtained from 1), i.e., V'_{out}, representing the case of a cascade of multiple neurons; and

3. diverse harmonic signals with varying frequencies ($0 - 5$Hz), amplitudes ($0 - 1$A), phases ($0 - 2\pi$), and their combinations, expressing the circuit behavior in other complex situations.

After obtaining the data from SPICE simulation, we split them into three sets, a training set (70%) to guide the training of the surrogate model, a validation set (20%) for stopping to avoid overfitting, and a test set (10%) to evaluate the effectiveness of the surrogate model. We use Adam [66] with its default setup as the optimizer and the Mean Squared Error (MSE) between the model output and the SPICE simulation as the metric. After hyperparameter tuning, we choose a Transformer with three causal attention layers as the final surrogate circuit model. Causal attention thereby ensures that outputs are only determined by the signals at previous time steps. Moreover, each layer has three attention heads. The MSE is 1.1×10^{-6} on the validation set and 9.7×10^{-7} on the test set; therefore, we conclude that the model is capable of sufficiently interpolating and accurately predicting the output voltages.

Using the crossbar and the SG model, the transfer characteristic of the printed spiking neuron can be expressed as

$$SG\left(\sum_{n=1}^{N} \mathrm{V}_{in}^{n}\left(w_n \cdot \mathbb{1}_{\{g_n \geq 0\}}\right) + inv(\mathrm{V}_{in}^{n})\left(w_n \cdot \mathbb{1}_{\{g_n < 0\}}\right)\right), \tag{3.13}$$

where $\mathrm{V}_{in}^{i} \in \mathbb{R}^T$ represents the i-th input voltage sequence. Moreover, by connecting multiple spiking neurons, more complex computing tasks can be implemented.

3.2.1.2. Training of P-SNN

Table 3.4.: Result of the Experiment on 13 Benchmark Datasets

Dataset	SNN	P-ANN		P-SNN	
	Accuracy	**Accuracy**	**Power** (mW)	**Accuracy**	**Power** (mW)
Acute Inflammation	1.00 ± 0.00	1.00 ± 0.00	5.51 ± 0.01	1.00 ± 0.00	1.83 ± 0.11
Balance Scale	0.87 ± 0.03	0.89 ± 0.02	4.41 ± 0.04	0.73 ± 0.03	1.42 ± 0.00
Breast Cancer Wisconsin	0.97 ± 0.02	0.96 ± 0.01	11.28 ± 0.03	0.97 ± 0.00	2.60 ± 0.01
Cardiotocography	0.85 ± 0.03	0.75 ± 0.02	19.45 ± 0.01	0.75 ± 0.06	5.30 ± 0.06
Energy efficiency (y_1)	0.99 ± 0.02	0.99 ± 0.01	7.88 ± 0.01	0.92 ± 0.01	2.32 ± 0.03
Energy efficiency (y_2)	0.90 ± 0.09	0.89 ± 0.02	8.33 ± 0.04	0.85 ± 0.06	2.24 ± 0.02
Iris	0.97 ± 0.10	0.65 ± 0.06	5.48 ± 0.08	0.87 ± 0.01	1.89 ± 0.02
Mammographic Mass	0.86 ± 0.12	0.83 ± 0.01	6.45 ± 0.02	0.81 ± 0.05	1.81 ± 0.02
Pendigits	0.32 ± 0.10	0.48 ± 0.06	16.11 ± 0.48	0.46 ± 0.10	4.49 ± 0.21
Seeds	0.95 ± 0.13	0.85 ± 0.09	7.72 ± 0.03	0.83 ± 0.02	1.71 ± 0.05
Tic-Tac-Toe Endgame	0.97 ± 0.05	0.84 ± 0.06	11.54 ± 0.03	0.85 ± 0.05	1.66 ± 0.04
Vertebral column (2 cl.)	0.84 ± 0.10	0.84 ± 0.01	5.84 ± 0.02	0.84 ± 0.04	1.52 ± 0.00
Vertebral column (3 cl.)	0.83 ± 0.11	0.84 ± 0.02	7.23 ± 0.02	0.82 ± 0.13	1.98 ± 0.02
Average	**0.87 ± 0.07**	**0.83 ± 0.03**	**9.02 ± 0.06**	**0.82 ± 0.04**	**2.37 ± 0.05**

Accuracy and power consumption of Printed Neural Network (p-NN), Spiking Neural Network (SNN) and Printed Spiking Neural Network (P-SNN) on the 13 benchmark tasks.

In the training of existing p-NN the cross-entropy loss $L(\cdot)$ is minimized w.r.t. the crossbar conductances g for maximizing the classification accuracy. However, given that the output of p-SNN is a temporal data

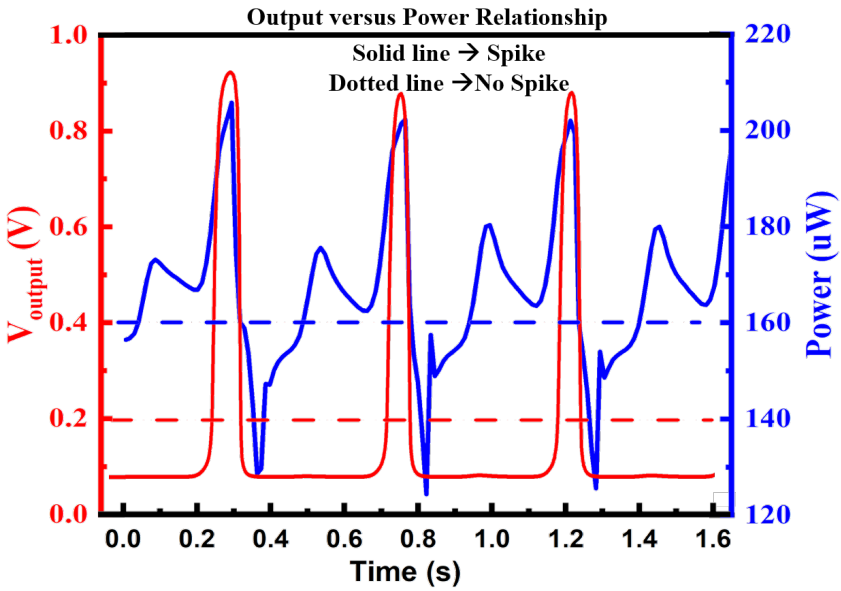


Figure 3.13.: Transient measurement results of the proposed printed spiking neuron.

series, temporal dynamics of the circuit output need to be considered. Therefore, to encourage the overall classification accuracy at every time step, a modified training objective can be formulated as

$$\underset{\mathrm{g}}{\mathrm{minimize}} \ \frac{1}{T}\sum_{t=0}^{T} L\left(\mathrm{x}_t, \mathrm{y}, \mathrm{g}\right), \tag{3.14}$$

where $\mathrm{x} \in \mathbb{R}^{B\times T}$ is input data series with batch size B, $\mathrm{y} \in \mathbb{R}^{B}$ denotes the corresponding classes, and g summarizes all the learnable conductances in the pSNN. Subsequently, as all the operations in Equation 3.18 are fully differentiable, the gradient serves for the parameter update can be calculated by $\frac{1}{T}\sum_{t=0}^{T}\nabla_{\mathrm{g}} L\left(\mathrm{x}_t, \mathrm{y}, \mathrm{g}\right)$. Consequently, gradient-based optimizers, such as Adam [66] and SGD [14], can be used for training.

3.2.1.3. Discussion

The P-SNN work demonstrates that:

- Printed SNNs can be competitive in accuracy with analog ANNs while offering substantial power savings due to event-driven operation.
- A task-specific surrogate modeling flow is crucial: it decouples expensive SPICE simulations from high-level training and allows realistic circuit non-idealities to be folded into learning.
- Nonetheless, the spike generation dynamics remain fixed once the circuit is chosen; they are not directly optimized during learning. This limits adaptability across tasks and prevents power/performance co-optimization at the spike generation level.

These limitations motivate the adaptive spike-generation approach taken by SpikeSynth.

3.2.2. SpikeSynth: Learnable Spike Generator and Robustness-Aware Training

SpikeSynth proposes a new analog spiking neuron with a *learnable spike generator* (LSG) whose temporal characteristics are parameterized and updated during training. Instead of a fixed RC-based spike generator, the neuron includes tunable components such as programmable resistors and capacitors or bias voltages

that shape the charging and discharging time constants and thresholds. These parameters are exposed to the optimization process just like synaptic weights.

At a high level, the LSG design extends the previous three-stage neuron:

- The synaptic crossbar performs the analog weighted sums of signal inputs.
- The charge and reset networks are enriched with learnable parameters: for example, adjustable resistances that control leak rates, reset slopes and spike width.
- The surrogate spike-generation model is extended to include these learnable parameters explicitly, enabling gradient-based tuning of spike timing and shape.

Crucially, the training framework is made robustness-aware. Process variations and device aging in PE are modeled as parameter perturbations (e.g., ±10% variation in key LSG components), and these variations are injected during training so that the learned parameters are robust to such deviations.The power of the LSG and the neuron is also estimated using surrogate models similar to the power-aware and power-constrained frameworks, and power is explicitly included in the loss function, leading to an energy-efficient, variation-aware spike generator.

Training proceeds over 13 benchmark datasets representative of printed edge applications. The network topology and evaluation flow are aligned with the P-SNN baseline to allow fair comparisons.

3.2.2.1. Results

Experimental evaluation shows that the learnable spike generator yields significant improvements over both P-SNNs and P-ANNs:

- On average, the LSG reduces its own power consumption by about 57.6% compared to previous fixed spike-generation circuits.
- By reducing capacitor sizes from the μF to the 10–100 nF range, the average circuit area is cut by roughly 89%, making the P-LSNN smaller than both the P-SNN and even the printed ANN baseline.
- The overall network reduces energy per inference by about 28.7% relative to the non-adaptive P-SNN, while increasing average classification accuracy from around 75% to 83% across the 13 benchmarks.

Under a ±10% process-variation test where noise is injected into both the network weights and LSG parameters, the P-LSNN experiences a modest accuracy drop (around 4.9%), yet its average accuracy under variation remains about 3.3% higher than the baseline P-SNN without variation considered.

The main cost of these gains is training time: offline training for the P-LSNN is roughly 16.4 hours on average, compared to 14.3 hours for P-SNNs and 1.3 hours for P-ANNs.

Overall, these two spiking works show how energy efficiency and robustness can be jointly addressed at the neuron-circuit level through learnable analog dynamics and variation-aware training.

3.2.3. Modeling and training of P-LSNN

By interconnecting multiple printed spiking neurons, the behavior of a SNN can be effectively realized in hardware, enabling a wide range of neuromorphic computations. To understand the architecture, however, it is essential to tailor the circuit-level parameters—such as the conductances in the resistor crossbar that encode synaptic weights—to the requirements of specific target tasks. Achieving this requires the formulation of a learnable optimization framework for the pSNN, allowing the component values to be systematically adjusted through task-driven training objective.

3.2.3.1. Modeling of P-LSNN

The behavior of the printed learnable spike generator (LSG) circuit is modeled using a transformer-based architecture that predicts the output voltage time series $V_{\text{out}}(t)$ as a function of the input voltage $V_{\text{in}}(t)$ and a set of hardware parameters expressed as the vector $\boldsymbol{v}$ where each component, such as the resistances or width and length of the transistors, is restricted with $v_i^{\min} \leq v_i \leq v_i^{\max}$. These parameters are constrained within physically realizable ranges to reflect the functionality within PE.

To respect hardware constraints, each unconstrained parameter $\zeta_i \in \mathbb{R}$ is passed through a scaled tanh function. The output of the surrogate model is given by:

$$\text{LSG}\left(V_{\text{in}}(t), \boldsymbol{v}\right) \tag{3.15}$$

where the components are given by

$$v_i = \frac{v_i^{\max} + v_i^{\min}}{2} + \frac{v_i^{\max} - v_i^{\min}}{2} \tanh(\zeta_i) \tag{3.16}$$

and $[v_i^{\min}, v_i^{\max}]$ defines the hardware-feasible interval for parameter v_i. These transformed values are provided as input to the surrogate model. The input voltage is given by the crossbar:

$$V_{\text{in}}(t) = \sum_{n=1}^{N} V_{\text{in}}^n(t) \left(w_n \cdot \mathbb{1}_{\theta_n \geq 0}\right) + \text{inv}(V_{\text{in}}^n(t)) \left(w_n \cdot \mathbb{1}_{\theta_n < 0}\right). \tag{3.17}$$

Here θ_n represents the learnable parameter where the absolute value is the crossbar conductance, i.e. $g_n = |\theta_n|$ and $\mathbb{1}_{\{\cdot\}}$ is the indicator function that returns 1 if its condition is met and is otherwise 0 and $\text{inv}(\cdot)$ denotes the negative function that is realized by the negation circuit. The weights w_n are given by the conductances normalized by all conductances $w_n = \frac{g_n}{\sum_m g_m}$.

3.2.3.2. Surrogate model for the LSG with hardware constraints

To facilitate gradient-based training through backpropagation [53], it is essential to construct a fully differentiable model that accurately captures the transfer characteristics of the printed spike-generator circuit. This work introduces a novel training methodology for pSNNs by integrating a differentiable, hardware-agnostic surrogate model directly into the optimization pipeline. Unlike prior approaches[253] that relied on surrogate modeling for behavioral approximation, we leverage the Transformer architecture to enable end-to-end gradient-based learning of both network and circuit-level parameters.

The proposed methodology treats these hardware parameters as trainable entities bounded within limits, allowing for simultaneous optimization of functional accuracy and physical realizability. By embedding the Transformer-based surrogate into the training loop, the model captures the dynamic response of the spike generator circuit from input voltages $V_{\text{in}}(t)$ to output voltages $V_{\text{out}}(t)$, while maintaining differentiability for backpropagation [53]. This closed-loop, hardware-constrained training flow allows for co-design of the neural architecture and its physical implementation, enabling adaptive and task-optimized behavior in printed spiking neuromorphic systems. Transformer architectures have demonstrated exceptional versatility across a wide range of domains, forming the backbone of state-of-the-art models like BERT [120] and GPT [217]. In this work, we leverage their sequence modeling capabilities to construct a fully learnable and differentiable approximation of the printed LSG's temporal dynamics.

Algorithm 1 End-to-End Training of P-LSNN with LSG Surrogate Model

Require: Training dataset $\mathcal{D} = \{\mathrm{x}, \mathrm{y}\}$
Require: SPICE simulation dataset $\mathcal{S}$
Train surrogate model LSG on $\mathcal{S}$ using MSE loss
Freeze surrogate model parameters
Initialize network weights θ and LSG parameters ζ
while lr > lr_min **do**
    for each (x, y) in $\mathcal{D}$ **do**
        Generate temporal input sequence $x^{(t)}$
        Run forward pass through the P-LSNN:
            Compute crossbar voltage V_{in} using Eq. (3.17)
            Compute LSG output V_{out} using Eq. (3.15)
        Compute loss using Eq. (3.18) or Eq. (4.5)
        Backpropagate loss w.r.t. θ and ζ
        Update parameters using Adam optimizer
    if Patience > 100 **then**
        lr ← lr / 2

To generate the dataset for training the LSG surrogate model, we conducted over 64K SPICE simulations using the well-established pPDK [129]. Each simulation was executed for a duration of 15 ms with a temporal resolution of 10 μs to capture fine-grained transient behavior of the circuit. The input voltage $V_{\text{in}}(t)$ was varied across multiple regimes to ensure that the surrogate model could generalize to both static and dynamic spiking conditions. The following input categories were considered:

- Constant voltages, ranging from 0 V to 2 V in steps of 0.2 V, which serve as baseline inputs for characterizing stable responses;
- Cascaded neuron outputs, where the output from a first neuron is used as the input to a second neuron, thereby modeling signal propagation through a multi-layer spiking network;
- A set of harmonic input voltage signals with varying amplitudes (A), DC offsets (B), frequencies (f from 0 Hz to 5 Hz), and phase shifts (0 to 2π), intended to emulate dynamic and noisy spiking activity.

Each simulation instance is parameterized by range of learnable parameters as tabulated in Figure 3.12. These parameters are constrained within the intended functionality of the spike generator and serve as the input features to the surrogate model. This comprehensive SPICE dataset uses more than ten times the size of that used in previous studies [253], which did not account for learnable hardware parameters. This enables accurate modeling of the spike-generator circuit under diverse operating conditions. A Transformer model with 3 encoder layers, 3 attention heads, and an embedding size of 48 achieved the best performance during evaluation, resulting in a mean squared error (MSE) of $7.3 \cdot 10^{-3}$ on the test set.

3.2.3.3. Training of the P-LSNN

After pretraining, the surrogate model is frozen and used to replace the LSG circuit in the full P-LSNN. The additional parameters ζ_i are now treated as independently trainable variables for each LSG circuit. These are optimized together with the rest of the P-LSNN using gradient descent.

The P-LSNN receives temporally extended input signals and produces spike-based outputs over time. To promote consistent prediction, the cross-entropy loss is computed at every time step and averaged across the sequence,

which leads to this optimization objective:

$$\underset{\theta,\zeta}{\mathrm{minimize}} \frac{1}{T} \sum_{t=0}^{T} L\left(\mathrm{x}_t, \mathrm{y}, \theta, \zeta\right), \tag{3.18}$$

where $x \in \mathbb{R}^{B \times T}$ is the input data series with batch size B, $y \in \mathbb{R}^{B}$ denotes the corresponding classes, θ is the vector of all the learnable parameters from the crossbar conductances and ζ is the vector of the learnable hardware parameters of the LSG circuit.

Since all components, including the LSG model with constrained v_i, are differentiable, the full system supports end-to-end training.

Table 3.5.: Accuracy Comparison Across Different Printed Nueromorphic Architectures across 13 benchmark datasets

Dataset	SNN[1]	P-ANN[2][243]	P-SNN[3][253]	P-LSNN[4]
Acute.	1.00	0.999 ± 0.002	1.000 ± 0.000	1.000 ± 0.000
Bal.	0.84	0.893 ± 0.020	0.430 ± 0.000	0.819 ± 0.073
Breast.	0.98	0.915 ± 0.020	0.975 ± 0.000	0.966 ± 0.007
Cardio.	0.84	0.824 ± 0.014	0.747 ± 0.000	0.755 ± 0.015
En(y1)	1.00	0.918 ± 0.014	0.848 ± 0.000	0.848 ± 0.000
En(y2)	0.98	0.892 ± 0.010	0.860 ± 0.000	0.869 ± 0.026
Iris	1.00	0.958 ± 0.009	0.710 ± 0.171	0.946 ± 0.019
Mamm.	0.82	0.789 ± 0.006	0.810 ± 0.007	0.810 ± 0.013
Pen.	0.38	0.371 ± 0.034	0.400 ± 0.070	0.536 ± 0.060
Seed.	1.00	0.900 ± 0.015	0.837 ± 0.070	0.915 ± 0.013
Tic-Tac.	1.00	0.818 ± 0.004	0.789 ± 0.019	0.873 ± 0.125
Vert.(2 cl.)	0.85	0.768 ± 0.038	0.635 ± 0.000	0.635 ± 0.000
Vert(3 cl.)	0.83	0.819 ± 0.004	0.683 ± 0.016	0.778 ± 0.057
Average	**0.886**	**0.836 ± 0.015**	**0.748 ± 0.027**	**0.830 ± 0.031**

[1] SNN: Software Spiking Neural Network [2] P-ANN: Printed Artificial Neural Network [3] P-SNN: Printed Spiking Neural Network [4] P-LSNN: Printed Learnable Spiking Neural Network.

This methodology enables full backpropagation-based training of P-LSNNs by introducing a differentiable surrogate for the LSG circuit, parameterized by constrained learnable variables, allowing both the circuit-level parameters and network weights to be jointly optimized in a hardware-aware adaptive manner. The entire end-to-end training of the P-LSNN is shown in algorithm 1.

3.2.4. Robustness-aware P-LSNN Training

In PE, the parameters associated with hardware components are highly susceptible to process-induced variations. These variations arise due to factors such as ink spreading, droplet jetting irregularities, and satellite droplet wetting on the substrate [98]. Specifically, printed n-type EGTs exhibit variability across multiple fabrication stages—including channel, dielectric, and top-gate formation—resulting in non-Gaussian distributions for electrical and structural properties [129].

To account for these fabrication non-idealities during training, we model both $\tilde{\theta}$ and $\tilde{\zeta}$ as random variables drawn from distributions $p(\tilde{\theta})$ and $p(\tilde{\zeta})$, respectively. $\tilde{\zeta}$ represents the parameters for the LSG circuit and $\tilde{\theta}$ the remaining parameters in the P-LSNN. These distributions capture the statistical nature of variation introduced during the printing process. Consequently, the training objective is defined over these distributions:

$$\underset{\tilde{\theta},\tilde{\zeta}}{minimize}\, L(\mathcal{D}, \tilde{\theta}, \tilde{\zeta}), \tag{3.19}$$

where the parameters $\tilde{\theta}$ and $\tilde{\zeta}$ are resampled from their respective distributions at each forward pass, allowing the model to encounter diverse perturbations over the course of training. Here, $\mathcal{D} = \{x, y\}$ denotes the training dataset, and $L(\cdot)$ is the task-specific loss function (e.g., cross-entropy [232]).

To make the formulation differentiable and trainable via gradient-based methods, we employ a reparameterization trick [56], expressing the random variables as:

$$\tilde{\theta} = \theta \odot \boldsymbol{\varepsilon}_{\theta}, \quad \tilde{\boldsymbol{\zeta}} = \boldsymbol{\zeta} \odot \boldsymbol{\varepsilon}_{\zeta},$$

where θ and $\boldsymbol{\zeta}$ are the nominal, learnable parameters, and $\boldsymbol{\varepsilon}_{\theta} \sim p(\boldsymbol{\varepsilon}_{\theta})$ and $\boldsymbol{\varepsilon}_{\zeta} \sim p(\boldsymbol{\varepsilon}_{\zeta})$ represent multiplicative noise sampled from the variation distributions, which in our case are uniform. Substituting this into the loss yields modified training objective:

$$\underset{\theta,\boldsymbol{\zeta}}{minimize}\, L(\mathcal{D}, \theta \odot \boldsymbol{\varepsilon}_{\theta}, \boldsymbol{\zeta} \odot \boldsymbol{\varepsilon}_{\zeta}). \tag{3.20}$$

3.2.5. Evaluation

To assess the effectiveness of the proposed pSNN with learnable spike-generation capabilities, we first designed the spike-generator incorporating synaptic inputs, and implemented the complete training pipeline using PyTorch [149].[3] We performed a comparative evaluation of our learnable pSNN against prior works [202], [253] and SNNs based on benchmark frameworks [188].

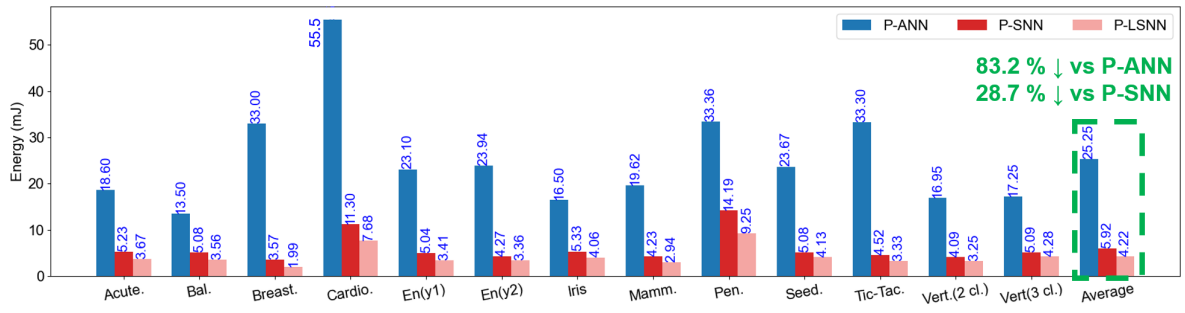


Figure 3.14.: Comparison of energy (mJ) utilization for P-ANN[243], P-SNN[253] and proposed P-LSNN across 13 benchmark datasets.

3.2.6. Experiment

To evaulate our proposed approach, we utilized the well-established n-EGT pPDK [129] to design both the synapses and the learnable spike-generator. To evaluate their spiking behavior, we conducted more than 60K SPICE simulations to understand the circuit spike-timing behavior w.r.t the components variation in Cadence Virtuoso[4], as shown in Figure 3.12.

3.2.6.1. Training and Evaluation Setup

Datasets and Hyperparameter choices Although SNNs or p-SNNs exhibit capabilities in processing temporal information datasets, most SOTA research on SNNs are typically employing "temporized" datasets. Here, temporized datasets refers to datasets that contain originally only non-temporal information. However, they are converted into temporal datasets through specific methods, e.g., encoding a spike train with the spike density proportional to the magnitude of the values in the datasets, as represented in Figure 3.15. To evaluate the proposed P-LSNN we used 13 benchmark datasets which are aligned to the applications in PE. We used a datasplit of 60%, 20% and 20% for the training, validation and test set, respectively. To optimize the model parameters including the learnable hardware parameters ζ_i we used the ADAM [66] algorithm with an inital learning rate of 0.1. This initial learning rate was halved if the

[3] https://github.com/Neuromophic-Computing/SpikeSynth
[4] https://www.cadence.com/en_US/home.html

Table 3.6.: Robustness-aware accuracy Across Models with ± 10% Component Variation across 13 Benchmark Datasets

Dataset	P-ANN[243]	P-SNN[253]	P-LSNN
Acute.	1.000 ± 0.012	1.000 ± 0.000	0.999 ± 0.007
Bal.	0.877 ± 0.008	0.457 ± 0.049	0.548 ± 0.147
Breast.	0.931 ± 0.039	0.974 ± 0.030	0.958 ± 0.016
Cardio.	0.763 ± 0.002	0.747 ± 0.000	0.757 ± 0.017
En(y1)	0.847 ± 0.012	0.857 ± 0.012	0.883 ± 0.052
En(y2)	0.867 ± 0.026	0.841 ± 0.026	0.854 ± 0.055
Iris	0.843 ± 0.045	0.637 ± 0.068	0.909 ± 0.076
Mamm.	0.766 ± 0.053	0.808 ± 0.010	0.808 ± 0.011
Pen.	0.548 ± 0.047	0.486 ± 0.058	0.466 ± 0.078
Seed.	0.820 ± 0.041	0.709 ± 0.042	0.856 ± 0.092
Tic-Tac.	0.660 ± 0.017	0.763 ± 0.026	0.804 ± 0.119
Vert.(2 cl.)	0.661 ± 0.000	0.635 ± 0.000	0.635 ± 0.000
Vert(3 cl.)	0.634 ± 0.075	0.677 ± 0.012	0.683 ± 0.050
Average	**0.786 ± 0.029**	**0.738 ± 0.024**	**0.781 ± 0.055**

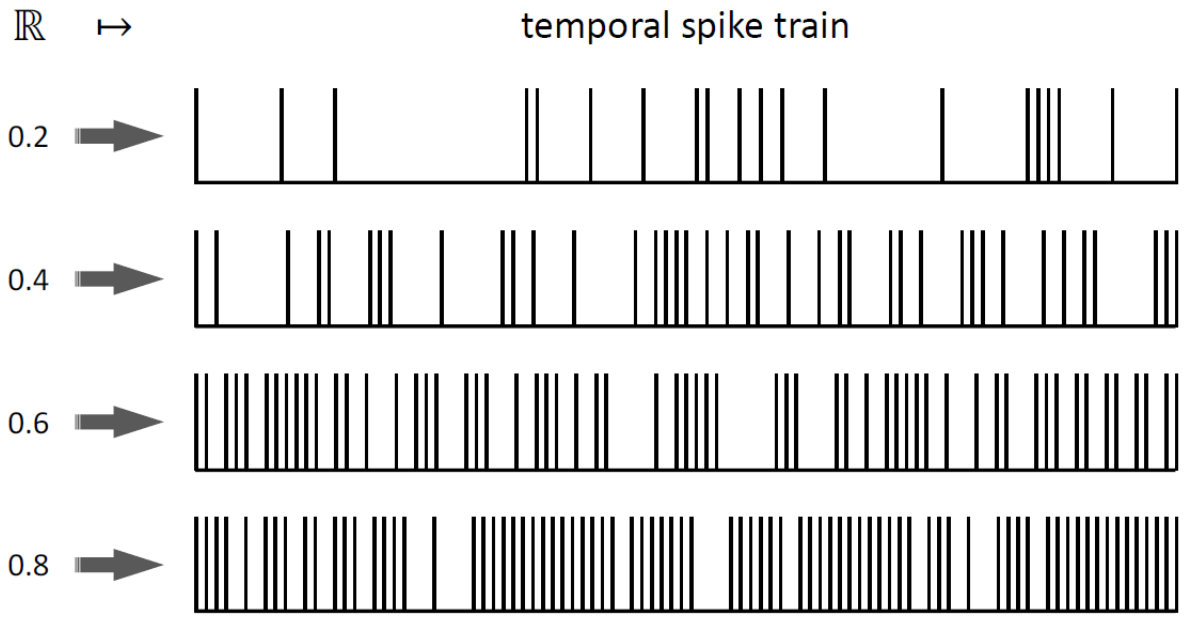


Figure 3.15.: Dataset temporization that encodes real numbers into the density of the temporal spike trains.

validation accuracy showed no further improvement for 100 epochs. This process was repeated 10 times until a sufficiently small learning rate was reached. To ensure to reduce the sensitivity to initialization, the process was executed 10 times with random seeds from 0 to 9, ensuring the reliability of the resulting solution.

Baselines We employed another two approaches as the baselines of P-SNNs to validate the major motivation of this work, that is, energy saving. *p*-NNs [202] with the same topologies as P-SNNs are trained on the corresponding datasets. Additionally, considering the target computing paradigm of this work, P-SNN is compared with its hardware-agnostic counterpart by conducting training on SNNs with the leaky-integration-fire mechanism [188].

Table 3.7.: Comparison of Power, Area, and Training Time with the Existing Methods on 13 Benchmark Datasets.

Dataset	Power (mW)					Area (cm²)			Training Time (hr)		
	P-ANN[243]	P-SNN[253]		P-LSNN		P-ANN[243]	P-SNN[253]	P-LSNN	P-ANN[243]	P-SNN[253]	P-LSNN
	Total[1]	SG[2]	Total[1]	LSG[2]	Total[1]						
Acute.	6.20	0.85	1.744	0.389	1.223	1.65	12.51	1.36	0.55	4.75	5.27
Bal.	4.50	1.02	1.694	0.426	1.186	2.68	15.03	1.80	1.69	9.85	16.17
Breast.	11.00	0.85	1.189	0.411	0.664	1.76	12.50	1.70	1.39	16.46	15.98
Cardio.	18.50	1.02	3.767	0.353	2.560	2.38	15.06	2.02	1.98	29.02	35.20
En(y1)	7.70	1.02	1.680	0.466	1.138	2.28	15.07	1.71	1.48	19.69	24.42
En(y2)	7.98	1.02	1.424	0.455	1.119	1.88	15.03	1.90	1.38	21.45	21.20
Iris	5.50	1.02	1.778	0.382	1.353	2.27	15.01	1.81	0.54	4.31	5.43
Mamm.	6.54	0.85	1.410	0.382	0.981	2.13	12.52	1.66	1.19	13.42	14.65
Pen.	11.12	2.21	4.730	0.878	3.083	5.51	32.54	1.71	3.27	32.69	40.45
Seed.	7.89	1.02	1.693	0.410	1.376	2.62	15.07	1.78	0.78	5.71	5.57
Tic-Tac.	11.10	0.85	1.506	0.366	1.109	1.99	12.53	1.40	1.26	13.78	13.57
Vert.(2cl.)	5.65	0.85	1.362	0.422	1.084	2.38	12.57	1.30	0.78	7.72	7.53
Vert(3cl.)	5.75	1.02	1.697	0.392	1.427	2.38	15.01	1.53	0.92	7.21	7.59
Average	**8.42**	**1.04**	**1.974**	**0.44↓**	**1.407↓**	**2.46**	**15.41**	**1.67↓**	**1.32**	**14.31**	**16.39**

[1] Total: Total Power [2] SG: Spike-generator power [3] LSG: Learnable SG power

3.2.6.2. Accuracy

For a robust estimation of accuracy, we selected the top-k models ($k = 3$) from 10 independent training runs with different random seeds. The average accuracy and standard deviation were then calculated from these models on the test set. For robustness-aware training, we injected noise onto the weights to emulate imprecise hardware components. Since the evaluation results on the test set are not deterministic, we sample each test dataset 10 times per model in the robustness case and average the results.

3.2.6.3. Power

To evaluate the efficiency of the proposed pSNN architecture, we estimated the total power consumption by summing the contributions of three distinct components: the crossbar array, the negation circuit (inverter), and the spike-generation (LSG) circuit. Each component is modeled based on its specific electrical behavior to ensure accurate and hardware-relevant power estimates. To estimate the power of the crossbar and the negation circuit we used similar methods as in prior works [243]. Estimating the power consumption of the LSG circuit is more complex due to its temporal and nonlinear behavior. To address this, we trained a feed-forward neural network to predict the LSG power based on the learned hardware parameters ζ_i. This power model was trained on a dataset of SPICE simulations with a wide range of parameter combinations and input patterns. The data was split into a training (70%), validation (15%) and test (15%) set. After hyperparameter optimization we trained a feed-forward network with 3 hidden layers (256, 128, 64) until the validation accuracy did not change for 100 epochs. The resulting test loss (MSE) was $7.0 \cdot 10^{-6}$.

3.2.7. Result

Table 3.5 reports per-dataset classification accuracy under standard (non–robustness-aware) training for four models: the reference SNN, P-ANN, baseline P-SNN, and proposed pSNN. Averaged across all datasets, the SNN achieves 88.6 % accuracy, followed by P-ANN at 83.6 %, pSNN at 83.0 %, and P-SNN at 74.8 %. To assess robustness under hardware variability, we trained and evaluated each model with ±10% random weight perturbations (Table 3.6). Under these perturbations, P-ANN achieves 78.6 %, pSNN 78.1 %, and P-SNN 73.8 %.

Figure 3.14 plots energy consumption in millijoules (mJ) for P-ANN, P-SNN, and pSNN across all datasets: P-ANN ranges from 13.50 mJ (Bal) to 55.50 mJ (Cardio); P-SNN ranges from 3.57 mJ (Breast) to 14.19 mJ (Pen); and pSNN ranges from 1.99 mJ (Breast) to 9.25 mJ (Pen).

Table 3.7 compares power consumption (mW), area (cm^2), and training time (hr) for P-ANN, P-SNN, and pSNN across all datasets. On average, P-ANN consumes 8.42 mW, occupies 2.46 cm^2, and requires 1.32 hr of training; P-SNN consumes 1.97 mW, occupies 15.41 cm^2, and requires 14.31 hr; and pSNN consumes 1.41 mW, occupies 1.67 cm^2, and requires 16.39 hr. Additionally, on average, the LSG circuit's power drops from 1.04 mW in the printed SG[253] to 0.44 mW.

Discussion The comparison between the proposed P-LSNN architecture and the baseline P-SNN model demonstrates notable improvements in accuracy, area, and energy efficiency. Incorporating learnable adaptive parameters into the LSG circuit significantly enhances performance across several datasets. In particular, certain benchmarks exhibit substantial accuracy gains, while others maintain performance levels comparable to the baseline.

On average, the P-LSNN achieves an 8.2 % increase in classification accuracy relative to the P-SNN model. In addition to improved accuracy, the P-LSNN also shows a considerable power reduction. Specifically, the average energy utilization of the P-LSNN (as shown in Figure 3.14) is reduced by 28.7 % compared to P-SNN and by 83.2 % when compared to the P-ANN baseline [243]. Our experiments also reveal that, on average, our proposed LSG reduced the power consumption by 57.6% compared to previous works[253] resulting in a highly energy-efficient printed SNN. Also, the large capacitors ($10\,\mu F$) in the P-SNNs[253] significantly increased area requirements. By reducing capacitor values from $10\,\mu F$ to $10 - 100\,nF$ range, the proposed P-LSNN's average circuit area is significantly reduced by 89%, which is much smaller than the P-SNN and even smaller than the P-ANN (2.46 cm^2). Thus, the remaining trade-off for the P-LSNN model is the increased training time—averaging 16.39 hours, compared to 1.32 hours for the P-ANN and 14.31 hours for the P-SNN. Despite this longer training duration, the considerable gains in power efficiency, reduced area footprint, and improved robustness justify the additional training cost.

Furthermore, the robustness was also evaluated under 10% process variation by injecting noise into the network weights and learnable circuit components. As expected, both P-SNN and P-LSNN models exhibited a slight degradation in performance. The P-SNN showed an average drop of 1.0%, while the P-LSNN experienced 4.9% reduction. This is attributed to the fact that, for P-SNN, the SG circuit components were excluded from noise injection, whereas in the P-LSNN, all learnable parameters were subject to variation. Despite this, the average accuracy of the proposed approach under variation remained 3.3% higher than that of the baseline without variation, underscoring the robustness and generalization capacity of the proposed approach.

Overall, the integration of adaptive parameters into the LSG not only improves accuracy and power efficiency but also improves the circuit adaptability with respect to the spike-timing events and also offers resilience against practical variability, making P-LSNN a promising candidate for ultra-low-power adaptive neuromorphic applications.

3.3. Temporal Processing and On-Sensor Intelligence

p-NNs must be not only energy efficient but also robust to time-series data and temporal distortions and sensor-level noise. Moreover, in many applications it is desirable to perform classification directly at the sensor to minimize communication energy. This section covers three complementary directions:

1. Temporal processing with learnable filters in *p*-NNs.
2. Robustness-aware adaptive temporal filters (ADAPT-*p*-NN).
3. On-sensor digital intelligence via bespoke ADC and decision-tree co-design.

3.3.1. Recurrent Neural Networks (RNNs)

RNN were primarily proposed to handle inputs with variable lengths, as the input dimensionality for an MLP is pre-determined. By incorporating an internal hidden state h, an RNN is then capable of

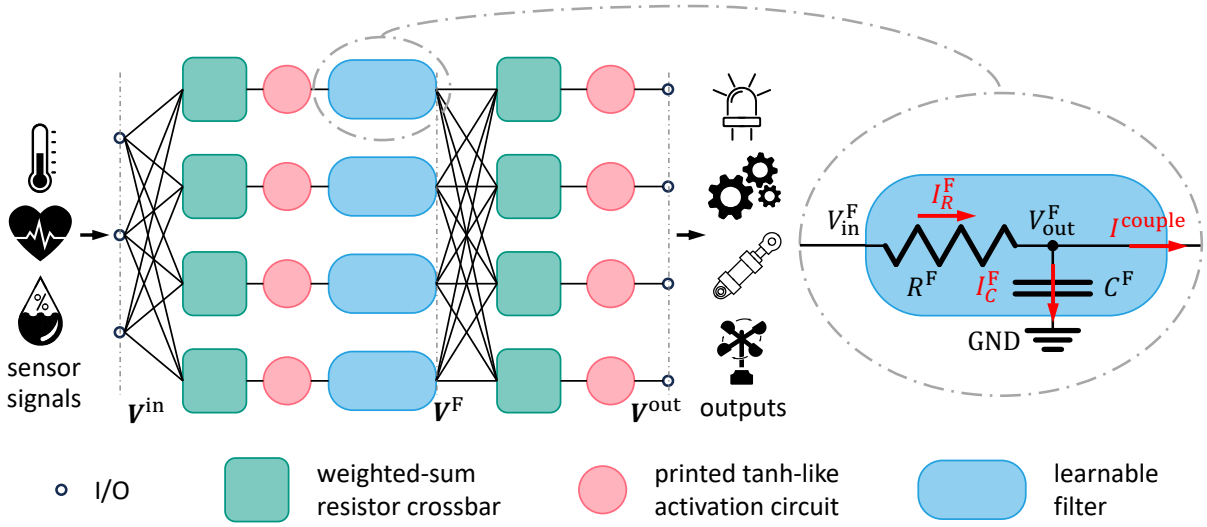


Figure 3.16.: Schematic of a 3-input 4-output *pTPB* that receives sensor signals and yields outputs to subsequent devices.

processing variable length inputs through repeated state updates. RNNs have demonstrated remarkable success in areas such as handwriting recognition [34]. Notably, RNNs are theoretically Turing-complete, meaning that they can execute arbitrary programs to process any given input sequences [16]. A general formulation of RNN state equations is given by

$$\begin{aligned} \mathrm{h}_k &= f_1(f_2(\mathrm{h}_{k-1}) + f_3(\mathrm{x}_k)), \\ \mathrm{y}_k &= f_4(\mathrm{h}_k), \end{aligned} \tag{3.21}$$

where the subscript $k \in \{0, 1, \cdots, K\}$ refer to the iteration (time step), h_k is the internal hidden state at the k-th step, x_k denotes the input at the k-th step, and y_k represents the output at the k-th step. Moreover, the functions $f_1(\cdot), \cdots, f_4(\cdot)$ are classic operations in ANNs, such as learnable affine mappings and/or activation functions. The specific choices of them vary across different network architectures, e.g., in Elman RNNs [11], $f_2(\cdot)$ and $f_3(\cdot)$ are weighted-sum operations with biases, while $f_1(\cdot)$ and $f_4(\cdot)$ are functions of learnable linear mappings with activation functions.

3.3.2. Modeling of Temporal Processing Unit

The structure of the *pTPB* is represented in Figure 3.16. It can be seen that the most essential part in the circuit are framed by the blue boxes, which consist of capacitors and resistors, resembling RC low-pass filters. Therefore, we will first model these filters while considering their coupling with the rest of the circuit. Afterwards, we will develop the model to cover the entire *pTPB*.

Modeling of single filter Initially, we concentrate on modeling the filter without considering its coupling to the successive circuit. Taking the first unit in Figure 3.16 as an example, we obtain:

$$\begin{aligned} I_R^{\text{F}} &= \left(V_{in}^{\text{F}} - V_{out}^{F}\right) / R^{\text{F}}, \\ I_C^{\text{F}} &= C^{\text{F}} \mathrm{d}V_{out}^{\text{F}} / \mathrm{d}t, \\ I_R^{\text{F}} &= I_C^{\text{F}}, \end{aligned} \tag{3.22}$$

where the superscript $(\cdot)^{\text{F}}$ indicates the values in this filtering unit. Therefore, the differential equation of the capacitor voltage V_{out}^{F} with respect to time can be expressed by

$$\frac{\mathrm{d}V_{out}^{\text{F}}}{\mathrm{d}t} = -\frac{1}{R^{\text{F}}C^{\text{F}}} V_{out}^{\text{F}} + \frac{1}{R^{\text{F}}C^{\text{F}}} V_{in}^{\text{F}}.$$

By using backward Euler integration, we obtain the update of the V^{F}:

$$\begin{aligned} V^{\mathrm{F}}_{outk} &= \underbrace{\frac{R^{\mathrm{F}}C^{\mathrm{F}}}{R^{\mathrm{F}}C^{\mathrm{F}}+\Delta t}}_{=:\beta} V^{\mathrm{F}}_{outk-1} + \underbrace{\frac{\Delta t}{R^{\mathrm{F}}C^{\mathrm{F}}+\Delta t}}_{=:1-\beta} V^{\mathrm{F}}_{ink} \\ &= \beta V^{\mathrm{F}}_{outk-1} + (1-\beta)V^{\mathrm{F}}_{ink}, \end{aligned} \tag{3.23}$$

where Δt refers to the step size of the temporal discretization, V^{F}_{ink} and V^{F}_{outk} are the input and output of the filter at time step k. Evidently, $\beta \in (0,1)$ depends on R^{F} and C^{F}, thus, by finding suitable R^{F} and C^{F}, a desired filtering behavior can be achieved. In this work, as these values will be learned jointly with the crossbar resistors to fit the specific tasks, we refer to these units as *learnable filters*.

Modeling of coupled filter To connect the learnable filters with the resistor crossbars, it is imperative to take their coupling into account. This coupling primarily results from the fact that the current flowing through the resistor R^{F} does not fully feed into the capacitor C^{F}, but is partially shunted towards the crossbar, see red arrows in Figure 3.16. To reflect this coupling in our model, we modify Equation 3.22 to

$$I^{\mathrm{F}}_{R} = I^{\mathrm{F}}_{C} + I^{couple} =: \mu I^{\mathrm{F}}_{C},$$

with I^{couple} refers to the coupling current flows towards crossbar, and $\mu := 1 + I^{couple}/I^{\mathrm{F}}_{C}$ is a decoupling factor. Consequently, Equation 3.23 is reformulated to

$$\begin{aligned} V^{\mathrm{F}}_{outk} &= \underbrace{\frac{\mu R^{\mathrm{F}}C^{\mathrm{F}}}{\mu R^{\mathrm{F}}C^{\mathrm{F}}+\Delta t}}_{=:\beta'} V^{\mathrm{F}}_{outk-1} + \underbrace{\frac{\Delta t}{\mu R^{\mathrm{F}}C^{\mathrm{F}}+\Delta t}}_{=:1-\beta'} V^{\mathrm{F}}_{ink} \\ &= \beta' V^{\mathrm{F}}_{outk-1} + (1-\beta')V^{\mathrm{F}}_{ink}, \end{aligned} \tag{3.24}$$

It is notable that μ is contingent on the values of R^{F}, C^{F} and R^{C}, which vary continuously during the training process. Additionally, μ is also determined by the frequency of the input signal, which is generally agnostic in the design stage. Therefore, to consider the circuit coupling in the training process, we determine the general range of μ through the SPICE simulation with the pPDK [129]. Specifically, by performing SPICE simulations on the target datasets (see subsection 6.1.4) with general printable resistances and capacitances, we empirically determined $\mu \in [1, 1.3]$ for the given applications. More details can be found in paragraph 3.4.5.

3.3.3. Modeling of Temporal Processing Block

Although Equation 3.24 emulates Equation 3.21, it possesses only one learnable parameter, i.e., β'. To expand the design space, and better mimic the expressiveness of classic RNNs, a more sophisticated combination of the learnable filters, crossbars, and ptanh circuits should be designed.

As sketched in Figure 3.16, to match analogous computational capabilities of classic Elman RNNs, we first pass the input voltages through resistor crossbars followed by ptanh activation circuits, before feeding them to the filters. Here, the ptanh circuits are introduced to decouple the learnable filters from the preceding crossbars, because proper weighted-sum computation through the crossbar necessitates a negligible output current I^{C}_{out}. However, the resistivity of the filter circuit is much lower than the crossbars. Additionally, we also apply an identical process to output voltages from learnable filters. In the rest of the work, we refer to this stack of primitive layers as a *pTPB*, i.e., the whole circuit exemplified in Figure 3.16. Consequently, the mathematical model of a *pTPB* can be described as

$$\begin{aligned} \mathrm{V}^{\mathrm{F}}_{k} &= \mathrm{'} \odot \mathrm{V}^{\mathrm{F}}_{k-1} + (1-\mathrm{'}) \odot \mathrm{p}tanh(\mathrm{W}_1\mathrm{V}^{in}_{k} + \mathrm{b}_1), \\ \mathrm{V}^{out}_{k} &= \mathrm{p}tanh(\mathrm{W}_2\mathrm{V}^{\mathrm{F}}_{k} + \mathrm{b}_2), \end{aligned} \tag{3.25}$$

where $\mathrm{V}_k^{in} \in \mathbb{R}^{N_{in}}$ and $\mathrm{V}_k^{out} \in \mathbb{R}^{N_{out}}$ vectorize the input and output voltages of the *pTPB* at time point k. $\mathrm{V}_k^{\mathrm{F}} \in \mathbb{R}^{N_{\mathrm{F}}}$ summarizes the output voltages of the filter layer and $' \in \mathbb{R}^{N_{\mathrm{F}}}$ collects the β' values of each filter. Moreover, $\mathrm{W}_1 \in \mathbb{R}^{N_{\mathrm{F}} \times N_{in}}$, $\mathrm{b}_1 \in \mathbb{R}^{N_{\mathrm{F}}}$, $\mathrm{W}_2 \in \mathbb{R}^{N_{out} \times N_{\mathrm{F}}}$, and $\mathrm{b}_2 \in \mathbb{R}^{N_{out}}$ denotes the weighted-sum operations emulated by the corresponding crossbars. Additionally, "$\odot$" indicates element-wise multiplication.

By comparing Equation 3.25 with Equation 3.21, we conclude that, the designed circuit layer represents an instance of an RNN with $f_1(\cdot)$ and $f_2(\cdot)$ being identity functions, while $f_3(\cdot)$ and $f_4(\cdot)$ are weighted-sums followed by activation functions.

Notably, a *pTPNN* may consist of multiple *pTPB*s connected successively for accomplishing more intricate computational tasks. In case of multiple *pTPB*s, we denote the initial input voltages (typically sensor signals) by x_k, and represent the final output of the last layer in the circuit by $\hat{\mathrm{y}}_k('$, g, $\mathrm{x}_k, \mathrm{x}_{k-1}, \cdots, \mathrm{x}_0)$, which is a function of $'$ in the learnable filters, the crossbar conductances g, and the input voltages at all time steps $\mathrm{x}_k, \cdots, \mathrm{x}_0$.

3.3.4. Training of pT*p*-NNs

In case of training basic *p*-NN (without *pTPB*), the cross-entropy loss $L(\cdot)$ can be minimized with respect to crossbar conductances g to decrease the mismatch between the label y and the circuit prediction $\hat{\mathrm{y}}$(g, x) for an input x, and thus improve, e.g., the classification accuracy. In contrast, the pT*p*-NNs is time-dependent and allows to obtain predictions for each time step y_k based on the current input x_k and previous inputs $\mathrm{x}_{k-1}, \cdots, \mathrm{x}_0$. We thus consider the temporal dynamics of the circuit output and, to encourage consistent correct classification at every point in time, the objective function can be modified to

$$\underset{',\mathrm{g}}{\mathrm{minimize}} \underbrace{\frac{1}{K}\sum_{k=0}^{K} L\left(\hat{\mathrm{y}}_k('\text{, g}, \mathrm{x}_k, \mathrm{x}_{k-1}, \cdots, \mathrm{x}_0), \mathrm{y}\right)}_{\mathcal{L}('\text{,g},\mathrm{x}_K,\cdots,\mathrm{x}_0,\mathrm{y})}. \tag{3.26}$$

Additionally, it is necessary to consider the dependency of the decoupling factor μ and the initial voltages of the capacitors. The former has been previously mentioned in subsection 3.3.2, while the latter is generally caused by the preceding input signal. To reduce the dependencies of the circuit coupling (μ) and the initial voltage $\mathrm{V}_0^{\mathrm{F}}$ on the results, we integrate our loss function over the value ranges for both variables, assuming $[1, 1.3]$ for μ, and $[0, 1]$ for $\mathrm{V}_0^{\mathrm{F}}$. Through this, we should achieve a configuration of that learnable parameters g and $'$ that is robust to the choice of either value, which leads to the training objective of

$$\underset{',\mathrm{g}}{\mathrm{minimize}} \int \mathcal{L}('\text{, g}, \mathrm{x}_K, \cdots, \mathrm{x}_0, \mathrm{y}, \mu, \mathrm{V}_0^{\mathrm{F}})\, \mathrm{d}\mu\, d\mathrm{V}_0^{\mathrm{F}}.$$

Unfortunately, no analytical solution for the integral (or its gradient with respect to the learnable parameters) exists. We thus rewrite the minimization of the training objective using equivalent formulation as an expected value

$$\underset{',\mathrm{g}}{\mathrm{minimize}}\ \mathbb{E}_{p(\mu),p(\mathrm{V}_0^{\mathrm{F}})}\left\{\mathcal{L}('\text{, g}, \mathrm{x}_K, \cdots, \mathrm{x}_0, \mathrm{y}, \mu, \mathrm{V}_0^{\mathrm{F}})\right\}, \tag{3.27}$$

which allows to obtain Monte Carlo estimates of the function value (and its gradients) whenever needed. Consequently, based on the ranges for $\mathrm{V}_0^{\mathrm{F}}$ and μ, we choose $p(\mathrm{V}_0^{\mathrm{F}}) = \mathcal{U}[0, 1]$ and $p(\mu) \sim \mathcal{U}[1, 1.3]$, i.e., uniform densities over their assumed ranges.

Notably, given that the circuit operates continuously on input signals rather than performing a one-time computing, the circuit latency is implicitly incorporated in the training objective Equation 3.27. By encouraging more correct classifications at each time step, the circuit should be trained to achieve correct output as fast as possible.

Discussion In this section, we modeled the learnable filters with consideration of the circuit coupling. Subsequently, we proposed the *pTPB* for temporal data processing, and shown that the proposed *pTPB* forms an instance of RNN. Finally, we formulated the optimization objective for the pT*p*-NNs. While

Dataset	RG	*p*-NN	Elman RNN	*pTPNN*
CBF	0.335	0.456 ± 0.038	0.683 ± 0.036	0.907 ± 0.015
DPTW	0.441	0.507 ± 0.006	0.764 ± 0.012	0.654 ± 0.007
FRT	0.520	0.597 ± 0.120	0.795 ± 0.030	0.761 ± 0.076
FST	0.492	0.509 ± 0.066	0.798 ± 0.068	0.765 ± 0.015
GPAS	0.390	0.452 ± 0.003	0.768 ± 0.023	0.682 ± 0.106
GPMVF	0.567	0.637 ± 0.054	0.829 ± 0.108	0.891 ± 0.163
GPOVY	0.557	0.540 ± 0.007	1.000 ± 0.000	1.000 ± 0.000
MPOAG	0.483	0.560 ± 0.042	0.708 ± 0.035	0.712 ± 0.006
MSRT	0.283	0.261 ± 0.008	0.625 ± 0.068	0.503 ± 0.208
PowerCons	0.445	0.651 ± 0.010	0.982 ± 0.008	0.801 ± 0.040
PPOC	0.655	0.711 ± 0.001	0.724 ± 0.006	0.743 ± 0.005
SRSCP2	0.464	0.489 ± 0.011	0.742 ± 0.010	0.782 ± 0.010
Slope	0.501	0.559 ± 0.002	0.963 ± 0.036	0.898 ± 0.159
SmoothS	0.268	0.447 ± 0.011	0.648 ± 0.010	0.694 ± 0.063
Symbols	0.152	0.141 ± 0.002	0.660 ± 0.049	0.670 ± 0.052
Average	0.437	0.501 ± 0.025	0.779 ± 0.033	0.764 ± 0.062

Table 3.8.: Results on 15 benchmark time-series datasets: mean and standard deviation of accuracy from random guess (RG), previous printed neural network (*p*-NN), Elman recurrent neural network (RNN), and printed temporal processing neuroal network (*pTPNN*).

this work does not currently consider variations in printed circuits into the training objective, similar variation-aware training as proposed in [242] could be introduced in the future work to overcome the sensitivity of analog computing systems to the variations.

Another challenge of this process pertains to the values of the decoupling factor μ. To minimize the coupling effect, the resistances in the filters are designed with lower values (<1kΩ) than that of the resistors in crossbars (100kΩ-10MΩ), while the capacitances are designed as high as the printing technology allows (100nF-100µF). In this work, we determined the range of μ by analyzing of signal frequencies in the experimental datasets (see subsection 6.1.4) and conducting SPICE simulations with pPDK [129]. Notably, this range of μ has taken into account the cases in which the filters might be connected to multiple (up to five) crossbars, which will aggravate the circuit coupling by elevating the coupling current I^{couple}.

3.3.5. Evaluation

To evaluate the *pTPNN*s, we implemented the proposed approach[5] with PyTorch [149] and conduct experiments on 15 benchmark time-series datasets. As the functionality of the printed neuromorphic hardware has been experimentally validated in [144], [202], [278], the experiment is conducted at simulation level based on the pPDK [129].

[5] The code is available at `https://github.com/Neuromophic/LearnableFilters`.

3.3.6. Experiment

First, we sourced all datasets from the UCR time-series classification archive [119]. Afterwards, we filtered out datasets based on their complexity. Only datasets with N_{in} and N_{out} below 10 are kept to match the typical complexity of the target applications of PE. Subsequently, we preprocess the datasets by resizing the series lengths uniformly to 128, normalized the signal values to the range of $[-1, 1]$, reshuffle and split the datasets into training (60%), validation (20%), and test (20%) sets. Then, we leveraged a 2-layer RNNs as a baseline to remove datasets whose difficulty surpassed the capabilities of general RNNs. Ultimately, the top 15 datasets with optimal RNN performance are retained for the further experiment.

3.3.6.1. Training Setup

For each dataset, we adopt a 2-layer *pTPNN* (consisting of two *pTPB*s) with the number of learnable filters N_F equaling to N_{out}. To minimize the objective function, we use the gradient-based Adam optimizer with default parameterization to conduct full-batch training to update the optimization parameters. The initial learning rate is set to 0.1 and get halved after every 100-epoch patience (consecutive updates without improvement) on the validation loss. The training process stops once the learning rate has been reduced below 10^{-5}. The entire training procedure is replicated 10 times, employing different random seeds ranging from 0 to 9. This aims to mitigate the potential impact of unfavorable initialization, and thus to ensures a sufficiently good solution.

3.3.6.2. Baselines

For comparison, we consider 2-layer basic *p*-NNs without *pTPB* as a baseline. The topology mirrors that of the *pTPNN*s, i.e., N_{in}-N_F-N_{out} with $N_F = N_{out}$. This comparison intends to assess the temporal processing ability of both *p*-NNs and *pTPNN*. Since *p*-NNs are unable to process temporal sensory data, the classification results should form random guesses (RG), which refers to always predicting the most probable class in training data. Besides, we also compare the *pTPNN*s with the RNNs that we strived to mimic. Specifically, we adopt the Elman RNNs provided in PyTorch, and analogously, we utilize 2-layer RNNs with the number of hidden states (equivalent to the number of learnable filters N_F) being equal to N_{out}. After hyperparameter tuning, we initiate the learning rate for RNNs to 0.01, while all other training setups are kept identical to those of the *pTPNN*s.

3.3.7. Result

After training, we select the top three models (trained with three different random seeds) for each dataset according to the accuracy on the validation set. Note that, in accordance with the objective proposed in subsection 3.3.4, we have computed the classification accuracy at every time step, and subsequently average these accuracies over time to yield the overall classification accuracy on a dataset. These selected models are then evaluated on the test set. Ultimately, for each dataset, we summarize its mean accuracy with respect to random seeds and the corresponding standard deviation. The result is presented in Table 3.1. To obtain a straightforward insight on the effectiveness of each models in various scenarios (datasets), we also averaged the accuracy and standard deviation with respect to datasets. The averaged values are reported in the last row of Table 3.8.

Hardware Cost To investigate the additional hardware resources required by the new circuit design, we collect the device counts and total power consumption of both the previous *p*-NNs and the proposed *pTPNN*s in different application scenarios (i.e., datasets). Analogously, we averaged the hardware costs across all datasets to report a comprehensive comparison regarding the hardware costs between the *pTPNN* and its *p*-NN counterpart. The results can be seen in Table 3.9.

Dataset	#Transistor		#Resistor		#Capacitor		#Total Device		Power (mW)	
	p-NN	*pTPNN*	*p*-NN	*pTPNN*	*p*-NN	*pTPNN*	*p*-NN	*pTPNN*	*p*-NN	*pTPNN*
CBF	18	24	57	84	–	6	75	114	0.471	0.653
DPTW	36	48	150	222	–	12	186	282	1.069	1.501
FRT	12	16	34	50	–	4	46	70	0.272	0.372
FST	12	16	34	50	–	4	46	70	0.276	0.342
GPAS	12	16	34	50	–	4	46	70	0.221	0.374
GPMVF	12	16	34	50	–	4	46	70	0.302	0.389
GPOVY	12	16	34	50	–	4	46	70	0.289	0.324
MPOAG	18	24	57	84	–	6	75	114	0.454	0.625
MSRT	30	40	115	170	–	10	145	220	0.862	1.188
PowerCons	12	16	34	50	–	4	46	70	0.312	0.363
PPOC	12	16	34	50	–	4	46	70	0.226	0.381
SRSCP2	12	16	44	60	–	4	56	80	0.294	0.472
Slope	12	16	34	50	–	4	46	70	0.320	0.388
SmoothS	18	24	57	84	–	6	75	114	0.436	0.610
Symbols	36	48	150	222	–	12	186	282	1.143	1.526
Average	**18**	**23**	**60**	**88**	**–**	**6**	**78**	**118**	**0.463**	**0.634**

Table 3.9.: Hardware costs of *p*-NN and printed temporal processing neural network (*pTPNN*).

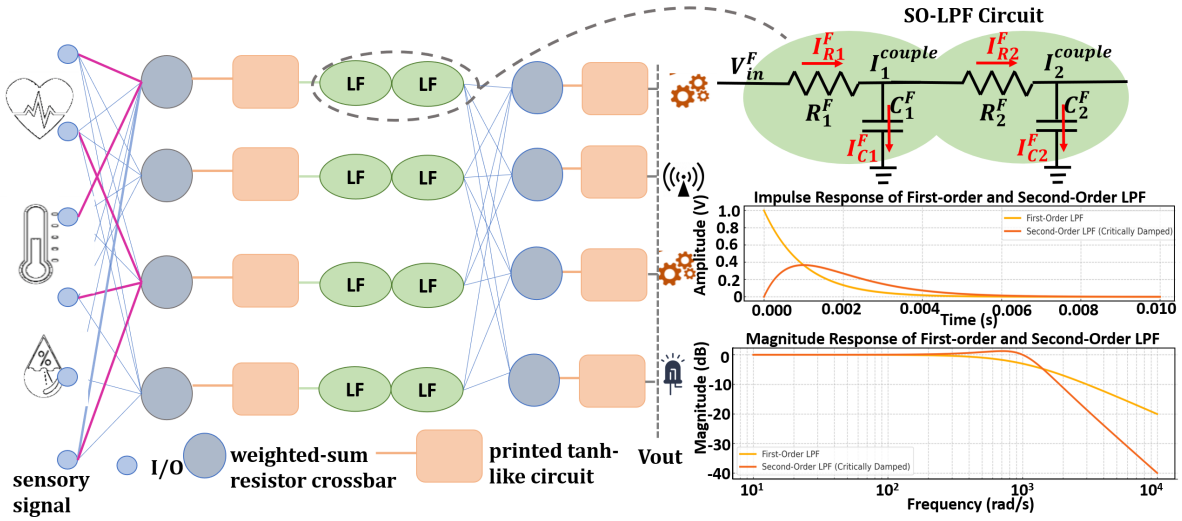


Figure 3.17.: Schematic of a 6-input 4-output pTPB including sensory signals from various inputs, processed through a weighted-sum resistor crossbar, second-order low-pass filters (LPF), and a printed tanh-like activation circuit. Signal flows are shown in both time and frequency domain of both printed filters.

Discussion As can be seen in Table 3.8, basic *p*-NNs without *pTPB* are unable to process temporal data, and thereby only achieve similar classification accuracy to that of the random guess. However, by comparison of averaged performance between *pTPNN*s and basic *p*-NNs, it reveals that *pTPNN*s are indeed capable of processing time-series data. Moreover, such capability for temporal signal processing requires only approximately 1.5× more devices and 1.3× more power consumption. By comparing the performance of *pTPNN*s with RNNs, we conclude that the *pTPNN*s can attain a comparable (98%) classification accuracy to their completely hardware-agnostic Elman RNN counterparts. Interestingly, a closer observation of Table 3.8 reveals that *pTPNN*s and RNNs do not consistently yield comparable performance on every dataset. Their accuracy differs significantly on datasets such as CBF, DPTW, PowerCons, and SmoothS. This may be due to the physical limitations of the circuits (and consequently their distinct computational models).

In summary, the temporal *pTPNN* work demonstrates that:

- Time-series tasks in PE can be tackled with bespoke temporal filters integrated directly into the analog NN, rather than resorting to external digital RNNs.
- Circuit-level coupling must be modeled explicitly; otherwise the interaction between filters and crossbars could lead to unstable or inaccurate behavior.
- However, the training objective does not yet include process variation or sensor noise; thus, *pTPNN*s can still be sensitive to the large device variations characteristic of PE.

These limitations motivate the robustness-aware ADAPT-*p*-NN framework.

3.4. ADAPT-*p*-NN: Robust Temporal Filters under Variability and Sensor Noise

p-NNs are designed to enhance silicon-based electronics, particularly in cost-effective edge computing applications like wearable devices or smart packaging. In some scenarios, such as stress detection [278], the absolute values of sensory signals may not provide significant insights due to individual variability. Instead, the temporal dynamics of these signals are more informative. Previous approaches have utilized first-order low-pass filters (LPFs) in *p*-NNs for basic input signal processing tasks. While the first-order LPFs can reduce high-frequency noise and capture signal trends, they are often inadequate for the dynamic nature of sensor signals, which require more complex analysis. This shortfall limits their utility in

accurately processing data for applications such as stress detection, where signals can vary widely between time and amplitude.

To overcome these limitations, we propose a learnable second-order filter (SO-LF) within p-NNs. Second-order filters offer a superior dynamic response compared to first-order filters, making them more adapted to handling the intricacies of temporal sensory data due to their sharper cutoff and better signal component separation, and are essential for capturing and analyzing both the temporal changes in input signals and also noisy input-data.

We achieve this by integrating two learnable printed resistors and capacitors connected back-to-back (as shown in Figure 3.17), thus enabling the circuits to retain and process temporal sensory information and emulating the characteristics of second-order RC low-pass filters. To design the bespoke signal processing behaviors for target tasks, we have developed the mathematical model for the proposed variation-aware second-order *pTPB* and the corresponding ADAPT-p-NN. This includes an optimization objective that allows the components within *pTPB*s (such as capacitance) to be fine-tuned along with the resistances in the crossbars, which act as weights and biases. This approach enables us to understand and optimize the performance of p-NNs and adapt to variations in input sensors and physical units. Additionally, in the proposed robustness-aware framework, we incorporate data augmentation techniques during both the training and testing phases.

3.4.0.1. Modeling of second-order filter

To model the second-order learnable filter (SO-LF), we consider it as two successive first-order learnable filters. Initially, we analyze the model of a first-order learnable filter without considering its coupling to the subsequent circuit. The relationship between the input and output voltage is derived and represented in Equation 3.28. The discrete-time update equations for $V^{F_1}_{\text{out},K}$ and $V^{F_2}_{\text{out},K}$ are shown in Equation 3.29 and Equation 3.30, respectively.

$$V^{F}_{\text{out},K} = \frac{R^F C^F}{R^F C^F + \Delta t} V^{F}_{\text{out},K} + \frac{\Delta t}{R^F C^F + \Delta t} V^{F}_{\text{in},K} \tag{3.28}$$

$$V^{F_1}_{\text{out},K} = \frac{R^F_1 C^F_1}{R^F_1 C^F_1 + \Delta t} V^{F_1}_{\text{out},K} + \frac{\Delta t}{R^F_1 C^F_1 + \Delta t} V^{F_1}_{\text{in},K} \tag{3.29}$$

$$V^{F_2}_{\text{out},K} = \frac{R^F_2 C^F_2}{R^F_2 C^F_2 + \Delta t} V^{F_2}_{\text{out},K} + \frac{\Delta t}{R^F_2 C^F_2 + \Delta t} V^{F_1}_{\text{out},K} \tag{3.30}$$

In these equations, Δt refers to the step size of the temporal discretization. The variables $V^{F_1}_{\text{in},k}$, $V^{F_1}_{\text{out},k}$, and $V^{F_2}_{\text{out},k}$ represent the input, intermediate, and output voltages of the filter at time step k, respectively. By appropriately selecting the resistor values R^F_1, R^F_2 and the capacitor values C^F_1, C^F_2, a desired filtering behavior can be achieved. Despite previous work, in our approach, the resistors and capacitors are trained separately. In this work, these values are learned along with the crossbar resistors to optimize performance for specific tasks.

3.4.0.2. Modeling of second-order coupled filter

To connect the learnable filters with the resistor crossbars, as shown in [260], it is imperative to consider their coupling effects. This coupling primarily arises because the current flowing through the resistor R^F_1 does not entirely feed into the capacitor C^F_1; similarly, the current through R^F_2 does not fully enter C^F_2. Instead, a portion of the current is shunted towards R^F_2 and the crossbar, respectively, as indicated by the red arrows (I^{couple}_1) and (I^{couple}_2) in Figure 3.17. Equation 3.31 and Equation 3.32 depict this decoupling, while Equation 3.33 and Equation 3.34 represent the decoupling factors.

Consequently, Equation 3.35 and Equation 3.36 are reformulated to account for these factors. Additionally, μ is influenced by the frequency of the input signal, which is typically unknown during the design stage. To incorporate circuit coupling into the training process, we determine the general range of μ

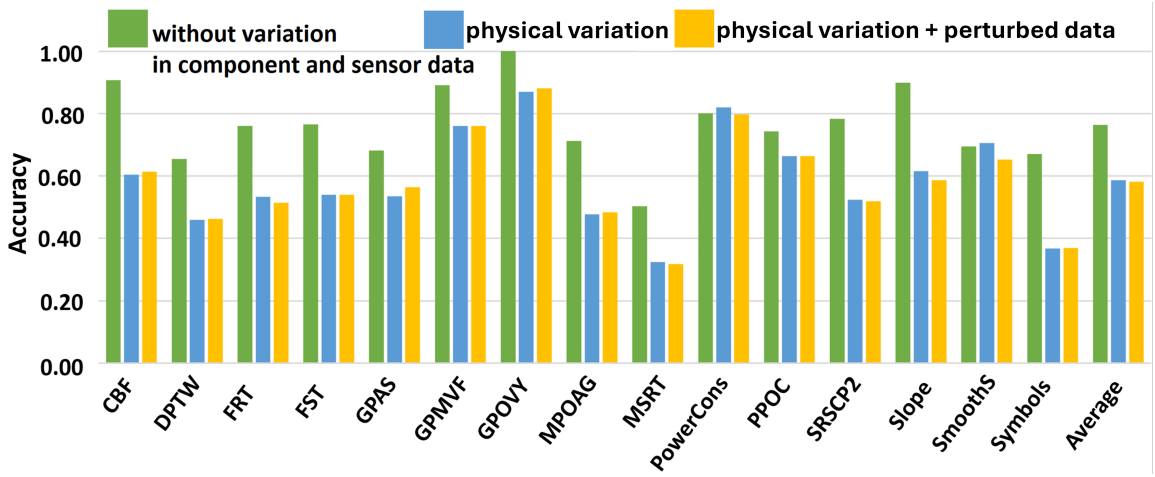


Figure 3.18.: Baseline tested under physical variations and perturbed sensor inputs.

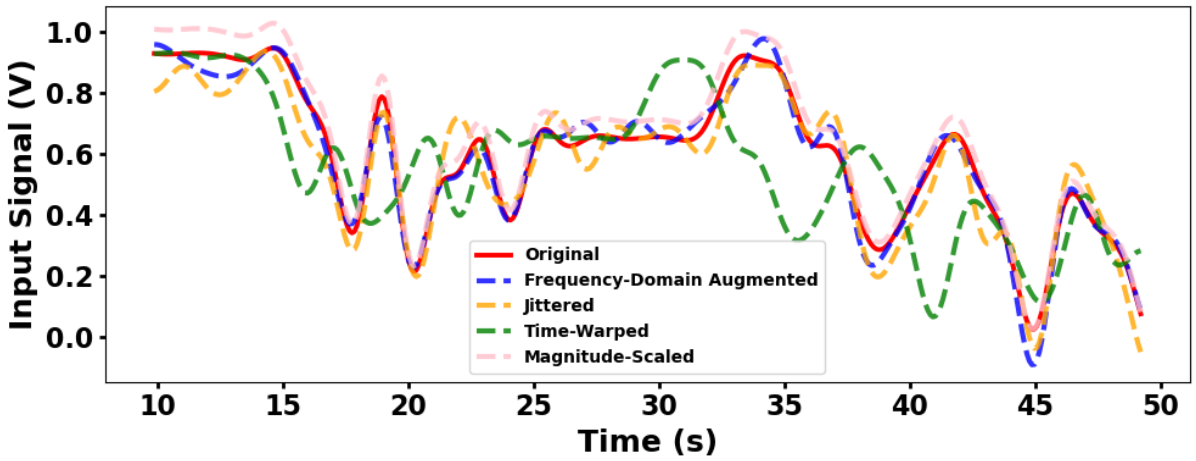


Figure 3.19.: Time-series augmentation techniques applied on PowerCons dataset: original, jittering, time-warping, magnitude scaling, and frequency-domain augmentation.

through SPICE simulations using the pPDK [130]. Specifically, by performing SPICE simulations on the target datasets with general printable resistances and capacitances, we empirically determined that $\mu \in [1, 1.3]$ for the given applications.

$$I_{R1}^{F} = I_{C1}^{F} + I_{1}^{\text{couple}} \tag{3.31}$$

$$I_{R2}^{F} = I_{C2}^{F} + I_{2}^{\text{couple}} \tag{3.32}$$

$$I_{R1}^{F} = \mu I_{C1}^{F} \tag{3.33}$$

$$I_{R2}^{F} = \mu I_{C2}^{F} \tag{3.34}$$

$$V_{\text{out},K}^{F_1} = \frac{R_1^F C_1^F}{\mu_1 R_1^F C_1^F + \Delta t} V_{\text{out},K}^{F_1} + \frac{\Delta t}{\mu_1 R_1^F C_1^F + \Delta t} V_{\text{in},K}^{F_1} \tag{3.35}$$

$$V_{\text{out},K}^{F_2} = \frac{R_2^F C_2^F}{\mu_2 R_2^F C_2^F + \Delta t} V_{\text{out},K}^{F_2} + \frac{\Delta t}{\mu_2 R_2^F C_2^F + \Delta t} V_{\text{out},K}^{F_2} \tag{3.36}$$

The SO-LF enhances the robustness of the model under variations. During training, we consider variations on both the components and the input data to further improve robustness.

3.4.1. Variation-Aware Training of proposed Second-Order Filter

In this framework, the parameters θ (corresponding to the printed resistances), R (resistors), and C (capacitors) of the SO-LFs are trainable and subject to process variations. These trainable parameters are modeled as random variables during training: $\theta \sim p(\theta)$, $C \sim p(C)$, and $R \sim p(R)$, with distributions representing the process variations. Furthermore, the decoupling factor (μ) and the initial voltage (V_0^F) are modeled as random variables but are not trainable.

To account for these variations, we use a reparameterization strategy: $\theta = \theta_0 \odot \epsilon_\theta$, $C = C_0 \odot \epsilon_C$, and $R = R_0 \odot \epsilon_R$, where ϵ_θ, ϵ_C, and ϵ_R are random variables that follow distributions $p(\epsilon)$. This allows us to model process variations while still optimizing the trainable parameters θ, R, and C.

The training objective is to minimize the expected loss:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{p(\theta),p(C),p(R),p(\mu),p(V_0^F)}\left[\mathcal{L}(\theta, C, R, \mathbf{x}_K, \cdots, x_0, \mathbf{y}, \mu, V_0^F)\right] \\ &= \int \mathcal{L}(\theta, C, R, \mathbf{x}_K, \cdots, x_0, \mathbf{y}, \mu, V_0^F)\, d\theta\, dC\, dR\, d\mu\, dV_0^F\end{aligned} \tag{3.37}$$

which is approximated via Monte Carlo sampling:

$$\mathcal{L} \approx \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\theta_i, C_i, R_i, \mathbf{x}_K, \cdots, x_0, \mathbf{y}, \mu_i, V_{0,i}^F) \tag{3.38}$$

This leads to the final training objective in Equation 3.39, which ensures optimal accuracy and robustness against variations.

$$\min_{\theta,C,R} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\theta_i, C_i, R_i, \mathbf{x}_K, \cdots, x_0, \mathbf{y}, \mu_i, V_{0,i}^F) \tag{3.39}$$

Table 3.10.: Result on 15 benchmark time-series datasets: (a) hardware-agnostic Elman recurrent neural network (RNN), (b) baseline printed temporal processing neural network (*pTPNN*), (c) robustness-aware ADAPT *p*-NN under precision printing (±10% variation) and perturbed input data.

Dataset	**Elman RNN (Reference Acc.)**	***pTPNN* (Baseline)**	**ADAPT-*p*-NN (Robustness-Aware)**
CBF	0.683 ± 0.036	0.615 ± 0.142	0.877 ± 0.006
DPTW	0.507 ± 0.006	0.462 ± 0.003	0.700 ± 0.008
FRT	0.597 ± 0.120	0.514 ± 0.003	0.677 ± 0.021
FST	0.509 ± 0.066	0.540 ± 0.000	0.591 ± 0.086
GPAS	0.452 ± 0.003	0.564 ± 0.012	0.568 ± 0.004
GPMVF	0.637 ± 0.054	0.760 ± 0.010	0.900 ± 0.010
GPOVY	0.540 ± 0.007	0.881 ± 0.030	1.000 ± 0.000
MPOAG	0.560 ± 0.042	0.483 ± 0.006	0.654 ± 0.021
MSRT	0.261 ± 0.008	0.317 ± 0.022	0.531 ± 0.004
PowerCons	0.651 ± 0.010	0.797 ± 0.865	0.880 ± 0.018
PPOC	0.711 ± 0.001	0.664 ± 0.005	0.660 ± 0.000
SRSCP2	0.489 ± 0.011	0.519 ± 0.010	0.525 ± 0.001
Slope	0.559 ± 0.002	0.587 ± 0.041	0.765 ± 0.016
SmoothS	0.447 ± 0.011	0.653 ± 0.015	0.864 ± 0.011
Symbols	0.141 ± 0.002	0.369 ± 0.050	0.697 ± 0.003
Average	**0.501 ± 0.025**	**0.582 ± 0.031**	**0.726 ± 0.014**

Table 3.11.: Comparison of Runtime (Average)

Models	**Elman RNN (Reference acc.)**	***pTPNN* (Baseline)**	**Robustness-Aware ADAPT-*p*-NN**
Runtime (avg.)	2.345 ms	0.230 s	2.537 s

3.4.2. Variation in Input Sensor

To improve robustness, we employ several data augmentation techniques during training, enhancing dataset variability and enabling the model to generalize to unseen conditions. The augmentations used for time-series data include frequency-domain noise to simulate signal distortions (applied to datasets like PowerCons and SmoothS), random cropping to mimic partial data availability (effective for datasets such as MSRT and Symbols), jittering to introduce sensor inaccuracies, time warping to alter the temporal dynamics, and magnitude scaling to simulate changes in sensor readings. These techniques collectively simulate real-world sensor variations, improving model robustness by creating a more diverse and resilient training set. Figure 3.19 shows the application of the combined data augmentation techniques on the PowerCons dataset.

3.4.3. Evaluation

The temporal processing block primitives in Figure 3.17 (top (c)-(g)) were designed using the well-established n-EGT pPDK [152]. We obtained the filter magnitude, impulse response and the cutoff frequencies from the SPICE simulations in Cadence Virtuoso[6]. The resistances in the filters are designed with lower values (<1kΩ) than that of the resistors in crossbars (100kΩ-10MΩ), while the capacitances are designed as high as the printing technology allows (100nF-100µF) to minimize the coupling effect.

3.4.4. Experiment

We have considered 15 datasets from the UCR Time Series Classification Archive [73]. The datasets were preprocessed by uniformly resizing the series lengths to 64, normalizing the signal values to the range of $[-1, 1]$, and reshuffling and splitting the datasets into training (60%), validation (20%), and test (20%) sets. We used a 2-layer RNN as a reference accuracy model and *pTPNN* as baseline.

For data augmentation, we employed the `tsaug` framework [186] to apply the augmentation techniques. The augmented data was combined with the original unaugmented data, and both were used during training, validation and testing.

Training Setup For each dataset, we utilize a 2-layer robustness-aware *pTPNN*, which includes two second-order *pTPB* layers per layer, with the number of learnable filters N_F matching the number of inputs for that layer. To optimize the objective function, we employ the AdamW optimizer[107] with default settings for full-batch training to adjust the optimization parameters. The initial learning rate is set at 0.1 and is halved after every 100 epochs of no improvement in the validation loss (patience). Training is terminated once the learning rate falls below 10^{-5}. This training process is repeated 10 times, each with a different random seed ranging from 0 to 9, to reduce the risk of poor initialization and ensure a robust solution.

For experiments involving data augmentation, we used Ray Tune [125] to tune hyperparameters such as crop size, noise level, and time warping. The optimal configurations for each dataset were applied during training. All implementations and training were done in PyTorch [149].

[6] `https://www.cadence.com/en_US/home.html`

Table 3.12.: Hardware Costs for baseline *pTPNN*[260] vs proposed Robustness-Aware ADAPT-*p*-NN

Dataset	**#Transistors**		**#Resistors**		**#Capacitors**		**#Total Devices**		**Power (mW)**	
	pTPNN	**Proposed**	***pTPNN***	**Proposed**	***pTPNN***	**Proposed**	***pTPNN***	**Proposed**	***pTPNN***	**Proposed**
CBF	24	59	84	147	6	24	114	230	0.653	0.06
DPTW	48	126	222	337	12	24	282	487	1.501	0.06
FRT	16	29	50	71	4	12	106	112	0.372	0.03
FST	16	30	50	75	4	12	70	117	0.342	0.03
GPAS	16	36	50	86	4	12	70	134	0.374	0.03
GPMVF	16	38	50	86	4	12	70	136	0.389	0.03
GPOVY	16	37	50	82	4	12	70	131	0.324	0.03
MPOAG	24	61	84	146	6	24	114	231	0.625	0.06
MSRT	44	127	170	335	6	60	210	522	1.188	0.15
PowerCons	16	34	50	79	4	12	70	125	0.363	0.03
PPOC	16	23	50	67	4	12	70	102	0.381	0.03
SRSCP2	16	32	60	89	4	12	70	133	0.472	0.03
Slope	24	27	50	72	6	12	114	111	0.388	0.03
SmoothS	16	59	84	148	4	24	70	231	0.610	0.06
Symbols	48	136	222	391	12	84	282	611	1.526	0.210
Average	**23**	**57**	**88**	**147**	**6**	**23**	**118**	**228**	**0.634**	**0.058**

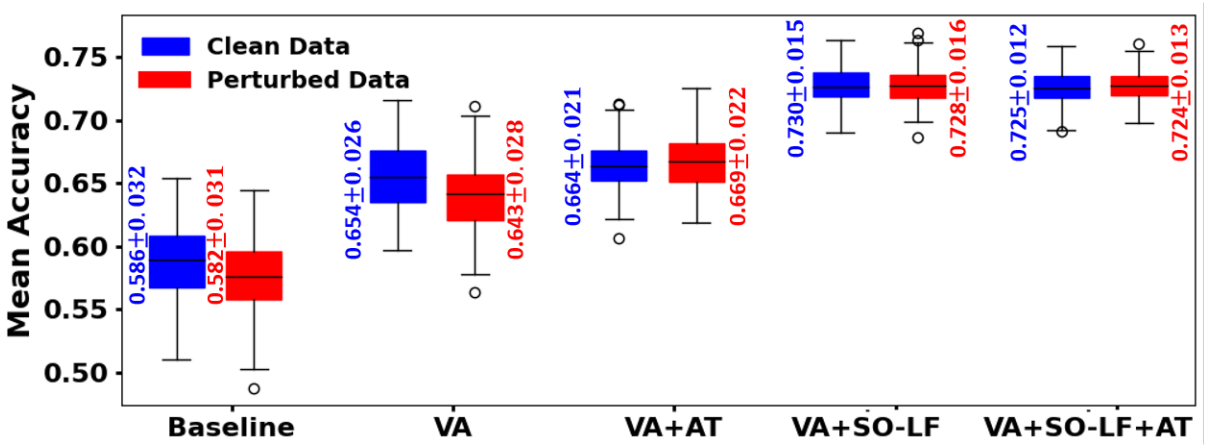


Figure 3.20.: Average accuracy comparison of variation aware (VA), augmented training (AT), and second-order learnable filters (SO-LF) and combined (VA +SO-LF+AT) with baseline using an ablation study.

3.4.5. Result

To evaluate the accuracy of our proposed robustness-aware ADAPT-*p*-NNs[7], we conducted a series of experiments and compared the results against a baseline model and a reference model. The baseline is a 2-layer no-variation-aware *pTPNN*, while the reference model is a 2-layer Elman RNN, as implemented in PyTorch[149].

After training, we selected the top three models for each dataset based on their accuracy on the test set. The selected models were evaluated on an augmented test set with a 10% variation in physical components. For each dataset, we report the mean accuracy across random seeds and the corresponding standard deviation, as shown in Table 3.10. The averaged accuracy and standard deviation across all datasets are provided in the last row. Additionally, the average runtimes for the proposed ADAPT-*p*-NN, baseline, and Elman are reported in Table 3.11. Table 3.12 presents the hardware component and power metrics for the proposed model. Furthermore, we conducted an ablation study to evaluate the effectiveness of different configurations under a 10% physical variation scenario: variation-aware (VA) during training, augmented training (AT), and second-order learnable filters (SO-LF).

Discussion The impact of manufacturing variability on PE cannot be overstated. Variations such as ink dispersion, droplet irregularities, and missing droplets [99], [181] introduce significant challenges. These variations are often modeled using uniform distributions and Gaussian Mixture Models at the device level [129]. As shown in Figure 3.18, a trained no-variation-aware *pTPNN*, when tested under these variations, exhibits a significant drop in accuracy. This clearly illustrates the necessity for a robustness-aware framework, which motivates our exploration in this direction.

The results in Table 3.10 highlight the accuracy of three models: the Elman RNN (reference model), *pTPNN* (baseline model), and the robustness-aware ADAPT-*p*-NN across 15 datasets. The Elman RNN and the baseline *pTPNN* achieve an average accuracy of 0.501 and 0.582, respectively, while the robustness-aware ADAPT-*p*-NN achieves the highest accuracy of 0.726, leading to $\approx$ 24.7% improvement from the baseline, showing the effectiveness of incorporating robustness-aware techniques. Furthermore, Table 3.11 also shows that while the proposed model requires more computational resources (2.537*s* on average, compared to 0.230*s* for the baseline), significant accuracy gains justify this increase. The trade-off between computational efficiency and accuracy is important for target application tasks[75], [219] where robustness to data variability is critical. Moreover, datasets CBF and GPOVY further highlight these improvements. In CBF dataset, Elman RNN shows an accuracy of 0.683, while the baseline *pTPNN* is reduced to 0.615. However, the robustness-aware ADAPT-*p*-NN improves to 0.877, reflecting the model's

[7] Code is available at `https://github.com/KIT-Neuromorphic-Computing/ADAPT-pNC`.

ability to handle complex variations. Similarly, for GPOVY, Elman RNN and baseline *pTPNN* achieve 0.540 and 0.881, respectively, but the proposed model reaches an impressive accuracy of 1.000. This demonstrates the model's capability to generalize effectively in challenging datasets.

Figure 3.20 demonstrates the improvement in mean accuracy across 15 benchmark datasets for different training configurations. In comparison with the baseline, which achieves ≈ 58% accuracy in both clean and perturbed data, adding variation-aware (VA) training improves accuracy by 11.6% for clean data and 10.5% for perturbed data. Augmented training (AT) further increases accuracy by 13.3% for clean data and 15% for perturbed data, accompanied by a decrease in standard deviation. The inclusion of SO-LFs results in a significant improvement of 24.6% for clean data and 25.1% for perturbed data, while also reducing both standard deviation and variability.

To investigate the additional hardware resources required by the new circuit design, we collect the device counts and total power consumption of both existing *pTPNN*[260] and the proposed ADAPT-*p*-NNs in different application scenarios (i.e., datasets). Analogously, we averaged the hardware costs across all datasets to report a comprehensive comparison regarding the hardware costs between the *pTPNN* and its *p*-NN counterpart. The results in Table 3.12 shows a significant power reduction (≈ 91%) at the expense of additional hardware (≈ 1.9×) compared to existing works[260].

In summary, our experiments show that the accuracy improvement from the Elman RNN to the *pTPNN* (baseline) is approximately 15%. For the robustness-aware ADAPT-*p*-NN, this improvement increases to approximately 45%. Additionally, ADAPT-*p*-NN requires approximately 1.9× more devices compared to the *pTPNN* (baseline). Also, the robustness-aware ADAPT-*p*-NN, particularly with the variation-aware (VA) + second-order learnable filter (SO-LF) + augmented training (AT) configuration, provides accuracy improvement and robustness to perturbations. Although it requires more computational time, the model's ability to handle real-world data variability makes it an ideal solution for target tasks that require high accuracy and reliability.

3.5. On-Sensor Intelligence via Bespoke ADC and Decision Tree Co-Design

The on-sensor co-design work targets a different, but complementary, problem: realizing digital printed ML classifiers that can operate entirely from printed energy harvesters, including the cost of analog-to-digital conversion. Printed decision-tree classifiers are chosen due to their simplicity and their suitability for modestly complex classification tasks in printed applications.

The methodology has three main elements:

- **Parallel unary decision trees:** Instead of conventional binary-encoded decision trees, the authors use a fully parallel unary representation of input features derived from flash ADC thermometer codes. Each tree node compares a single unary bit to a threshold, and tree outputs are implemented as two-level AND–OR logic. This drastically simplifies the digital logic.

- **Bespoke ADCs:** A conventional N-bit flash ADC uses $2^N - 1$ comparators and outputs a thermometer code that is then encoded to binary. Here, the encoder is removed, and only the unary digits required by the decision tree are generated. A bespoke ADC thus retains only those comparators corresponding to the needed thresholds ("output unary digits"), eliminating all others and the encoder.

- **ADC-aware tree training:** Since ADC cost depends on which unary digits are used, tree training is modified to prefer splits that minimize the number of needed unary outputs and favor lower-order comparators, which are cheaper in terms of power. This is achieved by augmenting standard Gini-based training with hardware-aware heuristics and a parameter τ that trades small accuracy losses for larger hardware savings.

The entire flow is implemented using a 4-bit flash ADC design in a printed n-EGT pPDK and synthesized decision-tree logic, evaluated on eight classification datasets representative of printed applications.

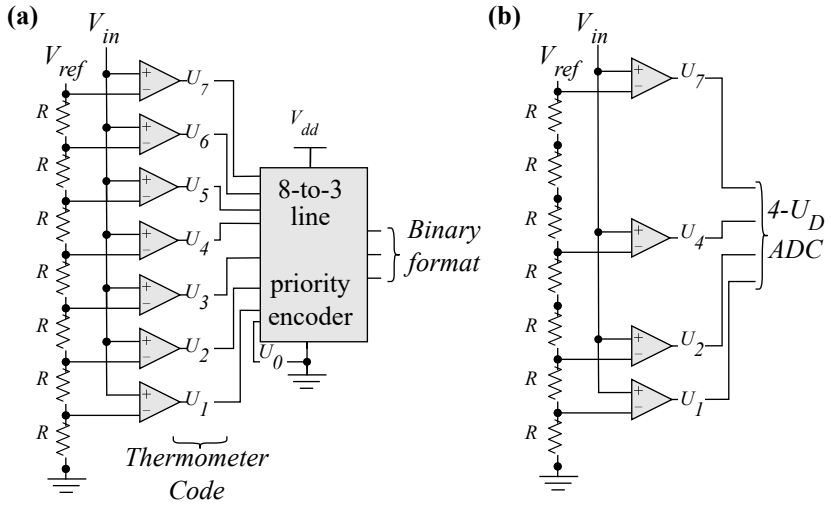


Figure 3.21.: Schematic of: a) conventional 3-bit Flash ADC and b) an example of an equivalent bespoke ADC with four unary digits of output.

3.5.1. Co-design Framework

This section describes our co-design framework for generating printed Decision Trees. In brief, we first introduce the architecture of a fully-parallel Decision Tree based on the unary representation and highlight its area and power benefits over non-unary approaches. Then, we analyze our flash ADC design tailored for such architectures, and we describe our ADC-aware Decision Tree training that enables minimizing the ADCs hardware cost while maintaining high accuracy.

3.5.2. Parallel Unary Decision Trees

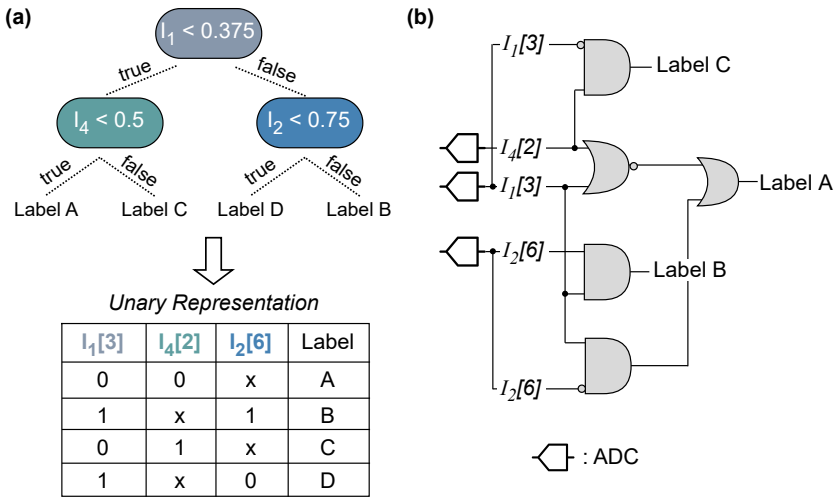
In unary coding (or else thermometer code), an N-bit binary number is represented by a code of length 2^N-1. The count of '1's in the unary code corresponds to the value being represented. Unary coding can express many types of numbers, such as integers, fixed-point, etc.

$$\begin{aligned} 0011111_{\mathrm{U}} &= 101_2 = 5_{10}, \\ 0.111_{\mathrm{U}} &= 0.11_2 = 0.75_{10} \end{aligned} \tag{3.40}$$

A parallel unary format is generally not preferred due to its increased size to represent a number (N vs. 2^N-1 bits) [229]. However, as we show hereafter, this does not hold in our case and thus we investigate and propose the implementation of parallel unary printed Decision Trees. As aforementioned, the unmatched customization in printed circuits enables hardcoding the model's parameters. As a result, a comparison $I \geq C$, where I is an input and C is a model parameter, becomes $I \geq 0.1011_2$ assuming that 0.1011_2 is the trained value of C. In unary format C can be written as $0.000111111111111_{\mathrm{U}}$. Therefore, although parallel unary representation does not typically make sense in conventional architectures due to the exponentially increased number of bits to be compared, in bespoke Decision Trees the comparator is essentially reduced to simply checking a bit from the input:

$$I \geq 0.1011_2 \xrightarrow{\text{Unary}} I \geq 0.000111111111111_{\mathrm{U}} \equiv I[11] \tag{3.41}$$

In other words, if the 11^{th} bit of I is 1 then the initial inequality $I \geq C$ is true. In general, if the most significant 1 of C is at bit position k, then $I \geq C \equiv I[k]$. In addition, the inequality is directly computed (since I is in parallel format), eliminating the need to wait for bit k to arrive from a serial input. Similar relations are derived for all comparisons: $I > C \equiv I[k+1]$, $I < C \equiv \neg I[k]$, and $I \leq C \equiv \neg I[k+1]$. Note that in unary format, if $I[k] = 1$, then $I[j] = 1$ for all $j \leq k$.



I_1[3]	I_4[2]	I_2[6]	Label
0	0	x	A
1	x	1	B
0	1	x	C
1	x	0	D

Figure 3.22.: Illustration of how a conventional Decision Tree (a) is translated into unary format, represented by a set of unary digits. This example assumes Q0.4 formatted values, e.g., $0.75 \rightarrow 6$. (b) depicts the simplified schematic.

The analysis above highlights that in a bespoke Decision Tree, if the inputs are provided in a parallel unary format, all Tree comparators can be removed. This is perfectly in line with the limited hardware resources of printed circuits. It is crucial to note that a serial (e.g., temporal) unary representation would not only enforce a printed-unfriendly multi-cycle operation but would also introduce a significant hardware overhead such as control circuitry and potentially numerous registers.

Figure 3.22 presents an example architecture of a Decision Tree when its inputs are available in a parallel unary representation. Instead of traditional comparisons, bespoke unary DTs can now be viewed as simple logic over a set of unary digits corresponding to the trained parameters (e.g., $I_1[3]$, $I_4[2]$, $I_2[6]$). The truth table for each label of the unary Decision Tree example is also depicted. As shown, only a few gates are sufficient. Each label is obtained through a simple two-level logic (e.g., `AND–OR`), and each input signal in this two-level logic denotes a node in the Decision Tree.

3.5.3. Bespoke ADCs

As shown in the previous section, ensuring the availability of inputs in a parallel unary representation minimizes the hardware overheads of printed Decision Tree classifiers. As shown in Figure 3.21a, the flash ADC calculates an intermediate result, which is the thermometer code of the input signal (each U_i denotes if V_{in} is larger than V_{ref}). As a result, by simply removing the encoder, we not only decrease the ADC's hardware requirements but also achieve our objective: transforming the sensor input into a parallel unary representation before supplying it to the Decision Tree classifier. This also further justifies our flash ADC consideration.

To further improve the hardware-efficiency of our ADCs, we also implement them in a fully customized manner. The simplified Decision Tree design described above (e.g., Figure 3.22) relies on a parallel unary representation of its inputs. However, as indicated by (3.41), each comparison requires only a specific input bit. Consequently, the remaining unary digits, if not needed for another comparison, can be discarded and do not have to be generated. An illustrative example of a bespoke ADC is presented in Figure 3.21b. In this example, we assume that the respective input is involved in multiple comparisons and is compared against four different parameters. Hence, only four unary digits need to be calculated. Without any loss of generality, this example assumes that the 1^{st}, 2^{nd}, 4^{th}, and 7^{th} unary digits are required for the comparisons. Figure 3.21b presents the architecture of the corresponding "4-U_D" (four output unary digits) ADC. To generate the specific ADC, we only need to retain the corresponding four comparators, and we can eliminate the remaining three comparators and the encoder. In general, bespoke ADC design entails retaining only the resistors and the bare-minimum comparators required.

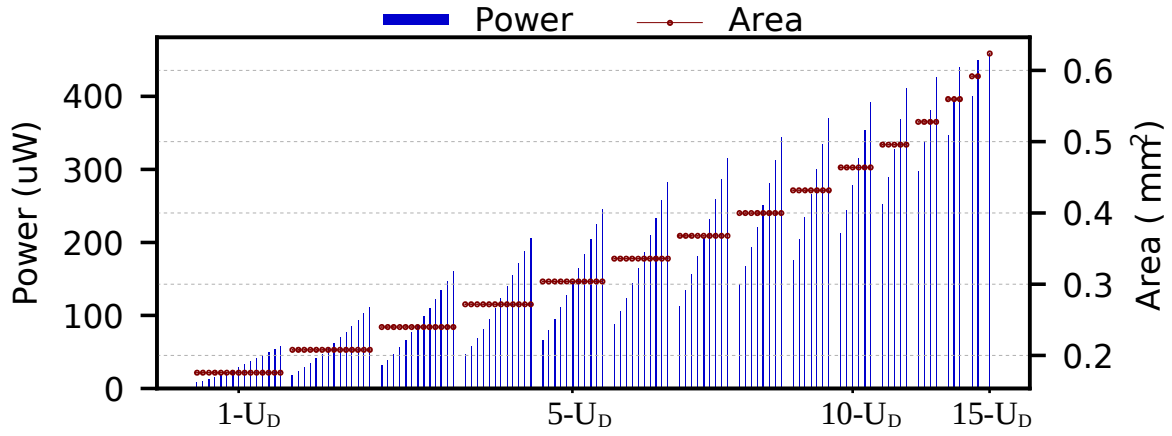


Figure 3.23.: The Area and power of (4-bit) bespoke ADCs w.r.t. their output unary digits. The first values correspond to 1-output ADCs, while the last one to 15-output ADC. Different points denote different output digits. Selected output digits are in sequential order, i.e., "U_1-U_2" 2-U_D ADC is followed by "U_2-U_3" 2-U_D ADC and so on, only to showcase the behavior of power.

The number of comparators, as well as the specific ones to be retained, is determined by the trained Decision Tree parameters. Considering a conventional 4-bit ADC, Figure 3.23 presents how the area and power characteristics of our bespoke ADCs scale w.r.t. the number and position of their output unary digits. In Figure 3.23, the number of output digits ranges from 1 to 15. For example, a 2-U_D ADC denotes a bespoke ADC with only two outputs, while a 15-U_D ADC indicates the retention of all comparators. To showcase the power behavior, a few representative examples are presented w.r.t. the specific outputs that are retained. The area and power of the conventional 4-bit ADC are 11mm^2 and 0.83mW, respectively. To obtain the area and power values, we designed the ADCs in Cadence Virtuoso using the n-EGT pPDK [158] and conducted SPICE simulations. The simulations are conducted with a voltage supply of 1V. As shown in Figure 3.23, the area of a bespoke ADC solely depends on the number of output unary digits of the ADC. Specifically, the area scales linearly along with the number of the comparators. On the other hand, power consumption also depends on which outputs are selected. For example, in a 4-U_D bespoke ADC, power consumption ranges from 47uW up to 205uW, i.e., 4.4× increase. It is observed that the power is substantially decreased when lower-order outputs are selected. This can be attributed to the lower Vref of the lower-order comparators. Figure 3.23 shows a linear increase in the comparators power consumption as we move towards higher-order outputs. The key takeaways from this analysis are twofold. First, bespoke ADCs provide significant hardware improvements over conventional ADCs. Second, the area and power consumption of a bespoke ADC are highly influenced by its outputs. By minimizing the number of outputs for each ADC, we can minimize its area, and by carefully selecting the specific outputs, we can further boost its power efficiency.

3.5.4. ADC-aware Decision Tree Training

As illustrated in Figure 3.22b, assuming parallel unary inputs, the logic of the Decision Tree is inherently simplified. Hence, since the hardware-efficiency of this two-level logic is mostly determined by the Decision Tree depth hyperparameter, our training methodology focuses on optimizing the cost of the ADCs. As demonstrated in subsection 3.5.3, the hardware efficiency of the ADCs, and consequently of the entire classifier, is determined by the trained parameters of the Decision Tree (i.e., output unary digits of each ADC). Hence, carefully selecting the Decision Tree parameters in an ADC-aware manner, while still achieving high classification accuracy, is essential for enabling power-autonomous printed Decision Trees.

To achieve this, we propose and implement an ADC-aware Decision Tree training approach. Our primary objective is to minimize the number of comparators induced by the ADCs. This is accomplished by minimizing the number of unique inputs involved among the total comparisons (i.e., minimizing the

Algorithm 2 ADC-aware Decision Tree Training Pseudocode

Input: 1) Dataset, 2) Tree Depth, 3) Gini Threshold τ
Output: Trained Decision Tree and Classification accuracy

$\mathcal{DT} = \emptyset$ *#selected split nodes*
for $0 \leq$ node $<$ Total nodes **do**
  for $\forall I_i \in$ Input Features **and** $\forall C$ value in dataset for I_i **do**
    calculate Gini(I_i, C)
  G = minimum Gini score
  $\mathcal{S} = \{(I_i, C) \mid \text{Gini}(I_i, C) \leq G + \tau, \forall (I_i, C)\}$
  $\mathcal{S}_\mathcal{Z} = \{(I_i, C) \in \mathcal{S} \mid (I_i, C) \in \mathcal{DT}\}$
  $\mathcal{S}_\mathcal{M} = \{(I_i, C) \in \mathcal{S} \mid \exists (I_i, C') \in \mathcal{DT}, C \neq C'\}$
  $\mathcal{S}_\mathcal{H} = \{(I_i, C) \in \mathcal{S} \mid \exists! (I_i, C') \in \mathcal{DT}, \forall C'\}$
  if $\mathcal{S}_\mathcal{Z} \neq \emptyset$ **then**
    g_m = calculate minimum Gini score $\forall (I_i, C) \in \mathcal{S}_\mathcal{Z}$
    $split$ = random$(\{(I_i, C) \in \mathcal{S}_\mathcal{Z} \mid \text{Gini}(I_i, C) = g_m\})$
  else
    if $\mathcal{S}_\mathcal{M} \neq \emptyset$ **then** $\mathcal{Z} = \mathcal{S}_\mathcal{M}$ **else** $\mathcal{Z} = \mathcal{S}_\mathcal{H}$
    $c_m = \min(\{C \mid \forall (I_i, C) \in \mathcal{Z}\})$
    $\mathcal{U} = \{(I_i, C) \in \mathcal{Z} \mid C = c_m\}$
    g_m = calculate minimum Gini score $\forall (I_i, C) \in \mathcal{U}$
    $split$ = random$(\{(I_i, C) \in \mathcal{U} \mid \text{Gini}(I_i, C) = g_m\})$
  $\mathcal{DT} = \mathcal{DT} \cup \{split\}$

number of ADCs) and, for each remaining input, by minimizing the number of different parameters, it is compared against, in all the comparisons involved in. Our secondary objective is to select more power-efficient values/parameters for each comparison (i.e., optimize the order of the output digits in the ADC). Since feasibility is the foremost requirement for printed ML circuits, prioritizing it over strict accuracy constraints is a typical procedure [224]. Consequently, we also explore the trade-off between some accuracy degradation and the potential for additional hardware gains.

Our ADC-aware training essentially trains a Decision Tree using the *Gini index* [123] cost function.

Gini is used to evaluate a split in the dataset. The latter involves one input feature (e.g., I_i in Figure 3.22a) and a trainable parameter C that will be compared with (i.e., output unary digit of the ADC of I_i).

We modify typical Gini-based Decision Tree training to incorporate ADC-awareness as follows. Initially, our algorithm seeks a split node and evaluates the Gini score for all possible combinations between input features and their corresponding values in the training dataset. At this point, ADC-unaware training would randomly select one combination among those with the best (minimum) Gini score. Assuming G is the best Gini score computed, we form a set of candidate split pairs $\mathcal{S} = \{(I_i, C) | \text{Gini}(I_i, C) \leq G + \tau\}$, with τ being a training hyperparameter. Next, we group the pairs $(I_i, C) \in \mathcal{S}$ into three sets based on the hardware induced from their selection.

1. $\mathcal{S}_Z$ (zero-cost): if (I_i, C) has been previously selected at a split node, it won't require additional hardware, only additional wiring.

2. $\mathcal{S}_M$ (medium-cost): if (I_i, C') with $C' \neq C$ has been previously selected at a split node, then the same ADC for I_i is reused, but a new comparator is added to that ADC because a different output digit is required. Since the area of our bespoke ADCs is linear with the number of comparators, all these pairs induce the same overhead, as each of them will add one comparator to one ADC.

3. $\mathcal{S}_H$ (high-cost): if no pair (I_i, C') has been previously selected at a split node, i.e., I_i is selected for the first time. These pairs induce the highest (and same) area overhead because a new ADC with one comparator is required.

Among these sets, we select the first non-empty one based on the order listed above. If $\mathcal{S}_M$ or $\mathcal{S}_H$ is chosen, we then identify the (I_i, C) pair in that set that results in the lowest power overhead. This can be accomplished by selecting the (I_i, C) pair with the minimum C value since it will necessitate the lowest-order output digit and, consequently, the induced comparator will have the lowest power consumption (see subsection 3.5.3). If multiple pairs feature the same minimum C value, or if $\mathcal{S}_Z$ is

Table 3.13.: Evaluation of the baseline bespoke Decision Trees.

Dataset	Acc (%)	#Comp.	#Inputs	Area (mm^2)		Power (mW)	
				ADCs	Total	ADCs	Total
Whitewine	52.8	207	11	17.3	261.3	5.4	14.6
Cardio	90.6	85	19	22.3	114.4	9.1	12.5
Arrhythmia	62.7	39	21	23.5	79.9	10.0	12.0
Balance-Scale	77.7	15	4	12.9	30.6	2.2	2.9
Vertebral-3C	86.0	7	5	13.6	16.8	2.5	2.8
Seeds	90.5	23	5	13.6	27.3	2.5	3.2
Vertebral-2C	87.1	7	5	13.6	16.4	2.5	2.8
Pendigits	95.0	215	16	20.4	268.7	7.7	17.2

chosen, we select the pair with the best Gini score, or random one if the Gini scores are equal. Our ADC-aware training assumes user-defined fixed depth and τ hyperparameters. Our algorithm identifies the most hardware-efficient split at each node. While our approach is inherently greedy, it introduces hardware-awareness into the traditionally employed, also greedy, Gini-based training. Still, alternative heuristic approaches could also be used. Algorithm 2 provides an abstract overview of our proposed training methodology.

The hyperparameter τ (if higher than 0) may lead to some accuracy degradation, as a pair with a Gini score higher than the best score might be selected. However, τ increases the size of the set $\mathcal{S}$, thereby increasing the chances of finding a more hardware-efficient (I_i, C). $\tau = 0$ will not affect the accuracy.

3.5.4.1. Results

In this section, we evaluate our printed Decision Tree classifiers, first examining hardware gains from our bespoke ADC design and ADC-aware Decision Tree training. Evaluation is based on 8 datasets listed in Table 3.13. These datasets are selected for two primary reasons: i) to facilitate direct comparisons with the state-of-the-art [170], [208], and ii) because these datasets utilize sensor inputs suitable for printed applications [170], [203]. The datasets are obtained from the the UCI ML repository[277]. Normalized inputs in the range $[0, 1]$ are used for training/testing with a random 70%/30% split. Synopsys Design Compiler and PrimeTime analyze the digital part of circuits, all operated at 20Hz, a common frequency aligned with typical performance of target PE applications [158], [224].

As our evaluation baseline, we consider the fully parallel bespoke Decision Trees designed as described in [170]. The minimum tree depth (up to 8) that achieves the maximum accuracy is used for each model and the input precision is set to 4 bits, since this is the value delivered close to floating-point accuracy for all datasets. Table 3.13 summarizes the accuracy and hardware overheads of the baseline Decision Trees [170]. Specifically, Table 3.13 reports the total area and total power requirements for each Decision Tree classifier as well as the respective values for the ADCs. As shown, the average total area and power consumption are $102mm^2$ and 8.5mW, respectively. Notably, all the classifiers exhibit power demands that exceed the capabilities ($>$ 2mW) of printed energy harvesters [182]. Consequently, none of these circuits can be self-powered. Finally, it's worth noting that, on average, for the Decision Trees in Table 3.13, 40% of the total area and 74% of the total power consumption is attributed to the ADCs.

Figure 3.24 depicts the area and power gains achieved by solely considering our bespoke ADCs along with the parallel unary Decision Tree design, i.e., the same ADC-unaware trained model used in [170]. In Figure 3.24, values are reported w.r.t. the baseline [170] (see Table 3.13). As shown, given that the overall area and power consumption of the printed Decision trees are predominantly governed by the ADCs, employing our bespoke ADC design delivers substantial hardware gains. Furthermore, as explained in subsection 3.5.1, using our ADCs streamlines the classifier's implementation, resulting in a simplified two-stage logic and additional hardware savings over the baseline [170]. Specifically, compared to [170], the achieved area and power reduction are 3.0x and 6.6x on average, respectively.

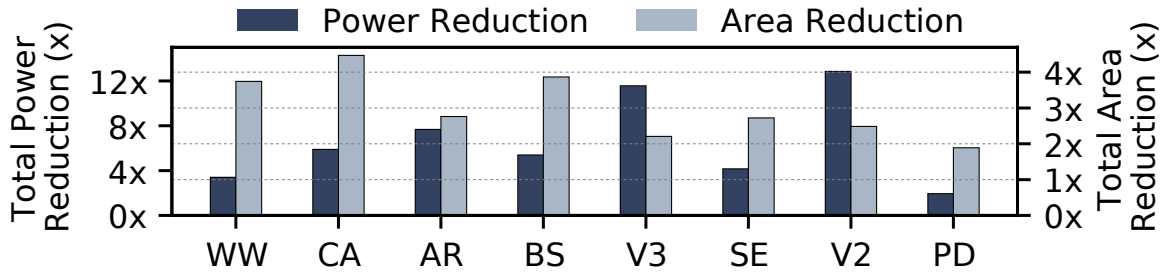


Figure 3.24.: Total area and power reduction (x) compared to the baseline designs [170] (i.e., vs Table 3.13). For our printed Decision Trees only our proposed bespoke ADCs and parallel unary architecture are considered.

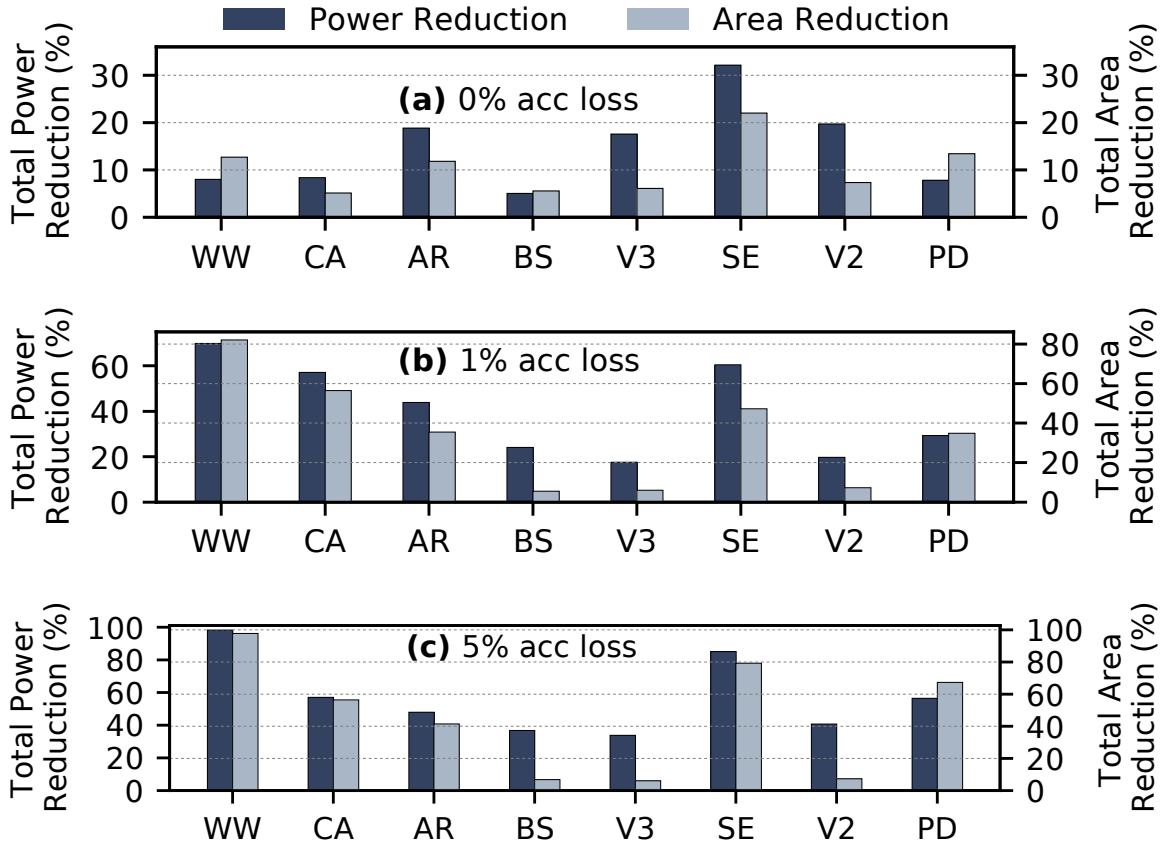


Figure 3.25.: Evaluation of the additional hardware gains delivered by our ADC-aware training. Total area and power reductions (%) of our printed DTs w/ and w/o our ADC-aware training are compared (i.e., vs Figure 3.24). Three accuracy loss constraints are considered: a) 0% (i.e., no accuracy loss), b) 1%, c) 5%.

Next, in Figure 3.25 we investigate the impact of our ADC-aware training. The area and power gains in Figure 3.25 are reported w.r.t. the area and power of the designs of Figure 3.24 (i.e., over the simplified Decision Trees with our bespoke ADCs). For this analysis, we consider three accuracy loss thresholds: 0%, 1%, and 5%. Specifically, we conduct a brute-force exploration of hyperparameters, including τ values ranging from 0 to 0.03 in increments of 0.005 and depth values from 2 to 8 with a step of 1. The rather simple classification tasks in PE [170] and the independence of different trainings (i.e., they can run in parallel), enable this exploration to be rapidly conducted. The average execution time is only 6 min on an Intel Xeon 6138 server with 256GB RAM. As demonstrated in Figure 3.25a, for *no accuracy loss*, our ADC-aware training leads to an average reduction of 11% in area and 15% in power when compared

Table 3.14.: Evaluation of our Decision Trees for up to 1% Accuracy Loss.

Dataset	Proposed		Reduction vs [170]		Reduction vs [208]	
	Area[1]	Power[1]	Area[1]	Power[1]	Area[1]	Power[1]
WhiteWine	11.99	1.26	21.8x	11.3x	10.5x	4.3x
Cardio	10.13	0.88	11.3x	14.1x	4.4x	2.4x
Arrhythmia	16.24	0.85	4.9x	14.1x	1.5x	1.3x
Balance-Scale	4.92	0.35	6.2x	8.2x	5.8x	3.6x
Vertebral-3C	2.71	0.17	6.2x	16.2x	3.4x	2.7x
Seeds	3.26	0.27	8.4x	11.9x	1.2x	1.1x
Vertebral-2C	2.22	0.15	7.4x	18.5x	-[2]	-[2]
Pendigits	89.00	6.12	3.0x	2.8x	4.2x	2.6x
Average	17.56	1.26	8.6x	12.2x	4.4x	2.6x

[1]Total area and total power, including ADCs.
[2]Not evaluated in [208].

to using the conventional ADC-unaware training. Similarly, for only 5% accuracy loss the average area and power reduction increase to 45% and 57%, respectively.

Finally, Table 4.10 evaluates the effectiveness of our co-design framework in generating self-powered printed Decision Trees, considering up to 1% accuracy loss. In Table 4.10, we also compare our Decision Trees against the baseline exact [170] (i.e., Table 3.13) and the approximate ones of [208] (with up to 1% accuracy loss). Note that for [170] conventional 4-bit ADCs are used, whereas for [208], since precision scaling is applied, the smallest suitable conventional ADC for each input is used. To the best of our knowledge, these are the only works that have investigated printed Decision Trees. As shown, our Decision Trees achieve on average 8.6x and 12.2x lower area and power, respectively, compared to [170]. Similarly, compared to [208], the corresponding gains are 4.4x and 2.6x, respectively. Note that in some cases (i.e., BS, V3, and PD), [208] features higher area and power consumption compared to our baseline [170] due to the use of deeper trees in [208] to compensate for the accuracy loss caused by their applied approximation. Concluding, as demonstrated by Table 4.10 all our classifiers except for Pendigits feature power consumption well bellow 2mW, even when accounting the significant cost of ADCs. Pendigits also adhered to the 2mW power constraint but at a 10% accuracy loss. This analysis suggests that our co-design framework can be used to effectively target printed applications with similar complexity to the datasets in Table 4.10, even after considering the cost of sensors, which is negligible compared to the hardware overheads of printed classifiers. For instance, relevant sensors reviewed in [158] for such printed applications consume only 5μW. Hence, for the examined datasets, the power increase due to sensors is less than 0.11mW. As a result, for less than 1% accuracy loss, our framework can efficiently produce printed classifiers, even with 207 comparators and 11 inputs, that demand less than 2mW of power, being thus suitable for printed energy harvester operation [182].

The proposed co-design delivers substantial area and power savings:

- Compared to a baseline of fully parallel bespoke decision trees with conventional 4-bit ADCs, the proposed ADC-aware co-design reduces area and power by averages of 8.6× and 12.2×, respectively, for up to 1% accuracy loss.
- Relative to a state-of-the-art approximate decision-tree design that ignores ADC cost, the gains are still about 4.4× in area and 2.6× in power.
- For seven out of eight datasets, the total power of the classifier—including bespoke ADCs—is well below 2 mW, making self-powered operation with printed energy harvesters feasible. Even for the most demanding dataset (Pendigits), the classifier can be kept under 2 mW at the price of around 10% accuracy loss.

The cost of printed sensors is negligible in comparison: typical printed sensors consume only a few μW, adding less than 0.11 mW to the total power for the examined datasets.

3.5.4.2. Discussion

Although this work targets digital printed classifiers rather than analog *p*-NNs, it provides important lessons for on-sensor intelligence in the same technology:

- ADCs can dominate area and power in on-sensor processing; any realistic printed design must co-optimize the classifier and ADC architecture.
- Unary representations, which would be prohibitively large in conventional silicon, become attractive in PE due to their simple logic and the possibility of bespoke ADCs that generate only the unary digits actually used.
- Hardware-aware training that knows the cost of each potential split is crucial for meeting strict power targets with minimal accuracy loss.

3.6. Chapter Summary

This chapter presented a comprehensive cross-layer methodology for designing energy-efficient and robust NNs in emerging PE platforms. The work unified algorithmic training, circuit modeling, and hardware co-design to address strict power budgets, device variability, temporal dynamics, and on-sensor intelligence.

First, power-aware (*P*-AT) and power-constrained (*P*-CT) training strategies were introduced for *p*-NNs. By developing differentiable analytical and surrogate power models for crossbars and nonlinear circuits, power was elevated from a post-design metric to an explicit optimization objective. *P*-AT enabled systematic exploration of accuracy–power Pareto fronts, while *P*-CT employed an augmented Lagrangian formulation to enforce strict power budgets during training. Together, these approaches demonstrated that substantial power reductions can be achieved with only modest accuracy degradation, making *p*-NNs compatible with printed batteries and energy harvesters.

Second, event-driven computation was explored through analog printed SNNs. The introduction of a learnable spike generator in the P-LSNN architecture enabled joint optimization of synaptic weights and spike dynamics under hardware constraints. By integrating surrogate circuit models and robustness-aware training, the proposed approach significantly reduced energy and area while improving classification accuracy and maintaining resilience under process variations.

Third, temporal intelligence was addressed through the *pTPNN* and the robustness-aware ADAPT-*p*-NN. Learnable first- and second-order temporal filters were co-designed with crossbar arrays, enabling effective processing of time-series data directly in the analog domain. Variation-aware training and data augmentation further enhanced robustness against manufacturing variability and sensor noise, yielding substantial accuracy gains with moderate hardware overhead.

Finally, on-sensor intelligence was investigated via bespoke ADCs and Decision Tree co-design. By leveraging unary representations and ADC-aware training, the proposed framework drastically reduced area and power consumption compared to prior printed digital classifiers. This co-design approach demonstrated that classifier architecture and conversion circuitry must be jointly optimized to achieve self-powered operation in PE.

Across all contributions, a unifying principle emerges: robust and energy-efficient printed intelligence requires tight integration of circuit physics, learning algorithms, and hardware-aware optimization. By treating power, variability, temporal dynamics, and sensing interfaces as first-class design constraints, this chapter establishes a scalable methodology for deploying reliable edge intelligence on flexible and ultra-low-cost printed platforms.

4. Architecture-Aware Bespoke NN Design

4.1. Neural Architecture Search for Variability-Aware *p*-NNs

The design of NNs for PE cannot directly follow conventional digital or CMOS-based architectures due to severe device-level constraints such as limited gain, large process variations, low mobility, and restricted supply voltages. Standard AFs implemented in software or silicon hardware often assume ideal nonlinear characteristics that are not physically realizable in printed technologies. As a result, a direct mapping of conventional NN architectures to PE leads to performance degradation and reduced reliability. To address these challenges, this chapter adopts an architecture-aware design methodology, where AFs and network structures are co-optimized with the underlying printed device characteristics. By tailoring the NN components, particularly learnable AF circuits, to the physical behavior of PE, we enable efficient, stable, and hardware-compatible neural computation.

In the following, we will provide a detailed description of learnable AFs and their transfer characteristic curves based on our circuit design in PE. Figure 4.1 (d, e, f) depicts the schematic of different learnable activation circuits, i.e., printed sigmoid (p-sigmoid), printed clipped ReLU (p-clipped_ReLU) and printed ReLU (p-ReLU) design.

4.1.1. Learnable Activation Function (AF) Circuits

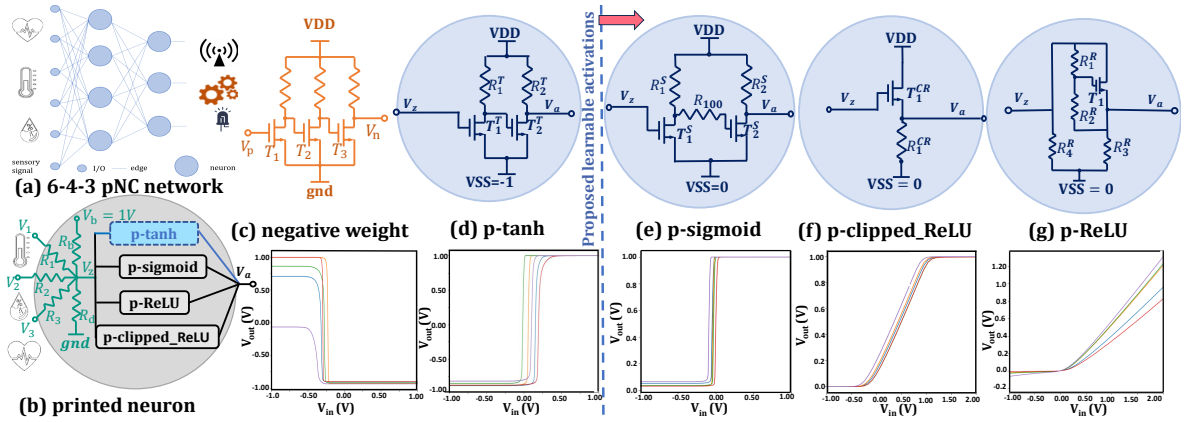


Figure 4.1.: Overview of (a) 6-4-3 printed neural network (*p*-NN). (b) printed neuron ;(c) negative weight circuit, activation circuit design and transfer characteristics curve of (d) p-tanh (e) p-sigmoid (f) p-clipped_ReLU and (g) p-ReLU.

Printed Sigmoid Circuit Similar to the learnable p-tanh AF circuit design in Figure 4.1 (c), the p-sigmoid AF can also be obtained from this configuration by replacing the supply voltage VSS = 0 and adding a small 100 Ω resistance as shown in Figure 4.1 (d) and can be modeled by a suitable mathematical equation with its own distinct parameters.

The equation for a printed sigmoid function is given by

$$V_{\mathrm{a}} = \eta_1^{\mathrm{S}} + \eta_2^{\mathrm{S}} \cdot sigmoid\left(\left(V_{\mathrm{z}} - \eta_3^{\mathrm{S}}\right) \cdot \eta_4^{\mathrm{S}}\right), \tag{4.1}$$

where $sigmoid(\cdot)$ function is defined by

$$sigmoid(x) = \frac{1}{1 + \mathrm{e}^{-x}}.$$

Table 4.1.: FEASIBLE DESIGN SPACE OF P-SIGMOID CIRCUIT

Range	R_1^S (kΩ)	R_2^S (kΩ)	W_1^S (µm)	L_1^S (µm)	W_2^S (µm)	L_2^S (µm)
minimal	350	40	80	80	500	40
maximal	750	80	600	200	800	80

where $\mathrm{J}^S = [\eta_1^S, \eta_2^S, \eta_3^S, \eta_4^S]$ are auxiliary parameters determined by the physical quantities $\mathrm{q}^S = [R_1^S, R_2^S, W_1^S, L_1^S, W_2^S, L_2^S]$ in the circuit. Adjusting these parameters allows to adapt the shape of the AF within the printed circuit.

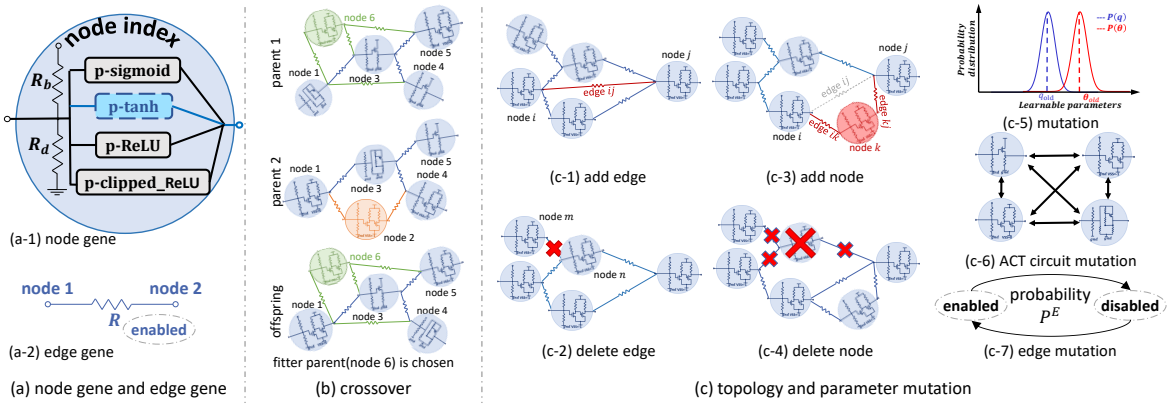


Figure 4.2.: Overview of the *EA*-based training of *p*-NNs. (a) Genes that encode nodes and edges. (b) Crossover from the parent genomes to offsprings. (c) Mutation of the topology and learnable parameters.

Printed Clipped ReLU Circuit A clipped ReLU activation circuit uses only one transistor and one resistor in series as shown in Figure 4.1 (e) and extends the basic ReLU by imposing an upper bound to the activation. The output voltage, V_a, is constrained between 0 and a predefined maximum value V_max.

Table 4.2.: FEASIBLE DESIGN SPACE OF P-CLIPPED RELU CIRCUIT

Range	R_1^{CR} (MΩ)	W_1^{CR} (µm)	L_1^{CR} (µm)
minimal	1	40	80
maximal	10	100	200

The mathematical representation of the clipped ReLU is:

$$V_\mathrm{a} = \begin{cases} \eta_1^{CR}, & V_\mathrm{z} < \eta_3^{CR} \\ \eta_2^{CR}, & V_\mathrm{z} > \eta_4^{CR} \\ \dfrac{\eta_2^{CR} - \eta_1^{CR}}{\eta_4^{CR} - \eta_3^{CR}} V_\mathrm{z} + \dfrac{\eta_1^{CR}\eta_4^{CR} - \eta_2^{CR}\eta_3^{CR}}{\eta_4^{CR} - \eta_3^{CR}}, & otherwise, \end{cases} \tag{4.2}$$

where $\mathrm{J}^{CR} = [\eta_1^{CR}, \eta_2^{CR}, \eta_3^{CR}, \eta_4^{CR}]$ are auxiliary parameters determined by the physical quantities $\mathrm{q}^{CR} = [R_1^{CR}, W_1^{CR}, L_1^{CR}]$. The printed clipped ReLU circuit thus models a modified ReLU function commonly

used in ANNs to prevent overactivation by limiting the maximum output value. The parameters can be tuned to fit specific requirements, providing flexibility in shaping the activation characteristics of the ANN. The ability to clip the activation at a certain level also helps with issues like exploding gradients during the training process of ANNs.

Printed ReLU Circuit The printed ReLU activation circuit as shown in Figure 4.1 (f), uses four resistors and one transistor and implements a piecewise linear function that maintains the linearity for positive inputs while nullifying negative inputs. As the transfer characteristic curve has a slope in the negative half and has a smooth transition at $V_z = 0$, neither the ideal ReLU function, nor its variation, e.g., LeakyReLU [179] function nor thesoftplus [94] function, is sufficient to precisely describe the printed ReLU circuit. Thus, we combine a softplus function to provide the smoothness at $V_z = 0$ and a constant linear function to provide the slope at negative half.

Table 4.3.: FEASIBLE DESIGN SPACE OF P-RELU CIRCUIT

Range	R_1^R (kΩ)	R_2^R (kΩ)	R_3^R (kΩ)	R_4^R (kΩ)	W_1^R (µm)	L_1^R (µm)
minimal	10	500	1	30	200	80
maximal	100	2000	20	100	800	120

Consequently, the function that describes printed ReLU circuit is : designed as

$$V_a = \eta_1^R \cdot (x - \eta_3^R) + \eta_2^R \cdot softplus(V_z - \eta_3^R, \eta_5^R) + \eta_4^R, \tag{4.3}$$

where the $softplus(\cdot,\cdot)$ function is expressed by

$$softplus(x,k) = \frac{1}{k} \cdot \log(1 + e^{k \cdot x})$$

where $\jmath^R = [\eta_1^R, \eta_2^R, \eta_3^R, \eta_4^R, \eta_5^R]$ are auxiliary parameters determined by the physical quantities $q^R = [R_1^R, R_2^R, R_3^R, R_4^R, W_1^R, L_1^R]$. This function has now been mostly used due to its simplicity and efficiency in promoting sparse activations in NNs. The component values can be tuned to modify the activation threshold, allowing for the emulation of various ReLU-like behaviors. This tuning capability makes printed ReLU versatile for different applications within ANNs, providing a more dynamic response compared to fixed AFs.

4.1.2. Evolutionary Architecture Search

We propose an innovative *EA* inspired by the classical NeuroEvolution of Augmenting Topologies (NEAT) [25] to design highly bespoke *p*-NN. The proposed algorithm not only tunes the component values within the AF circuits but also selects the most appropriate type of AF per layer per dataset to mitigate the effects of printing variation and outperform the error rate. In this problem statement, the parameters have certain constraints; conductances in the crossbar resistor arrays are limited to representing only positive weights. EAs are more convenient for considering the constraints on learnable parameters and are also suitable for non-differentiable problems, unlike gradient-based methods. As designing the circuit topology (i.e., the neural architecture) and selecting the optimal activation functional forms represent a discrete decision space, we can leverage the ability of EAs for such discrete problems.

We strategically encode the printed *p*-NNs so that the circuit topology are jointly optimized during evolution, leading to *NAS*. Additionally, the algorithm trains the crossbar conductances (i.e., weights) and optimizes the *type* of AF circuits for each printed neuron per dataset, along with the learnable parameters q in the AF circuits. With this enhanced search space, the resulting *p*-NNs are expected to be more robust

against printing variation, leading to a reduction in the post-mapping accuracy degradation compared to those trained by gradient methods, where only crossbar conductances and parameters in AF circuits are learned. The key components of the proposed algorithm are shown in Figure 4.2. Each genome represents a p-NNs and comprises two types of genes encoding the circuit specifications: node genes (Figure 4.2(a-1)) and edge genes (Figure 4.2(a-2)). Initially, a population of genomes is created and categorized into multiple species based on node and edge similarities. The number of offspring for each species is determined based on its mean fitness. Top-performing genomes are preserved unchanged for the next generation, while others undergo crossover and mutation. When the termination criteria are met, the most optimized solution is obtained according to the defined fitness function, as will be discussed in subsection 4.3.5. In the following, we explain how the genome encoding, as well as the processes of crossover and mutation for the proposed algorithm, are adjusted.

Encoding The **node gene** represents a neuron and consists of learnable parameters, including resistors R_b and R_d and learnable parameters of the candidate AF circuit and the negative weight circuit, i.e., q. Additionally, it also contains a learnable and discrete variable that determines which candidate activation circuit is selected to connect to the crossbar output. Each **node gene** is uniquely identified by a global index. On the other hand, the **edge gene** denotes connections between neurons and comprises a learnable resistance R in the crossbar for weights, along with a learnable boolean parameter indicating the connectivity state (enabled/disabled) for circuit topology. Each **edge gene** is distinguished by the indices of the connected nodes, and all edges are directional, meaning $(i, j) \neq (j, i)$. Here, we denote the set of genes from all the genomes in the population as 🧬.

Crossover In the proposed algorithm, unique genes (not shared between parents) of the fitter parent are directly inherited by the offspring, as illustrated by *node 6* in the offspring from *parent 1* in Figure 4.2(b). For common genes, crossover involves the random inheritance of genes from the shared topological structure (genes with identical global index) of both parent genomes, as demonstrated in Figure 4.2(b) for *node 3*, which is common between the two parents. During crossover, the features of each gene are exchanged randomly between the parents, with a higher probability of selection for the fitter parent. Subsequently, the mutation process is applied to the offspring.

Mutation Mutation, as depicted in Figure 4.2(c), is a two-stage process involving **genome-level** mutation for neural architecture and gene-level mutation for the parameters related to node and edge genes. At the **genome-level**, mutations can involve either the addition (Figure 4.2(c-1)) or deletion (Figure 4.2(c-2)) of edges between existing nodes, and similarly, nodes can be added (Figure 4.2(c-3)) or deleted (Figure 4.2(c-4)) at existing edges. When deleting a edge, a random edge is selected. Before deleting a node, edges associated with that node are removed to prevent disruption. Importantly, to avoid the extinction of new genomes, new structures should not influence genome fitness. Therefore, to maintain unchanged circuit output (and thus performance), the conductance of new edges should be initialized to zero when adding edges. Additionally, when adding a node to an edge, as shown in Figure 4.2(c-3), the existing edge is disabled (not deleted) and the new node is introduced with two connections to replace said edge. To preserve the output, the conductance on edge (k, j) should be initialized by that of the edge (i, j), whereas the output of the node k should be the same as that of node i. At the **gene-level**, mutation involves perturbing crossbar conductance θ and nonlinear circuit parameters q by adding scaled samples from a normal distribution $p(\theta)$ and p(q) respectively (Figure 4.2(c-5)). The type of selected activation circuit mutates randomly among [p-sigmoid, p-tanh, p-ReLU, p-clipped_ReLU] (Figure 4.2(c-6)), while the state parameter of the edge is determined by a Bernoulli variable (Figure 4.2(c-7)).

4.1.3. Variation-aware Training with NAS

Variation-aware training is critical in optimizing the reliability and performance of p-NNs, which often suffer from intrinsic printing variations.

We therefore integrate the proposed *NAS* to dynamically adjust the design of p-NNs, ensuring that they not only meet desired classification accuracy, but also demonstrate resilience to printing variations. This methodology not only enhances the adaptability of p-NNs but also their usefulness in real scenarios, where variations are a critical concern.

In this framework, all parameter corresponding to printed resistances, i.e., θ and transistors, i.e., q, are subject to processs variation arising from ink dispersion on the substrate, droplet jetting oddness and satellite drops wetting [98]. Each printing/processing step of the resistances and the n-EGTs (channel, dielectric, and top-gate) introduces variations resulting in non-Gaussian distributions for both the process and electrical parameters of this technology[129]. Consequently, these parameters are modeled as random variables to account for the inherent printing variations, i.e., $\theta \sim p(\theta)$ and $\mathrm{q} \sim p(\mathrm{q})$ respectively. These variables adhere to their respective probability distributions to mirror potential deviations arising from the printing process. To evaluate the robustness of different architectures against these variabilities and thus guide the evolution process, the expected loss with respect to parameter variation is used as the training objective, namely

$$\begin{aligned} \underset{\text{minimize}}{} \mathcal{L} &= \mathbb{E}_{\theta,\mathrm{q}} \left\{ L(\theta, \mathrm{q}, \mathcal{D}) \right\}, \\ &= \int_{\theta} \int_{\mathrm{q}} L(\theta, \mathrm{q}, \mathcal{D}) p(\theta) p(\mathrm{q}) \, \mathrm{d}\theta \, \mathrm{d}\mathrm{q}, \end{aligned} \tag{4.4}$$

where $\mathcal{D} = \{\mathrm{x}, \mathrm{y}\}$ refers to the target datasets, while $L(\cdot)$ refers to the cross-entropy loss [232], which is commonly used to improve classification accuracy. However, Equation 5.14 poses a challenge that the optimization variable θ and q will be integrated out. To facilitate the training of these parameters, we introduce a reparameterization strategy [56] to decouple the learnable parameters from the random variables expressing the variation. Consequently, $\theta = \theta_0 \odot \boldsymbol{\varepsilon}_\theta$ and $\mathrm{q} = \mathrm{q}_0 \odot \boldsymbol{\varepsilon}_q$, where θ_0 and q_0 denote target values to be optimized, while each element in $\boldsymbol{\varepsilon}_\theta$ and $\boldsymbol{\varepsilon}_q$ follows a distribution $p(\varepsilon)$ respectively. With this approach, the training objective can be reformulated as

$$\begin{aligned} \underset{\text{minimize}}{} \mathcal{L} &= \mathbb{E}_{\boldsymbol{\varepsilon}_\theta, \boldsymbol{\varepsilon}_q} \left\{ L(\theta_0 \odot \boldsymbol{\varepsilon}_\theta, \mathrm{q}_0 \odot \boldsymbol{\varepsilon}_q, \mathcal{D}) \right\}, \\ &= \int_{\boldsymbol{\varepsilon}_\theta} \int_{\boldsymbol{\varepsilon}_q} L(\theta_0 \odot \boldsymbol{\varepsilon}_\theta, \mathrm{q}_0 \odot \boldsymbol{\varepsilon}_q, \mathcal{D}) p(\boldsymbol{\varepsilon}_\theta) p(\boldsymbol{\varepsilon}_q) \, \mathrm{d}\boldsymbol{\varepsilon}_\theta \, \mathrm{d}\boldsymbol{\varepsilon}_q, \end{aligned} \tag{4.5}$$

With Equation 4.5, the parameters θ_0 and q_0 can be trained to guarantee the optimal classification accuracy under the expectation of given variation $p(\boldsymbol{\varepsilon}_\theta)$ and $p(\boldsymbol{\varepsilon}_q)$. However, the integration in Equation 4.5 still has no closed form, which poses challenge to its optimization. For this, we employ an estimation of the integration through Monte-Carlo sampling, i.e.,

$$\mathcal{L} \approx \frac{1}{N} \sum_{n=1}^{N} L(\theta_0 \odot \boldsymbol{\varepsilon}'_\theta, \mathrm{q}_0 \odot \boldsymbol{\varepsilon}'_q, \mathcal{D}), \tag{4.6}$$

where $\boldsymbol{\varepsilon}'_\theta \sim p(\boldsymbol{\varepsilon}_\theta)$ and $\boldsymbol{\varepsilon}'_q \sim p(\boldsymbol{\varepsilon}_q)$ are samples drawn from their respective distribution in each calculation of L. Moreover, N denotes the number of samples utilized to estimate the integration.

Finally, Equation 4.6 is utilized within the training objective (i.e., fitness function $f(x)$)) of the variation-aware training. For the training objective, we also consider the expected classification accuracy ACC, i.e.,

$$f(\mathbf{x}) = ACC - \mathcal{L}, \tag{4.7}$$

With this objective, *NAS* aims not only to improve the classification accuracy of p-NNs, but also improve the robustness against intrinsic stability with respect to variations. This process involves an assortment of optimization techniques, which may extend beyond conventional gradient-based methods, to seek out architectures that guarantee reliable performance despite the unpredictable nature of the printing process. The ultimate aim of using *NAS* in this context is to strike an optimal balance between performance and resilience, ensuring proper operation in real scenarios.

Table 4.4.: Simulation Result and Runtime of gradient-based approach without variation and comparison with *EA* with baseline in (i) high precision printing (5% variation) and (ii) low-precision printing (10% variation) on 13 Benchmark Datasets.

Dataset	**Reference accuracy (without variation)**	**High-precision printing (±5%)**		**Low-precision printing (±10%)**		**Runtime**	
		Baseline	***EA***	**Baseline**	***EA***	**Baseline (min)**	***EA* (min/pop)**
Acute Inflammation	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.999 ± 0.012	1.000 ± 0.000	183.9	9.5
Balance Scale	0.902 ± 0.017	0.880 ± 0.004	0.896 ± 0.008	0.877 ± 0.008	0.881 ± 0.012	205.9	21.6
Breast Cancer Wisconsin	0.971 ± 0.001	0.963 ± 0.008	0.966 ± 0.006	0.931 ± 0.039	0.949 ± 0.012	180.9	11.6
Cardiotocography	0.879 ± 0.007	0.774 ± 0.004	0.857 ± 0.005	0.763 ± 0.002	0.794 ± 0.007	178.0	19
Energy Efficiency (y_1)	0.915 ± 0.019	0.889 ± 0.032	0.916 ± 0.026	0.847 ± 0.012	0.866 ± 0.011	194.6	15.1
Energy Efficiency (y_2)	0.894 ± 0.016	0.883 ± 0.023	0.891 ± 0.038	0.867 ± 0.026	0.866 ± 0.021	189.4	10.3
Iris	0.965 ± 0.005	0.912 ± 0.034	0.923 ± 0.050	0.843 ± 0.045	0.882 ± 0.039	178.8	7.3
Mammographic Mass	0.788 ± 0.003	0.782 ± 0.017	0.810 ± 0.018	0.766 ± 0.053	0.764 ± 0.055	190.9	5.0
Pendigits	0.577 ± 0.054	0.554 ± 0.038	0.559 ± 0.039	0.548 ± 0.047	0.553 ± 0.050	198.3	14.2
Seeds	0.891 ± 0.031	0.820 ± 0.034	0.851 ± 0.023	0.820 ± 0.041	0.827 ± 0.007	176.0	6.4
Tic-Tac-Toe Endgame	1.000 ± 0.001	0.713 ± 0.012	0.765 ± 0.018	0.660 ± 0.017	0.716 ± 0.019	177.1	6.4
Vertebral Column (2 cl.)	0.830 ± 0.007	0.716 ± 0.007	0.794 ± 0.004	0.661 ± 0.000	0.685 ± 0.004	180.6	4.6
Vertebral Column (3 cl.)	0.811 ± 0.010	0.634 ± 0.086	0.791 ± 0.016	0.634 ± 0.075	0.734 ± 0.059	130.9	9.6
Average	**0.879 ± 0.013**	**0.809 ± 0.023**	**0.848 ± 0.019**	**0.786 ± 0.029**	**0.809 ± 0.023**	181.9	10.81

.

4.1.4. Evaluation

To assess the effectiveness of the proposed method, we utilized PyTorch to implement the algorithm[1] and carried out experiments on 13 benchmark datasets. These datasets are also used in other state-of-the-art studies on p-NNs [202], [220], and match the complexity within the application domains of PE.

4.1.5. Experiment

We conduct training on the p-NN utilizing the *EA* methodology and test it on 13 benchmark datasets against the established gradient-based optimization techniques as a baseline of this work.

Circuit Setup The AF circuits in Figure 4.1 (top (e), (f), (g)) were designed based on the well-developed n-EGT P-PDK [152] and the ranges of learnable parameters are determined by performing sweep analysis. We used Cadence Virtuoso[2] tool to simulate the transfer characteristics (as shown in Figure 4.1 (bottom (e), (f), (g))) in SPICE.

Initialization Drawing insights from other works on *EA* and guided by a series of preliminary trials, we have strategically initialized the network topologies for all datasets as unconnected networks, which consist solely of nodes corresponding to the number of outputs, featuring only $\#output$ nodes. The population for these experiments is robustly set at $1,000$ individuals. Each node is initialized to have a random AFs circuit among the given design.

In terms of the mutation mechanisms employed, we have defined specific probabilities for the genetic alterations within the network structures: the probability of introducing either a new node or a new connection is set at a substantial rate of 0.7, while the probability for the deletion of a node or a connection is comparatively lower, at 0.3. Moreover, there exists a 0.1 chance that any given edge within the network will toggle its state from enabled to disabled, or vice versa, as part of the mutation process. Moreover, the mutation rate of changing selected AFs circuit is 0.1.

In terms of variation, we take uniform distribution, i.e., $\boldsymbol{\varepsilon}_\theta \sim \mathcal{U}[1-\epsilon, 1+\epsilon]$ and $\boldsymbol{\varepsilon}_q \sim \mathcal{U}[1-\epsilon, 1+\epsilon]$, to reflect the printing variation, because the printing variation is primarily determined by the geometric variation of the printing shape which varies within one printing pixel. More specific, we select an $\epsilon = 5\%$ to simulate a relatively high printing precision, while another $\epsilon = 10\%$ to simulates a relatively low printing precision. This is because the typical printing resolutions range from 20 µm to 100 µm [65], whereas the component feature sizes in p-NNs are on the order of 1 mm [202]. Moreover, for Monte-Carlo sampling, we select $N = 20$ for numerical estimation of the integration, as it can already yield sufficiently precise estimation in our experiments.

Training In training (evolution) process, we utilize a full-batch training, with termination upon a patience threshold of 100 generations. This specific criterion hinges on observing no significant improvement in the performance metrics on the validation dataset over the aforementioned span of generations.

To ensure that our findings are statistically reliable and to mitigate the variability due to stochastic elements of the training process, we repeat the training sessions ten times for each value of γ, employing different random seeds for each session, varying from 1 to 10. This repetition ensures that we achieve sufficiently optimal and robust solutions.

Baseline To conduct experiment with baseline approach, we perform training with topologies initialized as $\#input$-3-$\#output$. We use the Adam [66] optimizer with default parameterization to train parameters. We start with an initial learning rate of 0.1 and halve it after a 100-epoch patience. Additionally, the

[1] `https://github.com/Neuromophic/eNAS_learnable_selectable_LNC`.

[2] `https://www.cadence.com/en_US/home.html`

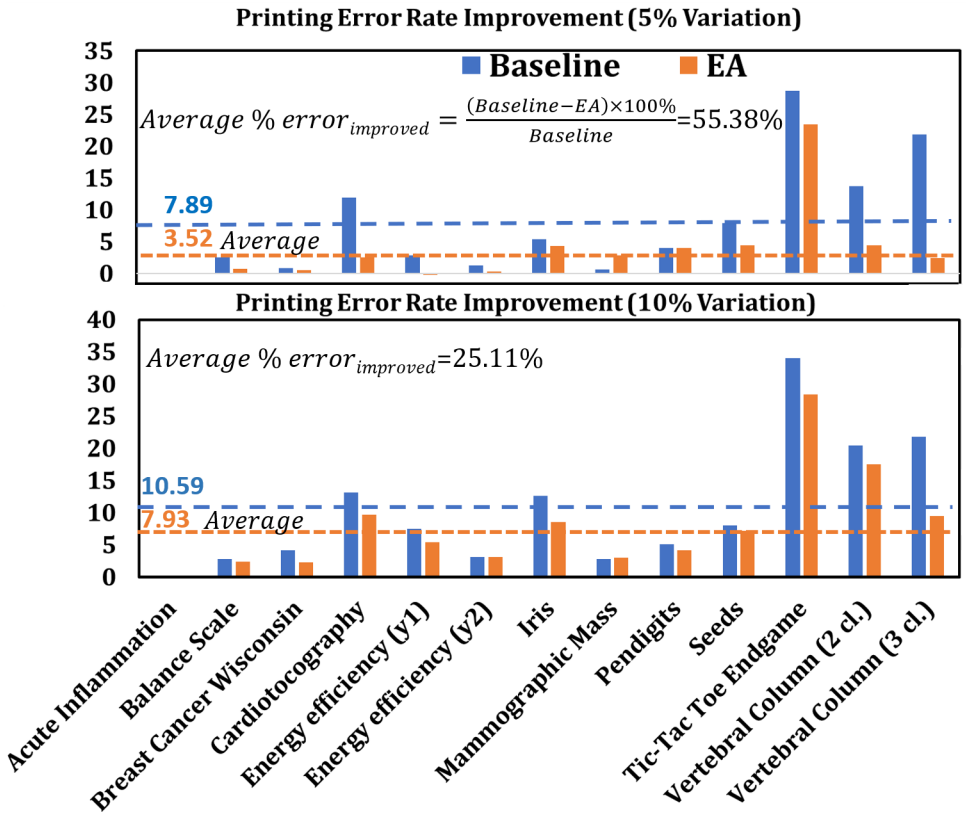


Figure 4.3.: Results of the printing error rate enhancement in 13 benchmark datasets with (i) ±5% and (ii) ±10% variation.

training process is stopped, when the learning rate was halved 10 times. Other setups are kept the same as its *EA* counterpart.

To provide an upper bound classification accuracy of each dataset, we also trained the p-NNs without any variation, i.e., $\epsilon = 0$. Because the accuracy in this case can be seen as the theoretically highest achievable values. We denote the accuracy in this case as the *reference accuracy*.

4.1.6. Result

After completing the training of all p-NNs, we carefully select the most optimal p-NNs w.r.t to the random seed in each experimental setup based on their performance on the validation loss. These selected circuits are the ones designed for physical realization. Subsequently, we evaluate their performance on the test sets. In testing, all p-NNs are tested and trained under the identical variations.

The mean, standard deviation of their accuracy and their corresponding runtime performances are presented in Table 4.4, showing a scalar average across all datasets for a clearer comparison of different training configurations. Additionally, Figure 4.3. illustrates the improvement in printing error rates. Also, Figure 4.4 details the selection percentages of various learnable AF circuits, highlighting their preferences during the training process.

Discussion Table 4.4 compares the performance of the EAs to a baseline gradient-based method across 13 benchmark datasets under conditions of high-precision (±5%) and low-precision (±10%) printing. The reference accuracy shows the performance without variation, providing a reference point for evaluating the resilience of the algorithms under variation. For both scenarios (±5% and ±10%), *EA* maintains comparable accuracy to the baseline across all datasets. Also, runtime analysis reveals that *EA*'s runtime depends on the population size and the extent of parallel processing utilized. *EA* can significantly reduce runtime through parallel processing, contrary to the baseline method's total sequential runtime and can offer more robust solutions per iteration.

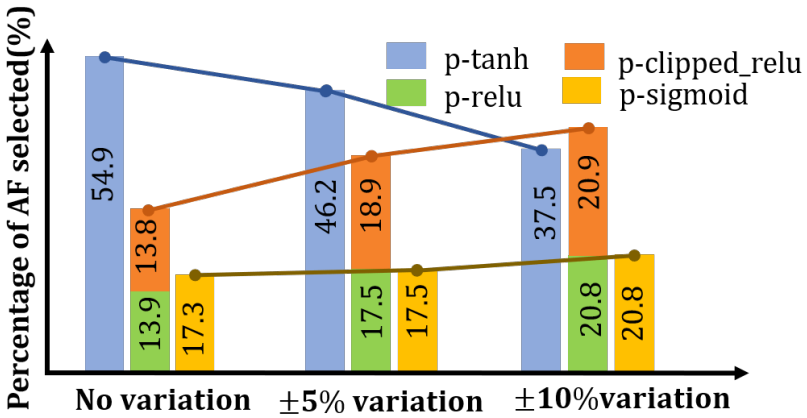


Figure 4.4.: Percentage of AF selected during (i) no (ii) ±5% and (iii) ±10% variation averaged over 13 benchmark datasets.

In Figure 4.3, EAs demonstrate robust error rate, with a significant enhancement of 55.38% in ±5% and 25.11% in ±10% scenarios, compared to the baseline, thus highlighting the EAs capability to effectively manage printing variations, thereby reducing error rates.

To the end, Figure 4.4 illustrates the selection percentages of different AFs used in *EA* under varying printing conditions. In worst-case scenario with ±10% variation, the ReLU family emerges as more favored, selected $\approx$ 21% of the time, up from $\approx$ 14% with no variation. This trend-line shift of the ReLU family suggests their ability to induce sparsity in NNs, encouraging the exploration of diverse solutions by focusing on relevant features. This leads to performance stability and better generalization, making ReLU-based NNs more suitable and robust in environments with higher uncertainty and variability. In contrast, p-tanh is the most preferred, chosen $\approx$ 55% of the time, due to its high sensitivity and effectiveness in stable conditions (no variation). However, p-sigmoid consistently maintains a steady selection rate across all levels of variation.

Therefore, it is worthy to conclude that the various learnable AF circuits and variation-aware training using *NAS* approach both contribute to a significant improvement in classification error-rate and robustness of *p*-NNs.

4.2. Neural Evolutionary Architecture Search for Compact *p*-NNs

The second contribution extends the variability-aware *NAS* framework to explicitly target *compact* analog *p*-NN by co-optimizing network topology and physical implementation parameters [259]. Instead of treating area and energy only implicitly through a size penalty, these quantities are modeled at circuit level and are optimized jointly with classification accuracy and robustness within a multi-objective evolutionary search.

To enable effective training for compact *p*-NNs, it is imperative to explicitly incorporate the circuit area into the training objective. In this work, the circuit area A is estimated through its components by $A = f(N_i, A_i)$, where $A_i \in \mathbb{R}^+$ denote the area of the individual devices, e.g., A_R denotes the area of a resistor, and $N_i \in \mathbb{N}$ denote their counts. Here, $f(\cdot)$ describes the area relationship between individual components and the overall circuit. Since N_i is an integer variable related to circuit topology, considering the area in the training objective necessitates training algorithms that are capable for discrete optimization. To this end, we propose an *EA*-based method for the simultaneous training of both crossbar conductances (weights) and circuit topologies (neural architecture).

The core parts of the proposed approach for training *p*-NNs is shown in Figure 4.5, involving *genes* that encode the circuit parameters and *genomes* that compose of multiple structured genes to represent the structure of *p*-NNs. They are optimized through the crossover and mutation during their evolution.

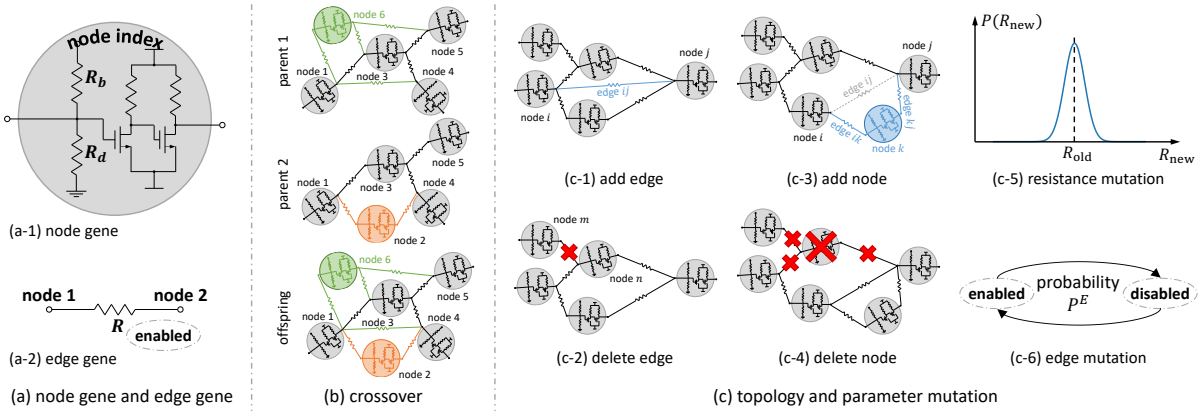


Figure 4.5.: Overview of the *EA*-based training of p-NNs. (a) Genes that encode nodes and edges. (b) Crossover from the parent genomes to offsprings. (c) Mutation of the topology and learnable parameters.

4.2.1. Encoding

The algorithm involves two gene types, namely **node genes** indicating neurons and **edge genes** denoting connections between neurons. As shown in Figure 4.5(a-1), every node gene holds a unique, fixed, and global index for identification. Moreover, each **node gene** includes an R_b and an R_d as learnable parameters (for weights and biases) followed by a fixed ptanh circuit as shown in Figure 4.1. These learnable parameters will mutate from generation to generation. Meanwhile, there are **edge genes** identified by the indices of the connected nodes, as shown in Figure 4.5(a-2), which are directional. Each edge gene contains a learnable resistance R, indicating the crossbar resistance for weights, and a learnable boolean parameter that indicates the state (enabled/disabled) for circuit connectivity (architecture). Similarly, these learnable parameters mutate during evolution.

Several node genes and their connections, i.e., edge genes, form a structured network that represents a p-NN. In this work, such a set of genes that represents a p-NN is referred to as a **genome**. Afterwards, a group of genomes can further form a **population**. We use 🧬 to denote the set of genes of all genomes involved in a population $\mathcal{P}$.

4.2.2. Evolution

In a population, the genomes are segregated into multiple **species** based on their similarities during the evolution. Each species undergoes origination, reproduction, and sometimes extinction. Here, in reproduction, well-performed genomes will crossover to produce offsprings and their genes will mutate. In this way, the fitness of the genomes will be gradually optimized, until a certain stop criterion is reached. The overview of this process is illustrated in 3, the detailed python implementation is available at our GitHub repository (given in subsection 6.1.4).

Speciation To protect novel genomes from immediate extinction without being evolved, the genomes within the population $\mathcal{P}$ are grouped into multiple species $\mathcal{S}$ based on their similarity. This speciation allows each species s to develop without impact from other species. Here, the similarity is determined by both common structures (exemplified as the gray part in Figure 4.5(b) in parent 1 and 2) and distinctive structures (exemplified as the green and orange parts in Figure 4.5(b) in parent 1 and 2). The distance of the former is quantified by the absolute difference in their learnable parameters, whereas the distance of the latter is measured by the absolute value of their parameters. For boolean variables, the distance between $True$ and $False$ is set to 1.

Extinction and Selection After speciation, the set of fitness values $\mathcal{F}$ of each genome is evaluated by the objective function $O(\cdot)$. Meanwhile, the average fitness within a species is calculated to represent the fitness of each species, denoted by F_s, and is summarized in the set $\mathcal{F}_S$ for all species. If the fitness F_s of a

Algorithm 3 Evolutionary Algorithm

Require: Dataset $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})\}$
Require: N (population size), $O(\cdot)$ (objective function), selection$(\cdot)$ (parent selection), adjust$(\cdot)$ (species size calculator), $\mathcal{K}$ (candidate parents for crossover), $\mathcal{K}_1$ (patience for species improvement), $\mathcal{K}_2$ (number of protected species), $\mathcal{K}_3$ (patience for evolution)
Ensure: Final population $\mathcal{P}$

Initialize population $\mathcal{P}$ with N genomes
stop $\leftarrow$ False
$\mathcal{S} \leftarrow \varnothing$
while *stop* = False **do**
    $\mathcal{S} \leftarrow$ speciation$(\mathcal{P})$
    $(\mathcal{F}, \mathcal{F}_S) \leftarrow O(\mathcal{S}, \mathcal{D})$
    $\mathcal{N}_S \leftarrow$ adjust$(\mathcal{F}_S)$
    for all $s \in \mathcal{S}$ **do**
        if NoImprove$(\mathcal{F}_S(s), \mathcal{K}_1)$ **then**
            if $\neg$ Protected$(s, \mathcal{K}_2)$ **then**
                Extinct$(s) \leftarrow$ True
        else
            for $n \leftarrow 1$ **to** $\mathcal{N}_S(s)$ **do**
                $(p_1, p_2) \leftarrow$ selection$(s, \mathcal{K})$
                $o_n \leftarrow$ crossover(p_1, p_2)
                $o_n \leftarrow$ mutation(o_n)
            $s \leftarrow \{o_1, o_2, \ldots, o_{\mathcal{N}_S(s)}\}$
    if NoImprove$(\max(\mathcal{F}), \mathcal{K}_3)$ **then**
        stop $\leftarrow$ True

certain species does not show improvement over $\mathcal{K}_1$ generations, it will become extinct unless it is one of the best $\mathcal{K}_2$ species. Subsequently, the top $\mathcal{K}$ genomes with respect to their fitness in each species are chosen as parents

$$P_s = \{p_1, \ldots, p_{\mathcal{K}}\}$$

for crossover.

Crossover Crossover refers to producing offsprings o that randomly inherits the genes from its parental genomes. This is a crucial process in producing evolved offspring while preserving well-performed structures. In this work, each offspring is produced from the crossover between two parents randomly selected from the parent candidates P_s. For common architectures in both parents (illustrated in Figure 4.5(b) as the gray part), the offspring inherits these structures directly, and the parameters for these structures are chosen from one of the parent based on a probability proportional to their fitness ratio. As for the distinct structures (depicted in Figure 4.5(b) as the green and orange parts), they are directly passed on to the offspring.

Mutation Mutation is another primary method for introducing new circuit architectures and serves as the essential source for circuit parameter evolution. As shown in Figure 4.5(c), in this work, mutation is a two-stage process consisting of genome-level mutation (targeting on neural architecture) and gene-level mutation (primarily for network parameters).

At the **genome-level**, mutation can either add or delete an edge between two existing nodes. Analogously, nodes can be added or deleted at an existing edge. Importantly, to prevent the extinction of newly mutated genomes, the structural changes introduced by mutation should not substantially affect the genome fitness. Therefore, to maintain the unchanged circuit output (thus fitness), when adding an edge in between two

nodes, as shown in Figure 4.5(c-1), the conductance of the new edge should be initialized to zero and be optimized during evolution.

Additionally, when adding a node to an edge, the existing edge will be disabled (not deleted) and the new node is introduced with two connections to replace said edge, as shown in Figure 4.5(c-3). To preserve the output, the conductance on edge (k, j) should be initialized by that of the edge (i, j), whereas the output of the node k should be the same as that of node i, i.e.,

$$\mathrm{p}tanh\left(\frac{g^{(i,k)}}{g^{(i,k)}+g_{\mathrm{b}}^{k}+g_{\mathrm{d}}^{k}}V_{\mathrm{a}}^{i}+\frac{g_{\mathrm{b}}^{k}}{g^{(i,k)}+g_{\mathrm{b}}^{k}+g_{\mathrm{d}}^{k}}V_{\mathrm{b}}\right)=V_{\mathrm{a}}^{i},$$

with the superscript denoting the gene index. For simplicity, we always initialize $g_{\mathrm{b}} = 0$ and $g_{\mathrm{d}} = 1$ for new nodes. Hence,

$$\eta_1^{\mathrm{A}}+\eta_2^{\mathrm{A}}\cdot tanh\left(\left(\frac{g^{(i,k)}}{g^{(i,k)}+1}V_{\mathrm{a}}^{i}-\eta_3^{\mathrm{A}}\right)\cdot\eta_4^{\mathrm{A}}\right)=V_{\mathrm{a}}^{i}.$$

Finally, the conductance on edge (i, k) is initialized to

$$g^{(i,k)}=\frac{M}{1-M},$$

with

$$M=\frac{1}{\eta_4^{\mathrm{A}}V_{\mathrm{a}}^{i}}\tanh^{-1}\left(\frac{V_{\mathrm{a}}^{i}-\eta_1^{\mathrm{A}}}{\eta_2^{\mathrm{A}}}\right)+\frac{\eta_3^{\mathrm{A}}}{V_{\mathrm{a}}^{i}}.$$

Note that, according to Equation 5.8, M is always real-valued. Furthermore, the algorithm is only negligible affected even if $M \approx 1$ or $V_{\mathrm{a}}^{i} \approx 0$, since g will not approach ∞ but is bounded by the range of printable conductances.

In contrast, the **gene-level** mutation is targeting to mutate circuit parameters (i.e., crossbar resistances) and is realized by a perturbation of the old values. This is implemented by adding a scaled sample from a standard normal distribution to the current resistance value, as shown in Figure 4.5(c-5). Additionally, the state parameter of the edge (i.e., enabled/disabled) is mutated through a variable drawn from the Bernoulli distribution, as shown in Figure 4.5(c-6).

Objective The training objective, i.e., the fitness function, considers both circuit area and classification accuracy. The former is assessed by counting the total number of nodes and edges and multiplied by their respective area. The latter is designed as a combination of the classification accuracy (ACC) and the cross-entropy (CE) loss function, which is a smooth and convex surrogate function for classification accuracy [232]. Although *EA* enables to directly employ accuracy (i.e., the actual classification metric) as the training target, integrating cross-entropy can offer smoother guidance during evolution and provides fine-grained feedback on improvements. This becomes particularly valuable when assessing minor perturbations in resistance values in gene mutations. Consequently, the combined metric for classification accuracy is

$$O(\mathrm{x},\mathrm{y},\text{🧬})=CE\left(\mathrm{y},\hat{\mathrm{y}}(\mathrm{x},\text{🧬})\right)-ACC\left(\mathrm{y},\hat{\mathrm{y}}(\mathrm{x},\text{🧬})\right), \tag{4.8}$$

where x, y are training examples provided by the target dataset, while $\hat{\mathrm{y}}(\mathrm{x},$ 🧬$)$ denotes the output of the genome. Then, the overall training objective, i.e., the fitness function, is given by

$$\underset{\text{🧬}}{minimize}\ (1-\gamma)O(\mathrm{x},\mathrm{y},\text{🧬})+\gamma\frac{A(\text{🧬})}{A'}, \tag{4.9}$$

where $\gamma \in \mathbb{R}^{+}$ expresses the balance between accuracy and area, and A' is a constant multiplier to calibrate the of area term to the similar magnitude of accuracy term $O(\mathrm{x}, \mathrm{y},$ 🧬$)$.
At the beginning of evolution, the *EA* starts with only output nodes. From there, it progressively increases the number of neurons and their connectivity. Over successive generational iterations, genome fitness improves progressively. Upon reaching the stop criterion (with which the genomes are sufficiently optimized), the associated topological structures and parameters can be mapped to the respective hardware primitives and fabricated.

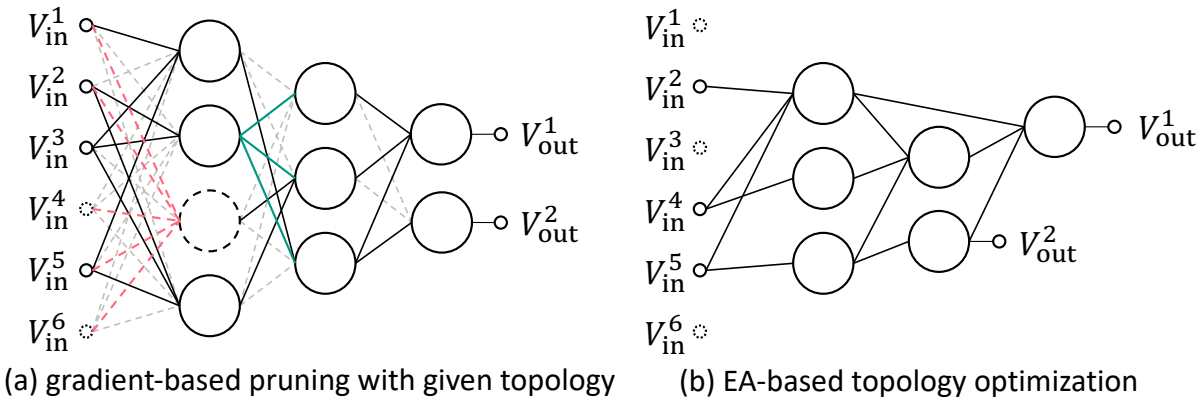


Figure 4.6.: Comparison of the circuit architecture from (a) gradient and (b) evolutionary approaches. In (a), the dashed edges and nodes are pruned. The red color refers to pruning a neuron when all its input weights are pruned. The green color represents the case of pruning a negation circuit when all the weights associated to a voltage are positive.

Baseline Methods This paragraph introduces the baseline we employed in our experiment. Different from *EA* that starts from a minimal architecture and gradually grows the network size, as the counterpart of *EA*, modern ML models are typically trained through gradient-based techniques. Gradient-based training usually provides much higher efficiency through backpropagation, but are generally less effective in topological optimization. To explore the effectiveness of the gradient approach in p-NN architecture and to provide a strong benchmark for evaluating the effectiveness of the proposed *EA*, we utilize a state-of-the-art pruning method. In addition, we introduce a task-specific *area-aware training* approach to especially encourage p-NNs with lower area footprint. These approaches start with a large and predefined circuit architecture, progressively removes circuit components, and finally results in an area-efficient circuit architecture.

Cutting Edge Approaches Gradient-based methods are more efficient than EAs in terms of training time, however, they are often inadequate to optimize network architectures because the architectures are usually discrete variables that do not produce an informative gradient to guide the training process. State-of-the-art gradient-based strategies for optimizing network architectures are *NAS* [135] and *network pruning* [193]. Unfortunately, *NAS* approaches are mainly designed for DNNs with block structures [88], [117], [132], [133] such as residual blocks with different kernel sizes or long-short-term-memory (LSTM) blocks, and they require hand-crafted architectures. For instance, in *differentiable architecture search* (DARTS) [126], several convolutional kernels with different sizes are pre-designed as candidates, and finally, the optimal one is chosen by learning the *importance factor* for each block. However, *NAS* essentially degrades to network pruning in the context of MLPs, because the *importance factors* for blocks in DNNs can be interpreted directly as the *weights* in MLPs.

Network pruning refers to remove parts of the network parameters to reduce the network size. Here, a regularizer (penalty function) is often employed to encourage higher sparsity of network parameters. Depending on the forms of regularization, pruning can be divided into *unstructured pruning* (targeting individual parameters) [74] and *structured pruning* (targeting groups of parameters) [93]. The former typically incorporates the ℓ_p norms of parameters into the regularizer independently, e.g., through

$$\|g_1\|_1+\|g_2\|_1+\cdots+\|g_i\|_1+\cdots \tag{4.10}$$

to promote increased number of zero-valued parameters In contrast, the latter applies penalties to the ℓ_p norms of grouped parameters, e.g., all weights associated with a neuron

$$\|[g_1,\, g_2,\, \cdots,\, g_i,\, \cdots]\|_2 \tag{4.11}$$

to foster the elimination of complete neurons. Therefore, the latter is also named grouped pruning **grouppruning**, **channelgrouping**.

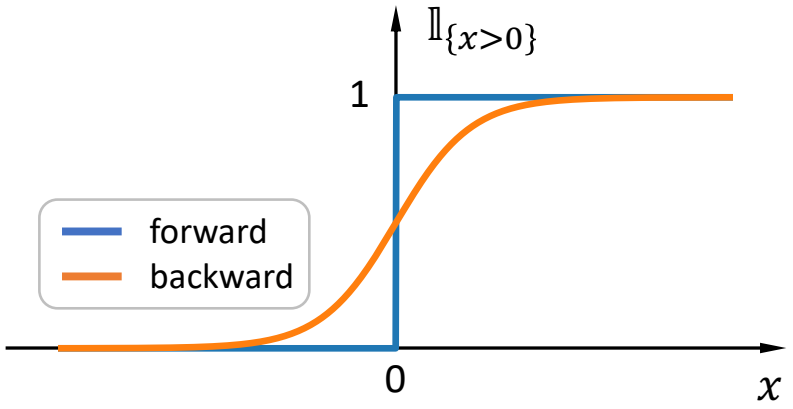


Figure 4.7.: Forward and backward pass of the soft count. The blue curve counts the number of x by 1 when $x > 0$, otherwise 0. In backward, the orange function is employed to derive the gradient information.

In ML, structured pruning is more favored as it streamlines the network architecture both from the algorithm and the hardware perspectives by simplifying the matrix multiplications. Unstructured pruning, on the other hand, does not provide obvious improvement due to the lack of conclusive tools to support sparse matrix multiplication. Conversely, in the context of p-NNs, both pruning approaches bring significant benefits. Because owing to the highly flexible and agile manufacturing process of PE, removing any component can contribute to the compactness of the circuits: Unstructured pruning can remove crossbar resistors, whereas structured pruning enables to remove entire printed neurons. Notably, this is a unique advantage of the additive PE.

In this regard, we employ combined unstructured pruning, Equation 4.10 and structured pruning, Equation 4.11 methods as a state-of-the-art baseline for the proposed evolutionary architecture algorithm.

4.2.3. Area-Aware Training with Pruning

In addition to existing network pruning methods, this work also enhances the existing pruning method to specifically encourage the compact design of p-NNs, namely the *area-aware training*.

Despite the large amount of research on the functional forms [44], [142], [261] of regularization functions, these studies typically employ simple and differentiable functions, that are not directly applicable to assess the circuit area. To address this issue and explore more potential of gradient-based pruning in compact p-NN design, we aim to incorporate circuit area of p-NNs directly as the regularization term in the training objective. This approach promotes the explicit optimization of compact p-NNs. Additionally, as the circuit area is significantly influenced by the circuit architecture, which is non-differentiable and fails to offer valid gradient information, we introduce gradient-relaxation methods to address these challenges and thus enable gradient-based training.

As illustrated by the dashed neuron in Figure 4.6(a), the presence of a neuron can be expressed by the existence of its input weights embodied by the conductance g_i, i.e.,

$$\max_i \left\{ [\mathbb{1}_{\{g_1>0\}}, \mathbb{1}_{\{g_2>0\}}, \cdots, \mathbb{1}_{\{g_i>0\}}, \cdots] \right\}. \tag{4.12}$$

This method belongs to structured pruning, as it aims to eliminate the entire neuron. Different from traditional regularization like Equation 4.11, Equation 4.12 only suppresses the largest input conductance in the crossbar, which avoids the impact on other input conductances and thereby minimizing the effect on classification accuracy caused by the regularization. However, the indicator function $\mathbb{1}_{\{\cdot\}}$ is a piece-wise constant function, as shown by the blue function in Figure 4.7, it has a gradient of zero almost everywhere. To address this issue and obtain gradient information that can guide the parameter update, we use the result of Equation 4.12 in the forward pass, while calculating gradients using a smooth relaxation called soft

counts [243], N^{soft}, in the backward pass. In this work, the $sigmoid(\cdot)$ function is used as the smoothing function, therefore, the function used for backpropagation of Equation 4.12 is given by

$$\max_i \{[sigmoid(g_1), sigmoid(g_2), \cdots, sigmoid(g_i), \cdots]\},$$

as shown in by the orange function in Figure 4.7.

Analogously, the presence of a negation circuit can be calculated through negative surrogate conductances, i.e.,

$$\max_j \left\{[\mathbb{1}_{\{\theta_1<0\}}, \mathbb{1}_{\{\theta_2<0\}}, \cdots, \mathbb{1}_{\{\theta_j<0\}}, \cdots]\right\}.$$

where θ_j refers to the succeeding conductances from a neuron, as shown by the green part in Figure 4.6. If none of the corresponding weights is negative, the output voltage from the preceding neuron does not need to be negated [243]. Similarly, the gradient of this function is relaxed by

$$\max_j \{1 - [sigmoid(\theta_1), sigmoid(\theta_2), \cdots, sigmoid(\theta_i), \cdots]\}.$$

Regarding the count of the resistors, which aligns with unstructured pruning, we employ $\mathbb{1}_{g_i>0}$ to count each crossbar resistor, while its gradient is given by

$$sigmoid(g_i)$$

With these soft counts, the area estimator adapted for gradient-based is expressed as

$$A^{soft} = f(N_i^{soft}, A_i).$$

Finally, we use the cross-entropy function [232] to guide the training for higher accuracy. Comparable to Equation 4.9, the training objective for the pruning is

$$minimize\ (1-\gamma)\, CE\,(\mathrm{y}, \hat{\mathrm{y}}(\mathrm{x},\ \theta)) + \gamma \frac{A^{soft}(\theta)}{A'}, \tag{4.13}$$

with θ summarizing all learnable conductances in Equation 2.13, and $CE(\cdot)$ refers to the cross-entropy loss function.

Fine-Tuning It is worthy to highlight that p-NNs trained with Equation 4.13 may not reach optimal trade-off between area and accuracy. Because the area term, functioning as a penalty, suppresses the conductances through *soft count* for decreasing the device counts, and thus circuit footprint. However, if, e.g., a conductance cannot be suppressed to zero for a given γ, the resistor can not be removed. In this case, such suppression not only fails to reduce the circuit area, but also diminishes the classification accuracy by forcing parameters from the optimal values for the cross-entropy loss.

To mitigate this problem, we introduce the fine-tuning process. After the main training process introduced above, we generate masks m for each surrogate conductance θ and multiply them to indicate the parameters after pruning, i.e.,

$$\theta \leftarrow m \cdot \theta,$$

where m is either 1 or 0, indicating whether the parameter is pruned. Regarding the pruning of neurons, if all input parameters of a neuron are pruned, the output voltage of this neuron will be multiplied with a mask value equaling 0. As for the negation circuits, we introduce

$$\theta \leftarrow \theta^+ \cdot (1 - m^{\mathrm{N}}) + \theta \cdot m^{\mathrm{N}}$$

to emulate the pruning of negation circuit. Here, $\theta^+ = \max\{0, \theta\}$, and m^{N} refers to the pruning of the negation circuit. $m^{\mathrm{N}} = 0$ marks the negation circuit is pruned, therefore, surrogate conductance θ can only be positive.

Subsequently, we take the cross-entropy loss as the objective function to train the pruned network towards higher classification accuracy. With this mask-based pruning emulation, the classification accuracy can be further improved without any area increase. In other word, the impact of the noneffective area penalty on the classification accuracy can be recovered in the fine-tuning stage.

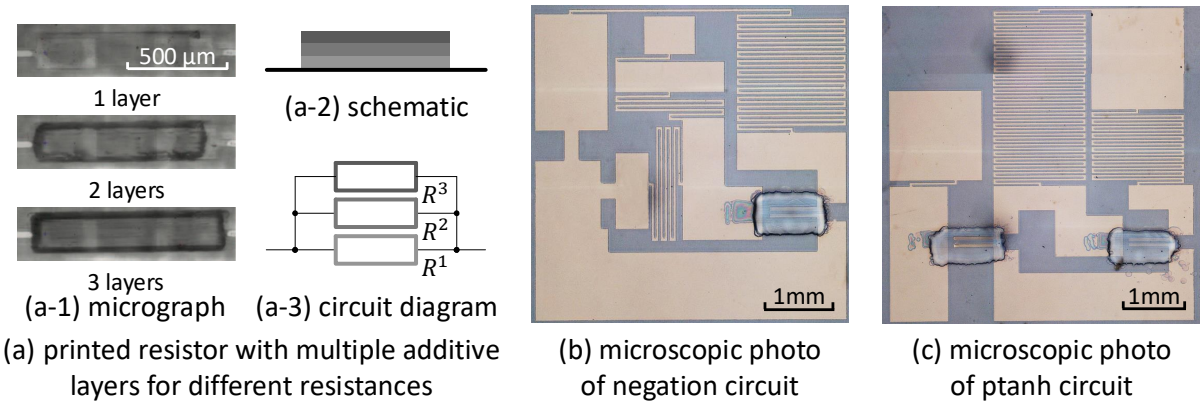


Figure 4.8.: Schematics and photos of the primitives in *p*-NNs. (b) and (c) sourced from [202] with permission for reprinting.

4.2.4. Circuit Area Model

By establishing the training methodologies that allow optimizing circuit architectures, we can leverage highly flexible and bespoke manufacturing process of PE to reduce circuit footprint and enhance circuit compactness. However, achieving this objective requires also to establish a precise model for estimating the circuit area in the regularization term in the objective function.

Area of Circuit Primitive The schematics of a printed resistor is depicted in Figure 4.8(a). Due to the additive manufacturing, the resistance values are progressively modulated by sequentially depositing resistive material on top of the existing resistors. Therefore, differences in printed resistors with different values primarily arise in their thickness, whereas their areas remain unchanged. Moreover, Figure 4.8(b) and (c) show the microscopic photos of the printed negation and ptanh circuits. As these circuits are predefined and fix during training, their respective areas remain also constant. Consequently, we read the area information directly from [202] as: area of the resistor (A_R) is $0.15\,\mathrm{mm}^2$, area of negation circuit (A_N) is $30\,\mathrm{mm}^2$ and area of activation (ptanh) circuit (A_A) is $22\,\mathrm{mm}^2$, correspondingly.

Software for Circuit Area Calculation To assess the area, we do **not** proceed to estimate the circuit area A directly through the summed area of each device. Because with the increasing number of devices, their connections will grow in a super-linear way, driving the routing problem more complicated and thus requiring more area than their linear relationship.

Rather, we utilize mature and well-developed commercial electronic design automation (EDA) tools to facilitate automatic placement and routing, serving as an estimator of circuit footprints. Although there are no specialized placement and routing tools developed for PE, the commonalities between PE and printed circuit boards (PCBs) suggests us to apply PCB-design tools for PE. Because both PE and PCBs are to place some predefined geometric components in a 2D surface, and their routing is featured by the connected pins on the given 2D space. In this context, we utilize EasyEDA[3] tool for the automatic placement and routing of *p*-NNs.

Although PE is predominantly a 2D technology, thanks to the additive manufacturing, the issue of wire-crossing can be simply addressed. As illustrated in Figure 4.9(c), we can print insulating materials (in our case, the dimethyl sulfoxide, DMSO) on top of the printed wires at the region that will be crossed. Subsequently, the second wire can be printed over the insulator. This approach is similar to multilayer PCBs with via holes, where the PCB substrates function as the insulator to avoid the intersection of wires, and the via holes provide connections across layers of substrates. Therefore, the automatic routing algorithm can natively support the tasks in PE. However, the additive PE offers significantly greater flexibility than PCBs in managing such issues.

[3] The software is available at `https://easyeda.com/`.

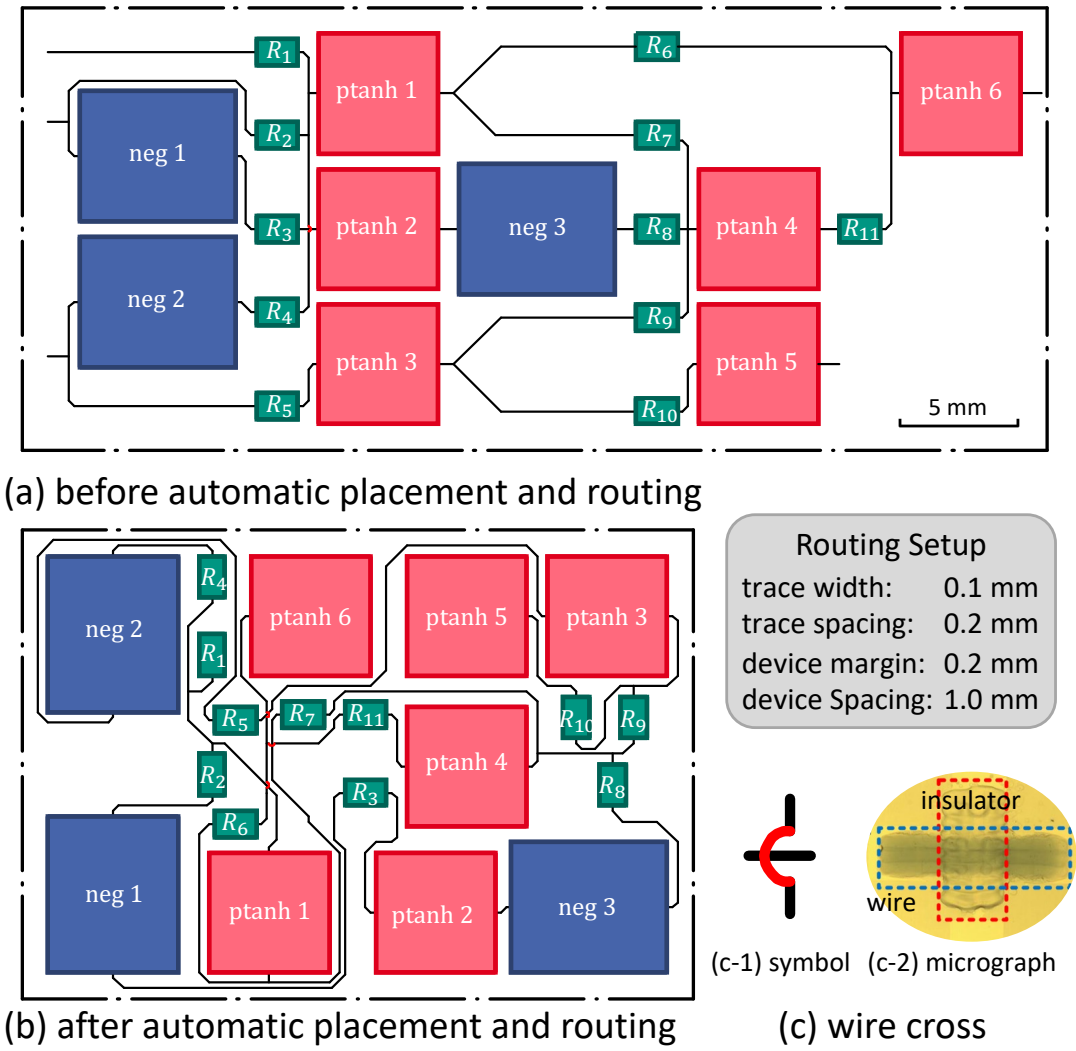


Figure 4.9.: Placement and routing of the analog p-NNs. (a) A naive placement that mimics the form of neural networks described in Figure 4.6(b), and (b) the solution of the automatic placement and routing from EasyEDA software. The gray box shows the major setups of the algorithm (with technology specification of PE). (c) illustrates the wire cross in PE: (c-1) is the symbol of cross that appears in (a) and (b), while (c-2) denotes the microscopic photo of a wire-cross with PEDOT:PSS as the conductive wire and dimethyl sulfoxide (DMSO) as the insulator, from **rasheed2020crossover**.

4.2.5. Circuit Area Estimator

As it is hard to integrate the EasyEDA into the training approach of the p-NNs, we have to develop a model that can estimate the circuit area during training and can be integrated into the algorithm. Given the sophisticated relationship between the circuit area and its architecture, we adopt an ANN-based model as the area estimator, because it is proven to be a universal approximator. The establishment of the area estimator comprised three stages: data acquisition, model design, and model training.

Data acquisition We first defined our customized library in EasyEDA based on the printed component geometry depicted in Figure 4.8, which includes dimensions and pin configurations. Subsequently, we randomly generate 500 different p-NNs architectures and convert them into netlists for importation into EasyEDA for automatic placement and routing. Key setups for placement and routing are reported in the gray box in Figure 4.9. After this, we use the minimum bounding rectangles for each circuit to denote the area footprint of the p-NNs. Since the algorithm provided by EasyEDA is not deterministic, i.e., each conduction may produce a different result, we repeated the algorithm 10 times per p-NN, and record their bounding box areas.

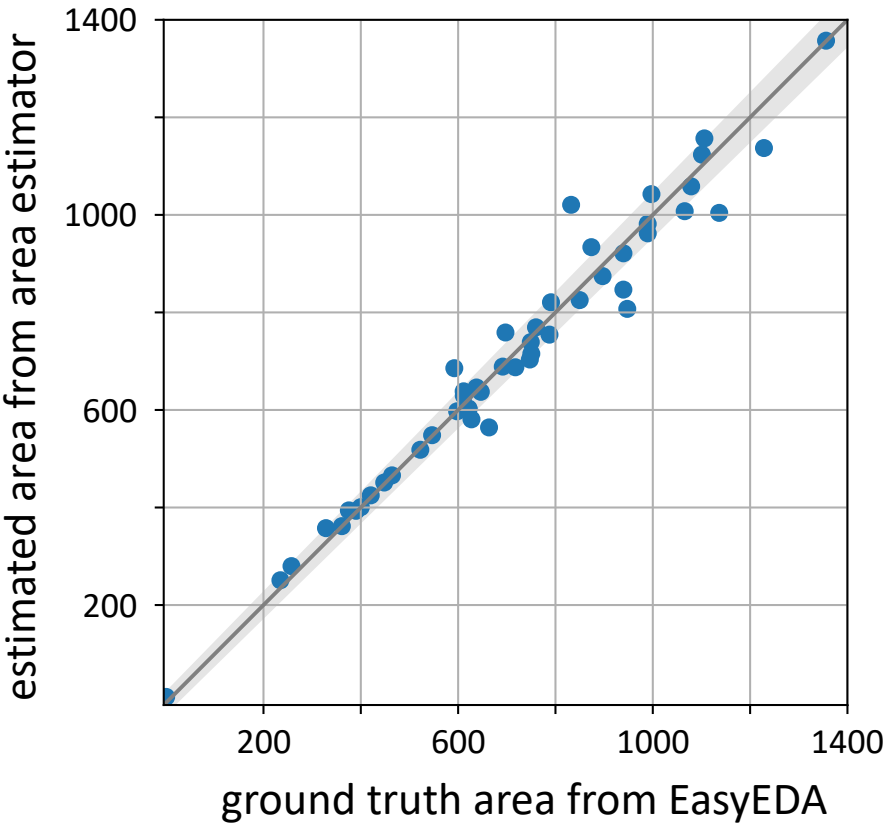


Figure 4.10.: Performance of the VAE-based area estimator on the test data. The x-axis is the ground truth area from the EasyEDA, while the y-axis denotes the estimated areas. Each blue point is a test data. The gray diagonal line refers to the ideal case where the estimation equals the ground truth. The gray area around the line represents the variation of the ground truth areas.

Area estimator model With the collected data, we constructed an ANN-based model that is capable of estimating the circuit areas A from their device counts N_i and device areas A_i, denoted by

$$A = AreaEstimator(N_i, A_i).$$

As the area generated by EasyEDA is not a deterministic value but rather follows a certain distribution, we employ a Variational Autoencoder (VAE) as the area estimator. Because VAEs natively support probability distribution as model output **vae** and widely used as generative AI models **vaegenerative**. In our area estimator model, input features (circuit netlists) are encoded to multiple latent distributions. The decoder then draws samples from the latent distributions to estimate the circuit areas.

Area estimator training To train the area estimator for precisely estimating circuit areas, the collected dataset is randomly divided into training (60%), validation (20%), and test (20%) sets. We used the training data to guide the training of the model, while employing the validation set to avoid overfitting through the early-stopping technique. Meanwhile, we utilized data normalization and hyperparameter tuning to enhance the precision of the area estimator. Afterwards, the final model with a 7-layer encoder and a 7-layer decoder is selected as the area estimator for p-NNs. Figure 4.10 illustrates the model performance on test data, which has not been used during training. The results suggest that, the area estimator can provide acceptable ($\leq$ 5% error) area estimation, and does not overfit the training data.

Finally, the well-trained area estimator is then used as the area terms in Equation 4.9 and Equation 4.13 to provide a precise circuit area.

4.2.6. Evaluation

To evaluate the efficacy of the proposed methods, we implemented both evolutionary and pruning algorithmswith PyTorch and conducted experiments on 13 benchmark datasets, which are recommended by the neuromorphic computing survey [111]. They are also employed in other state-of-the-art studies on *p*-NNs [202], [241] and aligned with the complexity of the application domains of PE. As the functionality of the printed *f*-NN hardware has been validated in [202], [218] and the contribution of this work, i.e., circuit area, can be completely verified at the algorithmic level, the experiment is conducted at simulation level based on the pPDK [129].

Circuit Design Setup In the following, we provide a detailed description of the negation circuit and ptanh activation circuit based on our feasible circuit design space in the pPDK [129]. It is worth note that, the printed EGTs are characterized by the length L and the width W of the channel size [195], i.e., the overlap between the top gate (PEDOT:PSS) and the semiconductor (In_2O_3) in Figure 2.4.

The device parameters used in the pPDK for the negation circuits and ptanh activation circuits are reported in Table 4.5. Regarding printed resistors, as they need to have different values to reflect specific weights, they are a range in pPDK rather than a fixed value. Specifically, they range from 100 kΩ to 10 MΩ.

Table 4.5.: Device Parameters in Nonlinear Circuits in pPDK
(VDD=2V, Threshold Voltage V_{th_EGT}=0.24V)

negation circuit	R_1^{N} (kΩ)	R_2^{N} (kΩ)	R_3^{N} (kΩ)	L_1^{N} (µm)	W_1^{N} (µm)	L_2^{N} (µm)	W_2^{N} (µm)	L_3^{N} (µm)	W_3^{N} (µm)
value	250	8	500	100	80	500	30	80	150
ptanh circuit	R_1^{A} (kΩ)	R_2^{A} (kΩ)	–	L_1^{A} (µm)	W_1^{A} (µm)	L_2^{A} (µm)	W_2^{A} (µm)	–	–
value	2050	7	–	80	80	480	40	–	–

4.2.7. Experiment

We train *p*-NNs with both *EA* and gradient approaches. For gradient-based training, we employ the existing (area-unaware) pruning as the baseline of this work. Meanwhile, we run the proposed area-aware pruning as another benchmark for compact *p*-NN design.

Initialization By referring to other works on *EA* and through a few preliminary trials, the population size is initialized to 1 000. Moreover, to preserve a minimum size of the circuits, the topologies for all datasets are initialized as unconnected networks, i.e., featuring only $\#output$ nodes.

Regarding network pruning via gradient-based approach, as pruning can only remove circuit components, it is critical to start with a larger architecture to ensure the competitive sub-architectures are included in the search space. Therefore, initialize the $\#input$-4-3-$\#output$ topology as a basis structure for pruning. This initial size is slightly more expansive than those typically employed in other works such as [241].

In addition, through few initial trials, the calibrating factor A' in Equation 4.9 is empirically set to 600 mm^2. Since this term simply aims to balance the loss term and the area term into a similar order of magnitudes, A' is not required to be a precise value, and will not impact the training results.

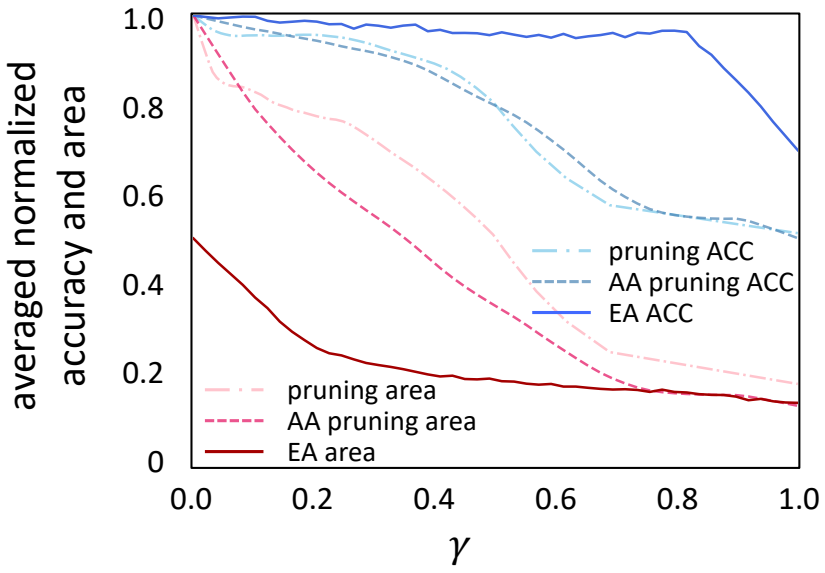


Figure 4.11.: Results of the experiment: averaged normalized accuracy (ACC) and area from three methods, namely the *EA*-based training, the area-aware (AA) pruning, and the existing pruning method.

Training During evolution, the top 10 best genomes in each species are chosen to be the candidate parents for crossover. The patience for species improvement, i.e., $\mathcal{K}_1$, is 20, while the number of species being protected, i.e., $\mathcal{K}_2$, is 2. In mutation, the probability of both adding node and adding edge in the mutation phase is 0.7, while that of deleting node and deleting edge is 0.3. In this way, the networks tend to have increasing size. Meanwhile, the edge state has 0.1 possibility to switch from enabled to disabled and vice versa. We employ full-batch training with the stop criteria being a 100-generation patience, i.e., $\mathcal{K}_3$, referring to consistently no improvement on the validation set. Moreover, to investigate the trade-off between accuracy and area, we run 50 evolutions with 50 uniformly selected values in $\gamma \in [0, 1]$. The whole process is repeated 10 times (with random seed varying from 1 to 10) for each value of γ to make sure to achieve a sufficiently good solution.

Regarding gradient-based pruning approach, we employ the Adam [66] optimizer in default parameterization to update parameters. To prevent overfitting, we use early-stopping [18], namely, we initialize the learning rate with 0.1 and halve it after a patience (updates without improvement on objective function) of 100-epochs on the validation set. The training process is stopped, when the learning rate was halved 10 times. Other setups are kept the same as its *EA* counterpart. Similarly, fine-tuning is conducted to further improve the classification accuracy of the pruned networks.

4.2.8. Result

After training, results are calculated on the corresponding test sets. It is evident that, with increasing γ, the training objective gradually transitions from prioritizing classification accuracy to minimizing circuit area. Conceptually, $\gamma = 0$ yields the highest accuracy and the largest area. As the objective in this case only focuses on the accuracy and ignores the circuit area, the network trained through gradient approach in this case is therefore referred to as the *reference*. We report the classification accuracy, circuit area, and training time in this case in Table 4.6.

Meanwhile, as γ increased, the both area and classification accuracy decreased. In this process, since the reduction ratio of accuracy and area holds more significance than their specific values, subsequent data will be normalized by the results of the *reference* values of the baseline, i.e., the existing pruning. The change of accuracy and area versus γ is described in Figure 4.11.

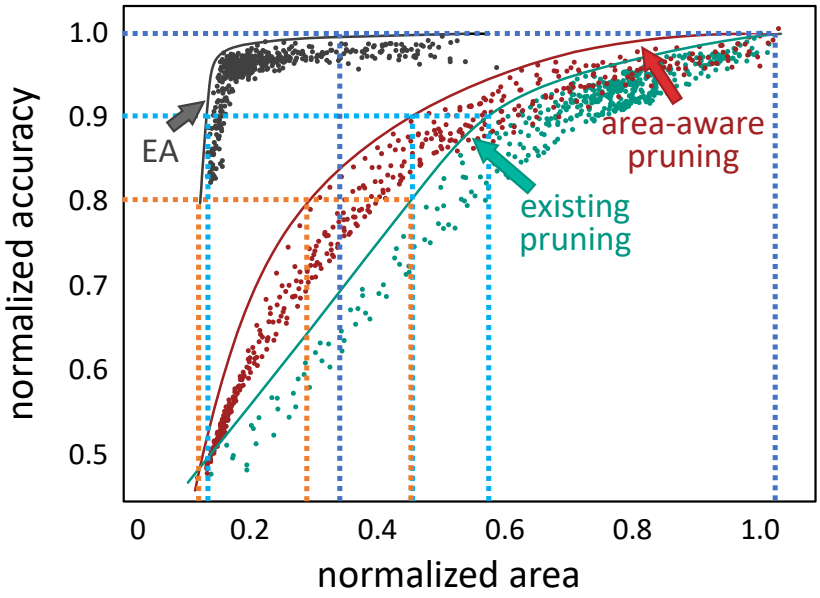


Figure 4.12.: Scatters and Pareto fronts of three methods, green for existing pruning that is unaware of circuit area, red for proposed area-aware pruning, and black for *EA*-based area-aware training.

Existing pruning vs. area-aware training By comparing the state-of-the-art pruning methods with the modified compactness-specific area-aware pruning, we conclude that, although they yield similar accuracies, the circuit area from existing pruning is consistently larger than the area-aware pruning. We speculate that, this is because of the unawareness of the circuit area of the existing pruning. For instance, the device counts of the negation circuits is not included in the regularization term. Consequently, the training will not encourage more positive weights to reduce the number of negation circuits.

***EA* approach vs. area-aware training** It can be seen that even with $\gamma = 0$, *EA* can already achieve competitive classification accuracy and a substantially smaller circuit area, when compared to the gradient approach. We speculate that, this is because the *EA* method starts the search from the minimal architecture, and thus may converge near it. Subsequently, as the γ increases, although both methods decrease the accuracy and area of p-NNs, the *EA* method produces a more modest degradation in accuracy. Because when $\gamma = 1$, the training objective focuses solely on minimizing area, disregarding classification accuracy. This should ideally produce the smallest circuit area, where only the output neurons are expected to be kept. In this case, *EA* enables input signals to be directly connected to output neurons to produce a reasonable classification result (even though the computing power of a single neuron is very weak). But the pruning-based approach can not even achieve this, because all neurons in the shallow layers are pruned, resulting in a forward propagation that was cut off from the input signal. Consequently, the output of the network behaves no better than a random guess.

To obtain the Pareto optimal trade-offs between accuracy and area, we plot the entirety of normalized areas versus their respective normalized accuracies for all runs and all values of γ in Figure 4.12 with the green (for existing pruning), red (for area-aware pruning), and black scatters (for *EA*). Based on the scatters, three Pareto fronts are drawn with their identical colors. Notably, the Pareto front of *EA* significantly outperforms that of the gradient-based training methods. Because *EA* natively support the optimization of circuit architectures. Moreover, the area-aware training yields better trade-offs compared to the area-unaware counterpart.

To provide more quantitative comparison, Table 4.7 displays several trade-off points from the Pareto fronts. It is evident that the *EA* approach offers 3.1× area-savings without any accuracy degradation compared to pruning. In case a 10% reduction in normalized accuracy is permissible, only 15% of the reference area is needed, which is 2.9× area reduction compared to area-aware pruning. Moreover, we

Table 4.6.: Simulation Result and Runtime of Three Approaches on 13 Benchmark Datasets with $\gamma = 0$

Dataset	Existing (area-unaware) Pruning			Area-Aware Pruning			Proposed *EA* Approach		
	Accuracy	**Area** (mm^2)	**Runtime** (min)	**Accuracy**	**Area** (mm^2)	**Runtime** (min)	**Accuracy**	**Area** (mm^2)	**Runtime** (min)
AcuteInf	1.000 ± 0.000	688 ± 47	8.3 ± 1.1	1.000 ± 0.000	743 ± 74	7.9 ± 1.5	1.000 ± 0.000	165 ± 17	32.8 ± 10.4
Bal.Sca	0.966 ± 0.065	890 ± 40	11.5 ± 2.4	0.930 ± 0.030	894 ± 28	11.5 ± 2.5	0.929 ± 0.019	269 ± 5	101.1 ± 31.1
BreastCanc	0.972 ± 0.003	827 ± 62	10.4 ± 2.2	0.970 ± 0.004	769 ± 136	11.3 ± 1.1	0.962 ± 0.013	176 ± 18	83.2 ± 13.7
Cardio	0.876 ± 0.006	1378 ± 97	12.4 ± 1.9	0.867 ± 0.010	1299 ± 94	11.4 ± 2.6	0.865 ± 0.032	238 ± 16	118.9 ± 26.0
En-(y_1)	0.969 ± 0.006	959 ± 29	15.6 ± 3.8	0.972 ± 0.017	970 ± 32	13.6 ± 4.8	1.000 ± 0.000	228 ± 32	88.9 ± 36.4
En-(y_2)	0.921 ± 0.011	965 ± 62	11.2 ± 0.8	0.917 ± 0.022	988 ± 82	10.4 ± 1.1	0.903 ± 0.015	188 ± 3	82.9 ± 26.3
Iris	0.975 ± 0.012	832 ± 71	8.6 ± 0.7	0.962 ± 0.006	859 ± 59	7.5 ± 0.8	0.935 ± 0.000	227 ± 6	32.6 ± 12.2
Mammo	0.857 ± 0.008	876 ± 40	9.8 ± 0.8	0.860 ± 0.012	827 ± 33	9.1 ± 1.2	0.865 ± 0.019	213 ± 2	71.3 ± 21.6
PenD	0.322 ± 0.013	1121 ± 30	21.6 ± 3.8	0.338 ± 0.070	1076 ± 177	22.2 ± 2.5	0.485 ± 0.102	551 ± 13	196.3 ± 66.5
Seeds	0.919 ± 0.023	957 ± 34	8.7 ± 0.8	0.936 ± 0.045	915 ± 62	8.9 ± 0.8	0.884 ± 0.023	262 ± 19	47.0 ± 19.5
TicTac	0.998 ± 0.004	883 ± 90	9.7 ± 0.5	0.974 ± 0.024	982 ± 36	9.4 ± 0.5	0.951 ± 0.015	199 ± 5	75.3 ± 26.3
Vert-2C	0.828 ± 0.006	811 ± 12	7.3 ± 0.5	0.825 ± 0.007	816 ± 2	7.9 ± 0.5	0.815 ± 0.009	238 ± 3	46.5 ± 20.3
Vert-3C	0.817 ± 0.017	978 ± 12	8.8 ± 1.2	0.810 ± 0.025	939 ± 60	8.6 ± 1.3	0.838 ± 0.009	238 ± 7	55.2 ± 16.8
Average	**0.878 ± 0.013**	**936 ± 48**	**11.2 ± 1.6**	**0.874 ± 0.021**	**929 ± 67**	**10.7 ± 1.6**	**0.879 ± 0.020**	**246 ± 11**	**79.4 ± 25.2**

also report the power consumption of the trained *p*-NNs through an established power consumption model from [243]. Specifically, the power are calculated from a hybrid model consisting of an analytical formula for resistor crossbars and a ANN-based surrogate power model for the nonlinear circuits. We observe that, as a byproduct of lower device counts, the power can also be greatly reduced compared to pruning. Specifically, the *EA* surpasses the area-aware pruning method by 3.0× and 2.9× power reduction respectively while providing 100% and 90% normalized accuracies.

Table 4.7.: Comparison of Accuracy-Area Trade-off

Normalized	**Area-Aware Pruning**		***EA* (superiority over pruning)**	
Accuracy (%)	**Area (%)**	**Power** (mW)	**Area (%)**	**Power** (mW)
100	100	78	32 (↓ 3.1×)	26 (↓ 3.0×)
90	44	37	15 (↓ 2.9×)	13 (↓ 2.9×)
80	31	22	12 (↓ 2.5×)	4 (↓ 5.5×)

Multivariate Regression Analysis To further assess the contribution of each circuit primitive in the circuit performance, we conducted multivariate regression analysis and reported the result in Table 4.8. It can be observed, each circuit primitive contributes to an increase in both area and power consumption, while also enhancing the classification accuracy.

Notably, the final circuit areas are contributed through resistor, negation circuit, and activation circuit by 3.19 mm^2, 42.2 mm^2, and 27.9 mm^2, respectively, which are slightly larger than the areas of the primitives themselves (0.15 mm^2, 30 mm^2, and 22 mm^2). Especially, while the nonlinear circuits contribute to the final area similarly to their individual area, the contribution of resistors is significantly larger than their own areas. This is because the resistors not only occupy their own area but also indicate the interconnection between neuron, which can lead to sophisticated placement and routing problems. This further justifies our work in establishing precise circuit area estimator based on EasyEDA and its placement & routing algorithm.

Algorithm Complexity Beyond the primary concerns, i.e., area and accuracy in this work, the complexity of the algorithms also draws our attention, because this is a notable distinction between gradient and evolutionary approaches. Thus, we summarize the training times for both methodologies in Table 4.6.

In our experiments, *EA* demands significantly more time (≈ 7×) than its gradient counterpart. However, we have the following comments on this issue: *(i)* The most time-consuming steps in *EA* are evaluating genomes and producing offsprings, which are strongly related (almost proportional to) the population size N. To fully explore the potential capabilities of *EA*, we have selected a large N in this work, nevertheless, it might be reduced in other scenarios. *(ii)* Genome evaluation and offspring production in *EA* are highly suitable to parallelization, through which the training can be accelerated. Although the parallelization is not included in this work, this can be done as a part of future work. *(iii)* Even though *EA* takes longer training time in our setup, the duration (ranging from 30 minutes to 3 hours) remains acceptable in the broader context of the product development cycle. This is not only because circuit optimization is only part of the non-recurring engineering (NRE), but also due to the target applications of PE that often require only small-scale circuits.

4.2.9. Case Study

To provide a clear and comprehensive overview of the effectiveness of our proposed method, we illustrate the algorithm performance with the *Energy Efficient* y_2 dataset as an example, as shown in Figure 4.13.

Table 4.8.: Coefficients in Multivariate Regression Analysis

Target\Factor	N_R	N_N	N_A
Area (mm^2)	3.1860	42.221	27.949
Accuracy	0.0158	0.0253	0.0790
Power (mW)	0.0339	0.3753	0.1553

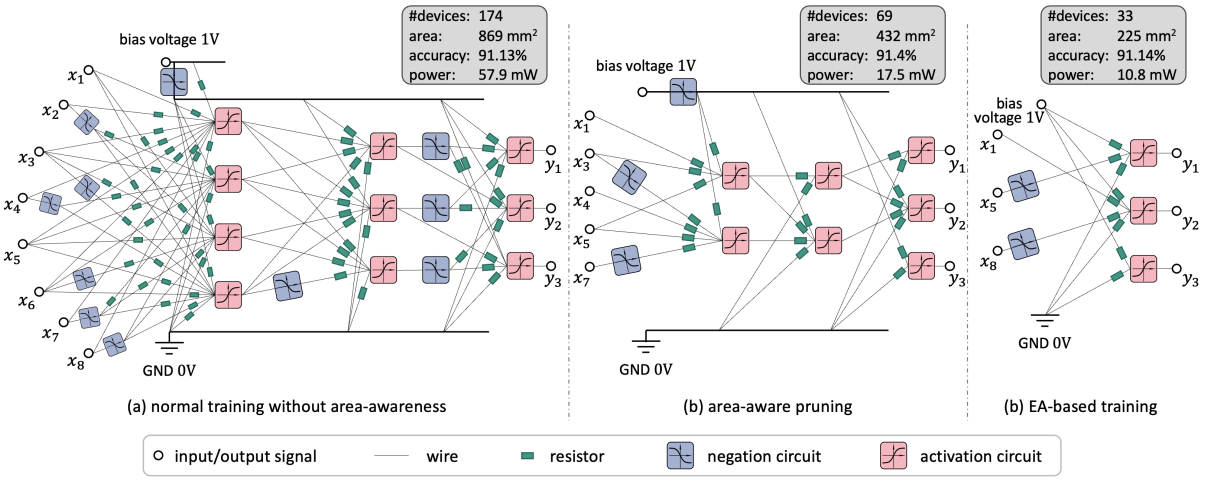


Figure 4.13.: Case study on the *Energy Efficiency* y_2 dataset with 8 input features and 3 output classes. Subfigures show the minimal area footprint to achieve ≈ 91% classification accuracy. The circuits are training from (a) previous area-unaware training, (b) area-aware pruning, and (c) *EA*-based area-aware training with architecture search. To maintain clarity, we illustrate only the symbolic diagram at the primitive-level without automatic placement. Nevertheless, we report the related attributes in the gray boxes.

From Figure 4.13(a) we know that, without the precise area estimator model, i.e., without area-aware training, the circuit architecture often fails to be optimized for compactness, resulting in complicated circuits structure with high power consumption and area footprint.

In contrast, with the established area model and pruning technique, the algorithm can effectively remove redundant components with awareness of circuit area (Figure 4.13b). This significantly simplifies the circuit architecture while maintaining unchanged classification accuracy. However, pruning still presents obvious limitations: 1) It is heavily dependent on the initial circuit architecture. Since pruning can only reduce circuit architecture from the original one, the initial design strongly restricts the upper bound of circuit scale, and thereby the circuit classification accuracy. 2) Meanwhile, the lower bound of pruning is also limited. For instance, pruning may not remove an entire layer, as it will interrupt the signal propagation, which could result in an output irrelevant to the input signal.

In opposite, our *EA*-based automatic architecture search addresses the limitations of pruning, and can therefore yield an optimal circuit architecture with significant smaller circuit area and power consumption, while preserving comparable classification accuracy.

Discussion The NEAS framework demonstrates that evolutionary search can reason jointly about algorithmic and physical design aspects of printed f-NN hardware. By embedding device-count, area, and energy models directly into the fitness evaluation, it becomes possible to automatically discover architectures that are not only accurate and robust under printing variation, but also match tight resource budgets of ultra-low-cost printed platforms.

The use of a multi-objective *EA* yields a diverse Pareto front of candidate p-NNs, ranging from extremely compact designs suitable for sub-cent smart labels to slightly larger but more accurate designs for higher-end flexible devices. This flexibility is crucial for future applications, where the same p-NN design methodology may have to serve a wide range of edge scenarios with very different requirements on cost, energy, and performance.

Finally, NEAS provides a natural foundation on which further cross-layer optimizations can be built, such as incorporating thermal models for flexible oxide TFTs (as addressed by Thermo-*NAS* in Section 4.3) or integrating application-specific constraints on latency and measurement interfaces.

4.3. Thermo-NAS for Flexible IGZO-based NN Hardware

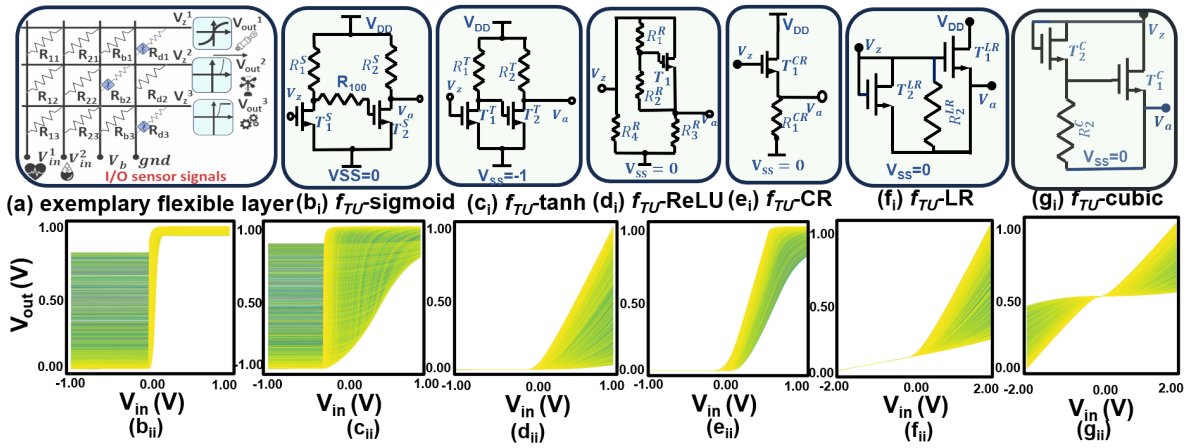


Figure 4.14.: (a) Bespoke flexible neuron layer; (b-g) thermal-unresilient (*TU*) activation circuits (f_{TU}*-sigmoid,* f_{TR}*-tanh,* f_{TR}*-ReLU,* f_{TR}*-clipped-ReLU,* f_{TR}*-Leaky-ReLU,* f_{TR}*-cubic*) with transfer curves within a temperature sweep from −20 °C to 185 °C.

In addition to printing-related variability, flexible NN hardware implemented in a-IGZOs TFT technology must operate reliably across wide ambient temperature ranges and under self-heating. Device parameters such as mobility, threshold voltage, and leakage currents vary strongly with temperature, which directly perturbs the gain and operating point of the analog neurons. Thermo-*NAS* is a thermal-resilient architecture search flow that co-optimizes NN topology and the parameters of temperature-aware analog building blocks on such flexible a-IGZO platforms[274].

The focus is on small a-IGZO-based f-NN classifiers that use operational transconductance amplifiers (OTAs) and inverter-based shaping circuits as their main analog primitives. A temperature-aware OTA and bias/reference infrastructure are embedded into the architecture-search loop, so that both the high-level network structure and the low-level analog parameters are chosen to minimize accuracy loss over a specified temperature range.

In the following, we will provide a detailed description of *TR*-crossbar arrays, thermal stability blocks and *TR*-learnable AFs and their transfer curves based on our circuit design.

4.3.1. Thermal-resilient (TR) resistor-crossbars

The resistor crossbar in the flexible neuron (left part of Figure 4.15(b)) exhibits temperature-dependent resistances, modeled by a negative temperature coefficient (NTCR) as: $R_T = R_0\,[1 - (\alpha_0 + \Delta\alpha_0(\sigma_\alpha))\,(T - T_0)]$, where R_0 and T_0 represent nominal resistance and temperature, respectively, α_0 is the nominal temperature coefficient, and $\Delta\alpha_0(\sigma_\alpha)$ denotes a stochastic variation of α_0 drawn from a normal distribution $N(0, \sigma_\alpha^2)$, capturing process-induced thermal fluctuations in a-IGZOs TFTs [79]. Thus, the resistor-crossbar output voltage: $V_z = \frac{g_1}{G_T}V_1 + \frac{g_2}{G_T}V_2 + \frac{g_3}{G_T}V_3 + \frac{g_b}{G_T}V_b = \sum_i V_i w_i + V_{\text{bias}} w_b$, models the effect of both thermal and manufacturing variations.

4.3.2. Thermal Stability Blocks

Operational Transconductance Amplifier (OTA) for Current Stability We propose an operational transconductance amplifier (OTA) for sigmoid and tanh AFs due to its ability to provide stable bias currents (I_{bias}),

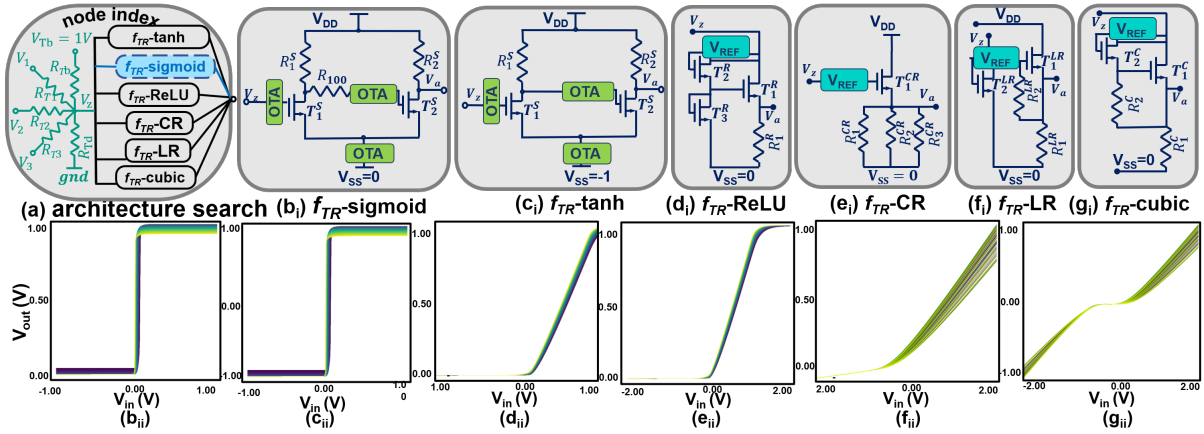


Figure 4.15.: (a) Proposed bespoke flexible neuron selected during *NAS*, (b-g) thermal-resilient (*TR*) activation circuits (f_{TR}-*sigmoid*, f_{TR}-*tanh*, f_{TR}-*ReLU*, f_{TR}-*clipped-ReLU*, f_{TR}-*Leaky-ReLU*, f_{TR}-*cubic*) with transfer curves within a temperature sweep from −20 °C to 185 °C.

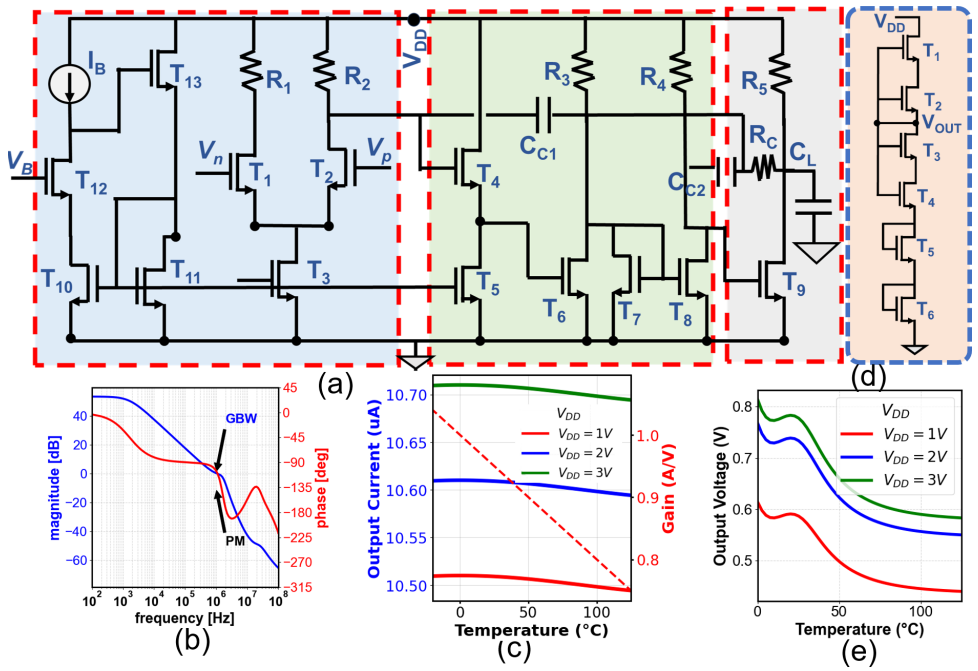


Figure 4.16.: Proposed three-stage OTA: (a) Circuit Schematic (b) Frequency response; (c) Current stability and gain w.r.t temperature; (d) Voltage reference circuit schematic; (e) voltage w.r.t temperature, from −20°C to 185°C.

thereby stabilizing transconductance (g_m) across temperature variations. By maintaining a consistent g_m, the OTA effectively compensates for temperature-induced variations in transistor mobility (μ) and threshold voltage (V_{th}), ensuring a stable V_{out}-V_{in}.

The proposed three-stage OTA (Figure 4.16) employs only n-type transistors, pull-up resistors, and capacitors. The pull-up resistors are selected such that $R_i \ll r_{oj}$, ensuring minimal voltage drop relative to the TFTs' intrinsic output resistance.

First and Second Stage The first stage consists of a differential pair (T_1, T_2) with a current source T_3, and load resistors R_1, R_2. A single-ended output is derived across R_2, given by: $V_1 = -\frac{g_{m1,2}}{2} R_2 (V_p - V_n)$. A unity-gain level shifter aligns the DC output of the differential stage to the gate of T_6, minimizing distortion. The second stage consists of a unity-gain inverter (T_6, T_7, R_3) followed by a common-source amplifier (T_8, R_4), producing: $V_2 = \frac{g_{m6}}{g_{m7}} \cdot g_{m8} R_4 V_1 \approx g_{m8} R_4 V_1$.

Third Stage and Biasing The third stage is a common-source amplifier comprising T_9 and R_5, yielding: $V_3 = -g_{m9}R_5V_2$. A Wilson current mirror (T_{10}–T_{12}) is used to provide stable current biasing. Transistor T_{12} equalizes V_{DS} across mirrored devices, minimizing early effect. The phase, gain and the current stability w.r.t thermal stability of the OTA are shown in Figure 4.16 (c)-(d). Aspect ratios are chosen to match V_{GS} across T_1, T_2, and T_{12}.

Bias Conditions A global bias current $I_B = 5\,\mu A$ sets $I_{D3} = 2I_B = 10\,\mu A$, and $I_{D5} = I_B$. Transistors T_6–T_8 are also biased at 5 µA, while T_9 is set to 20 µA to effectively drive a 50 pF load. All device dimensions are chosen to maintain the desired bias currents.

In f-NN-based analog computing, OTAs play a crucial role in stabilizing current and thereby thermal effects in AF circuits such as sigmoid and tanh, which exhibit smooth, continuous transitions rather than sharp threshold behaviors. Unlike AFs such as ReLU, which rely heavily on precise, fixed voltage references (V_{ref}), sigmoid and tanh depend primarily on current-based transconductance ($g_m = I_{\text{bias}}/V_{\text{eff}}$). In addition, the use of a separate voltage reference circuit is not required for these functions, as OTA-based current regulation alone provides sufficient stability under various thermal conditions.

Voltage Reference Circuit for Voltage Stability AFs such as ReLU, Clipped ReLU, and their variants exhibit distinct threshold behaviors that depend heavily on precise voltage thresholds. These functions thus require a stable reference voltage (V_{ref}) to maintain consistent and reliable activation points. Thermal variations significantly impact transistor parameters, such as threshold voltage (V_{th}) and carrier mobility (μ), thus shifting the threshold and inaccuracies in response.

To address this, we design a temperature-compensated voltage reference circuit that combines complementary-to-absolute-temperature (CTAT) and proportional-to-absolute-temperature (PTAT) behaviors which are well-known reference circuits for temperature compensation [81]. Specifically, CTAT and PTAT voltages are carefully balanced to produce a stable output voltage across a wide temperature range. The stability of the generated reference voltage ensures that AFs relying on fixed voltage thresholds remain unaffected by temperature-induced variations. The circuit consists of stacked NMOS transistors, where transistors T_1 and T_2 regulate the output voltage. The diode-connected transistors T_3 to T_6 introduce a temperature-dependent impedance, ensuring voltage stability. The output reference voltage is determined by:

$$V_{\text{TEMP}} = \underbrace{V_{TH_{123}} - V_{TH_{456}}}_{\text{CTAT}} + \underbrace{mV_T \ln \frac{C_{OX1}(W/L)_1}{C_{OX456}(W/L)_{456}}}_{\text{PTAT}}$$

The circuit operates on a wide supply voltage (V_{DD}) range from 0.5 V to 3 V and a temperature range till 185 °C. Thus, by choosing the correct stabilization, AFs remain robust against temperature, improving the reliability of f-NNs.

4.3.3. Thermal-resilient (TR) Learnable AF Blocks

Here, we detail *TR* learnable AFs and their transfer curves in Figure 4.15(b_i-g_i) and Figure 4.15(b_{ii}-g_{ii}) respectively. Thermal variations ($\mathcal{T}$) from −20 °C to 185 °C are considered, assuming component variation and enforcing a power constraint α to restrict high-power overloads. Self-heating effects were modeled by enabling the SHE flag in *FlexIC Gen3* Process Design Kit (PDK) and adding thermal resistances (R_{th}) and capacitances (C_{th}) coupled via temperature-dependent behavioral sources for resistors.

Flexible Sigmoid (f-sigmoid) Circuit The *f-sigmoid* is designed to emulate a smooth S-shaped, closely resembling the standard sigmoid function with the ground terminal biased at 0V. The circuit parameters (e.g., resistor values and transistor dimensions) are assumed to have minimal variations due to the high precision of the manufacturing process. However, temperature variation still influences the V_{th} and μ. To counteract these temperature-induced effects, an **OTA** is connected to the blocks as shown in Figure 4.16 (b). The OTA supplies a stable bias current (I_{bias}), ensuring that the effective transconductance ($g_m = I_{\text{bias}}/V_{\text{eff}}$)

remains consistent despite temperature fluctuations, thus maintaining a stable V_{out}-V_{in} for the *f-sigmoid* circuit. The mathematical equation of the circuit is: $V_{\text{a}} = \eta_1^{\text{S}} + \eta_2^{\text{S}} \cdot \text{sigmoid}((V_{\text{z}} - \eta_3^{\text{S}}) \cdot \eta_4^{\text{S}} \cdot \exp(-\frac{\alpha}{\mathcal{T}+10^{-8}}))$, with auxiliary parameters $\jmath^{\text{S}}$ defined by physical quantities $\mathrm{q}^{\text{S}} = [R^{\text{S}}, W^{\text{S}}, L^{\text{S}}, \alpha, \mathcal{T}]$.

Flexible Tanh (f-tanh) Circuit The *f-tanh* circuit employs balanced transistor-resistor configurations between a negative rail (e.g., $-1\,\text{V}$) and a positive supply (V_{DD}) to achieve a symmetric, smooth tanh-like transfer characteristic around $0\,\text{V}$. Resistor ratios and bias currents are optimized for stable operation across temperature variations. An **OTA** dynamically compensates bias currents to preserve performance despite thermal-induced parameter shifts. Unlike the *f-sigmoid*, a $100\,\Omega$ resistor is omitted to prevent unwanted offset shifts. The *f-tanh* is modeled as: $V_{\text{a}} = \eta_1^{\text{T}} + \eta_2^{\text{T}} \cdot \tanh((V_{\text{z}} - \eta_3^{\text{T}})\eta_4^{\text{T}} \cdot \exp(-\frac{\alpha}{\mathcal{T}+10^{-8}}))$, with parameters $\jmath^{\text{T}}$ determined by physical quantities $\mathrm{q}^{\text{T}} = [R^{\text{T}}, W^{\text{T}}, L^{\text{T}}, \alpha, \mathcal{T}]$.

Flexible ReLU (f-ReLU) Circuit In the *f-ReLU* circuit (Figure 4.15(d)), transistor T_1 acts as the primary switching element for positive input signals, while T_3 clamps negative inputs to ground. Transistor T_2 sets the bias and threshold voltage, driven by the proposed stable voltage reference (V_{ref}) to offset temperature-induced parameter variations, maintaining a stable ReLU threshold. The transfer characteristic is given by: $V_{\text{a}} = \eta_1^{\text{R}}(x - \eta_3^{\text{R}}) + \eta_2^{\text{R}} \cdot \text{softplus}(V_{\text{z}} - \eta_3^{\text{R}}, \alpha\mathcal{T}) + \alpha$, where $\text{softplus}(x, \mathcal{T}) = \frac{1}{\mathcal{T}} \log(1 + e^{\mathcal{T}\cdot(x-\eta_3^{\text{R}})})$ with auxiliary parameters $\jmath^{\text{R}}$ determined by $\mathrm{q}^{\text{R}} = [R^{\text{R}}, W^{\text{R}}, L^{\text{R}}, \alpha, \mathcal{T}]$.

Flexible Clipped ReLU *(f-CR)* Circuit In a flexible *clipped ReLU (f-CR)*, negative inputs are driven to zero, while positive inputs follow the input voltage until reaching a hardware-defined maximum threshold. Beyond this threshold, the output saturates, effectively clipping the signal. Also, a stable **reference voltage (V_{ref})** is connected to the transistor that governs the clip level, the circuit maintains a consistent upper bound over thermal variations, combining the rectifying behavior of ReLU with a robust hardware clamp. The mathematical model of the *f-CR* follows the given equation:

$$V_{\text{a}} = \begin{cases} \eta_1^{CR} + \eta_5^{CR} \cdot \tanh\Big(\mathcal{T} \cdot (V_{\text{z}} - \eta_3^{CR})\Big), & V_{\text{z}} < \eta_3^{CR}, \\ \eta_1^{CR} + \dfrac{\eta_2^{CR} - \eta_1^{CR}}{\eta_4^{CR} - \eta_3^{CR}} \cdot (V_{\text{z}} - \eta_3^{CR}) + \eta_5^{CR} \cdot \tanh\Big(\alpha\mathcal{T} \cdot (V_{\text{z}} - \eta_3^{CR})\Big), & \eta_3^{CR} \le V_{\text{z}} \le \eta_4^{CR}, \\ \eta_2^{CR} + \eta_5^{CR} \cdot \tanh\Big(\alpha\mathcal{T} \cdot (V_{\text{z}} - \eta_4^{CR})\Big), & V_{\text{z}} > \eta_4^{CR}. \end{cases}$$

where $\jmath^{CR}$ depend on physical values $\mathrm{q}^{CR} = [R^{CR}, W^{CR}, L^{CR}, \alpha, \mathcal{T}]$.

Flexible Leaky ReLU (f-LR) Circuit In this *f-LR* circuit, diode-connected transistor T_2 provides a conduction path for negative inputs, creating a non-zero slope instead of clamping the output strictly to ground. This ensures that when $V_{\text{in}} < 0$, the output follows a fraction of the negative voltage rather than dropping to zero. T_1 governs the main rectification for positive inputs, passing the signal to the output when $V_{\text{in}} \ge 0$. The resistor network (e.g., R_1, R_2) defines the exact slope of negative region and the bias conditions for both transistors. As temperature compensation is required, a stable **voltage reference (V_{ref})** is connected to the gate node of T_1, ensuring the transition remains consistent across varying thermal conditions. The mathematical expression is given by:

$$V_{\text{a}} = \eta_1^{LR} \cdot (x - \eta_3^{LR}) + \eta_2^{LR} \cdot \text{LReLU}(V_{\text{z}} - \eta_3^{LR}, \alpha\mathcal{T}) + \alpha.$$

$$\text{LReLU}(x, \eta_5^{LR}) = \begin{cases} x, & \text{if } x \ge 0, \\ \eta_5^{LR} \cdot x, & \text{if } x < 0. \end{cases}$$

where $\jmath^{LR} = [\eta_1^{LR}, \eta_2^{LR}, \eta_3^{LR}]$ are the auxiliary parameters determined by $\mathrm{q}^{\text{T}} = [R^{LR}, W^{LR}, L^{LR}, \alpha, \mathcal{T}]$ in the circuit.

Algorithm 4 ThermoNAS Evolutionary Algorithm

Require: $\mathcal{D} = \{(\mathrm{x}, \mathrm{y})\}$
Require: N (population size)
Require: $O(\cdot)$ (objective function)
Require: selection$(\cdot)$ (parent selection)
Require: adjust$(\cdot)$ (species size calculator)
Require: $\mathcal{K}$ (candidate parents for crossover)
Require: $\mathcal{K}_1$ (species patience)
Require: $\mathcal{K}_2$ (protected species)
Require: $\mathcal{K}_3$ (evolution patience)

Initialize population $\mathcal{P}$ with N genomes
$stop \leftarrow$ False
$\mathcal{S} \leftarrow \emptyset$
$\mathcal{F} \leftarrow \emptyset$
$\mathcal{F}_S \leftarrow \emptyset$
while $stop$ = False **do**
    $\mathcal{S} \leftarrow$ speciation$(\mathcal{P})$
    $(\mathcal{F}, \mathcal{F}_S) \leftarrow O(\mathcal{S}, \mathcal{D})$
    $\mathcal{N}_S \leftarrow$ adjust$(\mathcal{F}_S)$
    for all $s \in \mathcal{S}$ **do**
        if NoImprove$(\mathcal{F}_S(s), \mathcal{K}_1)$ **then**
            if $\neg$ IsBest$(s, \mathcal{K}_2)$ **then**
                Mark s as extinct
        else
            for $n \leftarrow 1$ **to** $\mathcal{N}_S(s)$ **do**
                $p_1, p_2 \leftarrow selection(s, \mathcal{K})$ ▷ Select parents based on fitness
                $o_n \leftarrow crossover(p_1, p_2)$ ▷ Crossover to produce offspring
                $o_n \leftarrow mutation(o_n)$ ▷ Mutation to introduce diversity
                ▷ Apply thermal-aware mutation $o_n \leftarrow thermal\_mutation(o_n, \mathcal{T}, \alpha)$ ▷ Adjust offspring with thermal resilience
                $s \leftarrow \{o_1, o_2, \ldots, o_{\mathcal{N}_S(s)}\}$
    if NoImprove$(\max(\mathcal{F}), \mathcal{K}_3)$ **then**
        $stop \leftarrow$ True

Flexible Cubic (f-C) Circuit The cubic activation (*f-cubic*) circuit employs transistors (T_1, T_2) and resistor networks (R_1, R_2) configured to produce an x^3-type response. Transistor conduction paths and resistor ratios define the polynomial output, while a **voltage reference ($V_{\mathbf{ref}}$)** maintains thermal stability by compensating temperature-induced parameter shifts. Mathematically: $V_{\mathrm{a}} = \eta_1^{\mathrm{C}}(x - \eta_3^{\mathrm{C}}) + \eta_2^{\mathrm{C}} \cdot \mathrm{softplus}(x - \eta_3^{\mathrm{C}}, \alpha\mathcal{T}) + \eta_5^{\mathrm{C}}(x - \eta_3^{\mathrm{C}})^3 + \alpha$, with parameters J^{C} derived from circuit parameters $\mathrm{q}^{\mathrm{C}} = [R^{\mathrm{C}}, W^{\mathrm{C}}, L^{\mathrm{C}}, \alpha, \mathcal{T}]$.

Note: All ReLU-family AF circuits employ a source degeneration resistor to stabilize gain, linearize transistor response, mitigate device variations and self-heating effects, and compensate thermal drift, thus ensuring consistent performance.

4.3.4. Thermal-aware Neural Architecture Search (NAS)

We propose a NEAT-inspired *EA* [25] to design thermal-resilient (*TR*) f-NN. Each AF and crossbar are modeled with thermal parameters $\mathcal{T}$ and α, capturing variations in device characteristics such as V_{th}, g_m, mobility, device aging and power constraints. Integrating α,$\mathcal{T}$ into the circuit models in section 6.1 ensures joint design of circuit topology and thermal-aware parameters, significantly enhancing robustness

and classification accuracy compared to methods that do not account for these effects. The training framework is shown in Algorithm 4[4].

4.3.4.1. Encoding, Crossover, and Mutation with Thermal Effects

Encoding Our encoding uses two gene types: **node genes**, showing neurons with unique indices, adjustable resistor parameters (R_{T_b}, R_{T_d}), AFs, power constraint (α) and thermal parameter ($\mathcal{T}$); and **edge genes**, defining directional neuron connections with learnable resistances and active/inactive neurons. These sets of genes mutate through evolution, forming **genome** and collectively form a **population** $\mathcal{P}$.

Crossover During crossover (Algorithm 3, line 9-11), offspring directly inherit unique genes from the fitter parent and randomly inherit common genes (matched by global indices) from both parents, favoring the fitter one. Thermal variation parameters ($\mathcal{T}$) and power constraint (α) ensure that the offspring retain thermal stability.

Mutation Mutation (Algorithm 3, line 12),introduces architectural and parameter refinements at two levels: genome-level (adding/removing nodes or edges initialized with thermal parameters $\mathcal{T}$ and power constraint (α) to ensure temperature stability) and gene-level (fine-tuning resistances and connection states via stochastic adjustments). Structural mutations preserve circuit functionality by matching original node outputs, while parameter mutations apply small variations respecting power constraints (α) to restrict heat build-up.

4.3.5. Thermal-resilient (TR) NAS Training

Thermal-resilient training is essential for f-NN, which are vulnerable to parameter drift under temperature fluctuations. We propose a *NAS* framework explicitly modeling circuit parameters (θ, q) as temperature-dependent random variables. The thermal-aware training objective is defined as the expected loss over these variations: $\mathcal{L} = \mathbb{E}_{\boldsymbol{\varepsilon}_\theta, \boldsymbol{\varepsilon}_q}[L(\theta_0 \odot \boldsymbol{\varepsilon}_\theta, \mathrm{q}_0 \odot \boldsymbol{\varepsilon}_q, \mathcal{D}) + \alpha \cdot \mathcal{T}(\theta, \mathrm{q})] = \int_{\boldsymbol{\varepsilon}_\theta} \int_{\boldsymbol{\varepsilon}_q} [L(\theta_0 \odot \boldsymbol{\varepsilon}_\theta, \mathrm{q}_0 \odot \boldsymbol{\varepsilon}_q, \mathcal{D}) + \alpha \cdot \mathcal{T}(\theta, \mathrm{q})] p(\boldsymbol{\varepsilon}_\theta) p(\boldsymbol{\varepsilon}_q) \, \mathrm{d}\boldsymbol{\varepsilon}_q \, \mathrm{d}\boldsymbol{\varepsilon}_\theta$. approximated via Monte-Carlo (MC) sampling as:

$$L \approx \frac{1}{N} \sum_{n=1}^{N} \left[L\left(\theta_0 \odot \epsilon'_{\theta,n}, q_0 \odot \epsilon'_{q,n}, D\right) + \alpha \cdot \mathcal{T}(\theta, \mathrm{q}) \right] \tag{4.14}$$

with samples $\boldsymbol{\varepsilon}'_{\theta,n} \sim p(\boldsymbol{\varepsilon}_\theta)$ and $\boldsymbol{\varepsilon}'_{q,n} \sim p(\boldsymbol{\varepsilon}_q)$. Here, the factor α enforces thermal stability, while $\mathcal{T}$ penalizes thermal deviations. This objective guides the *NAS* toward robust architectures balancing accuracy and thermal resilience. To simplify optimization, we apply reparameterization [56], expressing parameters as $\theta = \theta_0 \odot \boldsymbol{\varepsilon}_\theta$ and $\mathrm{q} = \mathrm{q}_0 \odot \boldsymbol{\varepsilon}_q$, where $\boldsymbol{\varepsilon}$ are random samples. The integral is approximated via MC sampling, and the fitness function is defined as $f(\mathbf{x}) = \gamma \cdot \mathrm{ACC} - (1 - \gamma) \cdot \mathcal{L}$, where $\gamma \in [0, 1]$ is a hyperparameter controlling the trade-off between accuracy (ACC) and thermal-aware loss ($\mathcal{L}$).

4.3.6. Evaluation

To assess the effectiveness of the proposed method, the algorithm was implemented in PyTorch and evaluated on 13 benchmark datasets commonly utilized in prior studies [202], [220]. SPICE simulations of AFs from Figure 4.15(b)-(f) were conducted in Cadence Virtuoso using Pragmatic *FlexIC* PDK [271], sweeping temperatures from $-20\,^{\circ}\mathrm{C}$ to $185\,^{\circ}\mathrm{C}$.

4.3.7. Experiment

Initialization Drawing insights from other works on *EA* and guided by a series of preliminary trials, we have initialized the network topologies for all datasets as unconnected networks, which consist solely of

[4] https://github.com/orgs/Reliable-Circuit-Design-and-Computing/

Table 4.9.: Accuracy comparison between baseline [259]: nominal and worst-case temperature variation; and using *TR* AF cells, *EA*-based training with three AF cell types: thermal-unresilient (*TU*), thermal-resilient (*TR*), and mixed version across 13 benchmarks.

Accuracy (%)	**Acu.**	**Bal.**	**Breast.**	**Cardi.**	**En(y1)**	**En(y2)**	**Iris**	**Mamm.**	**Pen.**	**Seed.**	**Tic.**	**Ver(2cl.)**	**Ver(3cl.)**	**Average**
Nominal[1]	100.0	90.2	95.1	86.8	90.5	89.4	96.5	78.8	53.4	88.1	100.0	82.1	81.1	**87.0**
Worst-case[2]	51.5	10.9	69.3	73.3	42.6	50.5	31.7	44.6	3.0	44.6	35.6	62.4	42.6	**43.2**
TR[3]	100.0	70.2	95.3	80.3	78.7	86.6	97.7	78.3	38.6	90.2	77.3	82.0	76.6	**80.9**
EA_TU[4]	60.9	7.5	65.0	79.1	47.6	47.6	48.8	55.9	10.0	38.7	65.2	69.6	68.5	**51.1**
EA_TR[5]	100.0	81.6	97.2	81.4	81.9	91.3	100.0	83.5	48.8	92.7	95.0	85.9	98.3	**87.5**
*EA*_Mixed[6]	98.0	79.9	95.2	79.7	80.2	89.4	98.0	81.8	47.8	90.8	93.1	78.6	75.6	**83.7**

[1] Nominal: Baseline accuracy under nominal temperature, [2] Worst-case: Baseline accuracy under worst-case temperature, [3] *TR*:Thermal-resilient [4] *EA_TU*: Evolutionary algorithm Thermal-unresilient, [5] *EA_TR*: Evolutionary algorithm Thermal-resilient [6] *EA*_Mixed: Evolutionary algorithm Mixed (*TU* and *TR*).

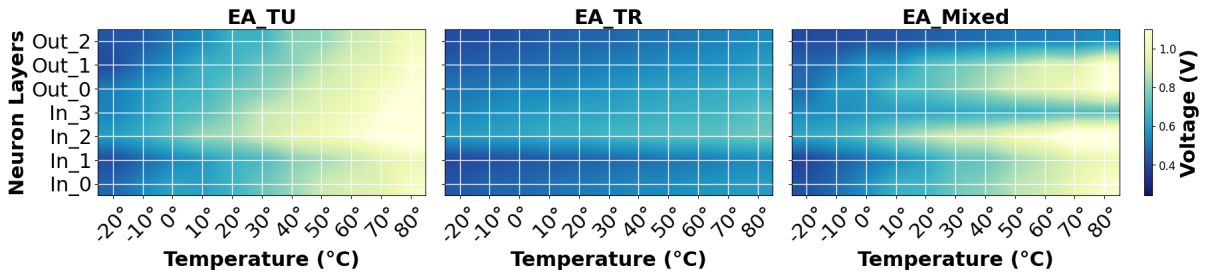


Figure 4.17.: Gradual voltage response of input-output neuron layers across temperature for *EA_TU*, *EA_TR* and *EA*_Mixed designs for Iris dataset. The color intensity reflects voltage levels, showing the thermal stability among designs.

nodes corresponding to the number of outputs, featuring only $\#output$ nodes. The population for these experiments is robustly set at 1000 individuals. Each node is initialized to have a random AFs circuit among the given design. In terms of the mutation mechanisms, we have defined specific probabilities for the genetic alterations within the network structures: the probability of introducing either a new node or a new connection is set at a substantial rate of 0.8, while the probability for the deletion of a node or a connection is comparatively lower, at 0.2. Moreover, there exists a 0.1 chance that any given edge within the network will toggle its state from enabled to disabled, or vice versa, as part of the mutation process. Moreover, the mutation rate of changing selected AFs circuit is 0.1.

In our framework, thermal fluctuations are modeled using a static uniform distribution to capture temperature-induced variations in device parameters. Specifically, we assume $\boldsymbol{\varepsilon}_\theta \sim \mathcal{U}[1-\epsilon, 1+\epsilon]$ and $\boldsymbol{\varepsilon}_q \sim \mathcal{U}[1-\epsilon, 1+\epsilon]$ in a uniformly defined range, where ϵ represents the magnitude of thermal variation. Thermal fluctuations, which stem from ambient or junction temperature changes, lead to variations in material properties over a limited range [227], [239]. Self-heating effects are included at both transistors and resistors through SPICE simulations, incorporating thermal resistances and capacitances coupled to temperature-dependent power dissipation.For the MC estimation of the expected loss, we employ $N = 50$ samples, which our experiments have shown to be sufficient for precise approximation.

Training Evolutionary training utilized full-batch optimization, terminating after 100 generations without validation improvement. Experiments were repeated ten times per configuration using random seeds (1-10) to ensure statistical reliability.

Baseline The baseline followed the methodology of [259] without thermal variations ($\epsilon = 0$), using Adam optimizer with default parameters, an initial learning rate of 0.1, and halving upon 100-epoch patience. Training terminated after the learning rate was reduced 10 times.

Post-Training Evaluation Optimal circuits selected based on minimal validation loss were evaluated in test sets under identical thermal conditions, using 100 MC samples ($N_{\text{test}} = 100$). Results including accuracy, area, and power, are summarized in Table 4.9 and Table 4.10. **Note:** Due to the bespoke nature of flexible circuits and technology limitations, model parameters are hardcoded post-deployment, constraining incremental or online parameter updates.

4.3.8. Result

The accuracy results summarized in Table 4.9 clearly illustrate the significant impact of thermal variations on f-NNs. Under nominal (ideal) conditions, the f-NNs achieve an average accuracy of $\approx$ 87.0%. However, under worst-case thermal variations, this accuracy (Table 4.10 *(col. 3)* sharply reduces to about 43.2%, showing a performance degradation of $\approx$ 50.3%. This shows that f-NNs are highly susceptible to temperature variations, highlighting the need for thermal resilience in real scenarios.

To mitigate this accuracy loss, we evaluate the use of thermal-resilient AFs in f-NNs. By integrating these TRs AFs, the f-NNs successfully regained most of accuracy loss, resulting in an average accuracy of

80.9%, thus recovering ≈ 87% of the accuracy initially lost due to thermal variations. However, as shown in Table 4.10, achieving this thermal resilience comes at notable hardware costs, with an area overhead of ≈ 89% and a power overhead of about 70.1%, showing a significant trade-off between accuracy and hardware efficiency.

Table 4.10.: Comparison of Average Accuracy, Area and Power w.r.t Different Cross-Layer Approaches

Metrics	Nominal	Worstcase	*TR*	*EA_TU*	*EA_TR*	*EA*_Mixed
Accuracy (%)	87.0	43.2	80.9	51.1	87.5	83.7
Area (sq. µm)	1.585	1.585	3.010	1.056	2.868	2.294
Power (mW)	1.379	1.612	2.743	0.919	2.240	2.397

To overcome hardware limitations, we further explore algorithmic approaches using *EA*s combined with *NAS*. Initially, applying these algorithmic optimizations to thermal-unresilient (*TU*) AFs alone (*EA_TU*) resulted in limited accuracy gains, improving from worst-case accuracy of 43.2% to only 51.1%, which remained significantly below nominal accuracy. We therefore adopt *EA*-based optimization directly on thermal-resilient AFs (*EA_TR*) to address this. This combined approach improve performance, achieving an average accuracy of 87.5%, thus fully recovering the accuracy loss due to thermal variations and slightly exceeding the nominal accuracy. However, this algorithmic strategy results in reduced area (≈ 5%) and power (≈ 18.3%) compared to only *TR* circuits but significantly higher area and power overhead of ≈ 81% and ≈ 39%, respectively, compared to nominal case.

Finally, to achieve an optimal balance between accuracy and hardware costs, we introduce a mixed AF strategy (*EA*_Mixed), selectively using both thermal unresilient and thermal-resilient AFs, optimized by *EA*-based *NAS*. This selective approach achieve an accuracy of 83.7%, incurring only a accuracy drop of about 4.3% compared to the *EA_TR* approach. However, it results in significant hardware efficiency, further reducing area overhead by ≈ 20% while maintaining a moderate power overhead of only about ≈ 7% compared to *EA_TR*. Despite the longer training duration for *EA* based approaches (≈ $8hr$ on average) compared to gradient (≈ $20min$), the considerable gains in accuracy, power efficiency, and reduced area footprints justify the additional training cost. This plot in Figure 4.17 further shows how output voltages of neurons shift with temperature. TRs cells maintain stable voltages, while unresilient cells show larger variations, especially in higher temperature ranges. While this work shows thermal-aware design for a-IGZOs TFT-based circuits, the methodology, i.e., evolutionary optimization of *NAS*, AFs, and circuit parameters, is generally technology-agnostic and suitable for various small scale NNs.

4.4. Chapter Summary

This chapter presented an architecture-aware design methodology for robust and efficient p-NNs implemented in emerging flexible and PE. The proposed framework jointly considers circuit-level nonidealities, neural architecture, and learning strategies to address key challenges arising from process variation, area constraints, and thermal instability.

First, we introduced learnable printed AFs and demonstrated how their circuit parameters can be co-optimized with network topology using an *EA*-based *NAS*. By explicitly modeling printing-induced variations within the training objective, the proposed variation-aware *NAS* significantly improved robustness and reduced post-mapping accuracy degradation compared to conventional gradient-based approaches.

Second, the framework was extended toward compact p-NNs through NEAS, which explicitly incorporates circuit area and power into the optimization process. By jointly evolving network topology and physical implementation parameters, NEAS achieved superior accuracy-area-power trade-offs compared to state-of-the-art pruning-based baselines, enabling highly compact printed neural implementations.

Finally, this chapter addressed thermal reliability challenges in flexible a-IGZO TFT-based f-NN hardware through Thermo-*NAS*. By integrating thermal-resilient circuit primitives and temperature-aware modeling into the evolutionary search, the proposed approach effectively mitigated severe accuracy degradation under wide temperature variations. A mixed activation strategy further balanced accuracy recovery and hardware overhead.

Overall, this chapter demonstrated that evolutionary, circuit-aware *NAS* provides a scalable and technology-agnostic methodology for designing p-NNs that are robust to process variability, compact in physical footprint, and resilient to thermal stress, thereby enabling reliable edge intelligence on flexible and printed hardware platforms.

5. Reliability, Endurance, and Test

PE promises extremely low-cost, mechanically flexible and customizable hardware, but these benefits come at the price of reduced reliability and limited testability. Process variations, parametric drift over time, and catastrophic manufacturing defects are all significantly more pronounced than in conventional silicon CMOS. In the *p*-NNs and other bespoke circuits developed in this thesis, these non-idealities directly affect the correctness of computations and the lifetime of the system.

This chapter brings together several contributions that address reliability and testability across different layers of the stack: circuit-level AEC for unreliable unary-encoded bitstreams, fault sensitivity analysis of printed classifiers, adaptive self-healing through fault-aware training and redundant circuit structures, and finally manufacturing test and diagnostics through delay-based and gradient-based test generation. Together, they form an end-to-end view of how to design, characterize, and test reliable printed systems at scale.

5.1. Fault Sensitivity Analysis of *p*-NN Classifiers

To assess the fault sensitivity of analog printed NN classifiers (a-*p*-NNs) circuits, a Monte Carlo-based fault injection methodology is employed. The first step in this work involves defining potential fault types, such as resistor open-short or transistor open or gate-drain (G-D) / gate-source (G-S) / drain-source (D-S) shorts-open, specifying their characteristic behaviors. For simulating the faults in the resistor crossbars, we utilized an analytical expression, i.e.,

$$\tilde{\mathrm{g}} = M_g \odot \mathrm{g}, \tag{5.1}$$

where g collects the conductance values and M_g functions as a mask to emulate by different faults by multiplying the conductances with 1 for fault-free, 0 for open, and ∞ for short. Moreover, $\odot$ denotes an element-wise multiplication, and therefore, $\tilde{\mathrm{g}}$ indicates the conductances with fault injection.

However, the impact of the fault in the nonlinear circuits, i.e., negative weight circuit and activation circuit, is a much more sophisticated mechanism. Even though there has been works on estimating the transfer characteristics of those nonlinear circuits through ML-based *surrogate nonlinear circuit* models [241], they can not be used for fault sensitivity analysis, they can only provide confident estimations when the component values are within a reasonable range. Therefore, we consider the nonlinear subcircuits as sub-systems and inject faults into different components. Thereafter, we conduct a SPICE simulation based on the pPDK to obtain the characteristic curves for the corresponding faults. Figure 5.1 (a) and Figure 5.1 (b) show the possible single fault that can occur in the negative weight circuit and ptanh circuit. In addition to being stuck at VSS/GND and stuck at supply voltage VDD, we also notice various unexpected faulty behaviors in the circuit which results in an abnormal rise in the negative voltage level when the resistor R_3 is kept open in Figure 5.1 (a) and R_2 in Figure 5.1 (b) respectively due to a discontinuity in the closed circuit connection, further disrupting the expected outputs. All these fault scenarios lead to the reduced classification accuracy of various datasets in the a-*p*-NN architecture and are therefore a critical concern.

Subsequently, we use Monte Carlo sampling to draw the fault components within a print a-*p*-NN. As the training framework of the a-*p*-NN is an ML-based model, we employ a top-down sampling strategy to keep a consistent fault rate for each component, i.e., we first draw N_l faults for each layer l with the probability

$$p(l) = \frac{N_l}{\sum_l N_l}, \forall l, \tag{5.2}$$

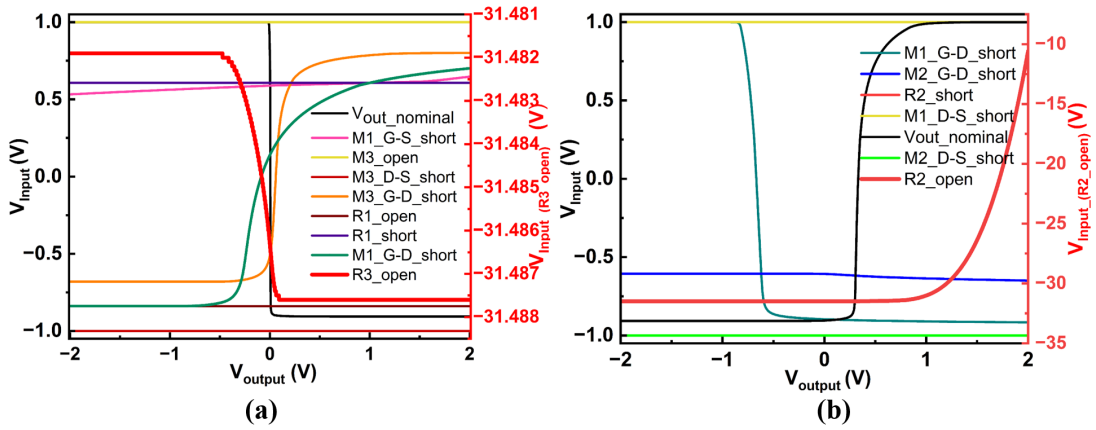


Figure 5.1.: SPICE simulated stuck-open and stuck-short faults in (a) negative-weight and (b) p-tanh activation circuit of a-*p*-NN hardware primitives.

with

$$N_l = 6N_l^N + 4N_l^A + N_l^R, \tag{5.3}$$

where N_l^N, N_l^A, and N_l^R refer to the number of negative weight circuits, ptanh activation circuits, and the number of resistors of the crossbar in the l-th layer. Afterward, each layer is requested to sample N_l faults for three circuit primitives proportional to their device counts. Note that, for each nonlinear circuit, we consider maximally one fault, justifying the ignorance of multiple faults occurring simultaneously in the same subcircuit.

5.1.1. Training approaches in analog-*p*-NN (a–*p*-NN) Classifier

Nominal Training We adopt nominal training as a design approach which is gradient-based training of a-*p*-NN [242]. It can efficiently train the parameters in a a-*p*-NN, fulfilling the constraints on the printed devices, e.g., limited printable resistances.

Variation-Aware Training In addition to the nominal training, variation-aware training takes the parametric faults (manufacturing errors) of printed components into account during the training [242] by modeling the fabrication through a stochastic variable. As it aims to improve the robustness against parametric faults, we tested the a-*p*-NNs from variation-aware training to analyze whether it can bring advantages for catastrophic faults.

Dropout Dropout is a common training method in deep learning [174]. As it randomly turns off some neurons/inputs, which is similar to the mechanism of a stuck-open, it is hypothesized that it can also improve the robustness of the circuits against catastrophic faults.

Variation-aware+Dropout Finally, a combined approach of variation-aware training and dropout is considered to comprehensively analyze their effectiveness in enhancing the performance of a-*p*-NN.

5.1.2. Fault Injection in digital-*p*-NN Classifier (d-*p*-NN)

Similar as explained in section 5.1, to assess a fault sensitivity simulation on the digital designs a Monte Carlo-based simulation is utilized. In this process, we obtain all the wires described in the post-synthesis netlist design, and randomly wires are selected. These obtained wires are the faulty wires of the circuit and are randomly stuck-at 0 or 1. Afterward, every defective wire is intentionally introduced into the gate-level netlist for each respective architecture. Then the gate-level simulation is performed to evaluate

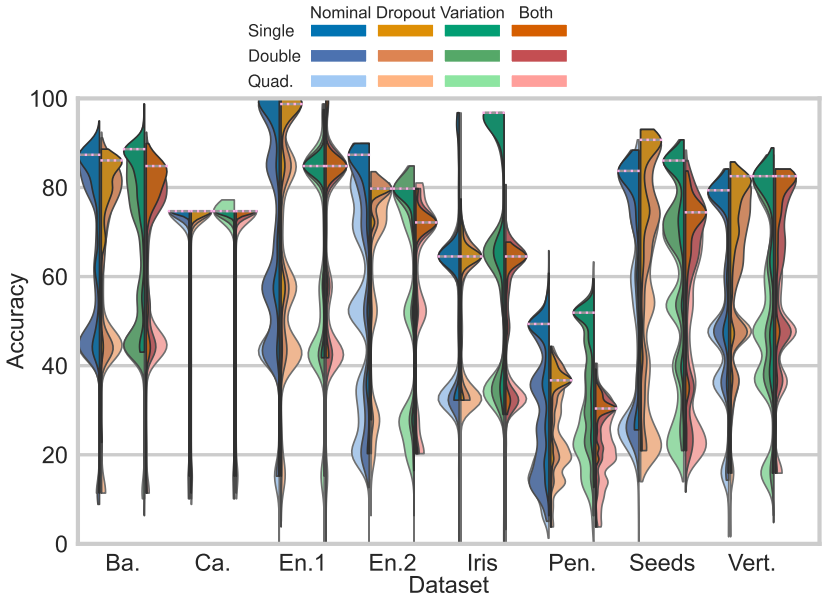


Figure 5.2.: Evaluation of various analog design approaches after single, double, and quadruple fault injection in a-*p*-NNs. The dotted line represents the fault-free accuracy.

the impact of these faults on the classification accuracy of the *p*-NN. Also, we simulate the entire test dataset on the fault-injected netlist to obtain the impact on the classification accuracy.

5.1.3. Architectures in d-*p*-NN Classifier

Different customization architecture We adopt two architectures of different customization levels; the conventional (i.e., generic) and the bespoke architecture. Both of these architectures use full fixed-point precision for their computations. The conventional architecture utilizes general-purpose multipliers with two operands; one for the input and one for the weights. On the other hand, the bespoke design paradigm utilizes multipliers specific for each weight, in the *p*-NN circuit. Subsequently, each multiplier produces a specific product based on the input for each weight within each neuron of the *p*-NN.

Different approximation tensity We consider three different bespoke approximate *p*-NN architectures with different approximation tensity. The first approximate *p*-NN architecture uses only power-of-2 (pow2) quantization of weights and thus, performs a multiplier-less inference. The second approximate *p*-NN architecture uses pow2 approximation and a fine-grain approximate accumulation (pow2+axAcc) [222]. The third *p*-NN architecture (axAll) [222], approximates all the components of the *p*-NN by using an argmax approximation with pow2+axAcc.

Dropout As mentioned before, dropout is a common training method in ML. This approach randomly nullifies some inputs during training, which in turn helps prevent over-fitting. After training, certain inputs have larger weights that dramatically change the outcome. Without dropout, the circuit is expected to be more prone to failure, if a stuck-at-fault occurs which could affect the outcome more.

5.1.4. Evaluation

While the previous section focused on correcting bit-level errors in unary-encoded outputs, reliability at the system level also depends critically on how neural-network classifiers implemented in PE respond to permanent faults. The ETS paper *Fault Sensitivity Analysis of Printed Bespoke Multilayer Perceptron Classifiers* investigates this question systematically for both analog and digital printed *p*-NNs, and across a broad set of datasets and training strategies.

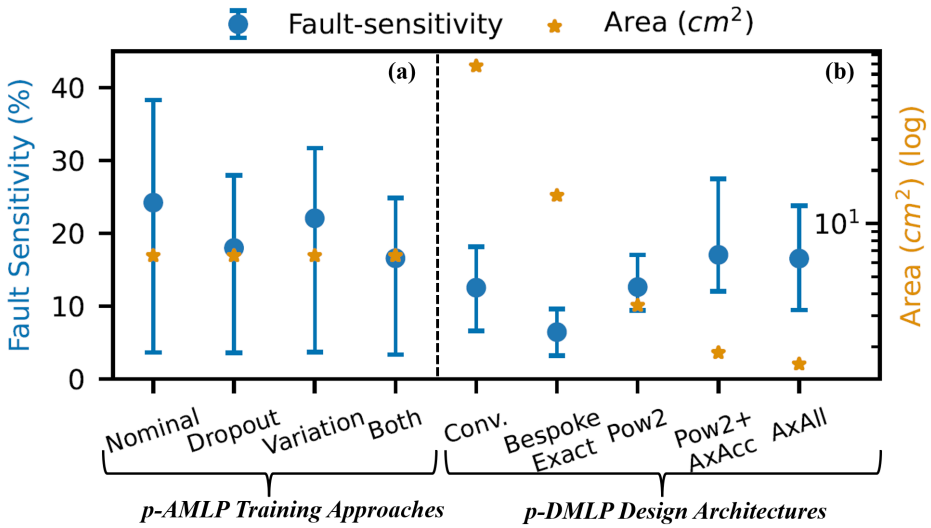


Figure 5.3.: Evaluation of fault sensitivity w.r.t (a) design approaches in a-*p*-NN (The area for a-*p*-NN is same in all training process) and (b) design architectures in d-*p*-NN. (Area is in logarithmic scale).

To analyze the fault sensitivity of both a-*p*-NN and d-*p*-NN, we conduct the following experiment. The code is available at GitHub repository[1].

5.1.5. Experiment

All analog hardware primitives were designed based on the n-EGT pPDK. We used Cadence Virtuoso[2] tool to simulate fault injection (Figure 5.1) in SPICE and trained the a-*p*-NNs afterward.

Digital circuits are synthesized using Synopsys Design Compiler S-2021.06 with the printed n-EGT library [158]. For simulation and power analysis, VCS T-2022.06 and PrimeTime T-2022.03 are employed. Accuracy is reported on the test dataset, with synthesis adhering to directives in [222] to align delay values with typical PE performance [102]. The digital printed NN classifiers (d-*p*-NNs) architecture mirrors **power**, [169], and bespoke exact d-*p*-NNs circuits, as outlined in [169], utilizes 8-bit fixed point weights and 4-bit inputs.

Training Setup We employed eight preprocessed datasets, namely Balance Scale (Ba.), Cardiotocography (Ca.), Energy1 (En1.), Energy2 (En2.), Iris, Pendigits (Pen.), Seeds, and Vertebral 3 Columns (Vert.), from [59] whose task complexity match the target applications of PE, and normalized their inputs to $[0, 1]$ to simulate the electrical signals from sensors with limited range. To this, we randomly split datasets into training (60%), validation (20%), and test (20%) sets.

5.1.6. Result

a-*p*-NNs We utilized various non-linear components in the a-*p*-NN circuit primitives. However, their reliable functioning in analog circuits is highly susceptible to faults due to several factors like input variations, printing process variations, device-geometry, and variations in ink compositions and substrates. The accuracy distribution with single, double and quadruple fault injection of a-*p*-NNs on 8 benchmark datasets using various training approaches are shown in Figure 6.3. It is evident that none of the methods can effectively mitigate the impact of catastrophic faults, resulting in a significant reduction in classification accuracy when faults are introduced. Additionally, with the increasing number of faults injected, the classification accuracy is reduced continuously.

[1] https://github.com/PrintedElectronics/Fault_Sensitivity_Analysis
[2] https://www.cadence.com/en_US/home.html

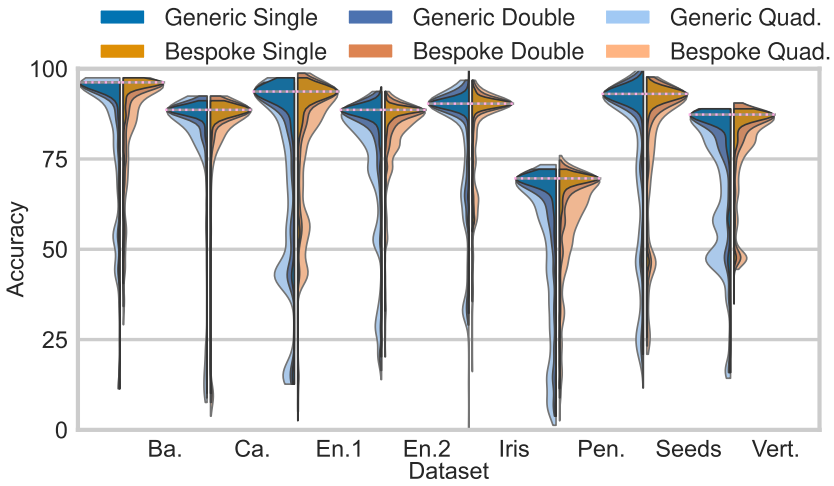


Figure 5.4.: Evaluation of conventional and bespoke architecture after single, double, and quadruple fault injection. The bespoke architecture is as in [169]. The dotted line represents the fault-free accuracy.

Both variation-aware training and dropout training can marginally contribute to robustness against fault and their effectiveness is dataset-dependent. However, variation-aware training stands out by achieving higher accuracy in fault-free testing scenarios, while dropout training yields lower accuracy under fault-free conditions. We speculate that, as variation-aware training introduces stochastic variables with continuous probabilistic distribution, the objective function was smoothed during training. As a result, the gradient can guide the training process more informatively. In contrast, as dropout introduces non-differentiability (turning-off vs. turning-on) into the training process, it creates difficulties for the gradient-based training process resulting in lower accuracies in fault-free cases.

Unexpectedly, the combination of both dropout and variation-aware training leads to even worse results than nominal training. In this regard, we argue that the introduction of both variation and dropout forces excessive perturbations to the parameters, so that the circuits are unable to produce promising accuracy while overcoming the large perturbations in the small optimization search space (solely 1 hidden layer with 3 neurons). Thus, a-*p*-NNs inherently operates with limited precision due to continuous voltage levels resulting in reduced classification accuracy for different training approaches.

Figure 5.3 depicts the summary of techniques used in this work, for both analog and digital *p*-NNs w.r.t. fault sensitivity and area. For each technique, Figure 5.3 presents the min-max range of fault sensitivity, the black dot represents the average of this range, while the yellow star is the area of the technique in cm^2. In this work, as fault sensitivity, we consider the average accuracy drop (classification miss) for each dataset and each fault injection type i.e., single, double, quadruple, normalized with the respective fault-free accuracy.

d-*p*-NNs Figure 5.4 depicts the accuracy distribution with single, double, and quadruple fault injection for both conventional and bespoke designs. The same distribution is shown for bespoke exact and approximate architectures in Figure 5.6. As discussed in paragraph 5.1.6, increasing the number of faults in the circuit results in a reduction of classification accuracy. In some cases, the approximate *p*-NN circuits achieve higher accuracy than their exact counterparts due to the benefits of approximation in terms of area, power gains, and improved generalization [223].

For different customization architectures, Figure 5.3 showcases that the bespoke architecture is more *FT* than the conventional model-agnostic architecture. As previously mentioned, the bespoke architecture uses fully customized multipliers that generate a specific product, while its conventional counterpart utilizes general-purpose multipliers with weight as one of its inputs. That means that if a fault occurs in the netlist of the conventional *p*-NN architecture, the multiplier will result in a different input's product. On the other hand, the bespoke netlist is more *FT* since the multiplier's netlist is dedicated to the specific value of the weight.

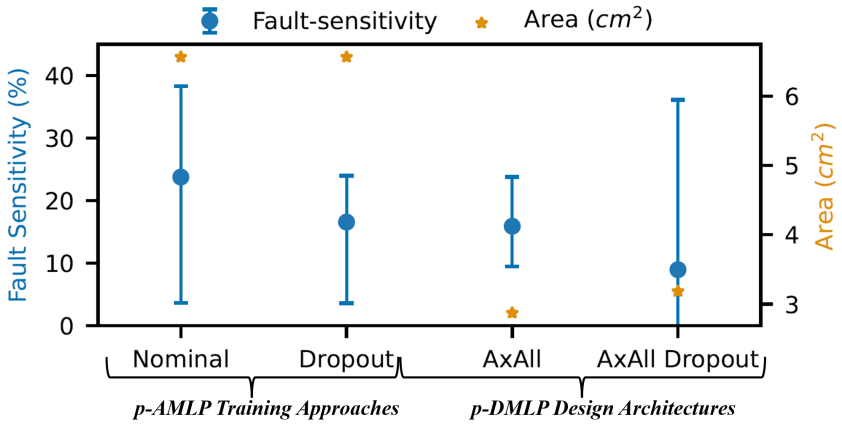


Figure 5.5.: Evaluation of fault sensitivity using dropout for both analog and digital *p-NN*. (The area for analog *p*-NN is same in all training process)

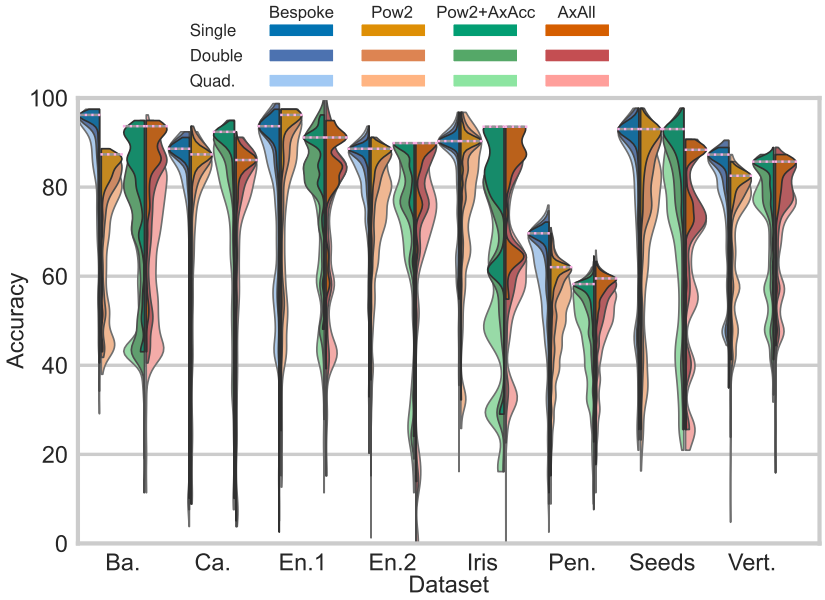


Figure 5.6.: Evaluation of approximate architecture with single, double, and quadruple fault injection. All the architectures follow [222]. The dotted line represents the fault-free accuracy.

Furthermore for different approximation tensity, Figure 5.3 showcases that pow2 is more *FT* than the other more fine-grain approximations. More precisely, while approximate circuits are generally assumed to be more fault tolerant than their exact counterpart [223], this is not the case for bespoke circuits as depicted in Figure 5.3. The approximate bespoke designs are more fault-sensitive than their bespoke exact counterparts. The aggressive fine-grain approximations in these designs result in significant area reduction, minimizing the gate count, meaning that any remaining redundancy is removed from the circuit and every gate is crucial for final classification. The decreasing area in Figure 5.3 demonstrates a trade-off between optimization intensity and fault sensitivity, emphasizing the criticality of even a simple permanent fault in highly optimized circuits.

Analyzing gate-level netlists post single fault injection, also reveals that errors near the output (argmax circuit) significantly degraded accuracy. A stuck-at-fault within the argmax circuit propagates to the output, causing one *p*-NN output bit to be stuck-at 0/1. Consequently, the classification output becomes confined to a specific subset of available *p*-NN classes, emphasizing the criticality of faults in proximity to the output for accuracy degradation.

Dropout Figure 5.5 shows the fault sensitivity for a-*p*-NN nominal, dropout, d-*p*-NN AxAll, and AxAll+dropout approaches. The a-*p*-NN operates on analog sensory inputs, and for a fair analog-digital area comparison, d-*p*-NN areas are reported with ADC areas included. The digital area is nearly half of its analog equivalent due to a more approximate design with all d-*p*-NN components approximated. Dropout during d-*p*-NN training alters model coefficients and circuits, affecting min-max fault-tolerance ranges on AxAll dropout. However, on average, AxAll d-*p*-NN dropout is more *FT* than without dropout during training. Overall, dropout increases the fault tolerance in both a-*p*-NNs and d-*p*-NNs on average.

Key Findings for Analog *p*-NNs The analysis revealed several important trends for analog printed *p*-NNs:

1. **High sensitivity to catastrophic faults.** Even a small number of stuck-open or stuck-short faults in critical locations (e.g., within the activation function circuits or important crossbar weights) can cause steep drops in accuracy. No training strategy could fully compensate for severe catastrophic faults once they occur.

2. **Variation-aware training improves nominal accuracy and mild-fault robustness.** When only parametric variations are present or when fault rates are low, variation-aware training yields the highest classification accuracy, as the network learns to operate in the presence of noisy device parameters.

3. **Dropout introduces robustness at the cost of baseline accuracy.** Dropout training sacrifices some accuracy in the fault-free case but generally improves average accuracy across fault scenarios, since the network learns to avoid over-reliance on any single neuron or path.

4. **Combining variation-aware training and dropout does not always help.** The combined scheme sometimes underperforms pure variation-aware training, suggesting a complex interaction between noise-aware optimization and structural redundancy.

These results highlight a fundamental tension: analog printed classifiers can be made tolerant to parametric spread, but catastrophic faults remain a major threat if not addressed at the circuit or architectural level.

Key Findings for Digital *p*-NNs For the digital implementations, the study compared generic, bespoke, and approximate designs, each subjected to random stuck-at faults. The main conclusions are:

- **Bespoke exact designs are relatively robust.** The full-precision bespoke d-*p*-NNs often show better fault tolerance than their approximate counterparts, because they retain more representational redundancy and do not aggressively compress internal arithmetic.

- **Approximate circuits increase fault sensitivity.** Architectures such as *pow2+axAcc* and *axAll*, which rely heavily on approximate operations to save area and power, tend to be more sensitive to stuck-at faults: the same structural redundancy that is removed for efficiency would have helped absorb faults.

- **Dropout helps both analog and digital.** Applying dropout during training improves average accuracy under faults for digital classifiers as well, confirming that redundancy at the algorithmic level is beneficial across implementation styles.

Implications This fault sensitivity study serves two purposes in the context of the thesis:

1. It provides a quantitative baseline for how fragile current printed analog and digital *p*-NN classifiers are to realistic PE fault models, motivating the self-healing approaches developed later.

2. It shows that training alone cannot guarantee fault tolerance; hardware-level measures (redundant structures, error correction) and test strategies (to screen grossly defective devices) are essential.

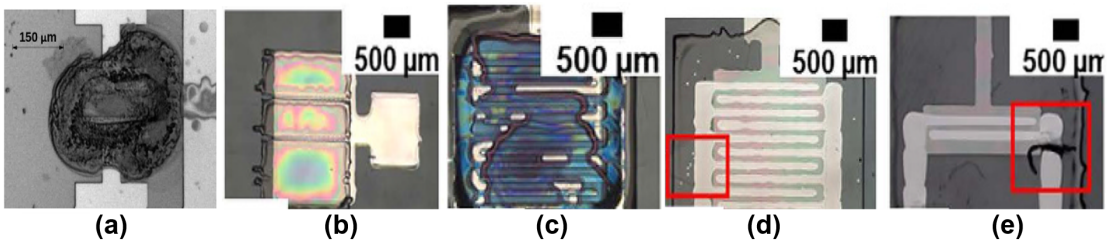


Figure 5.7.: (a) Transistor with exploded electrolyte (b) inkjet-nozzle clog (c) inhomogeneous layer formed (d) satellite drops of ink (e) short circuit between S-D (sourced from [98], [187]).

The PRINT-SAFE framework in the next section builds directly on these insights by combining redundant circuit structures with fault-aware training to achieve adaptive endurance.

5.2. PRINT-SAFE: Adaptive Endurance and Self-Healing

The primary benefit of additive printing processes is the significant cost reduction achieved through maskless manufacturing. However, these processes come with reduced process control, leading to a higher variability and defect rate during both manufacturing and runtime. Common defect mechanisms in PE include misprints, incomplete traces, broken or misaligned connections, crossovers, and material degradation [98], which can cause open-short circuits, unpredictable electrical behavior or even complete non-functionality of the circuits as shown in Figure 5.7. In addition, issues such as ink smudging, delamination, or void formation can further reduce the reliability of printed circuits [276]. Given these challenges, it is very critical to ensure high manufacturing yield while maintaining reliability in PE during in-field operation. Furthermore, the bespoke architectures used in *p*-NNs [206], [220], [241] introduce additional layers of complexity, further increasing vulnerability to faults[26]. Therefore, cost-effective fault-endurant custom hardwired (bespoke) architectures are needed to address these inherent defects, ensuring that printed circuits continue to function reliably despite the presence of manufacturing defects [187], [247], allowing the system to remain operational, thereby maintaining its overall quality and robustness. Although significant efforts have been made to implement various *p*-NNs [241], [253], [254], very few studies have focused on fault modeling and *FT FT* design for printed ANN architecture [254]. In this context, our work focuses on the X-design (co-design) and modeling of *FT* nonlinear activation circuits, fault injection, and FAT in the NN architecture.

In short, the contributions of this work are:

1. This work, for the first time, proposes the X-design (co-design) of ***FT* printed bespoke** nonlinear transformation circuits at both the circuit and the algorithmic level.

2. A gradient-based FAT approach is introduced to optimize *p*-NNs in a bespoke manner. Using *Gumbel-Softmax distribution*, the most *FT* AF is selected for each neuron, allowing differentiable backpropagation to dynamically adapt to printing defects.

5.2.1. Analog Printed Neural Networks (*p*-NNs)

Neuromorphic computing[3], driven by advancements in artificial intelligence, offers a powerful approach for solving complex tasks, with small neural networks enabled by *p*-NNs being especially cost-effective. These *p*-NNs are ideal for real-time sensor data processing in resource-constrained applications like wearable sensors, flexible displays, and smart bandages [184], [204], [219], where they outperform conventional silicon-based hardware. By utilizing basic operations like weighted-sum and nonlinear

[3] Here, in these works, the term "neuromorphic" is used broadly to indicate analog computational circuits inspired by neural network architectures, using nonlinear AFs and resistor crossbar arrays for parallel computation.

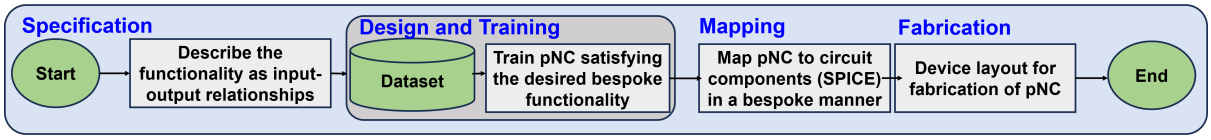


Figure 5.8.: On-demand design and fabrication given a specification of a desired functionality realized through training a *p*-NN. The derived design can be readily fabricated through the on-demand fabrication capabilities of inkjet-PE [202]

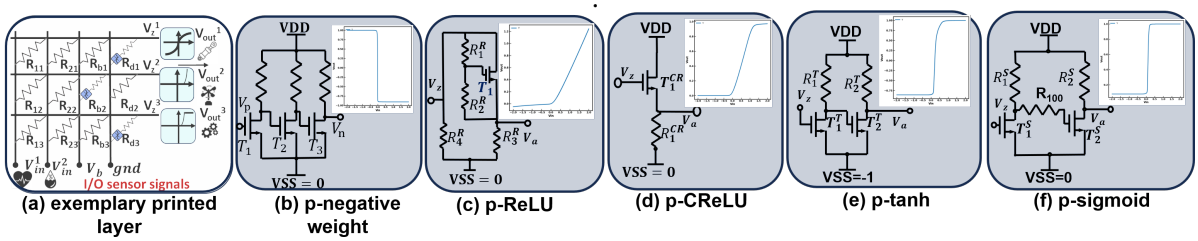


Figure 5.9.: Schematic of printed NN computing circuits. (a) example of a 3-input and 3-output printed layer based on crossbar array; (b) p-negative weight (c) p-ReLU (d) p-CReLU (e) p-tanh (f) p-sigmoid AF circuits[241]

activation, *p*-NNs offer efficient, on-demand computing for various PE target applications. Figure 5.8 shows the existing flowchart of a *p*-NN as available in the literature.

5.2.1.1. Hardware Primitives

Figure 5.9 (a) illustrates the circuit schematics of a neuron in *p*-NN. Some negative weight circuits are also incorporated in case required. Figure 5.9(b) and Figure 5.9(c)-(f) show the specific schematics of the negative weight circuit and the printed activation circuits [202]. The cross-sectional layout of the primitives are shown in Figure 4.8. In the following, we will provide a detailed introduction of these circuit primitives.

Resistor crossbars The resistor crossbar circuit, shown on the left side of Figure 5.9 (a), follows Ohm's Law and Kirchhoff's Laws to calculate the output voltage V_z as a weighted-sum of input voltages V_i, with the weights determined by the conductance ratios as shown in Equation 5.4.

$$V_z = \frac{g_1}{G}V_1 + \frac{g_2}{G}V_2 + \frac{g_3}{G}V_3 + \frac{g_b}{G}V_b, \tag{5.4}$$

where g_i signifies the conductances of the resistors R_i and G represents the aggregate conductance $\sum_j g_j + g_b + g_d$. This design enables the customization of conductances to achieve desired weights, analogous to training in ANNs.

Printed negative weight circuits Since the conductances in the crossbar resistor array can only represent positive weights, some resistors are paired with inverter-based circuits [243], as depicted in Figure 5.9(b), to enable negative weight representation.This setup allows for the emulation of multiplication with negative weights by inverting the input voltages V_i.

Printed activation circuits After passing through the crossbar, the signals can be processed by different printed activation circuits, as shown in Figure 5.9 (c-f), which emulate the AFs commonly used in ANNs. These AFs are crucial in introducing nonlinearity into the system, enabling the network to model complex functions. For example, **p-tanh** maps the input to a range [-1,1], which is useful for balanced and stronger gradient flow [241]. The **p-sigmoid** outputs values between [0,1], which is ideal for binary classification tasks [156]. Similarly, **p-ReLU** passes only positive inputs, promoting network sparsity and addressing gradient saturation issues [172], while **p-CReLU** limits the maximum output value, ensuring stability by

preventing overly large activations. Although the existing printed AFs are learnable and are designed to maintain ANNs performance, they are more sensitive to defects, which leads to unstable behavior in *p*-NNs [254].

5.2.2. Fault-Tolerant (*FT*) Printed Circuit Design

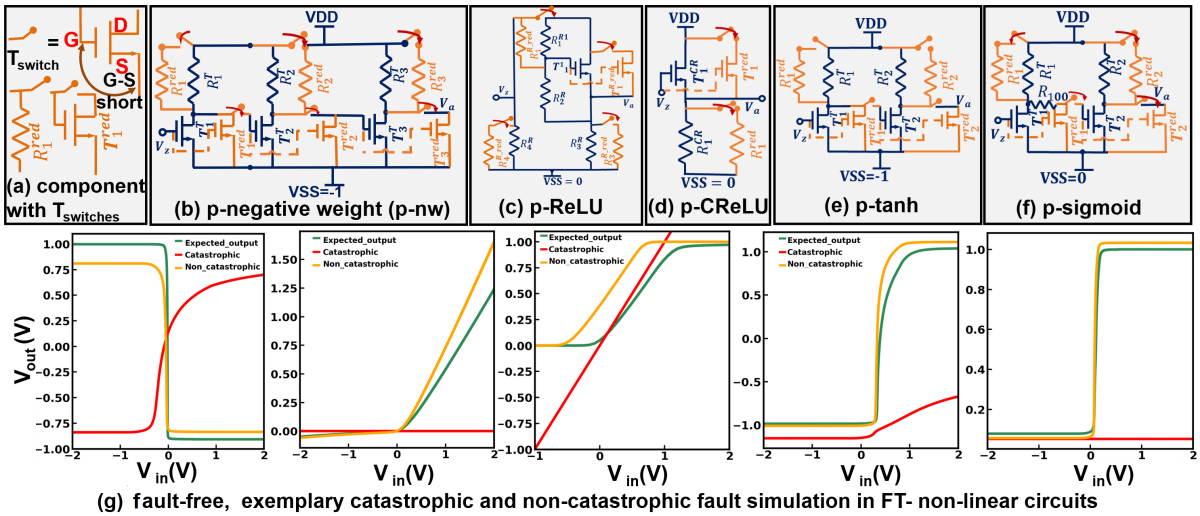


Figure 5.10.: Overview of (a) redundant transistor and resistor configuration for *FT* *p*-NNs (b) p-negative weight circuit; printed activation circuit designs of (c) p-ReLU (d) p-CReLU (e) p-tanh (f) p-sigmoid and (g) corresponding fault-free and faulty characteristics curves.

As shown in Figure 5.10, in this section, we describe the design of *FT* nonlinear circuits, aimed mainly at mitigating catastrophic faults by assuming single-component failures in either resistors or transistors. Furthermore, in subsection 5.2.3, we introduce a FAT) algorithm that dynamically selects the appropriate AF for each neuron. This bespoke approach aims to minimize accuracy degradation in *p*-NNs, ensuring robustness while reducing hardware costs, even in the presence of faults.

The printed *FT* nonlinear circuits are designed with redundant components to ensure that it continues to function properly even if certain components fail, and thus follows a consistent design principle. The main component of these circuits includes redundant resistors, transistors, and transistors configured as switches to reroute the current path in the event of any faults.

Circuit Working Principle The *FT* activation circuits use printed n-EGTs and resistor networks to implement a robust analog NN computing system. The core working principle relies on the inherent tunability of EGT threshold voltages (V_{th}) by simple adjustments during the printing process—such as varying electrolyte concentration or gate material composition—individual transistor switches can be precisely used to exhibit different conduction states. This simplifies the design of *FT* nonlinear transfer functions (e.g., ReLU, sigmoid, tanh), without requiring additional complex manufacturing steps.

Unlike CMOS fabrication, where individual threshold voltage adjustments are challenging due to specialized implants and costly processes, printed EGT allows easy per-device threshold adjustments. Additionally, printed resistors uniquely benefit from additive manufacturing, allowing precise deposition of individual resistors without added complexity. Although redundancy is common in traditional CMOS circuits, redundancy in PE has not been extensively explored due to its technology-related constraints. Thus, we introduce redundancy through multiple parallel EGT and resistor paths, each with slightly ***varied thresholds and conduction characteristics***. This provides inherent robustness against manufacturing variability, catastrophic faults (short/open circuits), and parametric deviations. Crucially, our circuits rely solely on built-in redundancy based on the concept of ***analog forward error recovery (FER)***, in which circuit components are designed with redundancy to inherently tolerate ***single-device defects (transistor***

or resistor). This approach does not require any form of reconfiguration after faults occur, significantly enhancing yield and reliability for ultra-low-cost scalable printed NN systems.

5.2.2.1. Fault-Tolerant Operation

In Figure 5.10 (a), the T_{switch} transistors act as two-terminal switches, with gate-source (G-S) shorted, operating based on the voltage difference between them and is connected with each redundant components. Under normal operation, they remain off, but if a catastrophic fault occurs in any component of the circuit, the G-S voltage of T_{switch} increases above a threshold voltage (V_{th}). The T_{switch} turns on, and allows the current to flow between the drain and the source (D-S) to either one of the redundant resistor's (R_1^{red}) or transistor's (T_1^{red}) paths. *It is worth mentioning that the threshold control in n-EGTs is easily achieved by transistor channel sizing [195], thus enabling multiple n-EGTs with varying thresholds in the same circuit.* So, even if the primary component fails, the current flow is maintained without disrupting the overall functionality. However, in any non-catastrophic fault, such as a partial short or open resistor R_2^R (as in ReLU), the circuit remains functional, without causing significant performance degradation and donot require redundant components. Also, the redundancy ensures automatic functionality without requiring explicit switches or external logic control. Under normal conditions, each transistor-resistor path, having slightly varied electrical characteristics (threshold voltage (V_{th})), together contributes to the intended circuit functionality (as shown in Figure 5.9).

***FT* Printed Negative Weight (Inverter) Circuit** The transfer characteristic of the *FT* negative weight (analog inverter) circuit (Figure 5.10 (b)) is characterized by:

$$\text{n}eg(V_{\text{z}}) = -\left(\eta_1^{\text{N}} + \eta_2^{\text{N}} \cdot \text{t}anh\left(\left(V_{\text{z}} - \eta_3^{\text{N}}\right) \cdot \eta_4^{\text{N}}\right)\right), \tag{5.5}$$

where $\jmath^{\text{N}} = [\eta_1^{\text{N}}, \eta_2^{\text{N}}, \eta_3^{\text{N}}, \eta_4^{\text{N}}] = [0.04, 0.95, 0.01, 142.20]$ are fixed auxiliary parameters determined by the physical quantities $\text{q}^{\text{N}} = [R_i^{\text{N}}, W_i^{\text{N}}, L_i^{\text{N}}]$, where $R_i^{\text{N}}, W_i^{\text{N}}, L_i^{\text{N}}$ are the resistances, width and length of the transistors respectively.

***FT* Printed ReLU Circuit** The printed ReLU activation circuit shown in Figure 5.10 (c) implements a piecewise linear function that remains linear for positive inputs and nullifies negative inputs. So, a combination of a softplus function for smoothness at $V_{\text{z}} = 0$ and a linear function for the negative slope is used to accurately describe the circuit's behavior. Consequently, the function describing p-ReLU circuit is:

$$V_{\text{a}} = \eta_1^{\text{R}} \cdot (x - \eta_3^{\text{R}}) + \eta_2^{\text{R}} \cdot softplus(V_{\text{z}} - \eta_3^{\text{R}}, \eta_5^{\text{R}}) + \eta_4^{\text{R}}, \tag{5.6}$$

and $\jmath^{\text{R}} = [\eta_1^{\text{R}}, \eta_2^{\text{R}}, \eta_3^{\text{R}}, \eta_4^{\text{R}}, \eta_5^{\text{R}}]$=[0.01, 0.66, 0.20, 0.003, 7.74] are fixed auxiliary parameters determined by circuit components $\text{q}^{\text{R}} = [R_i^{\text{R}}, W_i^{\text{R}}, L_i^{\text{R}}]$, where $R_i^{\text{R}}, W_i^{\text{R}}, L_i^{\text{R}}$ are the resistances, width and length of the transistors respectively.

***FT* Printed Clipped ReLU (CReLU) Circuit** The *FT* CReLU activation, as shown in Figure 5.10 (d), extends the basic ReLU by capping the output voltage so $V_{\text{a}} \in [0, V_{\text{max}}]$.

The mathematical expression of the CReLU is:

$$V_{\text{a}} = \begin{cases} \eta_1^{CR}, & V_{\text{z}} < \eta_3^{CR} \\ \eta_2^{CR}, & V_{\text{z}} > \eta_4^{CR} \\ \dfrac{\eta_2^{CR} - \eta_1^{CR}}{\eta_4^{CR} - \eta_3^{CR}} V_{\text{z}} + \dfrac{\eta_1^{CR}\eta_4^{CR} - \eta_2^{CR}\eta_3^{CR}}{\eta_4^{CR} - \eta_3^{CR}}, & o\mathit{therwise}, \end{cases} \tag{5.7}$$

with $\jmath^{CR} = [\eta_1^{CR}, \eta_2^{CR}, \eta_3^{CR}, \eta_4^{CR}] = [0.001, 0.96, 0.02, 1.17]$ are fixed auxiliary parameters determined by the physical quantities $\text{q}^{CR} = [R_i^{CR}, W_i^{CR}, L_i^{CR}]$, where $R_i^{CR}, W_i^{CR}, L_i^{CR}$ are the resistances, width and length of the transistors respectively.

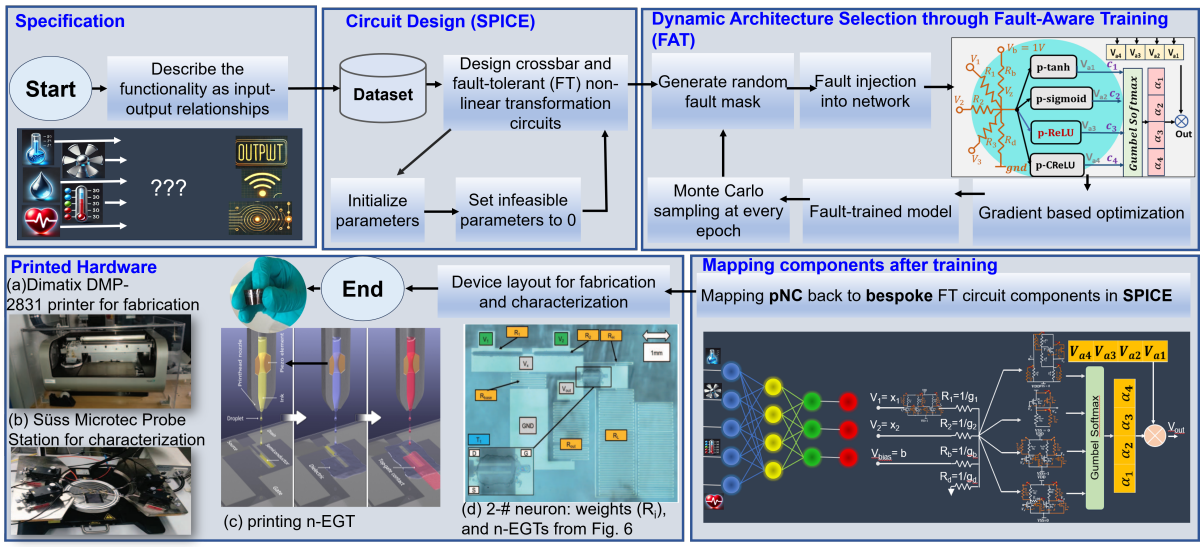


Figure 5.11.: Proposed implementation flow for an on-demand bespoke *FT* printed NN circuit (*FT*-*p*-NN) X-design given a specification of a desired functionality realized through Gumbel-Softmax distribution through fault-aware training (FAT).

***FT* Printed Tanh Circuit** Figure 5.10 (e) shows the schematic of a *FT* p-tanh circuit and is realized by:

$$V_{\mathrm{a}} = \mathrm{p}tanh(V) = \eta_1^{\mathrm{T}} + \eta_2^{\mathrm{T}} \cdot tanh\left(\left(V - \eta_3^{\mathrm{T}}\right) \cdot \eta_4^{\mathrm{T}}\right) \tag{5.8}$$

with the fixed auxiliary parameters $\mathrm{J}^{\mathrm{T}} = [\eta_1^{\mathrm{T}}, \eta_2^{\mathrm{T}}, \eta_3^{\mathrm{T}}, \eta_4^{\mathrm{T}}]$ =[0.05, -0.91, 0.23, -9.24] determined by $\mathrm{q}^{\mathrm{T}} = [R_i^{\mathrm{T}}, W_i^{\mathrm{T}}, L_i^{\mathrm{T}}]$,, where $R_i^{\mathrm{T}}, W_i^{\mathrm{T}}, L_i^{\mathrm{T}}$ are the resistances, width and length of the transistors respectively.

Printed Sigmoid Circuit Similar to the p-tanh AF circuit design in Figure 5.10 (f), the p-sigmoid can also be modeled by a suitable mathematical equation:

$$V_{\mathrm{a}} = \eta_1^{\mathrm{S}} + \eta_2^{\mathrm{S}} \cdot sigmoid\left(\left(V_{\mathrm{z}} - \eta_3^{\mathrm{S}}\right) \cdot \eta_4^{\mathrm{S}}\right), \tag{5.9}$$

where the $sigmoid(x)$ function is defined as $sigmoid(x) = \frac{1}{1+\mathrm{e}^{-x}}$,
and $\mathrm{J}^{\mathrm{S}} = [\eta_1^{\mathrm{S}}, \eta_2^{\mathrm{S}}, \eta_3^{\mathrm{S}}, \eta_4^{\mathrm{S}}]$ = [0.99, -0.91, 0.10, -48.71] are fixed auxiliary parameters determined by the physical quantities $\mathrm{q}^{\mathrm{S}} = [R_i^{\mathrm{S}}, W_i^{\mathrm{S}}, L_i^{\mathrm{S}}]$, where $R_i^{\mathrm{S}}, W_i^{\mathrm{S}}, L_i^{\mathrm{S}}$ are the resistances, width and length of the transistors respectively.

The normal and faulty transfer characteristics of all the AFs are shown in Figure 5.10 (g), assuming single-component failures in either resistors or transistors.

5.2.2.2. Redundancy Management and Complexity

In our *FT* approach, redundancy is realized through analog forward error recovery (FER). Unlike conventional digital redundancy, which requires active fault detection and complex reconfiguration, our printed circuits leverage passive redundancy directly embedded during additive manufacturing. Specifically, each printed AFs incorporates multiple parallel transistor-resistor paths, inherently managing faults passively by automatically rerouting current in the event of single-component-level faults such as open or shorts. This design significantly minimizes additional complexity as it eliminates the need for dedicated fault detection circuits, digital logic overhead or external runtime configuration.

Furthermore, the implementation uses bespoke *p*-NNs, in which neural network parameters are physically embedded during fabrication. While bespoke network architectures inherently risk fault vulnerability due to fabrication deviations affecting hard-coded parameters, our passive redundancy substantially mitigates this risk. Moreover, our approach utilizes *FT* architecture search, dynamically selecting robust combinations of normal and *FT* components during training. This co-optimization

ensures high fault tolerance and minimal hardware overhead, leveraging the neural network's inherent approximate computing nature and intrinsic redundancy. hardware.

5.2.3. Fault-Aware Training (FAT) Framework

In this section, we present our fault-aware training framework for modeling *FT-p*-NNs, which integrates three key aspects and is also shown in the proposed implementation flow Figure 5.11:

- **Initialization of Fault-Aware Printed Layer:** Our method uses printed neural network layer (`pLayer`) structure (Algorithm 5, Lines 1–5). Initially, each layer is configured with several trainable parameters:
  - **Conductances (θ):** These represent the crossbar weights and are initialized with small random values.
  - **Selection Logits (*coefficients_act*, *coefficients_neg*):** Trainable parameters controlling dynamic selection of activation functions and negative weight implementations via the Gumbel-Softmax mechanism.
  - **Temperature Parameters (*act_temp*, *neg_temp*):** Initially set to a higher value to encourage exploration in early training phases, and later gradually annealed during training iterations.
- **Fault Injection During Training:** We introduce defects (catastrophic: resistor open/short, transistor G-D/G-S/D-S shorts, non-catastrophic: resistance variation, transistor threshold voltage, width and length variations) into crossbars and nonlinear circuits, derived from SPICE-level simulations.
- **Dynamic Architecture Search:** We use Gumbel-Softmax to *dynamically select* AFs and inverters at each neuron, enabling a differentiable architecture search that adapts to faults.
- **Fault-Aware Classification Loss:** We incorporate these faults into the forward pass and compute an expected classification loss, ensuring the model learns to remain accurate under realistic manufacturing defects.

In summary, our FAT framework integrates realistic fault scenarios directly into the training phase of *p*-NNs. Using Monte Carlo-based fault injection, we randomly introduce realistic fault behaviors, derived from detailed SPICE simulations, into our neural network training process and dynamically optimizes the architecture, selecting bespoke AFs via a Gumbel-Softmax, ensuring maximum robustness against the injected faults. Although our framework considers AFs, negative weight circuit and crossbar conductances' faults, we place greater emphasis on faults in AFs due to their critical nonlinear behavior and inherent vulnerability to faults, which require more careful analysis and mitigation strategies. The following subsections detail each component, with *Algorithm* 5 summarizing the core pseudo-code.

Simulated Fault Injection During Training To evaluate *p*-NNs fault endurance, we employ a *Monte Carlo-based* fault injection approach during training, targeting both crossbars and nonlinear circuits. As illustrated in 5 (Lines 8, 12, 18), random masks $M_{\text{Res}}, M_{\text{INV}}, M_{\text{ACT}}$ alter the behavior of conductances, inverters, and AFs, modeling realistic printing defects. We consider resistor open/short conditions and transistor faults (gate-drain (G-D), gate-source (G-S), or drain-source (D-S) shorts/opens), each with specific behaviors derived from SPICE simulations.

Top-Down Sampling Strategy for Faulty Layers As shown in *Algorithm* 5 (Lines 21-32, 35-38), we simulate faults by selecting which layers are faulty, then deciding which components in those layers (AFs, inverters, conductances) will be corrupted. Let N_l denote the total number of components in layer l. We define a probability

$$p(l) = \frac{N_l}{\sum_{l=1}^{L} N_l}, \tag{5.10}$$

which ensures that layers with more components have a higher probability of receiving faults. After choosing e_{fault} layers based on $p(l)$, we further distribute faults among AFs, inverters, and conductances proportionally to their device counts. Specifically,

$$\begin{aligned} p(l) &= \frac{N_l}{\sum_{l=1}^{L} N_l}, \qquad p(\text{Type} = ACT \mid l) = \frac{N_l^{ACT}}{N_l}, \\ p(\text{Type} = INV \mid l) &= \frac{N_l^{INV}}{N_l}, \qquad p(\text{Type} = Res \mid l) = \frac{N_l^{Res}}{N_l}. \end{aligned} \tag{5.11}$$

where N_l^{ACT}, N_l^{INV}, and N_l^{Res} denote the counts of AFs, inverters, and resistors in layer l. This approach maintains a consistent fault rate across the network, reflecting the higher probability of defects in layers with more total components.

Masks for Crossbars and Nonlinear Circuits Once we determine which components are faulty, we apply specialized masks, as shown in *Algorithm* 5 (Lines 8, 12, 18):

- **Crossbar Conductances** (`Res`): A *ternary mask* M_{Res} multiplies each conductance g by $\{1,\ 0,\ \infty\}$, simulating no fault, open circuit, or short circuit, respectively. Hence, $g \to 0$ emulates an open fault, and $g \to \infty$ a short.
- **Nonlinear Circuits** (`ACT` or `INV`): We define a generic binary mask M_{NL} for any *nonlinear* subcomponent, whether it is an AF or an inverter. Concretely,

  $$M_{\text{NL}} \in \{M_{\text{ACT}},\ M_{\text{INV}}\},$$

  meaning that M_{NL} corresponds to M_{ACT} if the subcomponent is an AF, and M_{INV} if it is an inverter. Formally, if $M_{\text{NL},k} = 0$, then subcomponent k is fault-free, while $M_{\text{NL},k} = n \neq 0$ indicates that the n-th faulty transfer function is used instead of the normal function. In equation form:

  $$\begin{aligned} \text{NL}_{\text{faulty}}(x) = \sum_k \big(& \mathbb{I}(M_{\text{NL},k} = 0) \cdot \text{NL}_k(x) \\ & + \mathbb{I}(M_{\text{NL},k} = n) \cdot \mathit{FBDataset}_{\text{NL}}[k, n](x) \big), \end{aligned} \tag{5.12}$$

  where $\mathbb{I}(\cdot)$ is the indicator function. Thus, if M_{NL} corresponds to an AF, it acts as M_{ACT}, and if it corresponds to an inverter, it acts as M_{INV}. Either way, a nonzero mask entry means subcomponent k is replaced by the corresponding faulty behavior from the *FBDataset*.

During each forward pass, `MakeFault` updates these masks based on the sampled probabilities if $e_{\text{fault}} > 0$, while `RemoveFault` resets them otherwise as shown in *Algorithm* 5 (Lines 22-24). The masked values directly impact the MAC output (Line 13) and subsequent layer computations.

Faulty Behavior Dataset (*FBDataset*) Although prior works [254] have analyzed fault sensitivity in certain p-NNs, they didnot provide any solution to improve its fault-endurance. Our approach strengthens the p-NNs robustness at both the *design* and *algorithmic* levels. Specifically, we first characterize printed circuit blocks such as AFs and negative weight circuit (analog inverter) (as in Figure 4.8) through SPICE simulations to capture their response under multiple fault conditions, such as transistor open faults, transistor short faults, resistor open faults and parametric deviations (e.g. variations in transistor threshold voltages and resistor values). These faulty behaviors were recorded as input-output pairs representing how each fault impacted the circuit's analog transfer characteristics. These functions are stored in a *Faulty Behavior Dataset* (*FBDataset*). When M_{ACT} or M_{INV} flags a subcomponent as faulty, the network substitutes its normal function with the corresponding entry from *FBDataset*, accurately modeling real-world printing defects.

Forward Pass Under Fault Masks In each training iteration:

1. **Faulty Layer Selection**: We choose up to e_{fault} layers based on $p(l)$. Within those layers, the probability $p(\text{Type} \mid l)$ decides how many AFs, inverters, or conductances are marked faulty.
2. **Mask Application**: For crossbars, M_{Res} zeroes or inflates conductances to emulate open/short conditions. For nonlinear circuits, $M_{\text{ACT}}, M_{\text{INV}}$ dictate whether an AF/inverter is normal or replaced by a faulty counterpart from *FBDataset*.
3. **Fault-Aware Computation**: The model's forward pass reflects these masked values, ensuring outputs incorporate realistic defect behaviors. Each iteration thus presents a different fault scenario, improving the model's overall robustness through repeated exposure.

By combining this mask-based approach with our dynamic architecture search (Section 5.2.4), the network adaptively learns circuit configurations that minimize accuracy loss, even under repeated exposures to realistic manufacturing faults.

5.2.4. Bespoke Architecture Optimization (Dynamic Architecture Search)

To further enhance fault endurance, we perform a *dynamic architecture search* at the circuit level. As illustrated in *Algorithm* 5 (Lines 11, 17), each neuron's choice of inverter and AF is determined by trainable logit vectors, which we sample using the Gumbel-Softmax distribution:

$$\text{GumbelSoftmax}(c_i, \tau) = \frac{\exp((\log(c_i) + g_i)/\tau)}{\sum_{j=1}^{n} \exp((\log(c_j) + g_j)/\tau)}, \tag{5.13}$$

where $g_i = -\log(-\log(U_i))$ with $U_i \sim \text{Uniform}(0, 1)$, and τ is a parameter controlling exploration vs. exploitation. Initially, a higher τ yields *soft* selections, allowing gradients to flow to multiple AFs/inverters. Over time, we decay τ (e.g. $\tau \leftarrow \max(0.95 \times \tau, 0.1)$) to approximate a near-hard argmax.

Masking & Gumbel-Softmax Synergy. The Gumbel-Softmax distribution serves as a differentiable approximation to categorical selection, making the choice of AFs a continuous optimization problem that can be efficiently solved using standard gradient-based backpropagation in PyTorch. Initially, each neuron considers all AF candidates simultaneously, weighted by learnable probabilities. During training, these probabilities are gradually optimized, resulting in one dominant AF per neuron at convergence. This "bespoke" selection process ensures that each neuron's AF is individually tailored, maximizing fault endurance and accuracy based on the learned robustness characteristics specific to the printed NN hardware. As faults are injected randomly each iteration, the selected AF that performs better under these defects tends to achieve lower classification loss, driving up its logit value. Over repeated exposures, each neuron converges on the circuit component most resilient to the observed faults. Thus, the network effectively learns a *bespoke* circuit architecture, picking subcomponents that best mitigate printing defects.

Loss Function for Fault Endurance Finally, we use a fault-aware classification loss as shown in *Algorithm* 5 (Lines 33-44) that accounts for the random fault masks M. Let $f(x; \phi, M)$ be the network output under parameters ϕ and a mask M. We define:

$$\mathcal{L}_{\text{classification}} = \mathbb{E}_{p(M)}\Big[\mathcal{L}_{\text{primary}}(y,\ f(x; \phi, M))\Big]. \tag{5.14}$$

Because enumerating all faults is intractable, we sample $M^k \sim p(M)$ multiple times and average the resulting classification loss (cross-entropy). Formally,

$$\mathcal{L}_{\text{classification}} \approx \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}_{\text{primary}}(y,\ f(x; \phi, M^k)). \tag{5.15}$$

By combining fault injection, dynamic circuit selection, and this expected classification loss, our framework learns robust p-NNs that can tolerate real-world manufacturing variability without sacrificing accuracy.

Algorithm 5 Fault-Aware Printed Layer (`pLayer`) and Printed Neural Network (*p*-`NN`)

Notation:

L: # layers; N_l: total devices in layer l; N_l^{ACT}, N_l^{INV}, N_l^{Res}: AFs, inverters, resistors in l; e_{fault}: faults per forward

$p(l) = N_l / \sum_{j=1}^{L} N_j$, $p(\text{Type} \mid l) = N_l^{\text{Type}} / N_l$, $\text{Type} \in \{\text{ACT}, \text{INV}, \text{Res}\}$

Trainable Params: θ : *conductances*; *coefficients_act*, *coefficients_neg* : *activation/inverter parameters*

procedure pLayer_Init($n_{\text{in}}, n_{\text{out}}$, ACT, INV)
  Init $\theta \in \mathbb{R}^{(n_{\text{in}}+2)\times n_{\text{out}}}$ ▷ $n_{\text{in}}, n_{\text{out}}$: input and output dims
  Init *coefficients_act*, *coefficients_neg*
  `act_temp`, `neg_temp` ← 1.0 ▷ Gumbel-Softmax temperature parameters for AF and inverter selection
  Init all-ones FaultMaskRes, zeros FaultMaskACT, FaultMaskNEG
function MAC(**a**)
  $\theta_{\text{noisy}} \leftarrow \theta \odot (\text{noise}) \odot \text{FaultMaskRes}$ ▷ variation + fault aware training
  $\mathbf{W} \leftarrow \text{normalize}|\theta_{\text{noisy}}|$
  Split θ_{noisy} into $\mathbf{W}_+$, $\mathbf{W}_-$
  $\beta \leftarrow \text{GumbelSoftmax}(\textit{coefficients\_neg}, \texttt{neg\_temp})$
  $\mathbf{a}_{\text{neg}} \leftarrow \sum_i \beta_i \, \text{INV}_i(\mathbf{a})$ ▷ Each INV_i uses its own `FaultMaskNEG` internally
  return $\mathbf{a}\,\mathbf{W}_+ + \mathbf{a}_{\text{neg}}\,\mathbf{W}_-$
function forward(**a**)
  $\mathbf{z} \leftarrow \text{MAC}(\mathbf{a})$
  $\alpha \leftarrow \text{GumbelSoftmax}(\textit{coefficients\_act}, \texttt{act\_temp})$
  $\mathbf{a}_{\text{out}} \leftarrow \sum_{i=1}^{|\text{ACT}|} \alpha_i \, \text{ACT}_i(\mathbf{z})$ ▷ Each ACT_i uses its own `FaultMaskACT` internally
  return $\mathbf{a}_{\text{out}}$
procedure MakeFault(e_{fault})
  if $e_{\text{fault}} = 0$ **then**
    RemoveFault; **return** (FaultMaskRes, FaultMaskACT, FaultMaskNEG)
  for $i = 1 \ldots N_{\text{fault}}$ **do**
    Split $e_{\text{fault}} \sim (N_l^{\text{Res}} : N_l^{\text{ACT}} : N_l^{\text{INV}})$
    Pick random resistor indices; set $M_{\text{Res}} = 0$ or ∞
    Pick random AF indices; set M_{ACT} to nonzero code
    Pick random inverter indices; set M_{INV} to nonzero code
  return (FaultMaskRes, FaultMaskACT, FaultMaskNEG)

***p*-NN:** sequence of `pLayer` modules

function *p*-NN_forward(**x**)
  RemoveFault
  for $i = 1 \ldots e_{\text{fault}}$ **do**
    Sample layer $l \sim \text{Categorical}\{p(1), \ldots, p(L)\}$
    Count faults per layer and call MakeLayerFault(l, 1)
    for each layer ℓ **do**
      $\ell.\texttt{act\_temp} \leftarrow \max(0.95\,\ell.\texttt{act\_temp}, 0.1)$; $\ell.\texttt{neg\_temp} \leftarrow \max(0.95\,\ell.\texttt{neg\_temp}, 0.1)$
      $\mathbf{x} \leftarrow \ell.\text{forward}(\mathbf{x})$
    return **x**

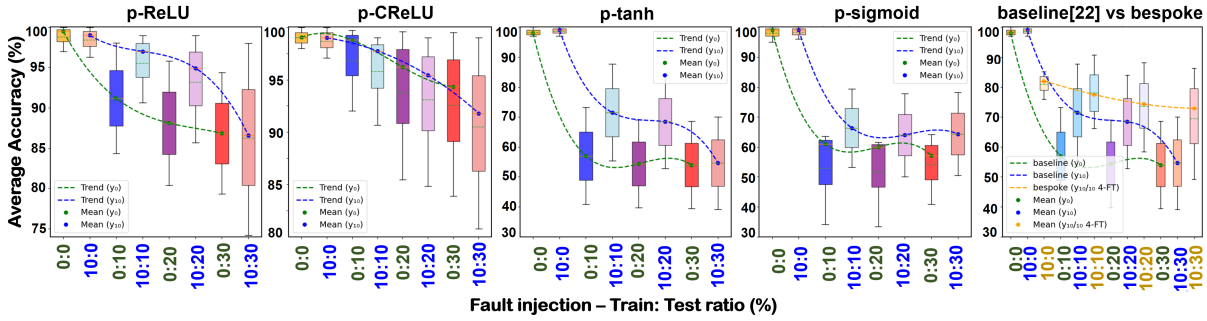


Figure 5.12.: (a)-(d) Evaluation of average accuracy of four single AF using *FT* p-NNs (e) effectiveness of bespoke *FT* AF over existing baseline [254] AF under 0% (no fault) and 10% (with fault) fault injection in training and up to 30% fault injection in testing averaged over 8 benchmark datasets. y_0 and y_{10} denotes 0% and 10% fault injection in training.

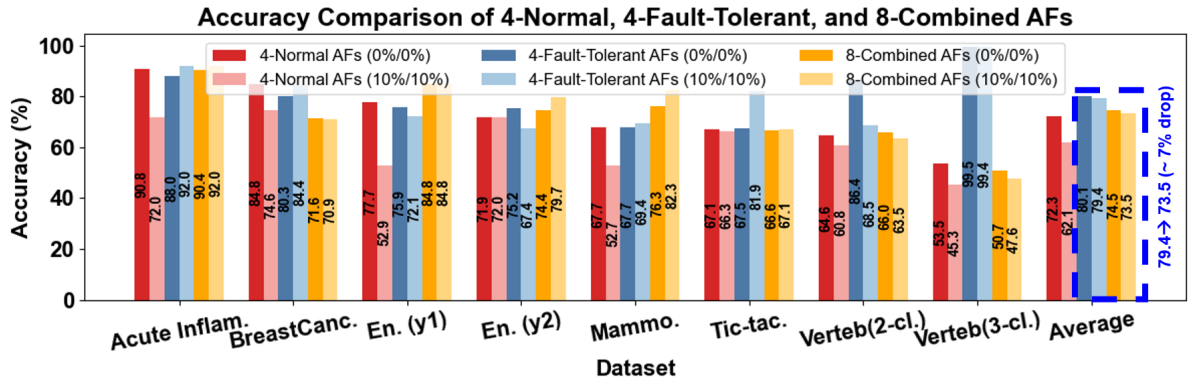


Figure 5.13.: (a)-(d) Evaluation of average accuracy of 4-Normal AFs, 4-Fault Tolerant AFs and 8-combined AFs under 0:0 and 10:10 train : test ratios over 8 benchmark datasets.

Runtime (Post-fabrication) Fault Management The *FT* designs inherently manage runtime faults passively. Redundant transistor-resistor pathways are integrated during additive printing. In operational scenarios, any component-level faults, such as opens or shorts, trigger automatic rerouting of current through available functional redundant paths. This automatic rerouting eliminates the necessity of active switching logic, multiplexers, controllers, or explicit fault classification circuits. Consequently, our printed circuits maintain functionality without additional runtime complexity, significantly reducing hardware overhead and simplifying operational reliability.

Dataset	4-Normal AFs			4-*FT* AFs			8-Combined AFs		
	Area (mm^2)	Power (mW)	Time (hr)	Area (mm^2)	Power (mW)	Time (hr)	Area (mm^2)	Power (mW)	Time (hr)
Acute Infl.	34.40	0.455	10.6	32.48	4.616	45.8	33.66	1.978	17.1
BreastCanc.	21.36	0.552	11.0	36.48	0.712	46.9	40.87	1.119	18.2
En. (y_1)	32.65	5.560	4.9	43.96	10.91	54.9	40.33	85.2	24.8
En.(y_2)	33.06	6.276	7.3	36.94	40.45	51.6	30.92	1.402	24.9
Mammo.	34.26	28.24	9.6	43.76	31.26	46.7	42.29	0.620	20.5
Tic-tac	32.09	31.00	8.5	55.07	79.5	44.5	47.74	0.615	20.7
Verteb(2-cl.)	26.22	94.19	11.6	37.82	288.9	38.8	34.81	124.1	17.7
Verteb(3-cl.)	41.83	1.939	7.02	47.33	18.83	37.6	41.43	0524	16.7
Average	**31.98**	**20.57**	**8.81**	**41.73**	**59.39**	**45.9**	**39.00 ↓(6.54%)**	**26.99 ↓ (54.5%)**	**20.07 ↓(56.2%)**

Table 5.1.: Hardware costs and training time (hours) comparison of Normal, *FT*, and Combined AFs over 8 benchmark datasets under train: test =10% : 10% ratios. (The reduction w.r.t 4-*FT* AFs at an acceptable accuracy drop (≈ 7%) is shown in blue.

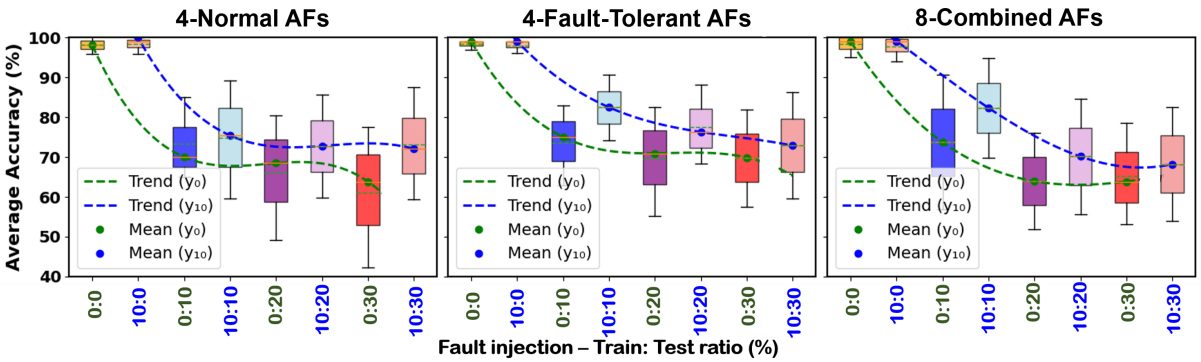


Figure 5.14.: (a)-(c) Evaluation of normalized accuracy of three bespoke architectures under 0% (no fault) and 10% (with fault) fault injection in training and upto 30% fault injection in testing averaged over 8 benchmark datasets. y_0 and y_{1_0} denotes 0% and 10% fault injection in training.

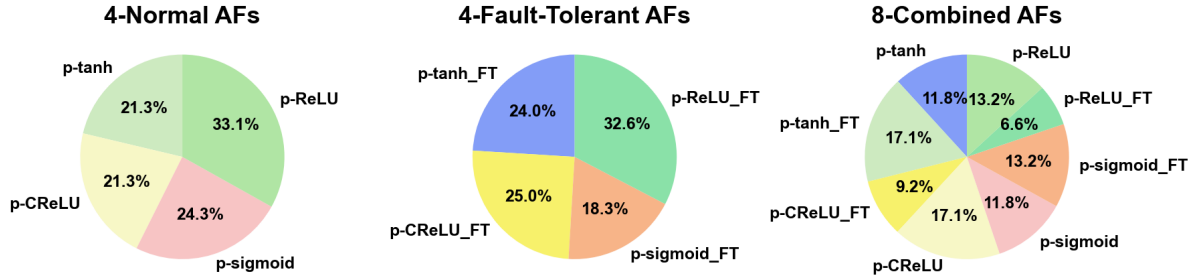


Figure 5.15.: Percentage of AFs used when (i) 4-Normal AFs (ii) 4-*FT* AFs and (iii) 8-Combined AFs are selected averaged over 8 benchmark datasets under train: test = 10% : 10% ratios.

5.2.5. Evaluation

To evaluate the effectiveness and robustness of our proposed fault-aware training method[4] in *p*-NNs, we employed standard ANN models, specifically MLPs, trained and tested using well-established PE benchmark datasets as in [202], [220], [243]. After training, we mapped the printed ANN weights and neuron functionalities onto multiple printed crossbar arrays, with each crossbar functioning as an analog computing block. We further assessed the generalization capabilities of our method by selecting fault injection ratios at different training-to-testing scenarios: specifically 0:0, 0:10, 0:20, 0:30, 10:10, 10:20, and 10:30 using PyTorch [149]. Ratios like 0:0 represent the ideal (fault-free) baseline scenario, matched ratios like 10:10 reflect realistic assumptions of similar fault occurrence between training and operation, and mismatched ratios (e.g., 10:20, 10:30) investigate model resilience under harsher defects than those explicitly faced during training. These assumptions are guided by practical considerations relevant to PE [187], where devices experience varying degrees of defects or degradation over their operational lifetime.

5.2.6. Experiment

Circuit Design and Training Configuration The *FT*-*p*-NNs model was trained using PyTorch [149], with datasets normalized to [0,1] and split into 60% training, 20% validation, and 20% testing. Fault injection was performed at two levels:

[4] Code is available at `https://github.com/KIT-Neuromorphic-Computing/Fault_Aware_\gls{pnn}`.

- **SPICE Level:** Fault injection in the non-linear AFs in Figure 5.10 (b)-(g) were first conducted at the SPICE simulation level using the n-EGT pPDK[152] in Cadence Virtuoso[5], ensuring an accurate representation of physical defects and their effects on circuit transfer and fault characteristics based on ***forward error recovery (FER)***.
- **Algorithmic Level:** The fault effects, encapsulated in the SPICE-generated *FBDataset*, were abstracted into a PyTorch-compatible format for FAT. During training, fault rates ranging from 0% to 10% were introduced using Monte Carlo sampling, enhancing the model's robustness against defect-induced variations. This approach enabled the model to effectively handle both typical and extreme fault scenarios (0% to 30% fault rates) during testing. The Adam optimizer [66] was used during training.

Bespoke Architecture In this section, we discussed the design of different AFs within the *p*-NN, i.e., different neurons utilizing different AFs during training: (a) using 4-normal-AFs, (b) using 4-*FT*-AFs, and (c) using 8-combined AFs (4-normal and 4-*FT*). The functionality of the baseline printed NN hardware and the printed defects have been validated in [187], [202], and the contribution of this work, i.e., the fault-endurant approaches has been verified at the algorithmic level, the experiment is conducted at simulation level based on pPDK [152].

5.2.7. Result

This subsection presents the experimental results of proposed *FT*-*p*-NNs, comparing its accuracy and hardware costs with the baseline and combined versions of AFs under various fault conditions on 8 benchmark datasets. The box plots in Figure 5.12 show that FAT enhances the robustness of *FT*-AFs across various *p*-NNs. In addition, for single *FT*-AFs, both p-ReLU and p-CReLU show the most notable accuracy with minimal variation under 0% till 30% fault conditions. Meanwhile, the ReLU family consistently demonstrated strong robustness when faults are injected during both training and testing, suggesting inherent characteristics that improve its fault resilience. However, they require high-value resistors ($\approx$ 1.5MΩ), which not only occupy a huge area but also consume more power.

As shown in Figure 5.13, the average accuracy across 8 benchmark datasets for the ideal fault-free scenario (0% fault during training and testing) is 72.3% when employing 4-Normal AFs. Under real fault conditions (10% faults during both training and testing), the accuracy of normal AFs significantly drops to 62.1%, while the 4-*FT* AFs improve this accuracy notably to 79.4%. Figure 5.13 and Table 5.1 further confirms that the 8-combined AF balances accuracy, robustness, area, and power, offering a viable solution for applications where these trade-offs are critical. For datasets, *verte3cl.* and *tictac.*, the 4-*FT*-AFs show an unusually high accuracy (81.9% and 99.4% respectively) at the 10%/10% train/test ratio, which is much higher than its accuracy in other datasets whereas *energy2.* dataset results in an accuracy drop to 67.4% in 4-*FT*-AFs. These suggest that certain AFs have struggled or excelled depending on the characteristics of the datasets. For some datasets, the accuracy of the *FT*-AF under the 10%/10% train/test ratio fault scenario exceeds that of 0%/0% train/test fault-free scenario. Fault injection during training has acted as a form of regularization, allowing the model to generalize better, even in a non-faulty environment. However, our evaluation clearly shows that adopting only 4-*FT* AFs introduces higher overhead compared to the baseline (4-normal AFs), increasing area by $\approx$ 30.5% (1.31 $\times$), power by $\approx$ 188% (2.89 $\times$), and training time by $\approx$ 420% (5.21 $\times$). To address this issue, we further adopt the combined fault-aware approach (8-Combined AFs), which effectively reduces these overheads, achieving significant reductions of 6.54% (1.23 $\times$) in area, 54.5% (1.31 $\times$) in power, and 56.2% (2.28 $\times$) in training time compared to the only 4-*FT* AFs while incurring only $\approx$ 7% classification accuracy drop. This $\approx$ 7% accuracy degradation for significant reductions in power (54.5%) and area (6.54%) is practically justified in resource-constrained applications where energy efficiency, cost, and size constraints outweigh the demand for maximum achievable accuracy. For e.g, in applications like disposable smart

[5] `https://www.cadence.com/en_US/home.html`

bandages or wearable health-monitoring sensors, maintaining adequate battery life, minimizing device footprint, and ensuring affordability are prioritized and moderate accuracy losses are acceptable, provided the system maintains sufficient reliability under real faults. It is also observed that the normal AF has slightly larger area (34.40 mm^2) than the FT-AF (32.48 mm^2) for the **"Acute Infl."** dataset which seem counterintuitive but can occur due to the bespoke (customized) selection of AFs. In some cases, this approach might choose AFs that inherently require fewer or smaller circuit components compared to the uniformly chosen normal AFs, resulting in a smaller overall area.

Also, the training time significantly increases from the normal AFs ($\approx 8.81 hours$) to the FT ones ($\approx 45.9 hours$), reflecting the additional computational complexity introduced by fault-aware optimization. However, using combined AFs moderately reduces the training overhead ($\approx 20.07 hours$) by $\approx 56.2\%$, showing a trade-off between accuracy, robustness, and computational cost.

Figure 5.14 compares the robustness of three bespoke AF architectures—4-Normal, 4-FT, and 8-combined—across different fault injection train-test ratios, averaged over 8 benchmark datasets. At low fault levels (e.g., 0:0 and 10:10), all architectures exhibit high accuracy and low standard deviations, indicating stable performance. However, as fault severity increases (e.g., 0:30 and 10:30 scenarios), the 4-Normal AFs show a sharp decline in average accuracy accompanied by notably higher standard deviations, indicating increased variability and instability. In contrast, the 4-FT AFs demonstrate significantly better robustness, maintaining higher mean accuracy with smaller standard deviations, showing consistent and reliable operation under increased fault levels. The 8-combined AFs give an intermediate solution, balancing accuracy and robustness, with moderate mean accuracy and lower standard deviation compared to the normal AFs.

The selection of AFs in each p-NN as in Figure 5.15 reflects their ability to handle faults in faulty scenarios. In both 4-normal-AF and 4-FT-AF, ReLU family is preferred for its simplicity, efficiency and better generalization in resource-constrained environments, while p-sigmoid is chosen for its robustness against severe faults. The 8-combined AF balances all the AFs leveraging their strengths to optimize area and power consumption, thus minimizing the drop in accuracy under various faulty conditions.

The PRINT-SAFE framework addresses long-term reliability and endurance of printed NN circuits by co-designing fault-resilient circuits and training algorithms. Instead of relying solely on one-time manufacturing tests or static redundancy, PRINT-SAFE aims to create p-NNs that can tolerate both catastrophic and parametric faults over their operational lifetime, and that can "self-heal" through algorithmic adaptation.

5.3. Delay-Based Testing and Reliable Design of ReFLEX-*LDO*

While p-NNs implement computation, reliable operation of any printed system also hinges on robust power delivery. This work on *ReFLEX-LDO: Reliable Design and Delay-based Testing of Flexible Low Dropout Regulators* tackles this problem for flexible LDOs fabricated with n-type flexible transistors.

5.3.1. Low Dropout Regulators (LDOs)

Low-dropout (*LDO*) regulators are fundamental power-management blocks that provide regulated DC supply rails under varying load and supply conditions. Unlike switching regulators, LDOs operate without inductors, making them highly attractive for compact, noise-sensitive, and area-constrained designs [31], [209]. Their ability to suppress supply ripple, minimize output noise, and achieve fast settling behavior is essential for analog front-ends, sensing circuits, and digital logic.

In the context of FE, these requirements become even more stringent. Target applications such as wearable healthcare devices, epidermal sensors, and large-area IoT nodes[166], [219] rely on stable supply rails to ensure correct functionality under mechanical bending, substrate variations, and runtime disturbances. Since FE technologies prioritize thin-film devices, lightweight form factors, and low-cost manufacturing, switching-based regulation is often impractical due to electromagnetic interference, efficiency loss at low load, and bulky passives. Consequently, LDOs emerge as the preferred solution,

where they serve as always-on regulators, post-regulators for energy harvesters and rectifiers, and critical enablers of robust operation across diverse application scenarios. Thus, LDOs are indispensable to ensure both reliability and system-level stability in FE systems. Recent works have demonstrated flexible LDOs, such as an InSnO TFT-based design on polyimide achieving $\approx$ 10 nA quiescent current with stable performance under bending down to 5 mm radius, highlighting their promise for future low-power, flexible biomedical electronics [136], [265].

Related Works on Analog Testing Testing of analog circuits is crucial for reliable operation, especially in biomedical, automotive, and IoT systems where power integrity directly impacts performance. Traditional *specification-based methods* measure parameters such as line/load regulation, power-supply rejection ratio (PSRR), transient response, or noise [22], [41], [198], [228]. While effective in CMOS circuits, they require complex monitors, incur design overhead, and fail to guarantee full structural fault coverage[221]. To overcome this, *structural approaches* perturb selected nodes and monitor invariants such as supply current, transfer-function poles/zeros, or delay signatures [21], [89], [124], [165], [257]. Among these, delay-based monitoring with ON/OFF keying has shown high coverage with mostly digital hardware [164], though its adaptation to FE remains highly underexplored.

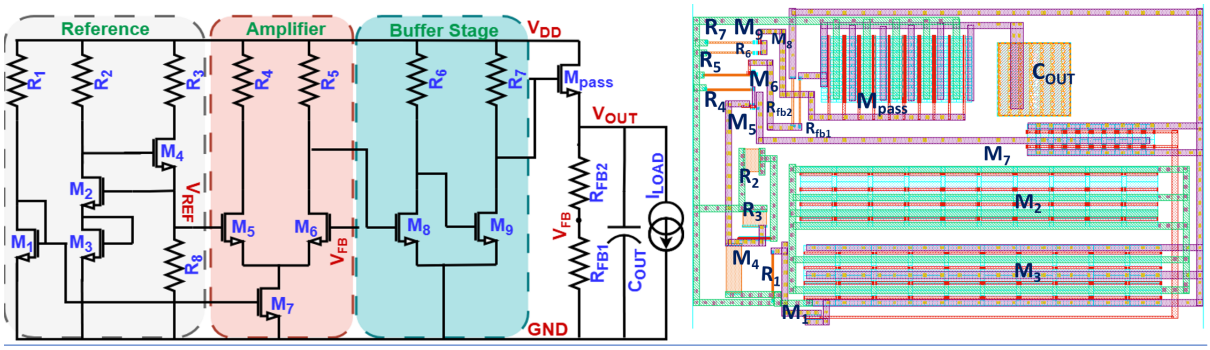
With the growing integration of LDOs in system-on-chip, analog characterization alone is insufficient and is thus driving interest in defect-oriented and BIST schemes [32], [83], [249]. Ince and Ozev [165] proposed a digital BIST using pseudo-random binary sequence (PRBS) injection and correlation analysis, achieving fault detection with low area cost. Similar defect-oriented test techniques have been applied to flipped-voltage-follower (FVF) LDOs for automotive-grade reliability [255]. Other efforts include non-invasive stability analysis estimating closed-loop phase margin without breaking feedback [141], and simulation-based frameworks like AnalogGym, which provide standardized analog test benches [257].

While these methods form a foundation for CMOS, applying them directly to FE is challenging: the n-type-only a-IGZO TFT process, low g_m, and large parasitics change which internal nodes can be safely perturbed, so their injection/observation schemes cannot be reused as-is. Specification-based BIST[32], [221] requires analog monitors that add significant overhead and is impractical for TFT with low density and high parasitics. Moreover, specification checks may miss structural defects (e.g., bias-branch opens masked at some operating points). FE further complicates testing due to low voltages/currents, high variability, and limited probe access, making on-chip PSRR or stability measurement extremely difficult without bulky, power-hungry analog monitors. Finally, specification-based approaches lack scalability, as each circuit demands bespoke tests (e.g., amplifier gain vs. *LDO* regulation). Combined with FE-specific issues, i.e., process spread, bending-induced defects, and restricted probing access limit conventional testing methods. At the same time, FE systems for wearables and biomedical use demand high reliability under mechanical stress and energy-constrained operation. This gap highlights the need for lightweight, embedded, defect-oriented strategies tailored to LDOs and analog circuits in FE.

5.3.2. Proposed *ReFlex-LDO* Architecture

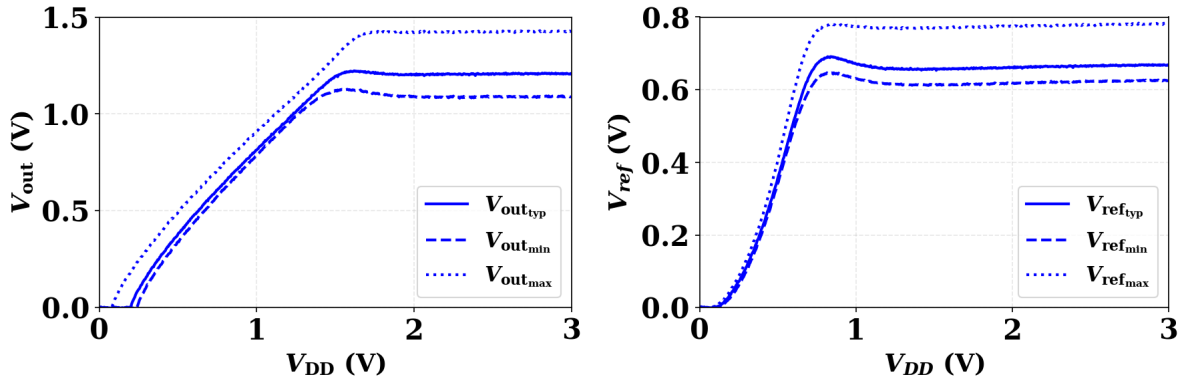
The proposed ***ReFlex-LDO*** is an all-NMOS low-dropout regulator designed in a flexible oxide-TFT process. The schematic and its layout is shown in Figure 5.16. Its architecture retains the four fundamental building blocks of a classical *LDO* but adapts them to address the absence of PMOS devices in the target technology, reduced intrinsic gain, and strong process variation in flexible substrates.

Error Amplifier (Stage 1) The regulation loop begins with the differential input pair (M5–M6), which senses the difference between the reference voltage V_{REF} and the divided feedback voltage V_{FB}. The pair is biased by the tail current source M7. The reference generation branch (M1–M3 together with resistors R_1–R_3 and R_8) establishes a stable V_{REF} across PVT and mechanical bending. Transistors M4–M5 provides high output resistance and forms the first high-gain stage. This stage amplifies the error between V_{REF} and V_{FB}, setting the accuracy of regulation.



W/L	M_1	M_2	M_3	M_4	R_1	R_2	R_3	R_8	M_5	M_6	M_7	R_4	R_5	M_8	M_9	R_6	R_7	M_{pass}	R_{fb1}	R_{fb2}	C_{out}
W(μm)	12	200	200	20	0.6	15	15	10	2	2	100	0.6	0.6	18	18	1	1	1000	1	1	14
L(μm)	1.2	1.2	1.2	0.6	30	9	9	30	1.2	1.2	1.2	30	30	1.2	1.2	30	30	1.2	30	30	15

Figure 5.16.: Schematic and layout of the proposed gain-boosted *ReFlex-LDO*, showing the reference, amplifier, buffer, and output stages, with design specifications.



(a) Line regulation at the output stage: V_{out} across PVT corners.

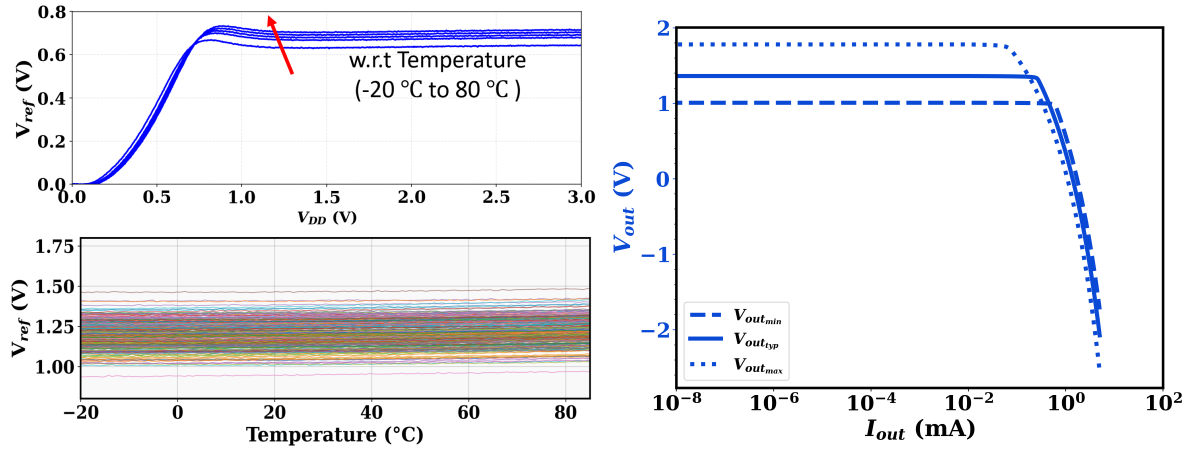
(b) Line regulation of the internal reference: V_{ref} across PVT.

Figure 5.17.: Line regulation of the proposed gain-boosted ReFlex-*LDO*. The left plot illustrates the regulated output voltage V_{out}, while the right plot highlights the stability of the internal reference voltage V_{ref} across process corners after **post-layout**.

Second Gain Stage (Stage 2) The amplified error signal is fed into the second gain/driver stage implemented using M8–M9. This stage boosts both voltage gain and output swing to ensure that the large pass transistor M_{pass} can be driven efficiently. By separating the differential pair from the driver, the design compensates for the limited intrinsic gain of a-IGZO TFTs and maintains adequate loop gain for tight regulation even with a pF-scale output capacitor.

Pass Device and Feedback Network (Stage 3) The output pass transistor M_{pass} provides the load current and regulates the output voltage V_{out}. Its sizing is optimized to achieve low dropout while accommodating the reduced mobility of TFT devices. A resistive feedback divider (R_{FB1}–R_{FB2}) samples V_{out} and generates the feedback signal V_{FB}, which is compared against V_{REF} in the error amplifier. This closes the regulation loop and determines the static accuracy and load-dependent behavior of the *LDO*.

Stability and Compensation (Stage 4) The *ReFlex-LDO* employs an output-capacitor–dominant-pole compensation strategy using a small on-chip capacitor C_0 at V_{out}. This creates the dominant pole $p_1 \approx 1/(R_{out}C_0)$. Because C_0 is only in the pF range, its ESR is negligible, pushing the ESR-induced zero $z_{ESR} = 1/(R_{ESR}C_0)$ far beyond the loop bandwidth. Thus, unlike CMOS LDOs, the design does not rely on ESR-zero compensation. Loop stability is instead ensured by placing all non-dominant poles (from the amplifier and buffer stages) well above the unity-gain frequency. The small C_0 additionally improves noise filtering and enhances PSRR, enabling a compact and robust *LDO* implementation.



(a) Line regulation of reference: V_{ref} versus V_{DD} and MC of 5000 runs for V_{ref} with temperature.

(b) Load regulation at the output stage: V_{out} versus I_{out} across process corner.

Figure 5.18.: Regulation characteristics of the proposed gain-boosted ReFlex-*LDO*. The left plot shows the line regulation of the internal V_{ref} and 5000 monte-carlo simulation across PVT corners while the right plot shows load regulation by showing variation of V_{out} with I_{load}.

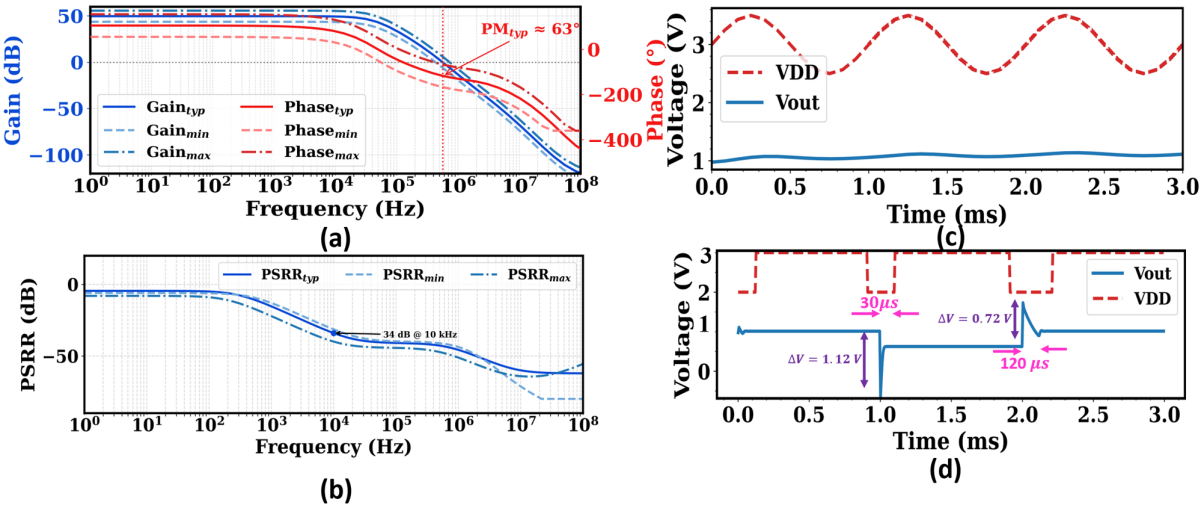


Figure 5.19.: (a) Loop gain and phase response across frequency for stable operation after **post-layout**, (b) power-supply-rejection ratio (PSRR) in dB (c) V_{out} under supply ripple V_{DD}, (d) line transient response of the *LDO* under varying V_{DD}.

Design Objectives. The *ReFlex-LDO* is designed to: (i) achieve a minimum feasible dropout voltage close to the minimum V_{GS} required for the rated load of M_{pass}, (ii) maintain loop stability across load currents with only the small on-chip capacitor C_0, (iii) minimize quiescent current by biasing the amplifier stages with low currents, and (iv) ensure robustness against bending-induced variations.

Regulation and Stability Analysis **Line regulation.** From Figure 5.17 (a), we observe that the regulated output varies weakly with supply, yielding $\mathrm{LR_{out}} \equiv \Delta V_{\mathrm{out}}/\Delta V_{DD} \approx 4.0\,\mathrm{mV/V}$ around nominal V_{DD}. The internal reference in Figure 5.17(b) shows $\mathrm{LR_{ref}} \approx 3.0\,\mathrm{mV/V}$. confirming a stable reference drive. Moreover, Figure 5.18 (a) and Monte-Carlo (MC) runs further shows the stability of V_{ref} across temperature and process.

Load regulation. From Figure 5.18(b), the small-signal slope near the operating point gives $\mathrm{LR_{load}} = \left.\frac{\partial V_{\mathrm{out}}}{\partial I_{\mathrm{out}}}\right|_{\mathrm{nom}} \approx 45\,\mathrm{mV/mA} \Rightarrow R_{\mathrm{out,cl}} \approx 45\,\Omega$. The knee at $I_{\mathrm{out}} \approx 1.0\,\mathrm{mA}$ marks the onset of dropout.

Stability (AC). The Bode plot in Figure 5.19(a) indicates a DC loop gain of 52 dB, unity-gain bandwidth $f_{\mathrm{UGB}} = 3.0\,\mathrm{MHz}$, and phase margin $\mathrm{PM} = 63°$, meeting the $\geq 60°$ target.

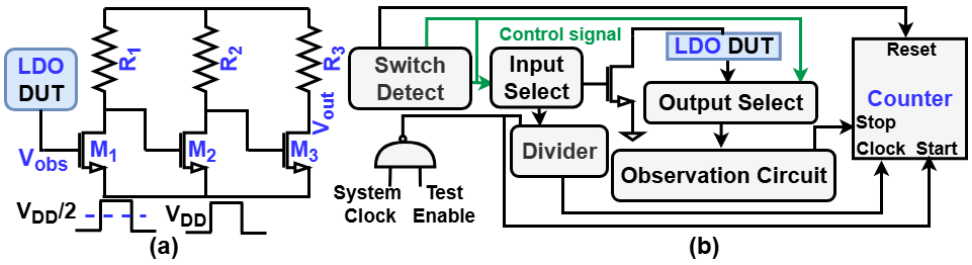


Figure 5.20.: (a) 1-bit *ReFlex-LDO* output using a chain of inverters; (b) Delay measurement using a counter.

Ripple/PSRR. With a supply ripple applied (Figure 5.19(b)), the time-domain attenuation corresponds to

$$\mathrm{PSRR}(f_r) = 20\log_{10}\left(\frac{\Delta V_{DD}}{\Delta V_{\mathrm{out}}}\right) \approx 34\,\mathrm{dB} \text{ at } f_r \approx 10\,\mathrm{kHz}.$$

Quiescent current. The design draws $I_q = 31\,\mu\mathrm{A}$ at no load.

The *LDO* regulator achieves tight line and load regulation with limited bias and a pF-scale output capacitor. The measured line regulation of $\approx 4\,\mathrm{mV/V}$ at V_{DD} (Figure 5.17a) implies that a 100 mV supply swing perturbs V_{out} by only $\sim 0.4\,\mathrm{mV}$ ($\sim 0.04\%$ for a 1 V rail); the reference shows an even smaller $\approx 3\,\mathrm{mV/V}$ (Figure 5.17b), ensuring a stable drive for the error amplifier. Also, the small-signal load regulation of $\approx 45\,\mathrm{mV/mA}$ (Figure 5.18a) yields $R_{\mathrm{out,cl}} \approx 45\,\Omega$, so a 100 µA load step causes only ~4.5 mV droop. The loop gain/phase (Figure 5.19a) show $A_0 \approx 52\,\mathrm{dB}$ ($\sim 4\times10^2$ V/V), giving a DC regulation error of $\varepsilon_{\mathrm{DC}} \le 1/(1+A_0) \approx 0.25\%$, with $f_{\mathrm{UGB}} \approx 3\,\mathrm{MHz}$ and $\mathrm{PM} \approx 63°$, ample margin for stable, well-damped transients. Despite the tiny $C_{\mathrm{OUT}} \approx 0.95\,\mathrm{pF}$, the time-domain ripple test as shown Figure 5.19(b-c) indicates $\mathrm{PSRR} \approx 34\,\mathrm{dB}$ at $\sim 10\,\mathrm{kHz}$ (about 50× attenuation), i.e., a 100 mV ripple on V_{DD} produces only $\sim 2\,\mathrm{mV}$ on V_{out}. The *LDO* also owns an outstanding transient response, as presented in Figure 5.19(d). It generates tiny overshoot voltages of 1.12 V and 0.72 V on falling and rising edges, and the settling times are only 30 µs and 120 µs, respectively. Finally, the quiescent current is just $I_q = 31\,\mu\mathrm{A}$, which corresponds to ~93 µW at 3 V and satisfies $I_q \le \eta I_{\mathrm{load,min}}$ with $\eta \approx 3.1\%$ for $I_{\mathrm{load,min}} = 1\,\mathrm{mA}$, which fits well for wearable/biomedical FE power budgets. Thus, these metrics, i.e., low mV/V line reg, low $R_{\mathrm{out,cl}}$, $\mathrm{PM} > 60°$, useful PSRR with pF-scale on-chip C_{OUT}, and 31 µA bias, demonstrate an *LDO* that is both energy-efficient and robust across corners.

5.3.3. Structural Delay-based BIST Methodology

We adopt a structural BIST approach inspired by [191], [249] that perturbs the circuit at an *injection node* and observes the resulting response at an *observation node* as a digitally measurable *time delay*. We redesign both the injection nodes and 1-bit observation chain so that a large digital disturbance can be applied without collapsing the *LDO* bias. Injection is realized by simple ON/OFF keying (e.g., shorting a passive or pulling a bias node). Observation is reduced to a 1-bit signal using an inverter chain; only lightweight switches and a counter are required, and a single observation circuit is shared across multiple injection points.

Node selection (injection/observation): Following the methodology, we select non-sensitive nodes for injection (bias rails and the feedback ladder) and use V_{out} as the observation node. Bypassing one feedback resistor with a switch creates a controlled step in the loop, which propagates to V_{out} and provides a clean, digitally detectable transition. This strategy was shown effective for an *LDO* by bypassing a divider leg and observing the output via a chain of inverters.

ON/OFF Keying Injection Circuits We use digital ON/OFF keying to inject a large, non-functional perturbation at selected, non-sensitive nodes. Because the goal is a *structural* delay (not a small-signal transfer function), the injected signal may change the operating point; this relaxation simplifies the hardware and makes the response easier to detect. Two practical templates are used: (i) bypassing a

passive element and (ii) bypassing a bias transistor. Both add only a single pass device per site, so multiple injection points is enabled to improve the fault coverage.

Bypassing passive components A resistor or capacitor can be shorted with a pass transistor placed across its terminals. Although the short is not ideal (finite R_{on}), the effect is equivalent provided $R_{\text{on}} \ll R_{\text{bypassed}}$. For capacitors, the response is exponential but still produces a large perturbation [249]. Shorting one feedback-divider leg may yield only a small output shift (set by the divide ratio), yet any detectable swing suffices for BIST.

Bypassing active components Bias rails work as convenient injection points: pulling a bias device's gate with a pass transistor perturbs multiple stages simultaneously and is among the least sensitive insert locations [249]. For an NMOS bias device M_{bias}, an injection NMOS M_{inj} pulls the gate low during the ON phase, limiting the M_{bias} current and shifting the bias voltage. The device M_{inj} must be sized so that its V_{DS} remains below the NMOS threshold in the ON state; even if M_{bias} is not fully shut off, the altered bias is sufficient to generate a strong disturbance.

Injection in ReFlex-*LDO* We use both templates: (i) bypass one leg of the feedback divider to produce a clear step at V_{out}, and (ii) pull selected shared bias gates to ground to perturb multiple sub-blocks at once. Both injection points are implemented with NMOS-only devices in the front end, and each point costs one pass device and a digital enable.

Observation Circuit To measure delay, the analog response at an observation node is first reduced to a 1-bit pulse. As the node's operating point is around mid-supply, a short chain of resistor-NMOS logic inverters (3 stages) (in Figure 5.20 (a) are connected directly to the node; a small change across the switching point produces a restored full-swing digital pulse. If a fault shifts the DC point so the inverters no longer toggle, the measured delay becomes effectively infinite (and is thus detectable). Inverters with shifted thresholds can extend this method beyond mid-supply, but increase sensitivity to process variation and make sharing across circuits harder. A counter as shown in Figure 5.20 (b) started at injection and stopped at detection measures the delay; the clock period T_{clk} bounds the resolution and is included in the guardbands for pass/fail. Specifically, thresholds are set as $\mu_{t_d} \pm 3\sigma_{t_d} \pm T_{\text{clk}}$.‘

5.3.4. Evaluation

We evaluate *ReFlex-LDO* using a structural delay-based BIST following the methodology in [249]. Transient SPICE simulations are run with mismatch-enabled Monte Carlo and PVT sweeps appropriate for the FE process. V_{DD} is swept over its operating range (e.g., ±10 %), load current over $I_{\text{load,min}} \rightarrow I_{\text{load,max}}$, and temperature over the qualified range (e.g., −20 °C to 80 °C). The on-chip output capacitor is $C_0 = 0.95\,\text{pF}$.

5.3.5. Experiment

Simulation flow All simulations are run in Cadence Spectre[6] with *Ocean scripts* after **post-layout** extractions using *PragmatIC's FlexIC-Gen3* PDK[272]. We evaluate: (i) **process corners** using the model sections `max/typ/min`; (ii) **Monte-Carlo** using the model's `mc` section (process and mismatch statistics enabled); (iii) **mechanical corner** for minimum bend radius $R = 5\,\text{mm}$ by overlaying an empirical bend model. For each condition we sweep V_{DD} and load current I_{LOAD} across the specified range and run N Monte-Carlo samples (*e.g.*, $N = 5000$).

[6] `https://www.cadence.com/en_US/home.html`

Fault list and injection Catastrophic faults are modeled as resistive opens/shorts on transistor terminals and passives: opens in $[1\,\mathrm{M\Omega},\ 1\,\mathrm{G\Omega}]$, shorts in $[1,\ 100]\,\Omega$. For each fault and each point, a transient run records t_d. Fault-free runs (across PVT and bend) form the nominal delay distribution.

Thresholding decision rule and fault simulation Delays for each injection/observation pair are characterized by 5000 Monte Carlo samples to obtain μ_{t_d} and σ_{t_d}; For each point, the FAIL window is a two-sided guardband

$$G = [\mu_{t_d} - 3\sigma_{t_d} - T_{clk},\ \mu_{t_d} + 3\sigma_{t_d} + T_{clk}],$$

computed from fault-free Monte-Carlo at each corner; production FAIL uses the intersection of guardbands across corners and bends. Faults are detected when t_d falls outside the guardband. Catastrophic faults are modeled as resistive opens/shorts at device terminals; both longer and shorter delays can indicate defects depending on the fault type. We apply a greedy selection of injection points to reach target coverage with minimal overhead.

FE mechanical corners Bending is modeled by perturbing the TFT parameters according to Table 5.2. For each radius r, we modify the nominal models as follows:

$$\mu \leftarrow \kappa_\mu(r)\,\mu_0, R_{\mathrm{sw1}} \leftarrow \kappa_R(r)\,R_{\mathrm{sw1,0}}, V_{\mathrm{th}} \leftarrow V_{\mathrm{th,0}} + \Delta V_{\mathrm{th}}(r),$$

Table 5.2.: Bending-induced parameter shifts predicted from reported TFT studies[91], [101]

Parameter	Flat (∞)	20 mm	10 mm	5 mm
μ	1.00×	0.95×	0.90×	0.70×
R_{sw1}	1.00×	1.05×	1.10×	1.30×
ΔV_{th} [mV]	0	±20	±40	±70

At circuit level this implies $g_m \rightarrow \kappa_\mu g_m$ and increased series resistance along the affected paths; r_o follows from the perturbed device model. Monte-Carlo runs are drawn around each radius by adding small zero-mean spreads to κ_μ, κ_R, and ΔV_{th}; the tabulated values serve as the corner means/bounds.

5.3.6. Result

Table 5.3 summarizes the fault coverage obtained for the proposed *ReFlex-LDO* under structural delay-based testing. A total of 140 structural and parametric faults were simulated across the circuit, including 73 faults within the *LDO* core, 62 faults in the integrated BIST circuitry, and 5 parametric variations. Out of these, 121 faults were successfully detected, yielding an overall coverage of 86.42%. For the *LDO* alone, 64 out of 73 faults were detected, corresponding to 82.19% coverage. The BIST circuitry achieved a comparable coverage of 91.94% (57 out of 62 faults), while parametric shifts showed slightly lower detectability at 80%. These results demonstrate that the structural delay-based methodology is capable of covering the majority of opens and shorts in both the analog *LDO* core and its digital-friendly BIST extensions, with coverage levels consistent with state-of-the-art analog test strategies.

Table 5.4 reports the impact of increasing the number of injection points on overall fault coverage of the *ReFlex-LDO*. The fault numbers were fixed at $N_{\mathrm{tot}} = 140$, and the set of injection locations was extended from a single feedback-leg bypass to eight distributed nodes across the reference, error amplifier, and output stages. With only one injection point, 121 faults were detected, corresponding to 86.4% coverage. Adding further injection points improved detection, reaching 125 (89.3%) with two points and 128 (91.4%) with three points. Beyond five points, the increase in coverage became marginal, saturating at 95.7% for seven and eight injection points. This trend highlights the trade-off between test complexity and coverage: while additional injection points improve detection of rare fault cases, most structural

defects are already detectable with the first few carefully chosen locations. The selected injection nodes including feedback-leg bypass, EA tail-starve, pass-gate discharge, V_{ref} pull, driver-node clamp, load step at V_{out}, shared bias-rail pull, and V_{DD} micro-droop were chosen to maximize structural delay defects with minimum hardware.

Table 5.3.: Fault coverage for *ReFlex-LDO*

type	simulated faults			detected faults			coverage
	total	shorts	opens	total	shorts	opens	
LDO	73	36	37	60	28	32	82.19%
BIST	62	30	32	57	28	29	91.94%
param	5	NA	NA	4	NA	NA	80%
Total	140	66	69	121	56	61	86.42%

Table 5.4.: Impact of multiple injection points(N_{tot} = 140).

Injection Points	1	2	3	4	5	6	7	8
Simulated	140	140	140	140	140	140	140	140
Detected	121	125	128	130	132	133	134	134
Coverage [%]	86.4	89.3	91.4	92.9	94.3	95.0	95.7	95.7

1) Feedback-leg bypass (R_9/R_{10}), 2) EA tail-starveM_7, 3) Pass-gate discharge M_{pass}, 4) V_{ref} pull $R_{8_{shunt}}$, 5) Driver-node clamp (M_6-M_9, 6) Load step at (V_{out}), 7) Shared bias-rail pull, 8) V_{DD} micro-droop.

Area Overhead and Test Time Table 5.5 shows that the proposed BIST adds only **4.2% area overhead** and consumes **1.8 mW** during test, which is almost 5.5% using multiple injection points, with a per-point time of **12 µs** and total test time of just ≈**1.84 ms**. This shows that delay-based BIST achieves high fault coverage at negligible cost in area, power, and time, making it practical for integration in *ReFlex*-LDOs.

Fault Coverage under PVT and Bending Figure 5.21 shows the impact of mechanical bending on the delay distributions and fault coverage of the proposed *ReFlex-LDO*. At the flat condition, the coverage is 95.5%, which gradually decreases as the bending radius reduces due to mobility degradation and increased parasitics. At 20 mm radius, coverage falls to 89.8%, while at 10 mm it remains 92.4%, and at the extreme 5 mm radius it drops to 86.7%. These results are consistent with reported TFT studies where mobility and threshold voltage vary under stress or strain, as summarized in Table 5.2.

Temperature variation was also evaluated, as shown in Figure 5.22. Starting from the baseline 27°C coverage of 86.4%, the distributions shift with temperature due to the negative temperature coefficient of the flexible resistors. Coverage remains above 88% at −20°C and 0°C, but decreases towards 82.0% at 80°C. Overall, the results indicate that while both bending and temperature variations affect fault coverage, the structural delay-based BIST maintains coverage above 95% without bending and 86.4% across nominal temperature.

Design-for-Test and Reliability Co-Optimization

The ReFLEX-*LDO* study emphasizes that reliable design and DfT must be co-optimized. Some design choices that improve *LDO* stability or regulation also make the delay signatures more distinguishable between healthy and faulty states, enhancing test coverage. Conversely, test structures must be designed

Table 5.5.: BIST overhead and test cost

Inj. Pts	1	2	3	4	5	6	7	8
ΔArea [%]	1.56	+2.54	+3.25	+3.96	+4.11	+4.23	+4.23	+4.24
ΔP_{test} [%]	+1.42	+1.45	+2.73	+3.92	+4.85	+5.50	+5.53	+5.56
ΔT_{tot} [ms]	1.11	1.12	1.24	1.36	1.48	1.60	1.72	1.84

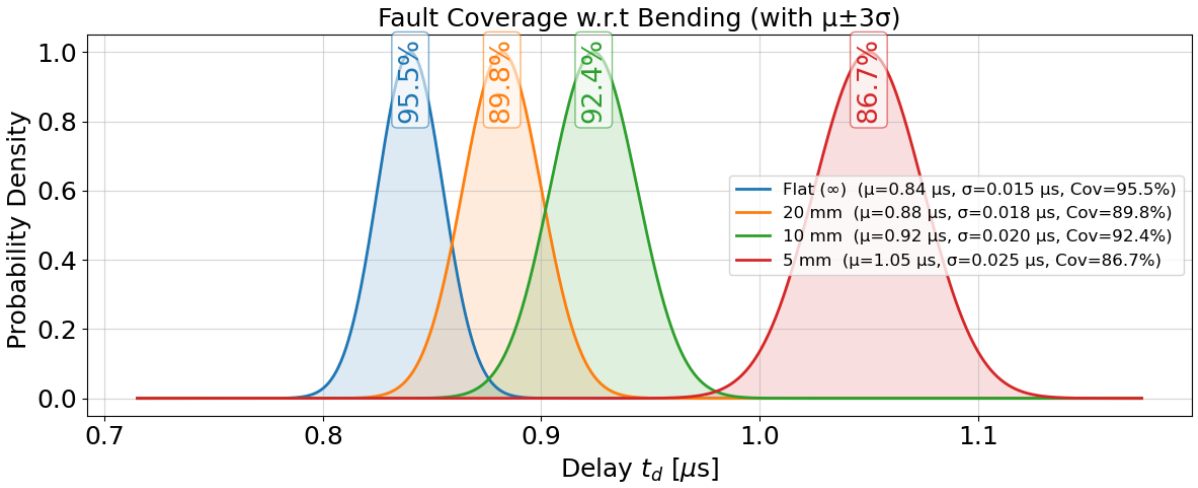


Figure 5.21.: Delay-based fault coverage under bending: PDF of path delay t_d for flat, 20 mm, 10 mm, and 5 mm; $\mu\pm3\sigma$ windows yield 95.5%, 89.8%, 92.4%, and 86.7% respectively.

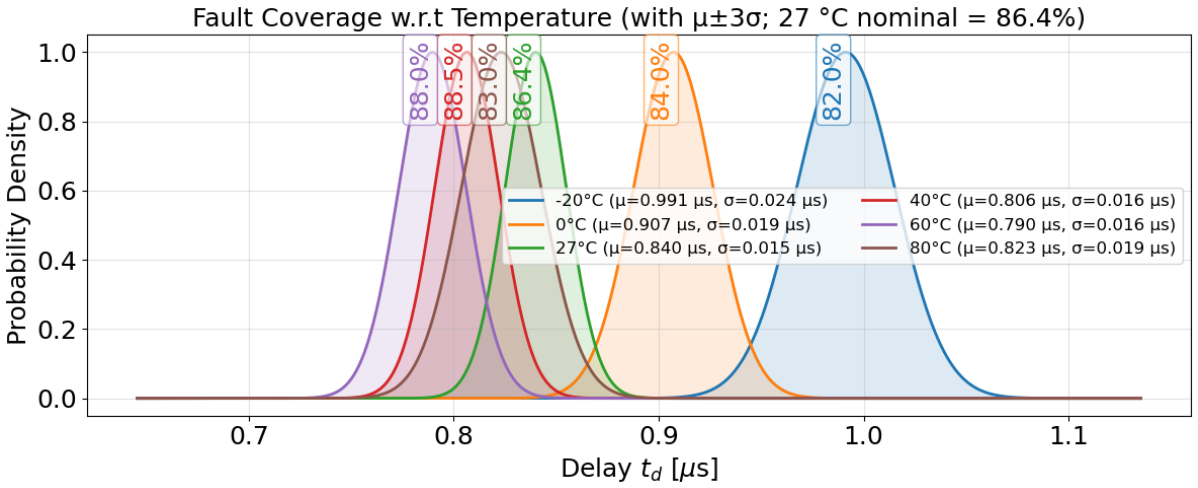


Figure 5.22.: Delay-based fault coverage vs. temperature: PDF of delay t_d at −20 °C, 0 °C, 27 °C, 40 °C, 60 °C, and 80 °C. Coverage is computed over $\mu\pm3\sigma$; the 27 °C is 86.4% respectively.

not to interfere with normal operation or significantly burden area and power. The final ReFLEX-*LDO* architecture demonstrates that both goals can be met simultaneously in printed technology.

5.4. Analog Error Correction (AEC) for Unary-Encoded Computing

Printed physical unclonable functions (p-PUFs) are attractive security primitives for PE because they exploit inherent process variations to derive unique device fingerprints. However, those same variations—

together with environmental changes and aging—severely degrade PUF reliability, i.e., the probability that a repeated challenge produces the same response bit. Reported reliabilities as low as 65% have been observed for some printed PUFs, far from the > 99% typically required in secure systems.

Conventional digital error-correcting codes (ECC), such as BCH or LDPC, are ill-suited for ultra-low-cost printed implementations: they require complex decoding logic, large numbers of devices, and substantial routing overhead. To address this, the work in *Efficient AEC for Printed Unary-Encoded Computing* proposes a mixed-signal AEC scheme that leverages unary-encoded bitstreams and a tiny analog majority voter to restore PUF reliability with much lower hardware overhead than digital ECC.

5.4.1. Proposed analog error correction Scheme

In this section, the proposed AEC scheme based on UE concept is described. At the final output stage, the output bitstream which are mapped to logic '1' and logic '0', should be stable sequence of 1's (all-1) or stable sequence of 0's (all-0). However, due to presence of defects and faults, there would be sporadic 0's or 1's in the respective sequences. This disrupts the stability and accuracy of the encoding. To address this issue, the proposed approach leverages a special case of unary encoding (UE), where the output is constrained to either an all-1 or all-0 sequence. By focusing on this scenario, AEC can be effectively implemented to eliminate sporadic errors (e.g., removing unexpected 0s from an all-1 sequence and vice versa. Our idea is to use some analog integrator based on the concept of TMV to correct such errors, and provide an efficient analog implementation of that scheme for PE. While our approach explicitly uses deterministic unary-encoded boundary cases (all-1 or all-0 sequences) for clear error detection, intermediate numeric values (e.g., 0.25 or 0.5) arising due to erroneous bit-flips carry meaningful probabilistic information. However, extension of our current AEC methodology to handle such intermediate values would require additional circuitry.

Please note that permanent failures, leading to always incorrect bitstreams or too high sensitivity to environmental conditions, can be detected using conventional testing techniques as a part of manufacturing test. Additionally, the unique feature of PE can be exploited in which optical inspection is utilized for manufacturing defect detection [187]. It is even possible to take advantage of another unique feature of this technology, namely post-fabrication tuning, as a unique feature of additive manufacturing to perform iterative refinements by printing additional resistive inks, to repair the printed devices [161], [212].

Therefore, the main focus of our proposed error correction scheme for printed UE is to correct sporadic bit errors which happen during runtime due to environmental noises. As a proof of concept, this approach is demonstrated for pPUFs. We apply a form of analog majority voter (integrator) which is applied on the output bitstream representing a deterministic logic 1 or 0, to filter out erroneous values in the sequence and clamping it to all-1 or all-0 sequence, accordingly. In order to keep the circuit complexity low, a mixed-signal approach is deployed, which combines digital components with an analog integrator, and finally generates a digital corrected bit sequence.

We use printed physical unclonable function (pPUF) as a case study. The idea of the proposed technique when applied to pPUFs is as follows. To correct bit errors of a single PUF output, the same PUF response is queried multiple times to generate a unary-encoded bit sequence. In case of ideal reliability (100%), this bit sequence should be all-1 (all-0), depending on the expected PUF response. However due to unreliability, this sequence contains erroneous values. Then the analog integrator circuit, as outlined above, is applied to that sequence to obtain a stable output. In case of multi-bit PUF response, the same principle is applied on individual PUF response bits to correct them individually. In other words, unlike traditional PUF error correction schemes which perform the error correction in space domain by using more PUF response bits (n) and extracting a smaller error-free response (k bits, $k < n$), here the redundancy is performed in the time domain and by generating PUF response multiple times and performing the correction in the time domain, an error-free PUF response of the same length ($k = n$) is obtained. Moreover, such time domain error correction is performed in an analog manner.

5.4.2. Unary bitstream generation

To generate the unary bitstream required for our AEC scheme, we utilize a simple and efficient method using a printed inverter-based 1-bit p-PUF [162]. The PUF is repeatedly switched on and off by connecting its supply voltage (V_{DD}) to a printed oscillator[134], which periodically activates the PUF and samples its output. Each power-on cycle yields a binary bit ('1' or '0') influenced by the inherent process variations and environmental conditions, resulting in a stochastic sequence.

The stochastic bitstream generated is then converted into deterministic unary bitstream using an analog integrator [246], which accumulates the voltage pulses corresponding to logic '1's in the stochastic sequence. As shown in Figure 5.23, the integrator's output voltage is compared against a threshold voltage using an analog comparator [246], producing a deterministic unary bitstream. The only assumption made about the PUF is that its output resistance should be lower than the input resistance of the transistors T_1 and T_2 of the bipolar encoder, and is satisfied due to the inherently high gate impedance of n-EGTs. To ensure proper signal propagation, the impedence should be matched, i.e., the subsequent logic circuitry processing the AEC output (AEC[X]) must possess a higher input resistance than the output resistance of the last inverter stage (formed by transistor T_7 and resistor R_4).

Method description In order to mitigate the runtime errors in the outputs of printed UE circuitry, our proposed method basically addresses such sporadic bit errors by counting the total bitstream over time, and performing a temporal majority vote among the bits [38]. A multi-bit output consists of several 1-bit output sequences, where X_i denotes the output bitstream at several subsequent points in time i, and the bitstream X is defined as:

$$X = (X_1, X_2, ..., X_L),$$

thus X is the bit sequence of a single output from time $i = 1 ... L$, where L denotes the bitstream length (or response readout time).

Furthermore, $j = 1 ... k$ refer to the k 1-bit output instances. In the following we consider only one 1-bit output instance, which has the correct value of logical '1'. The probability that we can correct the output using a temporal majority voting, with error probability p, can be calculated by the binomial distribution:

$$\begin{aligned} P(\mathrm{OUT}_j = \mathrm{OUT}_j^{\mathrm{ref}}) &= P(\#\text{'1's} > \#\text{'0's}) \\ &= \sum_{i=\lfloor \frac{L}{2} \rfloor + 1}^{L} \binom{L}{i} p^i (1-p)^{L-i} \end{aligned} \tag{5.16}$$

where OUT_j is the error corrected output bit, and OUT_j^{ref} is the initial, error-free bit (assumed to be '1' here: OUT_j^{ref}='1') and $\#'1's$ / $\#'0's$ are the number of measured '1's and '0's of this output stream OUT_j. Note that the binomial distribution assumption in Equation 5.16 explicitly relies on modeling each bit measurement as an independent Bernoulli trial. Practically, this means each sampled bit is assumed to have the same fixed bit-error probability p and is statistically independent from all other sampled bits. This assumption is justified in our scenario because the PUF is reset by cycling its power between subsequent measurements, effectively ensuring independent and identical conditions for each sampling event. Thus, the binomial distribution accurately describes the probability of successful error correction using TMV for the given bitstream length L and error probability p. So Eqn. (5.16) formulates an upper bound of the error correction capabilities of a TMV circuit, and increases with L, i.e. the length of the evaluated bitstream increases, thus, in theory, unlimited error correction can be performed for large L. This way, we can have a trade-off between the performance and the length of burst errors to be corrected. A more detailed discussion is provided in paragraph 5.4.4.

5.4.3. Circuit design

In general, capacitive circuits can be deployed to accumulate voltage pulses in the time domain. This can also be considered as an analog integration over a total bitstream, such as the outputs of an UE

Table 5.6.: Design parameters of the AEC circuit in n-EGT technology

$R_{1,2}$	R_3	R_4	$V_{th1,2,3,5}$	V_{th4}	V_{th6}	V_{th7}	C_1
100kΩ	850kΩ	200kΩ	126mV	250mV	312mV	2.1mV	100nF

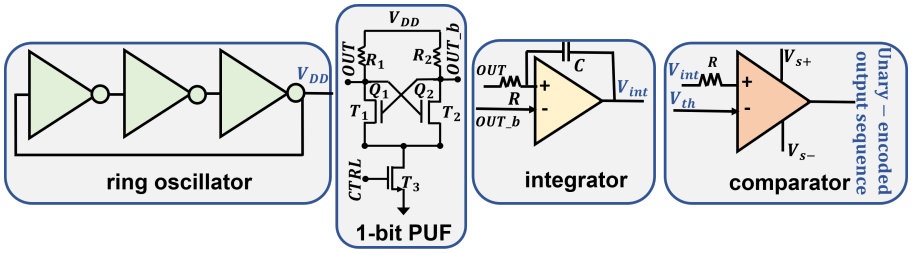


Figure 5.23.: Schematic of the unary bitstream generation using pPDK[134], [162].

circuit or sporadic (and thus erroneous) response of an unreliable PUF instance. In combination with an inverter-based step function, on average the errors are filtered out as long as the probability of the bit error in the bitstream is lower than 50%. The proposed AEC circuit is illustrated in Figure 5.24. The circuit components are explained in the following, step by step. Exemplarily, for the target technology we chose the n-EGT technology and consequently only n-type transistors are deployed[200]. However, the circuit can also be designed in another PE technology which enables complementary-logic, by replacing the pullup resistors R_1, R_2, R_4 with p-type printed transistors. The design parameters of the circuit are provided in Table 5.6. The design parameters were empirically found by circuit simulations using a process design kit for n-EGTs, which is based on measurements of printed n-EGTs and resistors [160][150]. The circuit was operated at $VDD = 1\text{V}$ and $VSS = -1\text{V}$.

Bi-polar encoder The bitstream X of the output response is first passed to the bi-polar encoder. Here, the bits '0' and '1' are mapped to '-1' and '1', using the following transfer function:

$$\widetilde{X}_i = 2X_i - 1$$

Due to this, an incoming logical '1' bit turns transistor T_2 on, and pulls the input of the discharge unit to VSS (discharging the capacitor C_1 to VSS). On the other side, if the output bit is '0', T_3 is turned on via the inverter (T_1,R_1) and pulls the bi-polar encoder output to VDD. It is important to note here, that this transformation actually maps '0' and '1' to '1' and '-1', respectively, and not to '-1' and '1' (which can be easily achieved by interchanging VSS and VDD in the bi-polar encoder circuit). The reason for this is part of an area reduction of the AEC, as explained in subsection 5.4.4.

By using this bi-polar encoding of the output bitstream, the switching threshold of the step function (step(z)), which is used to determine the AEC output, is set to 0V. This leads to a more reliable operation of the AEC, as it is insensitive to variations of the capacitance value C_1, the only analog component in the AEC circuit.

Discharge unit The purpose of the discharge unit is to reset the AEC to its initial state, i.e. discharging the capacitor voltage to 0V. The discharging cycle is performed when the reset signal RES is pulled up to VDD (see Figure 5.24). Subsequently, the transistor T_6 is switched on, connecting the capacitor C_1 to ground, and transistor T_4 is turned off through the inverter (T_5, R_2) to disconnect the transistor from the bi-polar encoder. During AEC operation, RES is constantly set to logical '0'.

Voltage integrator In the voltage integrator, the bitstream coming from the discharge unit is summed up in the capacitor C_1. The resistor R_3 can be used to tune the capacitor charge and discharge time. If VDD is applied through R_3, the capacitor is loaded to a positive voltage, and when VSS is applied, the capacitor is unloaded to a negative voltage. The absolute capacitor voltage level is proportional to

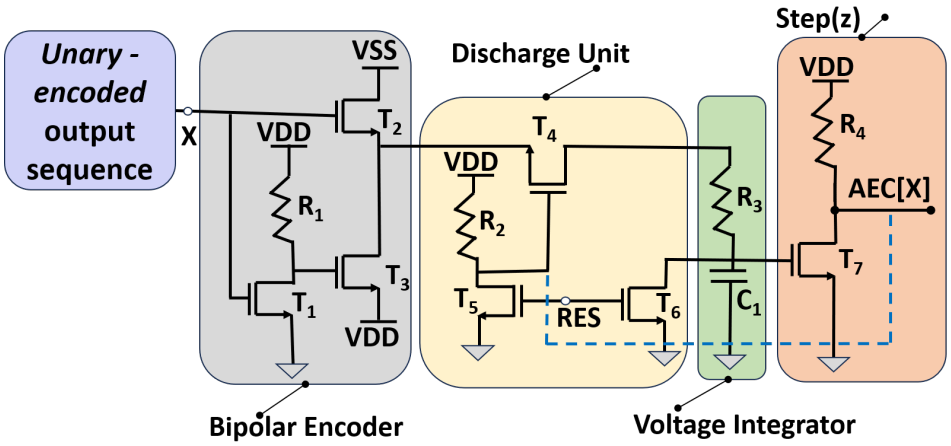


Figure 5.24.: Schematic of the AEC circuit for error correction connected to a 1-bit output (e.g. PUF instance).

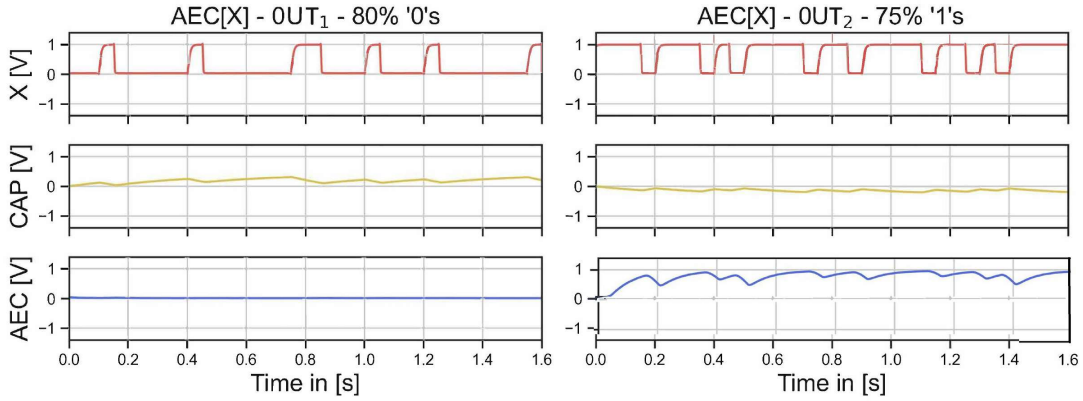


Figure 5.25.: Simulation of the printed AEC for two output bitstreams and L=32. The first output Figure 5.24 (a) (OUT_1) produces 80% '0's, which are applied as X to the AEC. As expected, the capacitor voltage CAP is positive and the AEC output OUT is pulled down to 0V. In contrast, in Figure 5.24 (b), the OUT_2 produces 75% '1's, and the AEC output is pulled up to VDD. Consequently, the majority vote is working in both examples.

the number of positive (or negative) voltage pulses. Again, the capacitance value is not crucial for the functionality of the circuit, as the threshold of the capacitor voltage is at 0V, but C_1 should be larger than the parasitic capacitance of the transistors (which is for n-EGTs about 10nF). The only requirement for correct operation is that the transistor T_7 of the output buffer has a transistor threshold voltage of 0V. The functionality of this device can be mathematically described as:

$$\mathrm{CAP} = \sum_{i=1}^{L} (2X_i - 1) \tag{5.17}$$

Step function The step function consists of one inverter, where the transistor threshold voltage of T_7 is 0V, which is achievable using transistor channel geometry scaling [195]. The output of the AEC can be described as:

$$\mathrm{AEC} = \overline{\mathrm{step}(CAP)},$$

where the step function is defined as:

$$\mathrm{step}(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases} \tag{5.18}$$

In this component, the final result of the majority vote is obtained. If more '1's in the PUF response bitstream are obtained, the capacitor voltage of the node *CAP* is negative, and the inverter output is at logic '1'. In the opposite case, if more '0's are received, the *CAP* voltage is positive, and the AEC output is at logic '0'. This resembles the behavior of a majority vote and enables the error correction of this circuit. Moreover, the closed-loop feedback path enhances stable circuit operation despite variation in circuit components and manufacturing processes.

Please note that the electrical behavior of the n-EGT is symmetric regarding the drain and source contacts as no bulk connection is deployed. In fact, flipping the n-EGT and reversing the drain and source contacts has no effect on the electric characteristics or on the surrounding circuitry it is connected to. Transistors T_2 and T_3 can be considered as transfer gates, which connect T_4 either to VSS and VDD. In a DC analysis, they can also be considered as voltage dividers as no current flows through T_4 during capacitor charging (T_6 is off, no current through the gate of T_7 (gate currents are very small) or C1 and thus no current through R_3 and T_4). The ability of the voltage divider to pull the node common with T_4 to VSS or VDD is determined by X. When X is '0', the node is pulled up more to VDD than when X is '1'. The node has then in the first case a negative voltage and in the second case a positive voltage, and C1 is discharged or charged. The correct operation of this circuitry was also validated using a simulation-based approach with a semi-empirical n-EGT model fit to actual measurements of printed n-EGTs in a laboratory setup [150].

Since in Table 5.6 different threshold voltages are considered for different transistors, it is worth mentioning that the controlling of the threshold voltages of n-EGTs is performed simply by transistor channel sizing [195]. Therefore having n-EGTs with different threshold voltages in same circuit is matter of channel sizing and quite feasible from fabrication point of view. Moreover, as the AEC circuit is a digital circuit, small variations on the threshold voltages do not affect the correct output behavior. While the design parameters in Table 5.6 were found empirically, setting Vth of T_7 to 0mV will not change the circuit behavior.

5.4.4. Circuit optimization

In order to reduce the transistor count of the $AEC[X]$ implementation, the inputs X_i and the output of the step function in the AEC circuit were inverted, otherwise an additional inverter had to be inserted at the output, requiring an additional transistor and resistor. Despite these modifications, still the correct output behavior is obtained:

$$AEC[X] = \overline{\text{step}\left(\sum_{i=1}^{L}(2\overline{X_i}-1)\right)} = \text{step}\left(\sum_{i=1}^{L}(2X_i-1)\right) \tag{5.19}$$

This equation is true, as the step function is point symmetric with respect to its argument ($\overline{\text{step}(z)} = \text{step}(\overline{z})$) and the summation over bi-polar encoded X_i is a linear function:

$$\overline{\left(\sum_{i=1}^{L}(2\overline{X_i}-1)\right)} = \left(\sum_{i=1}^{L}(2\overline{\overline{X_i}}-1)\right) = \left(\sum_{i=1}^{L}(2X_i-1)\right) \tag{5.20}$$

The input-output relation of the AEC is illustrated in Figure 5.25, where transient simulations of two output bitstreams (OUT_1 and OUT_2) in n-EGT technology are depicted. In the first output bitstream (OUT_1), the bit error probability is 20%, while in the second bitstream (OUT_2) the bit error is 25%. The correct response for OUT_1 is logical '0', which is correctly extracted by the AEC. The reference response for OUT_2 is logical '1', and the AEC pulls the output correctly up to VDD.

Increasing the unary-encoded bitstream length The error correction capability of the AEC can be improved, by increasing the length L of the output bitstream. For instance, this can be done by generating more instances of 1-bit PUF instance. Without loss of generality, we assume in the following that the

correct output response is '1' and the bit error probability is p, thus $\mathbb{E}[X_i] = 1 - p = q$. The expected probability that the output response is classified correctly is:

$$\begin{aligned}
\mathbb{E}[\,\mathrm{AEC}[X]] > 0 &\iff \mathbb{E}\left[\mathrm{step}\left(\sum_{i=1}^{L}(2X_i - 1)\right)\right] > 0 \\
&\iff \mathbb{E}\left[\sum_{i=1}^{L}(2X_i - 1) > 0\right] \\
&\iff \sum_{i=1}^{L}(2\mathbb{E}[X_i] - 1) > 0 \\
&\iff \sum_{i=1}^{L}(2q - 1) > 0 \\
&\iff L * (2q - 1) > 0
\end{aligned} \tag{5.21}$$

We assumed here that the subsequent bit errors are independent and identical binomial distributed random variables. From this equation it is obvious that an increase in L under the assumption that the error probability $p < \frac{1}{2}$ (and thus $q > \frac{1}{2}$), the expected value increases monotonically:

$L(2q-1) > (L-1) \times (2q-1) > (L-2) \times (2q-1) > ... > (2q-1)$.

The monotonically increasing expected function of a correct majority voting indicates that the probability of an uncorrected error decreases.

Moreover, our AEC method is only applicable when the bit error probability of the output is not higher and not equal to $\frac{1}{2}$ ($p < \frac{1}{2}$), which is reasonable as AEC is in essence a majority voter function.

The analysis above can be performed for the case where the expected response is logical '0', accordingly, just changing the criterion for successful error correction to: $\mathbb{E}[\,\mathrm{AEC}[X]] = 0$

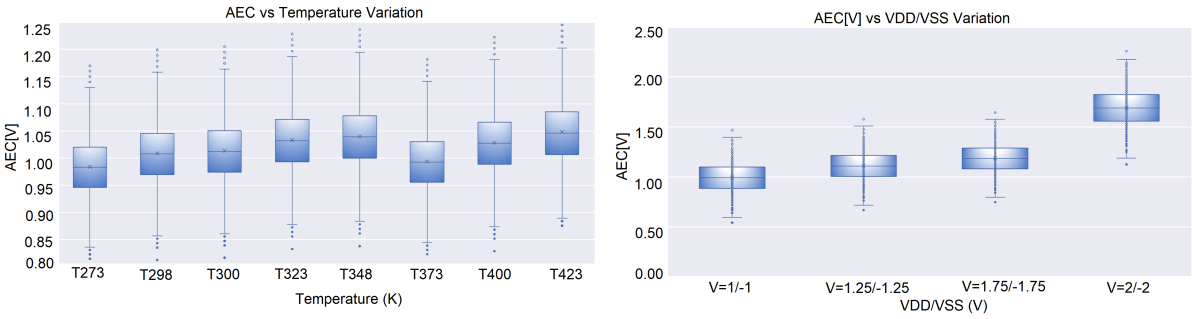


Figure 5.26.: Impact of (a) temperature (K) and (b) supply voltage (VDD/VSS) variation on AEC[V]

Burst Errors In general it is possible that the AEC circuit is subject to runtime variations such as voltage fluctuations or temperature variations, which can result in erroneous consecutive bits in the output bitstream. Such burst errors can have a substantial impact on the reliability depending on the probability of their occurrence and burst error length. In the worst case, the burst errors do not overlap over time and the number of total bit errors in the output can be computed by $Errors = B \cdot B_L$, where B is the number of bursts and B_L is the bit error length of the burst.

Although the underlying probability distribution of burst errors are different to the assumed binomial distribution described before, the expected reliability calculation in paragraph 5.4.4 can be reused for the worst case burst error scenario by replacing the bit error probability p by $p' = B_L \cdot p$. E.g., when $B_L = 3$, the bit error probability is three times higher: $p' = 3 \times p$. Also the bit error probability threshold under which a correct output response extraction by AEC can be performed is reduced to $p < \frac{1}{2 \times B_L}$. However, if this threshold criterion is met, the reliability of the AEC method can be arbitrarily increased by using larger bitstream length L as pointed out in paragraph 5.4.4. As the burst error problem can be

converted into a single-bit error problem by using the above transformation of the bit-error probability, the experimental evaluation in the following is limited to a bit-wise binomial distribution.

Please note that when the temperature variations cause permanent impact on the output bitstream, for instance switching from 273K to 423K forces the output to be always 1 instead of 0, such cases cannot be corrected by the proposed scheme. This is because the error correction is across the entire output bitstream and increasing the bitstream length does not change the outcome. These cases are categorized as permanent failures, which can be identified during manufacturing test or optical inspection [187], and afterwards repaired using post-manufacturing tuning [161], [212].

5.4.5. Variation Analysis of AEC Circuit

This section presents an evaluation of the proposed AEC method, focusing on its performance against temperature and supply voltage fluctuations, and transistors' threshold voltage variation. These performance analysis are critical in demonstrating the robustness and reliability of the circuit in practical scenarios. Environmental conditions were considered to study the circuit's robustness by varying the ambient temperature and supply voltage, and are evaluated for pPUFs. The AEC's efficacy was also evaluated through a series of Monte-Carlo simulations where the threshold voltage and the output (AEC[V]) were measured at a fixed time of 1s w.r.t. Figure 5.25 (b).

Temperature (T) The Figure 5.26 (a) illustrates the AEC circuit's performance across a range of temperatures. As shown, the AEC circuit maintains a relatively stable output (AEC[V]) despite variation in temperature from 273K to 423K. This stability is crucial indicating that the AEC can reliably compensate for the intrinsic variability and environmental sensitivities of PUFs, related to the operating conditions. While the medians are stable, there is noticeable variability in the spread of the plots at different temperatures. The outliers present and the width of the boxes suggest that there are still some temperature effects on the circuit's output. However, since the majority decision principle inherently averages out such deviations, the temporal majority voter design helps to ensure that these do not significantly impact the final output.

Supply Voltage (VDD/VSS) The graph in Figure 5.26 (b) shows the performance of the AEC circuit in response to supply voltage variation, which is critical for the reliability of PUFs that utilize AEC circuits. The data is represented across four different supply voltage pairings —1/-1, 1.25/-1.25V, 1.75/-1.75V, and 2/-2V—highlighting how the AEC [V] responds to changes in the VDD/VSS. As shown in Figure 5.26, at a lower voltage of 1/-1V, the output voltage exhibits minimal variability with a few outliers. As the voltage increases, there is a slight decrease in median at 1.25/-1.25V but with a consistent variability. At 1.75/-1.75V, the median output increases, and variability becomes more pronounced, with the trend continuing more at 2/-2V where both the median and the spread of AEC[V] significantly increase. This indicates that while the AEC circuit maintains a fairly stable output across standard operating voltages, higher voltages introduce greater variability and outliers, potentially challenging the PUF's reliability and security in extreme variation. Thus, we need to take care in the design and application, ensuring that they operate within a voltage range (preferably upto 1V) where the AEC circuit can effectively manage inconsistencies without compromising the outputs.

Threshold Voltage (V_t) We present an analysis of the AEC circuit's response to threshold voltage variations across multiple transistors. As shown in Figure 5.27, the study evaluates AEC output voltage as a function of threshold voltage for transistors T_1, T_2, T_3, T_4, T_5, T_6, and T_7. The results indicate a generally stable AEC output across varying threshold voltages, with median values maintaining output voltage close to 1V, particularly for transistors T_4, T_6, and T_7. This stability is crucial for ensuring the reliability of the error correction mechanism in printed UE, where component variability can significantly impact computational accuracy. Notably, all the transistors demonstrate a robust tolerance to a broader range of voltage thresholds, suggesting these components are particularly effective in maintaining circuit integrity

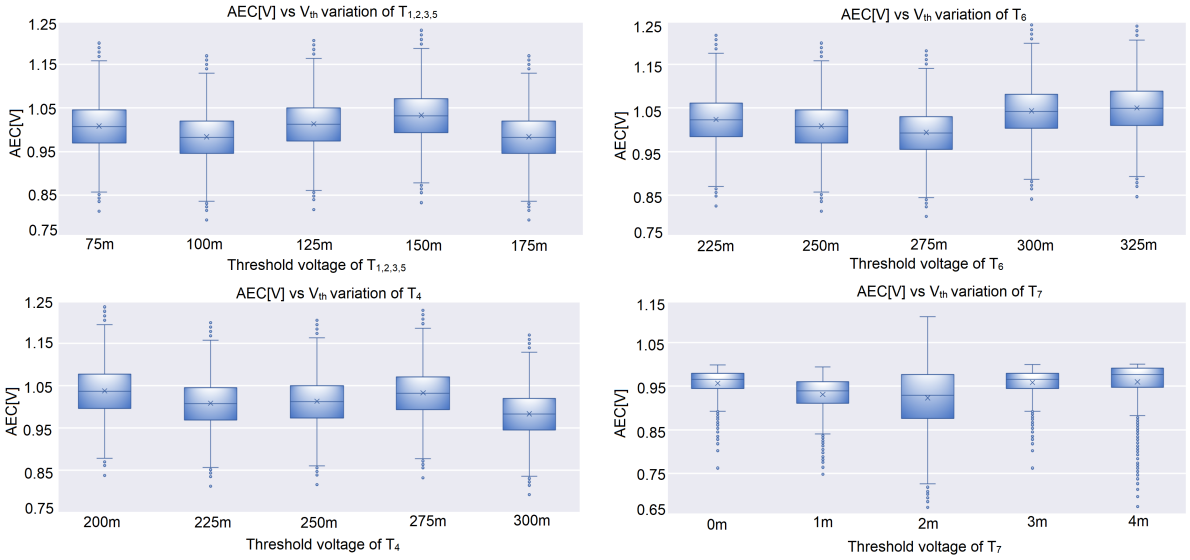


Figure 5.27.: Impact of (a) T_1, T_2, T_3, T_5, (b) T_4, (c) T_6, (d) T_7 transistor's V_{th} on AEC[V].

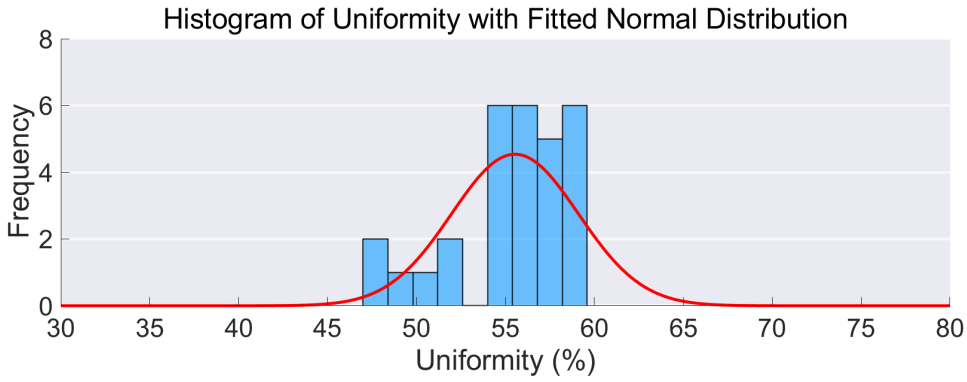


Figure 5.28.: Uniformity of pPUFs with distribution of 1's and 0's.

under various resource constrained environments. The interquartile range in these plots is relatively tight, indicating minimal variability, which is essential for the high reliable security-centric applications.

Table 5.7.: Evaluation of hardware implementation of 8-bit PUF and expected error probability at 20% BER for proposed and baseline approaches. Bitstream length is set to L = 8. Energy efficiency is abbreviated by "EE".

Metric (Unit)	No-ECC	PC (Baseline)	Proposed AEC
Reliability (%)	~75%	~85%	>99% (↑ **1.16**×)
Area (mm^2)	4	24	6 (↑ **0.25**×)
Power (μW)	800	6303	1282 (↑ **0.20**×)
Delay (ms)	120	158	400 (↓ **2.53**×)
Transistors (#)	40	222	56 (↑ **0.25**×)
EE (Bits/Js)	60,483	50,456	38,989 (↓ **0.77**×)

5.4.6. Case Study: Printed PUF (p-PUF)

In this section, the proposed AEC method is evaluated for pPUFs and compared with a conventional parity-check-based (PC) technique. PC was chosen as the reference approach, as it requires the lowest

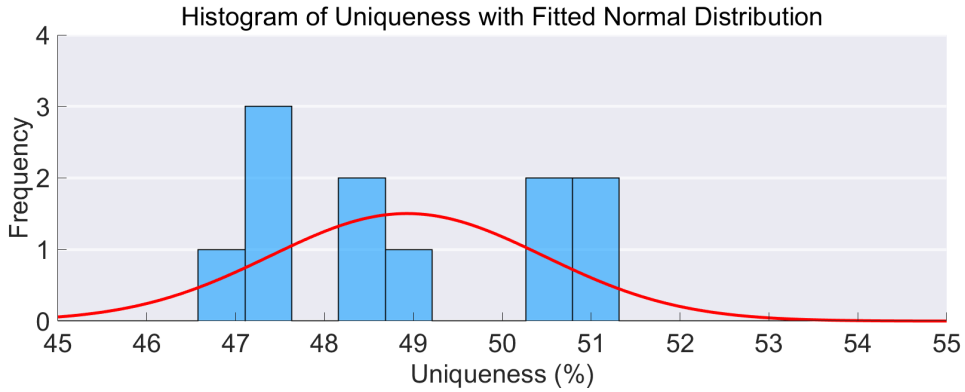


Figure 5.29.: Uniqueness of pPUFs (inter-HD)

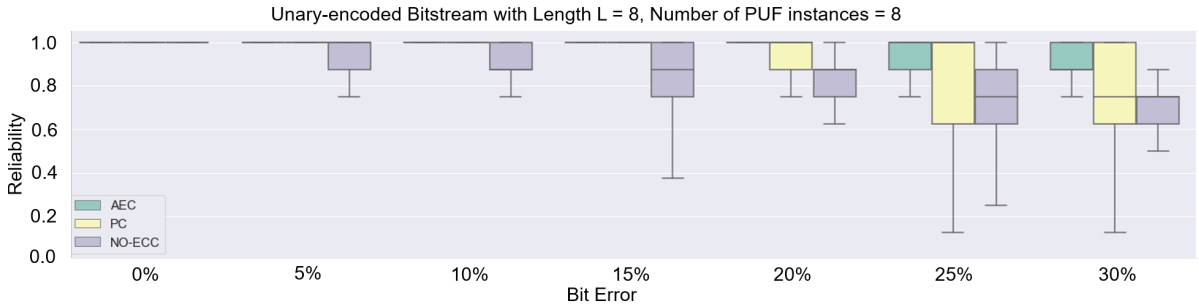


Figure 5.30.: Monte Carlo simulation (1000 samples for each bit error corner): distribution of the PUF reliability for proposed method vs conventional approach with unary-encoded bitstream length of L=8 and 8 1-bit PUF instances. Whiskers are related to the low and high quartiles, middle line represents the unary-encoded mean. The reliability for AEC is nearly 100% and is shown by grey lines.

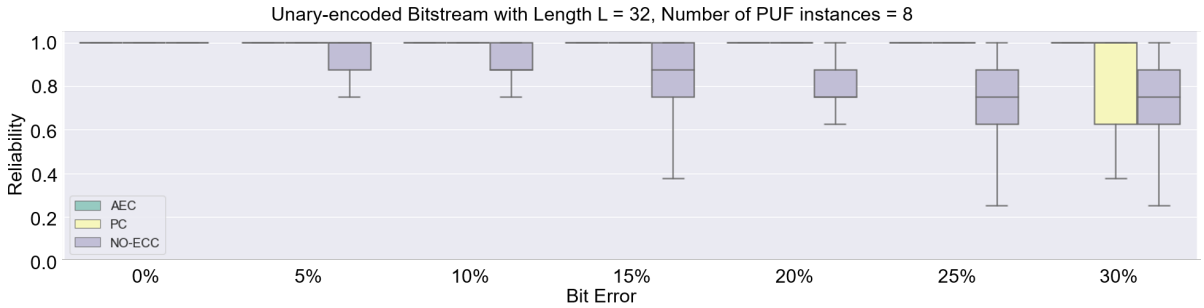


Figure 5.31.: Monte Carlo simulation (1000 samples for each bit error corner): distribution of the PUF reliability for proposed method vs conventional approach with unary-encoded bitstream length of L=32 and 8 1-bit PUF instances.

hardware footprint compared to other existing digital PUF error correction schemes. Two analyses have been carried out: first the hardware footprint of both techniques were estimated, and second, the reliability improvements were quantified with respect to different PUF bit error probabilities. For the evaluation, n-EGT-based circuits were chosen as the target technology. For the estimation of the required hardware resources, a high-level synthesis tool (Synopsis design compiler [50]) in combination with printed process design kit for n-EGT technology which contains a physical design standard cell library [160] was deployed. An 8-bit PUF, i.e., eight 1-bit PUF instances based on the design in [115] were synthesized with the target of maximum performance. Due to this, the error correction is computed in parallel among the PUF instances.

Given that a PUF is a highly design-specific hardware component, it is important to thoroughly assess its quality before integrating its output into an AEC system. This assessment is crucial to ensure that the PUF meets the necessary standards for reliable and secure operation. Consequently, we conduct a

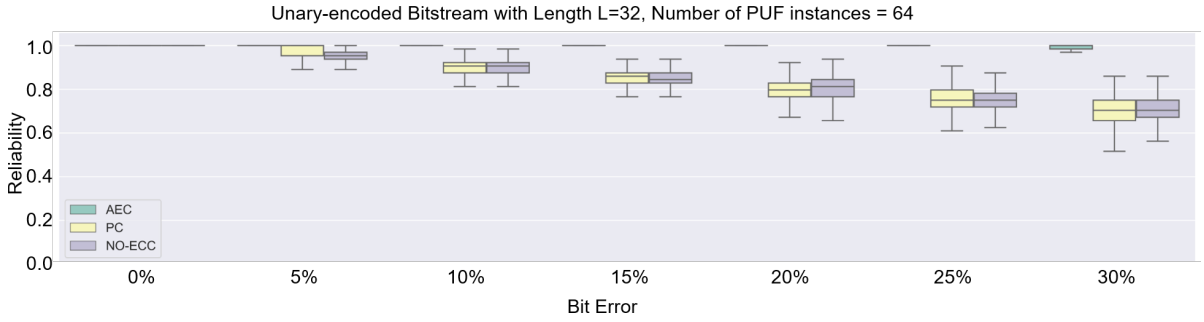


Figure 5.32.: Monte Carlo simulation (1000 samples for each bit error corner): distribution of the PUF reliability for proposed method vs conventional approach with unary-encoded bitstream length of L=32 and 64 1-bit PUF instances.

comprehensive analysis of the other two important metrics of the PUF except reliability: uniformity and uniqueness. This thorough evaluation guarantees that the PUF can deliver consistent and dependable outputs suitable for secure applications. Therefore, we show histograms, along with their fitted normal distributions and provide a quantitative basis for assessing the uniformity and uniqueness of PUFs, ensuring their robustness and reliability in practical applications.

The histogram plot shown in Figure 5.28 illustrates the distribution of uniformity percentages for a given PUF sample set, overlaid with a fitted normal distribution curve. The x-axis represents the uniformity percentage, ranging from 30% to 80%, while the y-axis indicates the frequency of observations. The uniformity data show a central tendency around 55%, indicating that most PUF instances have a uniformity near this value. The frequency peaks at around 6 observations for the central bins, with fewer occurrences as the uniformity percentage moves towards the extremes. In an ideal case, the value should be 50% but in practical, 55% is also acceptable. The overlaid normal distribution curve (in red) suggests that the data roughly follows a normal distribution, although there are some deviations, particularly in the lower uniformity range. This uniformity histogram helps in understanding how evenly distributed the output bits of the PUF are, which is crucial for ensuring the functionality of the PUF in cryptographic applications.

The histogram plot in Figure 5.29 shows the distribution of uniqueness percentages for the same PUF sample set, again with a fitted normal distribution curve. The x-axis represents the uniqueness percentage, ranging from 45% to 55%, and the y-axis represents the frequency of observations. The uniqueness data are centered on 50%, which is the ideal value indicating that each PUF instance is distinct from the others. The frequency distribution shows a peak at around 3 observations near the 50% uniqueness mark, with a rapid decline in frequency as the uniqueness percentage deviates from this central value. The fitted normal distribution curve (in red) closely matches the observed data, implying that the uniqueness distribution approximates a normal distribution. This histogram is essential for evaluating how uniquely each PUF instance can be identified, which is fundamental for the security properties of the PUF.

One AEC circuit is deployed for each PUF instance, which results in 8 AECs for the 8-bit PUF. For the PC we assumed that the bit error probability is 20%, as it determines the number and size of parity check units (one parity check unit can only reliably detect up to 2 bits of errors).

The hardware synthesis results, as illustrated in Table 5.7, clearly demonstrate the efficiencies of our proposed AEC method. Although, the No-ECC scenario reduces area by 83% (4 mm^2 vs. 24 mm^2), power by 87% (800 µW vs. 6303 µW), delay by 24% (120 ms vs. 158 ms), and transistor count by 82% (40 vs. 222) compared to PC, it suffers a significant reliability drop of 12% (75% vs. 85%). On the other side, our proposed approach uses only 25% area, consumes 20% power and requires 25% transistors compared to the conventional PC approach. Also, the bit-wise circuit propagation delay is 50ms, which corresponds to 400ms for a bitstream length of 8, 2.5× larger than the PC. We don't consider this as a major concern, as high-performance computing is not the primary objective in PE and the target applications of PE usually can tolerate higher latencies. For instance sensor readouts occur only periodically every 100 of milliseconds [61], [121], seconds[90] or even minutes [77]. Due to the larger PUF latency, the proposed method has 23% reduced energy efficiency compared to the baseline.

For the reliability assessment, 1000 Monte Carlo simulations were performed per PUF bit for both AEC and PC, with respect to different run-time induced bit errors, which we assumed to be independently distributed. In each Monte Carlo sample, the erroneous bitstream is drawn from a binomial distribution, where p equals the bit error probability. The results of the Monte Carlo simulations are presented in Figure 5.30, where the length of the bitstream was set to $L = 8$. Besides AEC and PC, also no error correction (NO-ECC) was included as a baseline. Several observations can be made from these results: First, both methods increase the PUF reliability for all bit error levels. Second, our method results in higher PUF reliability compared to PC. And finally, the reliability can be restored to virtually 100% ($> 99.9\%$ since no error was observed in 1000 simulations) for a bit error probability of up to 20%.

As discussed in paragraph 5.4.4, enhancing the length L of the bitstream significantly augments the capacity to restore reliability. We further validated this concept through a comprehensive experiment wherein we continued with the same Monte Carlo simulations, but this time using an extended bitstream length of $L = 32$. The results from these simulations are shown in Figure 5.31, where it is evident that the AEC method can now achieve a virtually perfect reliability level—exceeding 99.9%—even at a 30% bit error probability. This increase in reliability, however, comes at the cost of an extended PUF readout time, which is now four times longer than before, as calculated by the ratio $\frac{32}{8} = 4$.

On the other hand, the reliability observed with the Parity Check (PC) method under these conditions diminishes to 60%. This contrast in performance underlines the limitations of the PC method when dealing with higher bit error probabilities. In practical applications, the choice of bitstream length must be strategically determined based on several key factors: the anticipated bit error probability, the required level of reliability, and the constraints on the PUF readout time. This decision-making process highlights the inherent flexibility of the AEC method. Specifically, adjusting the bitstream length does not necessitate any modifications to the actual circuit design of the AEC. Therefore, this approach allows system designers to finely balance the trade-off between achieving optimal reliability and maintaining acceptable performance levels without necessitating structural changes to the error correction architecture. This adaptability makes AEC a versatile and potent solution for enhancing the reliability of systems susceptible to significant error rates.

Finally, Figure 5.32 illustrates the reliability outcomes for a PUF configuration consisting of 64 bits. Notably, for the proposed AEC method, we observe no significant difference in reliability when compared to the performance noted with the smaller 8-bit PUF size. This consistent reliability across different PUF sizes can be attributed to the fact that each PUF instance operates with a dedicated AEC unit, and error correction is effectively carried out over time. This creates a scenario where reliability is primarily dependent on the number of evaluations, denoted as L, rather than the size of the PUF itself.

Table 5.8.: Reliability (%) at different bit error rate (BER)

PUFs (#)	Length (L)	Scenario	BER (%)	Reliability (%)
8	8	No ECC	20/25/30	80/75/70
		PC	20/25/30	90/85/80
		AEC	**20/25/30**	>**99.9/98/9**5
8	32	No ECC	20/25/30	80/65/55
		PC	20/25/30	92/88/80
		AEC	**20/25/30**	>**99.9/99/98**
64	32	No ECC	20/25/30	80/65/55
		PC	20/25/30	88/85/78
		AEC	**20/25/30**	>**99.9/98/97**

Overall, our observations indicate that at higher bit error rates (20–30%), the reliability without error correction deteriorates significantly, dropping as low as 55–75%. However, the proposed AEC maintains

reliability consistently above 98%, clearly validating its efficacy even in extreme error cases as shown in Table 5.8.

In contrast to the scalability of the AEC method, the parity-check-based (PC) approach exhibits limitations when applied to larger PUF sizes. Specifically, the PC method is limited in its ability to detect and correct errors beyond a certain threshold per PUF instance. As clearly shown in Figure 5.30, Figure 5.31, Figure 5.32, the effectiveness of the PC method diminishes with increased PUF sizes, to the point where it mirrors the reliability levels of having no error correction mechanism in place at all.

To address this limitation, one potential strategy involves segmenting the larger PUF array into smaller sections, each managed by its own dedicated PC unit. For example, a 64-bit PUF could be divided into eight separate segments, each comprising 8 bits. Although this segmentation approach can enhance the overall reliability to levels comparable to those observed with the 8-bit PUF configuration, it necessitates a significant expansion in the hardware footprint. This increase in complexity underscores a critical trade-off between achieving desired reliability through segmentation and managing the expanded hardware footprint. Thus, while the proposed AEC method showcases robust scalability and maintains effectiveness regardless of PUF size, the PC method requires careful consideration of its limitations and the potential need for more complex system architectures to maintain reliability. Additionally, our proposed AEC explicitly leverages simple analog hardware components (e.g., bipolar encoders, discharging unit and voltage integrators), allowing linear scalability to larger systems without significant complexity overhead. Practical implementation such as impedance matching between stages, transistor sizing, and component variability was verified using existing PE processes[242], thus further enhancing the practical feasibility of our approach in large-scale applications.

5.5. Chapter Summary

This chapter addressed one of the fundamental challenges of PE: how to build reliable, endurant, and testable computing systems on a technology platform that is inherently stochastic, variation-prone, and defect-sensitive. Unlike silicon CMOS, where reliability is often enforced through tight process control and mature test infrastructure, printed systems must instead embrace variability and integrate robustness mechanisms across circuit, architecture, and algorithmic layers.

We first quantified the intrinsic vulnerability of printed analog and digital p-NNs through systematic fault sensitivity analysis. Monte Carlo-based fault injection revealed that while variation-aware training and dropout can partially mitigate parametric deviations, catastrophic faults remain a dominant reliability bottleneck, particularly in nonlinear activation circuits and bespoke arithmetic structures. These results established a clear need for hardware-level redundancy and defect-aware co-design rather than relying solely on training-based robustness.

Building upon this insight, the PRINT-SAFE framework introduced an end-to-end methodology for adaptive endurance and self-healing in printed neural hardware. By embedding passive redundancy directly into printed activation circuits and combining it with gradient-based FAT using Gumbel-Softmax-driven architecture selection, the proposed approach enabled bespoke, neuron-level fault resilience. Instead of static redundancy, the system dynamically learned which nonlinear primitives provide the highest robustness under realistic manufacturing defect distributions. This cross-layer co-design demonstrated that reliability can be significantly improved with moderate hardware overhead, achieving graceful degradation rather than catastrophic failure under increasing defect rates.

At the manufacturing-test level, this chapter proposed structural and delay-based test methodologies tailored to the physics of printed devices. For power management units, the proposed delay-based BIST methodology for *ReFlex-LDO* converted analog degradation into digitally measurable timing signatures, enabling structural fault coverage above 85% with minimal area and power overhead. These contributions demonstrate that test strategies must be co-designed with circuit topology and technology constraints in printed electronics.

Finally, reliability at the bitstream level was addressed through an efficient mixed-signal AEC scheme for unary-encoded computing. By leveraging temporal majority voting implemented with a simple

analog integrator, the proposed AEC restored PUF reliability to above 99% even under high runtime bit-error probabilities, while requiring only a fraction of the hardware resources of digital ECC schemes. Importantly, this time-domain redundancy preserves output length and scales linearly with system size, making it particularly attractive for ultra-low-cost printed security primitives.

Together, the contributions of this chapter establish a holistic reliability stack for printed and flexible computing systems. Reliability is addressed at multiple abstraction levels: (i) algorithmic robustness through fault-aware training, (ii) circuit-level resilience through passive redundancy, (iii) architectural endurance through bespoke co-optimization, (iv) structural test through delay-based methodologies, and (v) runtime stabilization through analog error correction.

The main message is that reliability in PE/FE cannot be an afterthought. It must be embedded into the design methodology, from device physics to learning algorithms and test infrastructure. By explicitly modeling defects, embracing stochasticity, and co-optimizing across layers, it is possible to construct printed and flexible NN systems that remain functional, diagnosable, and trustworthy despite the intrinsic imperfections of additive manufacturing technologies.

6. Security Analysis and Trust

The convergence of AI with novel hardware form factors is paving the way for ubiquitous, intelligent systems. Applications in wearable health monitoring, soft robotics, and bio-electronic interfaces demand computing platforms that are not only energy-efficient but also mechanically compliant, lightweight, and low-cost. FE, often based on TFTs like amorphous indium-gallium-zinc oxide (a-IGZO), have emerged as a primary enabler for this vision, allowing computation to be integrated directly onto conformal and even transparent substrates [250].

A critical yet often overlooked security concern in TFT-based FE systems is their susceptibility to side-channel attacks (SCAs), which exploit unintended information leakage such as power consumption or electromagnetic radiation during intermediate computations. Due to their minimal protective packaging, mechanical flexibility, and sensitivity to environmental variations, TFT-based analog flexible NN computing hardware (a-f-NN) classifiers are inherently more exposed to power-based SCAs compared to conventional silicon systems. In analog designs, the continuous-valued signals further amplifies leakage potential, creating substantial security and privacy risks for sensitive domains such as healthcare, wearables etc[184], [219], [256].

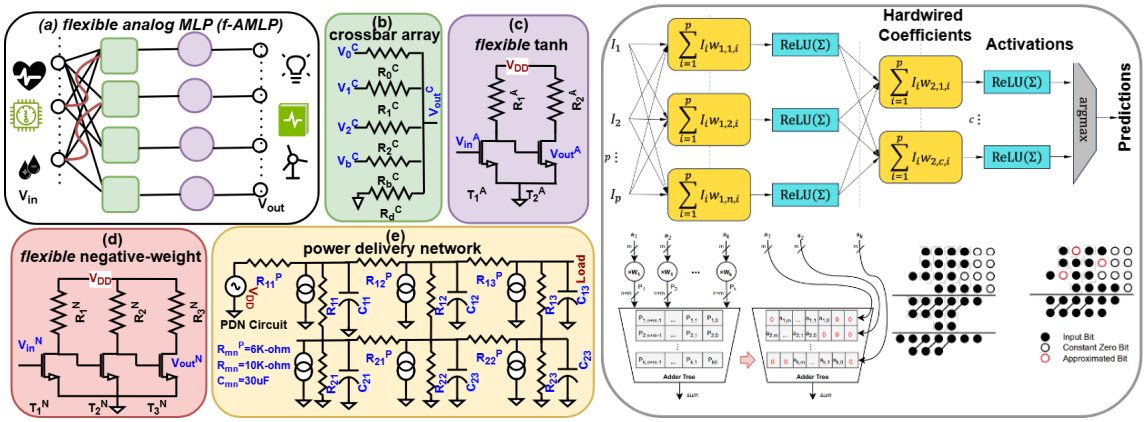
Security evaluation of digital silicon-based neuromorphic circuits is a mature field, offering well-established methodologies such as correlation power analysis (CPA), to be directly adapted to digital flexible neuromorphic circuit (d-f-NN) classifiers. However, extending these methods to a-f-NN is significantly more challenging. The intrinsic nonlinear transistor characteristics, device-specific threshold voltage shifts, parasitic effects, mechanical stress-induced variations, and strong sensitivity to environmental conditions such as temperature and humidity produce highly complex and noisy leakage profiles. These characteristics limit the effectiveness of traditional statistical models and necessitate the use of advanced, data-driven approaches to capture and exploit subtle correlations in leakage. By applying standard digital SCA techniques to d-f-NN and contrasting them with tailored NN-based attacks for a-f-NN counterparts, our work provides a comparison of attack success, complexity, and trace efficiency, offering new insights into the distinct security challenges of FE. In short, the contributions of this work are as follows:

- Side-channel vulnerability analysis of TFT-based flexible bespoke classifiers, in both analog (a-f-NN) and digital (d-f-NN) forms, using realistically simulated power traces incorporating device-level nonidealities and noise sources.
- CNN-based regression attack framework tailored to the continuous, nonlinear, and noise-distorted leakage of a-f-NN circuits, and benchmark its effectiveness against a standard CPA attack on the digital counterpart.
- Evaluation in benchmark datasets shows that the d-f-NN reaches a high attack success rate (ASR) within 4k to 5k traces, while the a-f-NN, despite slower initial growth, achieves a higher ASR (>90%) after ≈8.8k traces, showing the different leakage behavior and attack complexity.

6.1. Side Channel Attack Methodology on NN Classifiers

6.1.1. Threat Model

We consider an adversary performing a non-invasive power-based side-channel attack on flexible bespoke ML classifiers. We assume the attacker can physically access the targeted FE device during normal operation and passively measure power consumption traces using standard measurement equipment (e.g.,



(a) Schematic of a bespoke a-p-NN [241] that receives sensor signals and yields outputs to subsequent devices. Circuit primitives: (b) 3-input, 1-output resistor crossbar.(c) Inverter-based negative weight circuit. (d) Tanh-like activation. (e) Power-delivery network (PDN) for stable power supply.

(b) Bespoke architecture of a digital f-NN [225]; Impact of power-of-2 weights on a bespoke MAC circuit. On the left bespoke multipliers and a generic adder tree are used. With power-of-2 weights, only a simpler and narrower adder tree is required ; example of accumulation approximation.

Figure 6.1.: Bespoke architecture of analog and digital f-NN [225], [241].

oscilloscope). The attacker does not have direct, invasive access to the device's internal circuitry or memory. Furthermore, we assume the attacker possesses knowledge about the general architecture of the classifier and the ability to collect a sufficient number of power measurements under varying operational conditions.

The primary goal of the attacker is to recover sensitive intellectual property (IP) embedded within the device, specifically the analog input voltage and inference outcomes. Successful extraction of these parameters and intermediate results from the measured power traces would compromise both the confidentiality of the processed data and the proprietary nature of the classifier design. The practical relevance of this threat model aligns with realistic scenarios involving flexible ML classifiers used in sensitive applications, such as medical diagnostics, wearable technologies, and secure IoT devices.

6.1.2. CPA-based Attack on d-f-NN Classifiers

6.1.2.1. Characterization of Side-Channel Leakage

Digital implementations of d-f-NN classifiers on FE platforms inherently simplify computational tasks through techniques such as multiplier approximations, gate pruning, and weight quantization. These simplifications lead to distinct and predictable power consumption patterns that strongly correlate with logical state transitions in digital circuits. Consequently, the leakage characteristics in d-f-NN classifiers manifest as deterministic switching activities, inherently less complex and more directly observable compared to the continuous and nonlinear leakage in analog implementations.

6.1.2.2. CPA Attack and Leakage Extraction

For CPA attacks, we adopt the Hamming distance (HD) leakage model, which accurately captures the power consumption variations resulting from transitions between digital logic states. In our CPA-based attack methodology, we record power measurements from the FE-based d-f-NN classifiers during their operation. These power traces inherently encode the switching activities that correspond directly to internal computational states.

Mathematically, the CPA procedure involves computing Pearson correlation coefficients between the collected power traces $P(t)$ and the predicted Hamming distance values HD_i for each hypothesized state transition.

$$\rho_i = \frac{\mathrm{Cov}(P(t), HD_i)}{\sigma_{P(t)} \cdot \sigma_{HD_i}}, \tag{6.1}$$

where ρ_i represents the correlation coefficient for the i-th hypothesis, Cov denotes the covariance, and σ indicates the standard deviation. High correlation values clearly indicate successful identification of internal digital states or parameter values. This direct correlation approach efficiently extracts leakage information without the computational overhead associated with ML techniques. The detailed correlation results validating this methodology are presented in subsection 6.1.4.

6.1.3. ML-based Attack on a-f-NN Classifiers

Profiling of Side-Channel Leakage Flexible analog classifiers implemented with TFT technology inherently exhibit intricate leakage characteristics arising from complex nonlinear transistor behaviors, including device-specific threshold voltage shifts, nonlinear current-voltage (I-V) characteristics, and parasitic capacitances. Additionally, the leakage signals from these devices are significantly influenced by mechanical deformation and environmental factors such as temperature and humidity, adding to their unpredictability and complexity.

Unique Exploitation of Leakage via CNN-based Regression To effectively exploit such subtle and highly nonlinear leakage signals, traditional statistical methods such as CPA or ML models such as logistic regression or support vector machines (SVM) are insufficient due to their inherently linear or shallow nonlinear decision boundaries. Conversely, our regression model leverages convolutional neural networks' intrinsic hierarchical structure and deep feature extraction capability to capture complex correlations within the leakage signals effectively.

Moreover, the CNN regression model uniquely identifies critical leakage signatures that correspond to specific input analog voltages processed by the TFT-based classifier. By employing convolutional layers, our model extracts spatial-temporal leakage patterns at multiple resolutions. This capability can be mathematically described as follows:

$$\mathbf{Y}^{(l)} = f\left(\mathbf{W}^{(l)} * \mathbf{Y}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{6.2}$$

where $\mathbf{Y}^{(l)}$ represents the extracted feature maps at layer l, $\mathbf{W}^{(l)}$ denotes convolutional kernels optimized to isolate leakage-specific features, $\mathbf{b}^{(l)}$ represents biases, and $f(\cdot)$ denotes nonlinear activation functions. Through successive layers, the CNN transforms noisy power traces into a refined set of features directly correlated to analog inputs, which a simpler ML model cannot achieve due to their limited modeling capabilities.

Training, Modeling, and Validation for CNN-based Attack Our CNN architecture for FE leakage signals uses several convolutional layers with increasing feature extraction capabilities, nonlinear ReLU activation functions, and dropout layers with a selected dropout rate of 0.3 to mitigate overfitting. During training, we employ a supervised regression approach using a comprehensive dataset generated from SPICE simulations, which accurately represent realistic leakage patterns including controlled Gaussian and PDN-induced noise.

The training process involves iterative back-propagation using the Adam optimizer, optimizing network parameters (kernels, biases, and dropout probabilities) to minimize the mean squared error (MSE) between the predicted and actual input voltages. A validation subset of our generated traces is consistently monitored to prevent overfitting and ensure robust generalization of the trained model. After training, the model's effectiveness is quantitatively validated through extensive testing, showing high regression accuracy and minimal residual error distribution. This rigorous training, modeling, and validation process ensures the CNN's precision and reliability in recovering inputs from noisy and distorted a-f-NN leakage, confirming our successful CNN-based side-channel attack.

6.1.4. Evaluation

To assess the effectiveness of the proposed method, the experimental analysis of a-f-NN was performed using SPICE simulations in Cadence Virtuoso[1] using the Pragmatic *FlexIC Gen3* PDK [271] combined with Python [149]-based processing and modeling scripts. The d-f-NN were gate-level netlists mapped to flexible logic cells; dynamic power was obtained from vector-based (VCD/FSDB) switching activity. Hardware costs are tabulated in Table 6.1.

6.1.5. Experiment

PDN modeling: We included a PDN as a distributed RC ladder (Figure 6.1 (e)), capturing rail resistances and on-board decoupling. We used $R^{\mathrm{P}}_{mn} = 6\,\mathrm{k}\Omega$, $R_{mn} = 10\,\mathrm{k}\Omega$, and $C_{mn} = 30\,\mu\mathrm{F}$ for the supply path and decoupling network, and sweep these within process tolerances during sensitivity analysis. The PDN was placed in series with *VDD* and the classifier load; instantaneous supply current $i_{DD}(t)$ and power traces were recorded at the PDN output node.

Table 6.1.: Hardware Costs of Bespoke *f-NN* Classifier

Dataset	Topology	Analog			Digital)		
		Area (mm^2)	Power (μW)	Acc. (%)	Area (mm^2)	Power (μW)	Acc. (%)
Iris	4-8-3	0.04	3.39	96.51	4.831	72.01	95.43
Cardio	21-12-3	0.09	8.81	86.81	1.60	240.40	85.11
Pendigits	16-10-10	0.07	7.17	53.44	3.21	410.01	89.60
Seeds	7-4-3	0.04	3.14	88.10	6.20	110.70	86.21

6.1.5.1. Attack Setup of d-f-NN

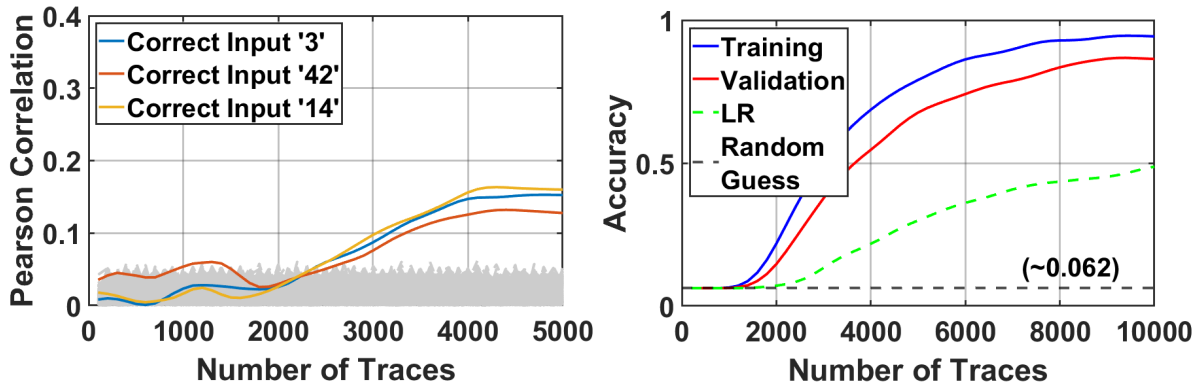
We performed CPA on the d-f-NN classifier. Initially, the design was constrained to incorporate multiplier approximations, gate pruning, and weight quantization[222], inspired by standard techniques employed in silicon-based digital designs to simplify the computational overhead. Power traces were captured under controlled experimental conditions, ensuring consistency in environmental parameters and measurement setups. Using the HD leakage model, we computed predicted switching activities corresponding to state transitions within the digital circuits. Pearson correlation coefficients were then calculated between these predicted switching behaviors and the experimentally captured power traces. The CPA parameters, such as the trace length, the number of traces, and statistical significance criteria, adhered closely to standard practices from digital VLSI security evaluations.

6.1.5.2. Attack Setup of a-f-NN

Dataset Generation We extracted data dependent power measurement consisting of 10,000 samples to realistically emulate side-channel leakage from a-f-NN classifiers. Each input sample represented analog input voltages uniformly distributed between 0 and 1. Corresponding power traces were modeled using nonlinear leakage functions characteristic of a-f-NN circuit primitives, i.e. nonlinear circuits (see Figure 6.1), including polynomial relationships. To reflect realistic measurements and environmental variability, we injected controlled Gaussian noise and introduced random outliers into the dataset, comprising $\approx 5\%$ of total samples.

Training Setup of a-f-NN The dataset was divided into train (60%), test (20%) and validation (20%) sets to facilitate unbiased evaluation of the side-channel attack. For modeling, we developed a CNN-based

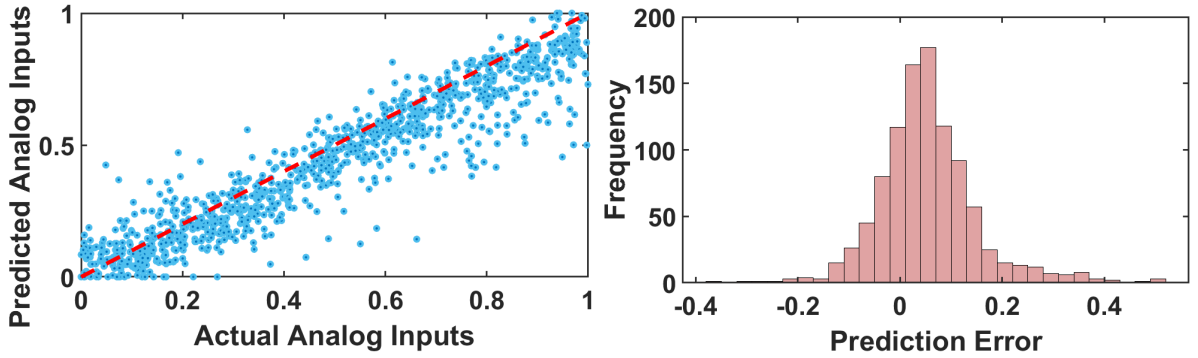
[1] `https://www.cadence.com/en_US/home.html`



(a) Correlation progression on digital *f-DMLP* classifiers, illustrating gradual separation of correct inputs' (3 among 21) correlations from incorrect inputs (shown in grey), highlighting the attack.

(b) CNN-based regression accuracy (training vs. validation) on analog *f-NN* classifiers, demonstrating effective generalization and robustness despite measurement noise for Iris dataset.

Figure 6.2.: Comparison of side-channel attack effectiveness on analog and digital FE classifiers (Iris dataset).



(a) Correlation showing successful recovery of input voltages from side-channel power traces, highlighting the vulnerability.

(b) Distribution of residual prediction errors showing concentrated accuracy around zero, validating model precision and reliability.

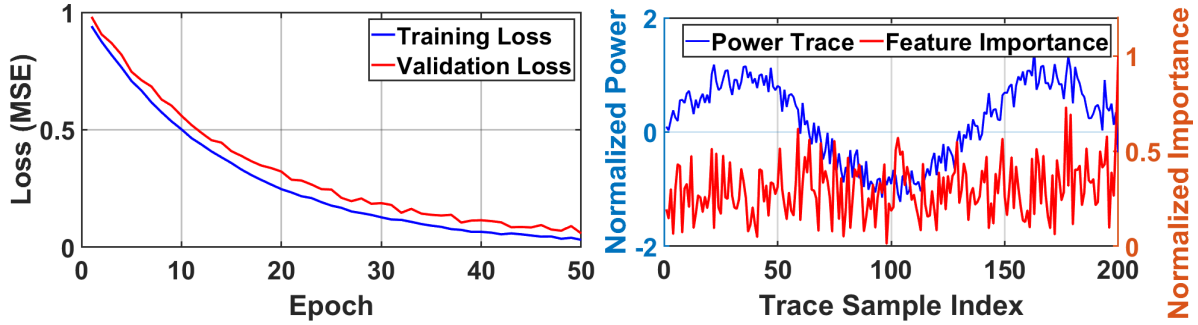
Figure 6.3.: Robustness of CNN-based side-channel regression attack on analog *f-NN* implementation (Iris dataset).

regression attack comprising two convolutional layers (kernel size 5) followed by Rectified Linear Unit (ReLU) activations and dropout layers (0.3 dropout rate) to enhance model robustness and generalizability. The CNN was trained using the Adam optimizer (learning rate 0.001) with a mini-batch size of 64 over 50 epochs, monitoring training and validation loss to ensure stable convergence and evaluate model effectiveness.

6.1.6. Result

CPA based SCA results on d-*f*-NN classifiers Correlation plots of the CPA-based attack for d-f-NN classifiers (Figure 6.2a) illustrate how clearly the model exploits power-leakage signals. Pearson correlation values for correct input predictions initially exhibit minimal improvement due to noise and fewer traces. However, after $\approx$ 1,500 traces, they distinctly separate from incorrect inputs, rapidly increasing and gradually saturating near moderate correlation values (around 0.2-0.25). The clear separation after sufficient trace collection demonstrates CPA's effectiveness in identifying switching activities associated with internal logic states. These results validate that digital implementations exhibit predictable and effectively exploitable leakage characteristics via CPA, thus highlighting their practical vulnerability.

CNN-based SCA on *a-f-NN* classifiers In contrast to the discrete switching behavior of digital designs, a-f-NN implementations exhibit complex, nonlinear leakage patterns shaped by TFT nonidealities, parasitic effects, and environmental variability. Figure 6.2b shows training and validation accuracy trends for our CNN-based regression model attacking an a-f-NN classifier. The model gradually learns to



(a) Learning curves showing CNN training and validation loss across epochs during regression-based modeling. Rapid initial convergence demonstrates effective learning of analog leakage patterns.

(b) Saliency map illustrating feature importance of CNN regression model on input power traces. High-importance regions reveal specific segments of the power trace most influential for accurate analog input.

Figure 6.4.: Attack progress on *a-f-NN* classifiers (Iris dataset).

separate correct analog input predictions from incorrect ones despite substantial noise, achieving robust generalization beyond 8,000 traces. This slow initial accuracy rise, followed by rapid escalation, highlights the need for advanced profiling techniques to extract weak but information-rich features in FE-specific measurement noise.

Further validation is provided in Figure 6.3, where the scatter plot (Figure 6.3a) presents the predicted versus actual analog input voltages obtained from our CNN-based regression model for the a-f-NN classifier. The dense clustering of points along the diagonal line indicates that the predicted values closely match the ground truth throughout the input range, with minimal bias or variance. Even at the extremes of the analog input domain, where non-idealities and PDN-induced voltage drops are most pronounced, predictions remain closely aligned with the ideal diagonal, reflecting the model's ability to learn highly non-linear leakage-to-value mappings.

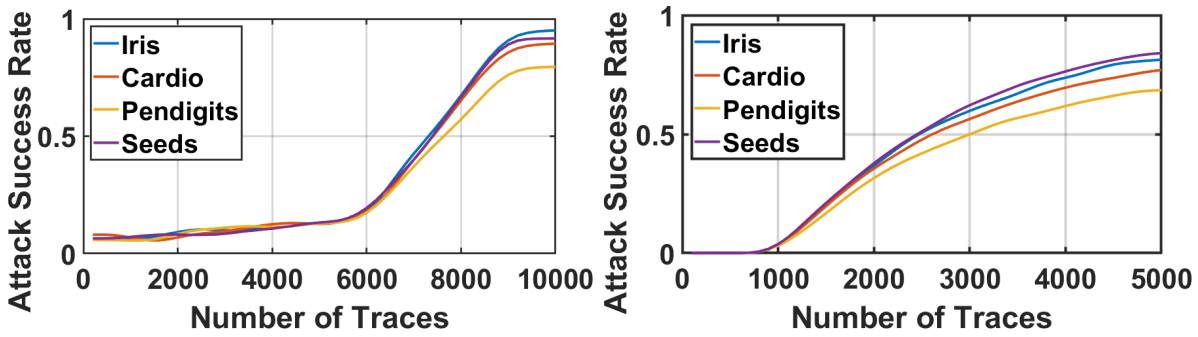
The residual error histogram (Figure 6.3b) provides complementary insight into prediction accuracy and error distribution. The narrow, Gaussian-like distribution sharply centered at zero demonstrates the absence of significant bias in the regression output, while the low spread confirms strong prediction consistency across all tested inputs. The lack of heavy tails in the distribution suggests that the CNN avoids large outlier errors, even under noisy measurement conditions.

Attack Progress and Feature Attribution To better understand how the CNN learns to exploit analog leakage, we examine intermediate training behavior and feature attribution. The loss curves in Figure 6.4a show rapid reduction of both training and validation MSE within the first $\approx$ 20 epochs, indicating fast convergence despite high measurement noise. Validation loss closely follows training loss, confirming minimal overfitting and strong generalization.

The saliency map in Figure 6.4b provides complementary insight by highlighting power-trace segments most influential to input prediction. High-importance intervals coincide with analog signal transitions in the crossbar summation and activation stages, where nonlinearities and circuit-specific switching generate distinctive leakage signatures. This targeted exploitation of specific time windows underscores the CNN's capability to focus on the most informative segments of the trace, enabling successful attacks in scenarios where traditional statistical methods fail.

Attack Success Rate Comparison: Analog vs. Digital The cumulative attack success rate (ASR) trends for the analog f-NC and digital f-DMLP implementations across four benchmark datasets (Iris, Cardio, Pendigits, Seeds) are shown in Figure 6.5a and Figure 6.5b, respectively. The progression curves highlight a fundamental trade-off between attack speed: the number of traces required to reach a given success probability, and the final achievable accuracy.

For the analog case, CNN-based regression must learn and generalize from complex, noise-rich leakage patterns arising from TFT device nonlinearities, PDN noise, and measurement distortions. Consequently, the ASR grows slowly during the initial $\approx$2,000–3,000 traces, reflecting the difficulty of extracting reliable



(a) ASR progression for CNN-based attacks on analog *f-NN* classifiers, highlighting slow initial growth followed by rapid escalation after a few hundred traces, indicative of complex but exploitable analog leakage. **(b)** CPA-based cumulative ASR on digital *f-NN* classifiers, demonstrating distinct dataset-specific trends and faster initial convergence to moderately high values, showing simpler leakage characteristics.

Figure 6.5.: Attack success rate comparison across diverse datasets.

features under high noise conditions. After this slow start, the model enters a rapid escalation phase between ≈3,000 and ≈4,700 traces, where learned features begin to strongly correlate with the target inputs. The ASR then continues to improve, eventually saturating at ≥90.

In contrast, the digital f-DMLP, targeted with CPA, exhibits a faster initial ASR growth, with most datasets achieving over 45% success within ≈1,500–2,000 traces. This rapid convergence is attributed to the discrete nature of switching activity in digital logic, where leakage correlates strongly and consistently with Hamming distance predictions. Saturation occurs earlier at ≈4,000–5,000 traces, but at lower final ASR values (70–85%) compared to the analog case. This reflects the ease of early exploitation due to simpler leakage models, yet a ceiling in achievable accuracy since digital leakage is more structured and contains fewer exploitable nonlinear components. This trade-off in attack speed versus final ASR has direct implications for threat modeling in FE, as the optimal defense strategy must consider both early-stage and long-term attack resilience.

In this work, we introduce a robust CNN-based regression attack model used to exploit subtle, nonlinear leakage patterns in flexible analog MLP classifiers and achieve accurate recovery of input voltages, and also compared with state-of-the art digital flexible classifiers. Our results demonstrate that while digital designs can be compromised with relatively few traces, their exploitable leakage limits the attacker's ultimate success. In contrast, analog implementations resist early exploitation but yield stronger information over time, making them more vulnerable to high-trace profiling attacks.

6.2. Security Frameworks for Flexible Spiking Neuromorphic Hardware

In parallel, biologically inspired SNNs have garnered significant interest as a power-efficient computing paradigm for these edge applications. Unlike conventional ANNs, SNNs operate on event-driven, discrete spikes, performing sparse, asynchronous computation. This model drastically reduces redundant switching activity, making f-SNN an ideal match for the stringent power and thermal budgets of on-skin or implantable devices.

However, this combination of flexible substrates and event-driven computing introduces a new, largely unexplored security vulnerability. The very attributes that make FE attractive are, from a security perspective, critical weaknesses:

- **Reduced Physical Protection:** Flexible devices inherently lack the rigid packaging, metallic shielding, and complex multi-layer ground planes of conventional silicon chips. This thin encapsulation significantly increases exposure to physical leakage.
- **Simplified Power Delivery:** To maintain low cost and flexibility, these systems often rely on limited I/O and shared power supply rails, restricting the use of common side-channel defenses like split-domain power or on-chip filtering.

- **Data-Correlated Activity:** The event-driven nature of SNNs, while efficient, directly correlates power consumption with the data being processed. Each spike, or lack thereof, represents a discrete computational event, yielding current and timing signatures that are rich with information.

This research confronts the hypothesis that these factors combine to make f-SNN highly susceptible to power-based Side-Channel Analysis (SCA). While SCA on rigid CMOS neural accelerators is a well-established field [196], the unique device physics of a-IGZOs TFTs and the analog, continuous-time dynamics of flexible circuits present a distinct and uncharacterized threat landscape [234].

To bridge this critical gap, this chapter introduces *FlexSpy*, a comprehensive, design-time side-channel framework specifically engineered for flexible neuromorphic hardware. *FlexSpy* provides the first end-to-end pipeline to model, simulate, and quantify power-based information leakage in f-SNN *before* fabrication.

6.2.1. Flexible Spiking Neural Networks (f-SNN).

Spiking neurons [250], [251], [253] exchange discrete events and remain quiescent between spikes. As shown in Figure 6.6 (left), a typical f-SNN cell integrates (i) a synaptic input stage that forms a current or conductance sum

$$i_{\text{syn}}(t) = \sum_i w_i \, s_i(t),$$

(ii) an *RC* integrator that accumulates charge to a threshold, and (iii) reset-discharge control that enforces a refractory interval. These blocks map well to TFT physics and enable low-power operation on bendable substrates.

However, their continuous conduction paths and shared supply rails create data-dependent shifts in the supply current: quasi-DC offsets during spike epochs with sharp edges at spike onsets/offsets. These offsets later provide alignment anchors and classification features in side-channel traces, as exploited by FlexSpy.

6.2.2. Flexible Recurrent Neural Networks (f-RNN).

Flexible recurrent designs realize smooth state dynamics without discrete spikes. Each hidden state is stored on a capacitor, while first- or second-order *RC* sections set learnable time constants $\tau_i = R_i C_i$. Weighted sums and activation functions use resistive networks and transconductance stages, forming a continuous-time approximation to recurrent models, as shown in Figure 6.6 (right). Compared with sparse, event-driven f-SNN, f-RNNs produce smoother supply-current envelopes with weaker spike-aligned features; leakage concentrates at low frequencies and reflects interactions between state voltages, bias currents, and the PDN.

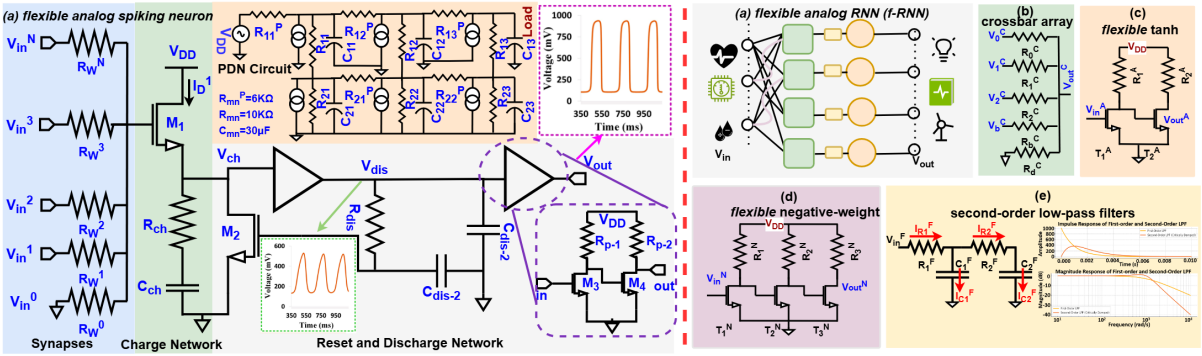
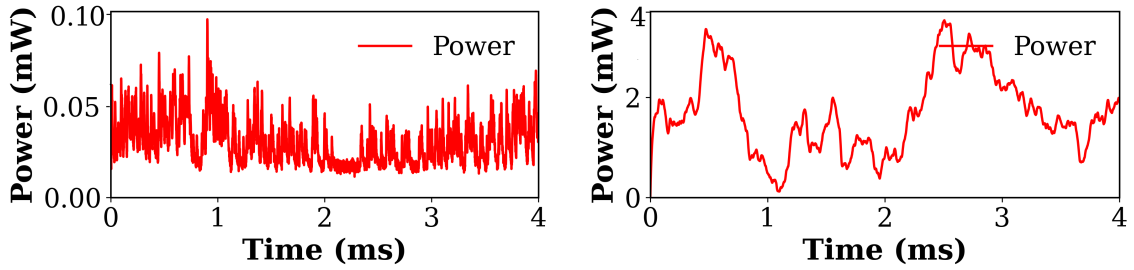


Figure 6.6.: Circuit-level implementations of f-NNs analyzed in this work. Left: f-SNN cell with Synapse, Charge/Integrate, and Reset/Discharge stages. Right: f-RNN cell with recurrent *RC* dynamics for continuous-time state evolution.



(a) Raw traces of f-SNN showing spike-triggered current steps. **(b)** Raw traces of f-RNN showing smooth oscillatory envelopes.

Figure 6.7.: Distinct leakage primitives from raw power traces: spike-driven quasi-DC offsets in f-SNN (left) vs. smoother low-frequency RC oscillations in f-RNN (right).

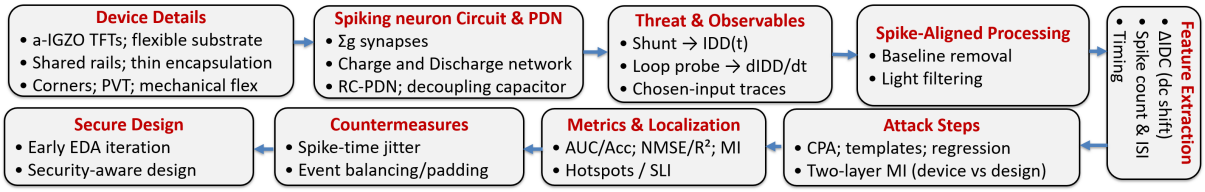


Figure 6.8.: Overview of the FlexSpy framework pipeline. FlexSpy provides a complete design-time flow: from technology-calibrated device simulation and PDN modeling to spike-aligned feature extraction, a calibrated attack suite (CPA, templates, regression, MI), and quantitative localization (SLI), enabling in-loop evaluation of countermeasures.

As summarized in Figure 6.7, the f-SNN supply current exhibits step-like, spike-synchronous offsets, whereas the f-RNN output is smoother and less sharply aligned to individual events. This qualitative difference is the root cause of richer, more easily exploitable leakage in f-SNN.

6.2.3. The FlexSpy Framework Methodology

FlexSpy is a design-time, simulation-based framework for predicting, localizing, and mitigating power side-channel leakage in f-NNs. The complete end-to-end procedure is illustrated conceptually in Figure 6.8 and detailed in Algorithm 6. The pipeline begins with device models and a circuit netlist, synthesizes power traces, extracts spike-aligned features, and then executes a calibrated attack suite to quantify and localize leakage.

6.2.4. Threat Model and Leakage Observables

We assume a practical, non-invasive adversary with the following capabilities:

- **Measurement:** The attacker has physical access to the device and inserts a small (1–10 Ω) shunt resistor in series with the main V_{DD} supply (Supply Voltage (VDD)). They measure the voltage drop across this shunt to record a single, aggregate power trace, $I_{DD}(t)$.
- **No Internal Access:** The attacker has no internal probes, no access to on-chip clocks, and no dedicated "trigger" signal indicating the start of computation.
- **Alignment:** To overcome the lack of a trigger, we introduce a **virtual trigger**, $V_{\text{trig}}(t)$, which the attacker computes numerically from the *same* power trace. This virtual trigger is used only to segment the trace into spike epochs for analysis.

6.2.5. Substrate-Aware Leakage Model for *f*-SNN

Understanding *what* leaks is the first step. Based on the a-IGZOs TFT physics and the circuit topology in Figure 6.6, the total supply current $I_{DD}(t)$ can be modeled as the sum of three primary components:

$$I_{DD}(t) \approx I_{bias} + V_{head} \sum_i g_i s_i(t) + \sum_k C_k \frac{dV_k}{dt} + n(t), \tag{6.3}$$

where:

- I_{bias} is the static, idle current of the circuit.
- $V_{head} \sum_i g_i s_i(t)$ is the **quasi-DC offset term**. This is the dominant leakage source. g_i represents synaptic conductances (weights), and $s_i(t)$ is synaptic activity (spike rate). During a spike epoch, active synapses create continuous conduction paths, drawing a sustained current proportional to the total active conductance. This creates an observable DC-level shift in the power trace, consistent with the step-like behavior in Figure 6.7.
- $\sum_k C_k \frac{dV_k}{dt}$ is the transient **displacement current** from charging/discharging internal node capacitances C_k. These currents correspond to sharp spike edges.
- $n(t)$ is device and instrumentation noise.

At the millisecond/kHz scales of SNNs, the quasi-Direct Current (DC) offset term dominates the observable leakage, and it is this feature that FlexSpy primarily targets when building leakage models and attacks.

6.2.6. Trace Synthesis and Spike-Aligned Feature Extraction

Trace Synthesis. The framework uses a PDK-calibrated transient simulation (e.g., in Cadence Spectre) to generate power traces. Neuron and synapse blocks are implemented with a-IGZOs TFT models, poly-resistors, and Metal–Insulator–Metal (MIM) capacitors. A compact model of the PDN, including series rail resistance and decoupling capacitors, is crucial for capturing the non-ideal effects of a flexible backplane (e.g., rail droop and coupling between distant cells). Traces are simulated across various process, voltage, and temperature (Process, Voltage, and Temperature (PVT)) corners to emulate realistic manufacturing variation and environmental conditions.

Spike-Aligned Feature Extraction. We apply a three-step process to isolate information-rich features from the raw $I_{DD}(t)$ trace:

1. **Baseline Removal & Filtering:** The idle baseline I_{bias} is estimated from quiescent segments and subtracted. A light low-pass filter suppresses measurement noise while preserving the ms-scale dynamics of the quasi-DC offsets.
2. **Virtual Trigger Alignment:** The attacker computes the virtual trigger $V_{\text{trig}}(t)$ to identify spike epochs. Since the quasi-DC offsets create sharp *edges* at their onset and offset, their derivative is large. The virtual trigger is thus computed as the time-derivative of the supply current (or the shunt voltage):
$$V_{\text{trig}}(t) = \alpha \frac{d}{dt} I_{DD}(t) \equiv \frac{d}{dt} V_{shunt}(t). \tag{6.4}$$
By thresholding $V_{\text{trig}}(t)$, the attacker robustly detects the rising and falling edges of spike windows $[t_0, t_1]$ without any internal trigger.

3. **Per-Window Feature Computation:** For each segmented window w, the framework computes the key leakage feature, the average DC-level shift:

$$\Delta I_{DC}^{(w)} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} (I_{DD}(t) - I_{bias})\, dt, \tag{6.5}$$

which directly measures the quasi-DC term in Equation 6.3. Additional features such as spike count (estimated from V_{trig}), inter-spike intervals (Inter-Spike Interval (ISI)), and coarse timing statistics are also extracted to form a feature vector $Z^{(w)}$ per window.

For downstream attacks we typically concatenate window-wise features into a design-level vector $\mathbf{z}_{\text{design}}$, optionally augmented by device-level indicators (e.g., idle noise, static offsets).

Calibrated Attack Suite After feature extraction, FlexSpy applies a suite of attacks (summarized in Algorithm 6) to quantify the exploitable information.

Correlation Power Analysis (CPA). Correlation Power Analysis (CPA) is used to localize *when* leakage is strongest. We correlate the feature vector $Z^{(w)}$ (primarily $\Delta I_{DC}^{(w)}$) with a linear predictor of rate-weighted activity,

$$x = \sum_i g_i \bar{s}_i,$$

and compute the Pearson correlation across windows. Peaks in the sliding-CPA trace identify dominant leakage windows and validate the quasi-DC leakage model [27].

Template Profiling (Gaussian). To perform label inference (classifying which input y was processed), we train Gaussian templates (e.g., Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA)) on the feature vectors Z from a profiling set. At attack time, per-window log-likelihoods are summed to form a global score for each candidate label, yielding a powerful profiled attack.

Regression for Continuous Recovery. To recover continuous values such as layer-wise spike rates, we train ridge regressors mapping the feature vector Z to per-layer rate vectors $r^{(l)}$. High R^2, low NMSE, and high cosine similarity indicate that an attacker can accurately estimate intermediate firing rates from power alone.

6.2.7. Leakage Quantification and Localization

Two-Layer Mutual Information Accounting. To distinguish device-specific leakage from design-level leakage, we use a two-layer Mutual Information (MI) decomposition. Let S denote the secret (e.g., label y), L_{device} capture device-specific features (e.g., idle noise, static offsets), and L_{design} denote the spike-window features (e.g., ΔI_{DC}, counts). The total information satisfies

$$I(S; L_{\text{device}}, L_{\text{design}}) = I(S; L_{\text{device}}) + I(S; L_{\text{design}} \mid L_{\text{device}}), \tag{6.6}$$

which we estimate empirically following standard information-theoretic practice [28]. This decomposition reveals how much information stems from the neuromorphic design versus the underlying substrate and fabrication variations.

Spike-Leakage Index (SLI) for Hotspot Ranking. To provide actionable feedback to designers, we must localize *where* and *when* leakage occurs. We define the **Spike-Leakage Index (SLI)** as a normalized MI for a specific circuit block b (e.g., Synapse, Charge, Reset) and time window w:

$$\text{SLI}(b, w; S) = \frac{I(S; Z_b^{(w)})}{H(S)} \in [0, 1], \tag{6.7}$$

where $Z_b^{(w)}$ is the feature vector extracted from block b in window w, and $H(S)$ is the entropy of the secret S. High SLI highlights spatio-temporal hotspots, enabling targeted countermeasures.

Algorithm 6 FlexSpy End-to-End Procedure

Inputs: Netlist N, Device Models M, Stimuli X, Targets O, PDN parameters
Outputs: CPA peaks, AUC/Accuracy, R^2/NMSE, MI, SLI map, Security–Power trade-offs

Step 1: Synthesize Traces
for all $(x, \text{corner}) \in X \times M$ **do**
    Run transient simulation on N with PDN
    Record $I_{DD}(t)$ and compute $V_{\text{trig}}(t) \leftarrow \frac{d}{dt} I_{DD}(t)$

Step 2: Build Features
$I_{DD}(t) \rightarrow$ baseline removal $\rightarrow$ light filtering
Use $V_{\text{trig}}(t)$ to segment spike windows $[t_0, t_1]$
Compute per-window features $Z \leftarrow \{\Delta I_{DC}, \text{count}, \text{ISI}, \text{timing}\}$
Standardize features by corner

Step 3: CPA
Correlate windowed Z with hypothesis $x = \sum g_i \bar{s}_i$
Record peak correlation and window index w_{peak}

Step 4: Profiling (Templates)
Fit Gaussian templates (e.g., LDA/QDA) for targets $o \in O$
At attack time, sum per-window log-likelihoods for classification

Step 5: Regression
Train ridge model $Z \mapsto r^{(l)}$ (layer-wise rates)
Report R^2, NMSE, cosine similarity on held-out traces

Step 6: MI / SLI
Estimate $I(S; L_{\text{device}})$ and $I(S; L_{\text{design}} \mid L_{\text{device}})$
Compute $\text{SLI}(b, w; S) \leftarrow I(S; Z_b^{(w)})/H(S)$ for all b, w

Step 7: Evaluate Countermeasures
Sweep jitter, balancing, and PDN settings
Re-run Steps 1–6
Report leakage reduction vs. power/latency/area overhead

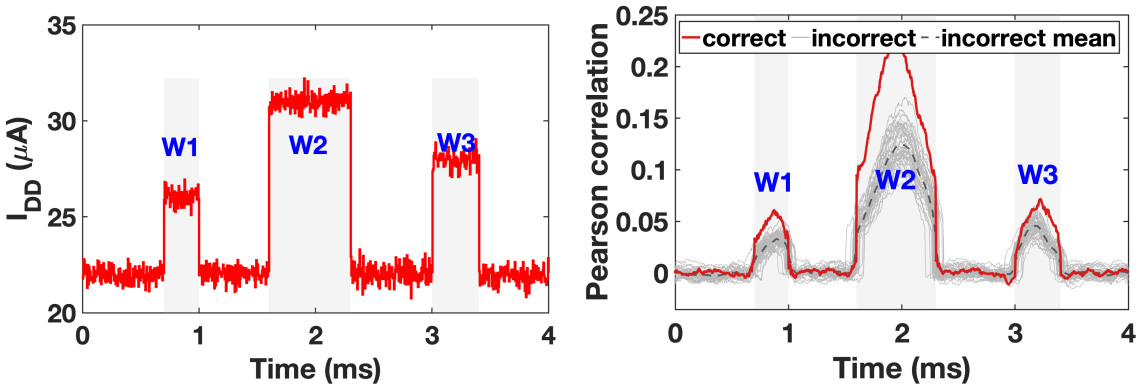
6.2.8. Evaluation

During evaluation, we quantify (i) design-level leakage induced by spike-driven activity and (ii) the effect of lightweight circuit countermeasures. Transient simulations ran in *Cadence Virtuoso Spectre* with technology-calibrated device models. The traces were analyzed by the integrated Python pipeline (6).

6.2.9. Experiment

We instantiated the f-SNN and f-RNN circuits in Cadence Virtuoso Spectre using technology-calibrated a-IGZOs TFT models. The PDN model included typical parameters for a large-area flexible substrate ($R_{\text{rail}} \approx$ 6–10 kΩ, $C_{\text{decoupling}} \approx 30$ μF). We simulated millisecond-length traces in the 1–5 kHz regime across nominal, low-V_{DD}, and high-temperature corners.

For workloads, we used time-series datasets from the UCR Archive [72] with a 70/15/15 split for profiling, validation, and attack. Reported metrics include ROC–AUC (ROC-AUC) for label inference,



(a) Measured $I_{DD}(t)$ with spike windows (W1–W3); quasi-DC offsets in f-SNN appear during spike epochs.

(b) Sliding-CPA localization: the correct hypothesis (red) peaks in window W2, incorrect ones (gray) near zero.

Figure 6.9.: Leakage localization in time for f-SNN on P-Cons. Left: quasi-DC ΔI_{DC} offsets in $I_{DD}(t)$ during spike epochs. Right: sliding-CPA shows that leakage is maximized in W2.

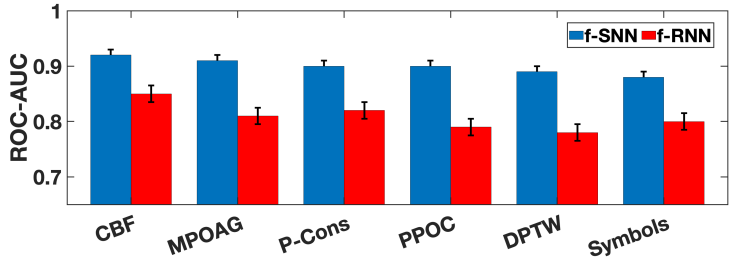


Figure 6.10.: Cross-dataset leakage at the nominal corner: ROC–AUC for label inference using spike-window features on six workloads. Blue bars: f-SNN; red bars: f-RNN; the dotted line indicates chance (AUC = 0.5). The f-SNN consistently leaks more than the f-RNN.

normalized MSE (*NMSE*) and cosine similarity for spike-rate regression, and mutual information (MI) for leakage quantification.

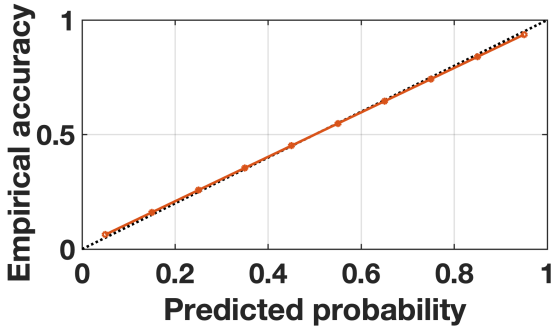
6.2.10. Result

We first evaluated the ability to infer the input class label y from the spike-window feature vector $\mathbf{z}_{\text{design}}$.

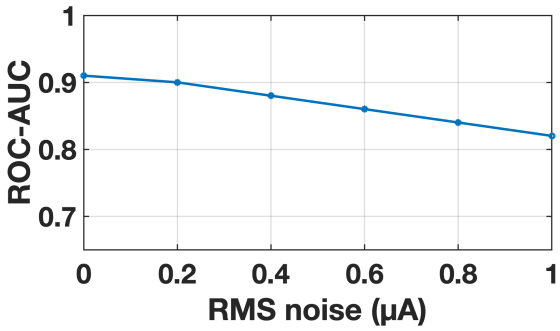
At the nominal corner, a logistic classifier trained on $\mathbf{z}_{\text{design}}$ achieves an ROC–AUC of ROC-AUC = 0.91 with low calibration error (ECE $<$ 5%), confirming a strong, class-dependent signal. At low V_{DD} and high temperature, the AUC degrades gracefully to 0.85 and 0.82, respectively. Augmenting the feature vector with simple device-variation indicators (e.g., idle noise) improves AUC by only $\approx$ 0.02, indicating that class information is dominated by the design-level spike-window features.

Figure 6.9 shows how sliding-CPA localizes the leakage in time. The left panel illustrates the measured supply current $I_{DD}(t)$ with shaded spike windows (W1–W3), where quasi-DC offsets appear during spike epochs in the f-SNN. The right panel plots the sliding-CPA correlation for multiple hypotheses: a bundle of incorrect hypotheses (gray) stays near zero except for small fluctuations, while the correct hypothesis (red) exhibits a sharp correlation peak confined to the dominant spike window W2. This behavior agrees with the quasi-DC leakage model in Equation 6.3 and highlights that ΔI_{DC} drives most of the measurable signal.

Across datasets, the same pattern holds. Figure 6.10 summarizes cross-dataset leakage at the nominal corner: the f-SNN consistently exhibits higher label leakage (higher ROC-AUC) than the f-RNN, with error bars showing 95% confidence intervals across runs. The f-RNN's smoother dynamics and weaker spike alignment yield lower instantaneous leakage, although some information still accumulates over longer time horizons.



(a) Reliability diagram: the model is slightly under-confident with ECE $< 5\%$, device-variation indicators yields only minor benefit.



(b) AUC vs. added measurement noise (normalized): smooth roll-off, consistent with dominance of spike-window features.

Figure 6.11.: Model reliability and measurement robustness for label inference in f-SNN on P-Cons.

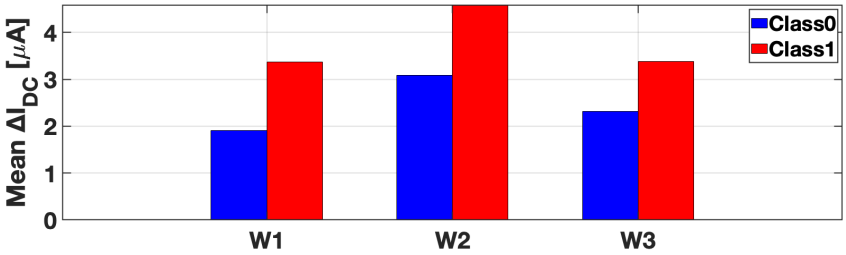


Figure 6.12.: Windowed quasi-DC current shift ΔI_{DC} by class for f-SNN on P-Cons. Distinct class clusters in the dominant W2 window visualize the rate-weighted $\sum_i g_i s_i$ leakage mechanism.

Calibration and robustness to measurement noise are shown in Figure 6.11. The classifier is slightly under-confident but maintains ECE $< 5\%$, and AUC degrades smoothly under added measurement noise, again indicating that spike-window observables dominate over device noise.

Finally, Figure 6.12 visualizes class-dependent leakage for P-Cons. The windowed current shift $\Delta I_{DC}^{(w)}$ in the dominant W2 window differs measurably between classes, directly reflecting the rate-weighted term $V_{head} \sum_i g_i s_i(t)$ in Equation 6.3. This separation explains the high AUC and provides an intuitive picture of the leakage source.

Spike-Rate Recovery and Mutual Information Beyond classification, we evaluate whether an attacker can reconstruct continuous-valued intermediate data such as layer-wise spike rates. A ridge regressor is trained to map $\mathbf{z}_{\text{design}}$ to the firing-rate vectors $\mathbf{r}^{(\ell)}$ of each layer.

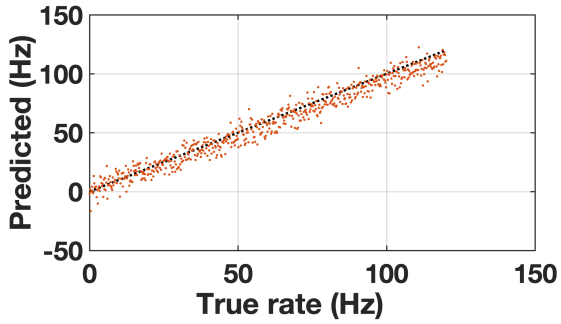
Across PVT corners, we obtain $NMSE = 0.14$ and cosine similarity of 0.93 between predicted and true spike-rate vectors. Figure 6.13 (a) plots predicted vs. true rates on held-out traces, showing a near-linear trend. This linearity reflects the quasi-static term $\sum_i g_i \bar{s}_i$ in Equation 6.3 captured by ΔI_{DC}: average firing rate in each window is essentially encoded in a rate-weighted current offset.

We also apply the two-layer MI decomposition in Equation 6.6. For label leakage, we measure

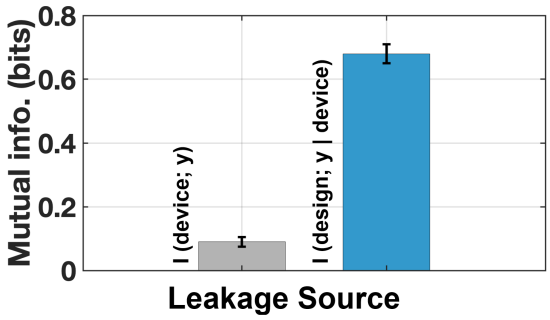
$$I(y; L_{\text{device}}) \approx 0.09 \text{ bits}, \qquad I(y; L_{\text{design}} \mid L_{\text{device}}) \approx 0.68 \text{ bits},$$

which implies that roughly 88% of class-related information stems from spike-window features, not device-specific fingerprints. For rates and multiplicity, conditional MI lies in the range 0.72–0.81 bits at nominal conditions and drops by $\sim 20\%$ at low V_{DD}. Overall, most recoverable information is attributable to design-level spike windows.

Structural Profiling from Power Traces We next assess whether power traces leak structural information about the neuromorphic circuit, beyond labels and rates. Gaussian templates are trained on concatenated spike-window features to recover two design-level targets that arise naturally in flexible analog neuromorphic circuits:

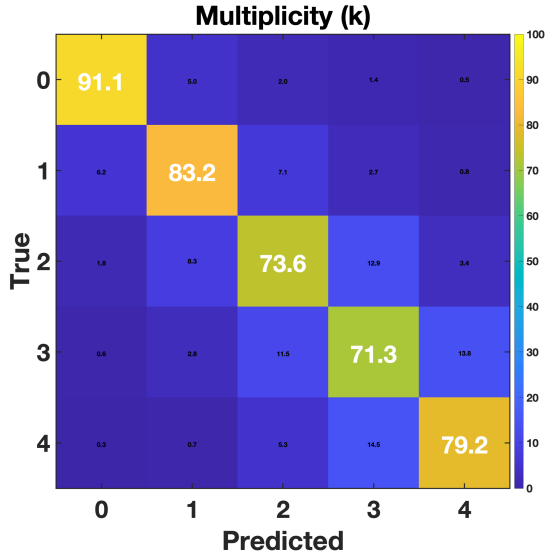


(a) Layer-wise rate recovery on held-out traces (P-Cons); dashed line is identity.

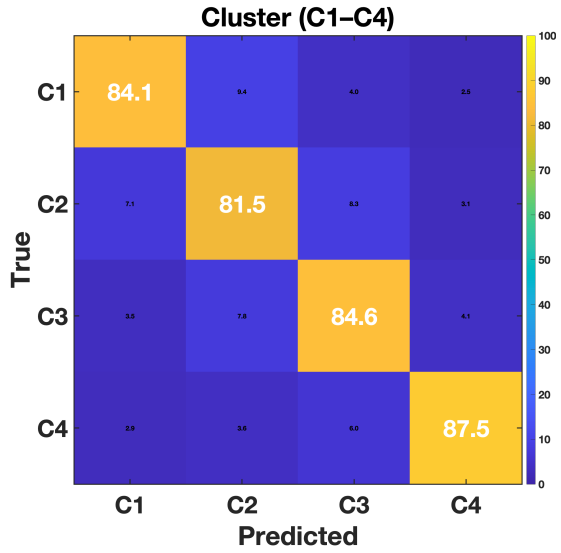


(b) Two-layer MI decomposition: labels leak primarily via design-level spike-window activity.

Figure 6.13.: Spike-rate recovery and mutual information analysis for f-SNN.



(a) Profiling synapse multiplicity k for f-SNN on P-Cons; mistakes are mostly off-by-one.



(b) Source-cluster profiling (C1–C4); errors concentrate on neighboring clusters.

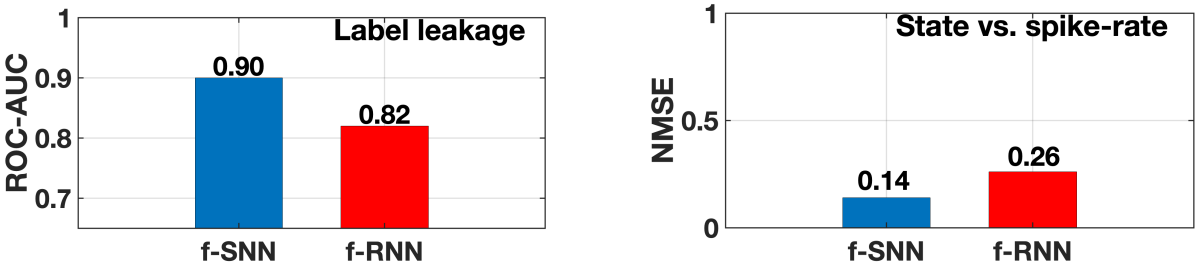
Figure 6.14.: Confusion matrices for structural profiling from power traces in f-SNN. Gaussian templates trained on spike-window features can recover both multiplicity and input source clusters.

- **Synapse Multiplicity k:** the number of active conductance paths contributing to a spike window ($k \in \{0, \ldots, 4\}$ in our experiments).
- **Source Clusters (C1–C4):** groups of synaptic inputs defined by connectivity or receptive-field origin.

At nominal conditions, synapse multiplicity reaches 87.3% accuracy (F1-score 85.9%, ECE 3.2%), with confusions concentrated on adjacent k values, as shown in Figure 6.14 (a). Source-cluster inference (C1–C4) achieves 83.1% Top-1 and 95.2% Top-2 accuracy when incorporating simple timing context (Figure 6.14 (b)). Corner degradation is modest (e.g., 82% / 79% at high-T/low-V_{DD}), confirming the robustness of structural profiling.

These results show that power traces leak not only *what* the network is computing (labels, rates), but also *how* it is structured (multiplicity and connectivity pattern), enabling reverse-engineering of neuromorphic architectures.

Comparative Study: f-SNN vs. f-RNN To understand how flexible dynamical primitives differ in leakage behavior, we repeat the analysis for f-RNN circuits that implement second-order RC filters. As already



(a) Label leakage (ROC–AUC): f-SNN 0.90 > f-RNN 0.82.

(b) State/rate NMSE: f-SNN 0.14 < f-RNN 0.26.

Figure 6.15.: Direct security comparison of f-SNN vs. f-RNN on P-Cons. The f-SNN's spike-window ΔI_{DC} offsets produce stronger instantaneous leakage than the f-RNN's smoother envelopes.

hinted by Figure 6.7, the f-RNN produces low-frequency oscillatory envelopes, whereas the f-SNN exhibits quasi-DC steps aligned with spike epochs.

Under identical rails, corners, and workloads, f-SNN achieves higher AUC and lower rate-recovery NMSE due to stronger static-current coupling. Figure 6.15 summarizes the comparison on P-Cons: the f-SNN exhibits stronger label leakage (AUC 0.90 vs. 0.82) and lower NMSE in recovering internal activity (0.14 vs. 0.26). The f-RNN hides instantaneous spikes but accumulates leakage more slowly over time, making it inherently more robust to the spike-window style of attack that FlexSpy targets.

Overall, the comparative study confirms our hypothesis: the event-driven f-SNN is the more vulnerable architecture when leakage is dominated by rate-weighted quasi-DC offsets.

6.2.11. Countermeasures and Mitigation

A key goal of FlexSpy is to enable *in-loop* evaluation of defenses, allowing designers to iteratively refine circuits at design time. We analyze two lightweight, circuit-level countermeasures that specifically target the quasi-DC leakage mechanism identified in Equation 6.3.

Countermeasure 1: Spike-Time Randomization (Jitter) The first defense aims to break the adversary's ability to align traces. We introduce bounded jitter (typically ±5%) into the neuron's reset path, implemented by modulating the effective threshold with a small pseudo-random current source. This jitter desynchronizes spike epochs across measurements, reducing the sharpness of CPA peaks and smearing the ΔI_{DC} features across time. As a result, profiling templates and regression models must average over more variability, weakening both label inference and rate recovery.

Countermeasure 2: Event Balancing (Dummy Conductance) The second defense directly attenuates the quasi-DC amplitude of ΔI_{DC}. We add a balancing branch (a switched TFT resistor or conductance element) that sources a small, calibrated counter-current whenever a synapse is active. The balancing element is sized to partially cancel the rate-weighted current offset, making the total current draw of an active epoch closer to that of an idle epoch. In feature space, this pulls the class clusters for ΔI_{DC} (cf. Figure 6.12) closer together, reducing separability.

6.2.12. Evaluation of Defenses

We re-run the full FlexSpy pipeline with spike-time jitter and event balancing enabled, both individually and in combination. The SLI metric provides a fine-grained view of hotspot mitigation, while aggregate metrics (AUC, NMSE, MI) quantify global security improvements.

Table 6.2 reports the SLI for the dominant window W2 in the f-SNN on P-Cons. In the unprotected baseline, the Synapse block exhibits an SLI of 0.55 in W2. Enabling only jitter reduces this to 0.43, while balancing alone cuts it to 0.22. The combination of jitter and balancing is most effective, suppressing the

Table 6.2.: Spike-Leakage Index (SLI) in dominant window W2 for f-SNN on P-Cons. Both countermeasures (CMs) are effective, and the combination provides the strongest hotspot suppression.

Block (W2)	Baseline	+Jitter	+Balancing	+Both
Synapse	0.55	0.43	0.22	**0.17**
Charge/Integrate	0.40	0.31	0.16	**0.12**
Reset/Discharge	0.18	0.14	0.07	**0.05**

Table 6.3.: Per-dataset countermeasure (CM) overheads and leakage reduction, grouped by architecture. Values are relative to each model's unprotected baseline; medians across parameter sweeps are reported.

Dataset	f-SNN				f-RNN			
	Area↑ (%)	**Power↑ (%)**	**Acc. Δ (pp (%))**	**Leak↓ (%)**	**Area↑ (%)**	**Power↑ (%)**	**Acc. Δ (pp (%))**	**Leak↓ (%)**
CBF	3.5	7.0	−0.2	60	2.8	5.5	−0.2	29
P-Cons	4.0	8.0	−0.3	56	3.0	6.0	−0.2	24
PPOC	4.5	8.5	−0.4	53	3.2	6.2	−0.3	23
DPTW	5.0	9.0	−0.5	50	3.5	6.5	−0.3	22
MPOAG	4.0	8.0	−0.3	55	3.0	6.0	−0.2	25
Symbols	3.5	7.0	−0.2	70	2.8	5.5	−0.2	38

CM = spike-time jitter (±5%) + event balancing. f-SNN benefits more because its leakage is dominated by spike-window ΔI_{DC} offsets, while f-RNN has smoother, lower-contrast envelopes.

primary Synapse leak by approximately 70% to an SLI of 0.17. Charge/Integrate and Reset/Discharge blocks exhibit similar trends, though with smaller absolute SLI values.

Table 6.3 summarizes area/power overhead and leakage reduction across datasets for both f-SNN and f-RNN. For f-SNN, the combined countermeasures reduce total leakage by 50–70% with area overheads of 3.5–5.0%, power overheads of 7.0–9.0%, and accuracy impact below 0.5 percentage points. The f-RNN, which exhibits weaker baseline leakage, still benefits from 22–38% leakage reduction at slightly lower overheads.

Overall, the countermeasures offer a highly favorable security–power trade-off. FlexSpy thus not only diagnoses f-SNN vulnerabilities but also validates practical, substrate-aware defenses.

6.3. Chapter Summary

This chapter introduced FlexSpy, the first substrate-aware, design-time security framework for flexible spiking neuromorphic hardware. We demonstrated that the unique combination of flexible substrates (thin, unshielded, shared rails) and SNN computational dynamics (event-driven, sparse) creates a significant and previously unquantified power side-channel vulnerability.

The core technical contribution is a unified device-to-design-to-network model that links the physics of a-IGZOs TFTs to observable, network-level information leakage. We identified the dominant leakage primitive: a rate-weighted, quasi-DC current offset stemming from active synaptic conductances during spike epochs. Using only a single, non-invasive power measurement and a self-generated virtual trigger, an attacker can exploit this primitive to infer labels with ROC-AUC = 0.91 and recover layer-wise spike rates with $NMSE = 0.14$.

Our comparative analysis showed that the f-SNNs event-driven model is substantially more "leaky" than the smoother dynamics of an f-RNN, both in terms of label inference and internal state reconstruction. To provide actionable design guidance, we developed the Spike-Leakage Index (Spike-Leakage Index (SLI)), an MI-based metric that successfully localizes security hotspots in time and across circuit blocks.

Finally, we demonstrated that this vulnerability is not fundamental. Two lightweight countermeasures, spike-time jitter and event balancing, were proposed and validated. Together, they suppress quasi-DC leakage by 38–70% with modest power overhead ($\leq$ 9%), small area cost, and negligible accuracy

degradation. FlexSpy thus provides a necessary, early-stage assessment tool and a pathway towards building a new generation of flexible neuromorphic devices that are *secure by design*. Future work will extend this framework to other channels (e.g., EM emissions) and explore co-design of training algorithms and circuit primitives to further harden f-SNN against side-channel threats.

7. Conclusion and Future Outlook

7.1. Conclusion

This dissertation presented a cross-layer methodology for designing dependable AI hardware on printed and flexible large-area electronic platforms. Unlike conventional CMOS technologies, these emerging substrates exhibit strong process variability, limited device performance, higher defect rates, and pronounced environmental sensitivity. These intrinsic characteristics invalidate many traditional hardware assumptions and necessitate a fundamentally different co-design philosophy in which learning algorithms, circuit primitives, architecture synthesis, reliability mechanisms, and security evaluation are tightly integrated.

The work first addressed the challenge of energy efficiency in bespoke neural network design under strict power constraints. By deriving hardware-aware power models for analog primitives and embedding them directly into the optimization process, power and energy consumption were treated as primary design objectives rather than post-layout metrics. This enabled explicit accuracy–energy trade-offs during training and architecture synthesis. The framework further demonstrated how temporal processing, event-driven computation, and mixed-signal co-design reduce interface overheads and improve end-to-end efficiency in large-area AI systems. These results establish that energy-efficient intelligence on flexible substrates must be co-optimized with the physical hardware characteristics from the earliest design stages.

Building upon this foundation, the dissertation introduced architecture-aware bespoke neural network design tailored to the non-ideal transfer characteristics of printed and flexible devices. Hardware-compatible activation functions and topology co-optimization strategies were developed to ensure functional robustness despite analog nonlinearities and device constraints. Automated architecture exploration, including variation-aware and evolutionary search methodologies, enabled joint optimization of network topology and circuit-level parameters. By incorporating environmental effects such as temperature-dependent drift into the design loop, the framework ensured stable performance across realistic operating conditions. This demonstrated that reliable inference in emerging technologies requires explicit alignment between algorithmic structure and device physics.

Beyond architectural robustness, the dissertation systematically investigated reliability, endurance, and test in large-area neural hardware. Fault sensitivity analyses revealed the limitations of purely variation-aware training when confronted with structural and catastrophic defects. To address runtime errors under severe resource constraints, lightweight mixed-signal error mitigation techniques were proposed, leveraging temporal redundancy and analog integration instead of conventional digital error correction. Adaptive endurance strategies further enabled graceful degradation and extended operational lifetime under realistic defect densities. In addition, technology-compatible structural test methodologies were developed to convert analog degradation effects into measurable digital signatures, thereby bridging the gap between large-area analog circuits and scalable manufacturability. Together, these contributions established that reliability must be embedded structurally into both the learning framework and the circuit architecture.

Finally, the dissertation examined security and trust in flexible and printed neuromorphic systems. By characterizing side-channel leakage mechanisms and evaluating realistic attack surfaces specific to analog and substrate-dependent implementations, the work highlighted the physical vulnerabilities inherent to large-area AI hardware. Security analysis frameworks were introduced to systematically evaluate leakage and fault-based attack resilience, alongside lightweight mitigation strategies that preserve energy and area efficiency. These findings emphasize that trust in emerging AI accelerators cannot be assumed; it must be explicitly analyzed and co-optimized alongside performance and reliability.

In summary, this dissertation demonstrates that dependable printed and flexible AI hardware is achievable when power efficiency, architectural compatibility, reliability, testability, and security are treated as unified and interdependent objectives. Rather than forcing silicon-centric methodologies onto fundamentally different substrates, the presented cross-layer framework embraces variability, non-idealities, and physical constraints as design parameters. By elevating robustness and trust to first-class optimization goals, this work advances a holistic methodology for constructing efficient and resilient intelligent systems beyond conventional CMOS paradigms.

7.2. Future Outlook

While this dissertation establishes foundational principles for reliable printed/flexible AI hardware, several research directions remain open and promising such as:

1. Cross-Layer Co-Optimization with Security Guarantees

Future work can extend architecture-aware *NAS* to jointly optimize for robustness, area, power, and *security*. As shown in prior chapters, analog and flexible neural accelerators exhibit side-channel leakage and fault-injection vulnerabilities. Integrating leakage-aware or fault-aware security metrics directly into evolutionary search would enable simultaneous optimization for reliability and resistance against physical attacks. This direction is particularly relevant for edge devices using Physical Unclonable Functions (PUFs) and bespoke printed accelerators.

2. Online Self-Healing and Adaptive Calibration

PRINT-SAFE demonstrated passive redundancy and fault-aware training at design time. A natural extension is dynamic self-healing during runtime. Emerging additive manufacturing technologies allow limited post-fabrication tuning (e.g., re-printing resistive elements or bias trimming). Coupling lightweight on-chip monitors with adaptive recalibration algorithms could enable long-term endurance against aging, bending-induced drift, or environmental degradation.

3. Scalable Mixed-Signal Test Architectures

The delay-based BIST methodology for *ReFlex-LDO* suggests that analog degradation can be mapped to digital observables. Extending this concept to full printed AI systems—including crossbars and nonlinear activation circuits—could yield scalable structural test frameworks for large-area printed/flexible systems. Furthermore, integrating test structures into architecture-aware design loops would allow co-optimization of functional and test coverage objectives.

4. Reliability Modeling Under Mechanical Stress and Aging

While thermal and process variations were explicitly modeled, long-term mechanical fatigue, substrate cracking, and bias stress in a-IGZO TFTs require deeper investigation. Coupling physics-based aging models with learning-aware optimization would allow predictive lifetime design for wearable and biomedical applications.

5. Standardized Design and Test Frameworks for PE/FE

Finally, the field would benefit from standardized reliability benchmarks, fault models, and open-source toolchains for printed/flexible AI hardware, analogous to what exists in silicon design. Establishing such infrastructure would accelerate adoption and industrial translation of reliable printed computing platforms.

Printed and flexible electronics challenge decades of CMOS-centric design philosophy. Devices are slower, noisier, and more variable, yet cheaper, lightweight, and mechanically adaptable. This dissertation

demonstrates that by embracing stochasticity, leveraging evolutionary design, and tightly coupling circuits with learning algorithms, it is possible to construct robust, testable, and secure AI accelerators in such emerging technologies.

Rather than attempting to force silicon methodologies onto printed systems, the future lies in developing technology-aware co-design frameworks where variability, redundancy, and learning are treated as complementary tools. In this vision, reliability is not enforced through rigidity, but achieved through adaptability.

Bibliography

[1] G. Kirchhoff, "On a deduction of ohm's laws", *Philosophical Magazine*, 1850.

[2] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[3] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, 1943.

[4] A. M. Turing, "Computing machinery and intelligence", *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[5] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current", *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[6] E. T. Jaynes, "Information Theory and Statistical Mechanics", *Physical Review*, vol. 106, p. 620, 1957.

[7] F. Rosenblatt, "The perceptron", *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[8] J. Von Neumann and R. Kurzweil, *The Computer and the Brain*. Yale University Press, 1958.

[9] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane", *Biophysical Journal*, vol. 1, no. 6, pp. 445–466, 1961.

[10] L. Milor et al., "Detection of catastrophic faults in analog integrated circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 2, pp. 114–130, 1989.

[11] J. L. Elman, "Finding Structure in Time", *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[12] J. L. Elman, "Finding structure in time", *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[13] R. Hecht-Nielsen, "Theory of the backpropagation neural network", in *Neural Networks for Perception*, Elsevier, 1992, pp. 65–93.

[14] S.-i. Amari, "Backpropagation and Stochastic Gradient Descent Method", *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[15] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1996, ISBN: 1886529043.

[16] H. Hyötyniemi, "Turing Machines are Recurrent Neural Networks", *Proceedings of step*, vol. 96, 1996.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] L. Prechelt, "Automatic Early Stopping Using Cross Validation: Quantifying the Criteria", *Neural networks*, vol. 11, no. 4, pp. 761–767, 1998.

[19] I. Bayraktaroglu, A. S. Ögrenci, G. Dündar, S. Balkır, and E. Alpaydın, "Annsys: An analog neural network synthesis system", *Neural Networks*, vol. 12, no. 2, pp. 325–338, 1999.

[20] N. Ibaraki, "Low temperature poly-si tft technology", in *SID Symposium Digest of Technical Papers*, vol. 30, 1999, p. 172.

[21] A. Devices, *An-83: Fundamentals of low noise analog circuit design*, Application Note, 2000. [Online]. Available: `https://www.analog.com/en/resources/app-notes/an-83.html`.

[22] M. Burns and G. W. Roberts, *An Introduction to Mixed-Signal IC Test and Measurement*. New York: Oxford University Press, 2001.

[23] W. Gerstner and W. M. Kistler, *Spiking Neuron Models*. Cambridge University Press, 2002.

[24] W. Gerstner and W. M. Kistler, *Spiking Neuron Models*. Cambridge University Press, 2002.

[25] K. O. Stanley et al., "Efficient Evolution of Neural Network Topologies", in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, IEEE, vol. 2, 2002, pp. 1757–1762.

[26] O. S. Unsal, I. Koren, and C. M. Krishna, "Towards energy-aware software-based fault tolerance in real-time systems", in *Proceedings of the 2002 International Symposium on Low Power Electronics and Design*, ser. ISLPED '02, Monterey, California, USA: Association for Computing Machinery, 2002, pp. 124–129, ISBN: 1581134754. DOI: 10.1145/566408.566442. [Online]. Available: https://doi.org/10.1145/566408.566442.

[27] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model", in *CHES 2004*, ser. LNCS, vol. 3156, Springer, 2004, pp. 16–29. DOI: 10.1007/978-3-540-28632-5_2.

[28] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley, 2006.

[29] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2e. New York, NY, USA: Springer, 2006.

[30] E. M. Ortigosa, A. Cañas, E. Ros, P. Martínez Ortigosa, S. Mota, and J. Díaz, "Hardware description of multi-layer perceptrons with different abstraction levels", *Microprocessors and Microsystems*, vol. 30, no. 7, pp. 435–444, 2006.

[31] R. J. Milliken, J. Silva-Martinez, and E. Sánchez-Sinencio, "Full on-chip CMOS low-dropout voltage regulator", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 9, pp. 1879–1890, Sep. 2007. DOI: 10.1109/TCSI.2007.904747.

[32] A. Zjajo, M. J. B. Asian, and J. P. de Gyvez, "Bist method for die-level process parameter variation monitoring in analog/mixed-signal integrated circuits", in *2007 Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1–6. DOI: 10.1109/DATE.2007.364477.

[33] Y. Dong, Y. Wang, Z. Lin, and T. Watanabe, "High performance and low latency mapping for neural network into network-on-chip architecture", in *IEEE International Conference on ASIC*, 2009, pp. 891–894.

[34] A. Graves et al., "A Novel Connectionist System for Unconstrained Handwriting Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[35] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks", *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009.

[36] D. Sung, A. de la Fuente Vornbrock, and V. Subramanian, "Scaling and optimization of gravure-printed silver nanoparticle lines for printed electronics", *IEEE Transactions on Components and Packaging Technologies*, vol. 33, no. 1, pp. 105–114, 2009.

[37] S. An et al., "47.2: 2.8-inch wqvga flexible amoled using high performance low temperature polysilicon tft on plastic substrates", in *48th Annual SID Symposium, Seminar, and Exhibition 2010 (Display Week 2010)*, vol. 2, 2010, pp. 706–709.

[38] F. Armknecht et al., "Memory leakage-resilient encryption based on physically unclonable functions", in *Towards Hardware-Intrinsic Security*, Springer, 2010, pp. 135–164.

[39] S. J. Johnson, *Iterative Error Correction: Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes*. Cambridge University Press, 2010, Key kept as johnson2004ldpc for consistency with thesis text; please adjust year or details if you use a different LDPC reference.

[40] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress", *Neurocomputing*, vol. 74, no. 1–3, pp. 239–255, 2010.

[41] R. Mozuelos, Y. Lechuga, M. Martínez, and S. Bracho, "Test of embedded analog circuits based on a built-in current sensor", in *2010 15th IEEE European Test Symposium*, 2010, pp. 164–169. DOI: 10.1109/ETSYM.2010.5512765.

[42] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling", in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 1947–1950.

[43] J. V. Arthur and K. A. Boahen, "Silicon-neuron design: A dynamical systems approach", *IEEE Transactions on Circuits and Systems I*, vol. 58, no. 5, pp. 1034–1043, 2011.

[44] R. Mazumder et al., "SparseNet: Coordinate Descent with Nonconvex Penalties", *Journal of the American Statistical Association*, vol. 106, no. 495, pp. 1125–1138, 2011.

[45] D. Zhou, M. Wang, and S. Zhang, "Degradation of amorphous silicon thin film transistors under negative gate bias stress", *IEEE Transactions on Electron Devices*, vol. 58, no. 10, pp. 3422–3427, 2011.

[46] E. Fortunato, P. Barquinha, and R. Martins, "Oxide semiconductor thin-film transistors: A review of recent advances", *Advanced Materials*, vol. 24, no. 6, pp. 2945–2986, 2012.

[47] M. Hiller et al., "A survey of physically unclonable function-based security solutions", *IEEE Design & Test*, 2012, Placeholder survey entry; please replace with the exact PUF survey you intend to cite (authors, volume, pages).

[48] M. Hiller, G. Sigl, et al., "Complementary IBS: Application-specific error correction for physically unclonable functions", in *Proc. Design, Automation & Test in Europe (DATE)*, Exact pages and author list to be updated according to the specific IBS/PUF paper you use., 2012.

[49] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 2012.

[50] P. Kurup et al., *Logic synthesis using Synopsys®*. Springer Science & Business Media, 2012.

[51] T. Tieleman, "Rmsprop", 2012.

[52] Y. Bengio et al., "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation", *arXiv preprint arXiv:1308.3432*, 2013.

[53] Y. Chauvin and D. E. Rumelhart, *Backpropagation: Theory, Architectures, and Applications*. Psychology press, 2013.

[54] A. M. Gaikwad et al., "A Flexible High Potential Printed Battery for Powering Printed Electronics", *Applied Physics Letters*, vol. 102, no. 23, p. 104, 2013.

[55] S. K. Garlapati et al., "Electrolyte-gated, high mobility inorganic oxide transistors from printed metal halides", *ACS Applied Materials & Interfaces*, vol. 5, no. 22, pp. 11 498–11 502, 2013. DOI: 10.1021/am403131j.

[56] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes", *arXiv preprint arXiv:1312.6114*, 2013.

[57] P. Koeberl, J. Li, and W. Wu, "A spatial majority voting technique to reduce error rate of physically unclonable functions", in *Proc. Int. Conf. Trusted Systems*, 2013, pp. 36–52.

[58] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation", in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[59] M. Fernández-Delgado et al., "Do We Need Hundreds of Classifiers To Solve Real World Classification Problems?", *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[60] S. Gong et al., "A flexible and printed pressure sensor for wearable and large-area applications", *Sensors and Actuators A: Physical*, 2014.

[61] S. Gong et al., “A wearable and highly sensitive pressure sensor with ultrathin gold nanowires”, *Nature communications*, vol. 5, no. 1, pp. 1–8, 2014.

[62] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, “Physical unclonable functions and applications: A tutorial”, *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[63] D. Hu, X. Zhang, Z. Xu, S. Ferrari, and P. Mazumder, “Digital implementation of a spiking neural network (snn) capable of spike-timing-dependent plasticity (stdp) learning”, in *Proc. IEEE Int. Conf. Nanotechnology*, 2014, pp. 873–876.

[64] S. Khan et al., “Technologies for printing sensors and electronics over large flexible substrates: A review”, *IEEE Sensors Journal*, vol. 15, no. 6, pp. 3164–3185, 2014.

[65] S. Khan et al., “Technologies for Printing Sensors and Electronics over Large Flexible Substrates: A Review”, *IEEE Sensors Journal*, vol. 15, p. 3164, 2014.

[66] D. P. Kingma et al., “Adam: A Method for Stochastic Optimization”, *arXiv preprint arXiv:1412.6980*, 2014.

[67] D. P. Kingma and J. Ba, “Adam”, *arXiv:1412.6980*, 2014.

[68] D. Monroe, “Neuromorphic computing gets ready for the (really) big time”, *Communications of the ACM*, 2014.

[69] R. A. Nawrocki et al., “Neurons in polymer: Hardware neural units based on polymer memristive devices and polymer transistors”, *IEEE Transactions on Electron Devices*, vol. 61, no. 10, pp. 3513–3519, 2014.

[70] H. Ohshima, “Value of ltps: Present and future”, in *Digest of Technical Papers - SID International Symposium*, vol. 45, 2014, pp. 75–78.

[71] M. Abadi et al., *Tensorflow*, 2015.

[72] Y. Chen, E. Keogh, et al., *The ucr time series classification archive*, `https://www.cs.ucr.edu/~eamonn/time_series_data/`, Accessed: 2025-11-18, 2015.

[73] Y. Chen et al., *The ucr time series classification archive*, `www.cs.ucr.edu/~eamonn/time_series_data/`, Jul. 2015.

[74] S. Han et al., “Learning Both Weights and Connections for Efficient Neural Network”, *Advances in neural information processing systems*, vol. 28, 2015.

[75] S. Li, L. D. Xu, and S. Zhao, “The Internet of Things: A Survey”, *Information systems frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[76] Y.-C. Lin et al., “Role of interface traps in IGZO thin-film transistors”, *IEEE Transactions on Electron Devices*, 2015.

[77] P. Mostafalu et al., “Wireless flexible smart bandage for continuous monitoring of wound oxygenation”, *IEEE Transactions on BioCAS*, vol. 9, no. 5, pp. 670–677, 2015.

[78] H. Qian et al., “Temperature-dependent characteristics of amorphous IGZO thin-film transistors”, *IEEE Electron Device Letters*, 2015.

[79] H.-M. Qian et al., “Temperature-dependent bias-stress-induced electrical instability of amorphous indium-gallium-zinc-oxide thin-film transistors*”, *Chinese Physics B*, vol. 24, no. 7, p. 077 307, May 2015. DOI: `10.1088/1674-1056/24/7/077307`. [Online]. Available: `https://dx.doi.org/10.1088/1674-1056/24/7/077307`.

[80] A. Tisan and J. Chin, “An end user platform for implementing artificial neural networks on fpga”, in *IEEE International Conference on Industrial Informatics (INDIN)*, 2015, pp. 856–859.

[81] C.-C. Wang, W.-J. Lu, and T.-C. Wu, “Wide-range ctat and ptat sensors with second-order calibration for on-chip thermal monitoring”, *Microelectronics Journal*, vol. 46, no. 9, pp. 819–824, 2015, ISSN: 1879-2391. DOI: `https://doi.org/10.1016/j.mejo.2015.06.004`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0026269215001378`.

[82] M. Wang et al., “Liquid-metal-based stretchable conductors for wearable and large-area electronics”, *Advanced Functional Materials*, 2015.

[83] M. J. Barragan, H.-G. Stratigopoulos, S. Mir, H. Le-Gall, N. Bhargava, and A. Bal, “Practical simulation flow for evaluating analog/mixed-signal test techniques”, *IEEE Design & Test*, vol. 33, no. 6, pp. 46–54, 2016. DOI: 10.1109/MDAT.2016.2590985.

[84] Y.-H. Chen et al., “Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks”, *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016.

[85] K. Chen et al., “Printed carbon nanotube electronics and sensor systems”, *Advanced Materials*, vol. 28, no. 6, pp. 4397–4414, 2016.

[86] Z. Cui, *Printed Electronics: Materials, Technologies and Applications*. John Wiley & Sons, 2016.

[87] Z. Cui, *Printed Electronics*. Wiley, 2016.

[88] K. Greff et al., “LSTM: A Search Space Odyssey”, *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[89] T. Instruments, *Power management lab kit: Low dropout regulator experiments*, User Guide SSQU011, 2016. [Online]. Available: https://www.ti.com/lit/ug/ssqu011/ssqu011.pdf.

[90] J. Kim et al., “Noninvasive alcohol monitoring using a wearable tattoo-based iontophoretic-biosensing system”, *Acs Sensors*, vol. 1, no. 8, pp. 1011–1019, 2016.

[91] Y. C. Kim, S. J. Lee, I.-K. Oh, S. Seo, H. Kim, and J.-M. Myoung, “Bending stability of flexible amorphous igzo thin film transistors with transparent izo/ag/izo oxide–metal–oxide electrodes”, *Journal of Alloys and Compounds*, vol. 688, pp. 1108–1114, 2016, ISSN: 0925-8388. DOI: https://doi.org/10.1016/j.jallcom.2016.07.169. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925838816322034.

[92] S.-H. Lee et al., “Quality control of inkjet-printed electronics using image-based defect detection”, *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 6, no. 12, pp. 185–194, 2016, Representative quality-control reference; adjust to the exact Lee et al. paper you are citing.

[93] H. Li et al., “Pruning Filters for Efficient Convnets”, *arXiv preprint arXiv:1608.08710*, 2016.

[94] Q. Liu and S. Furber, “Noisy softplus: A biology inspired activation function”, in *Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part IV 23*, Springer, 2016, pp. 405–412.

[95] L. A. Pastur-Romay, F. Cedrón, A. Pazos, and A. B. Porto-Pazos, “Deep artificial neural networks and neuromorphic chips for big data analysis”, *International Journal of Molecular Sciences*, vol. 17, no. 8, p. 1313, 2016.

[96] M. Shafique et al., “Cross-layer approximate computing: From logic to architectures”, in *Proc. Design Automation Conference (DAC)*, 2016, pp. 1–6.

[97] E. Sowade et al., “All-inkjet-printed silver lines: Correlation of printing orientation and electrical performance”, *Flexible and Printed Electronics*, vol. 1, no. 1, p. 015 001, 2016, Title/details are indicative of Sowade’s work on orientation- dependent inkjet printing; update to the exact paper you are using.

[98] E. Sowade et al., “All-inkjet-printed thin-film transistors: Manufacturing process reliability by root cause analysis”, *Scientific reports*, vol. 6, no. 1, p. 33 490, 2016.

[99] E. Sowade, M. Polomoshnov, and R. R. Baumann, “The design challenge in printing devices and circuits: Influence of the orientation of print patterns in inkjet-printed electronics”, *Organic Electronics*, vol. 37, pp. 428–438, 2016, ISSN: 1566-1199. DOI: https://doi.org/10.1016/j.orgel.2016.07.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566119916303068.

[100] P. Vilmi et al., “Fully printed memristors for a self-sustainable recorder of mechanical energy”, *Flexible and Printed Electronics*, vol. 1, no. 2, p. 025 002, 2016.

[101] M. M. Billah, M. M. Hasan, and J. Jang, “Effect of tensile and compressive bending stress on electrical performance of flexible a-igzo tfts”, *IEEE Electron Device Letters*, vol. 38, no. 7, pp. 890–893, 2017. DOI: 10.1109/LED.2017.2707279.

[102] G. Cadilha Marques et al., “Digital power and performance analysis of inkjet printed ring oscillators based on electrolyte-gated oxide electronics”, *Applied Physics Letters*, vol. 111, no. 10, p. 102 103, 2017.

[103] J. S. Chang, A. F. Facchetti, and R. Reuss, “A circuits and systems perspective of organic/printed electronics: Review, challenges, and contemporary and emerging design approaches”, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 7, no. 1, pp. 7–26, 2017.

[104] P. Heremans et al., “Flexible metal-oxide thin film transistor circuits for rfid and health patches”, in *Technical Digest - International Electron Devices Meeting (IEDM)*, 2017, pp. 6.3.1–6.3.4.

[105] K. Kuribara et al., “Organic physically unclonable function on flexible substrate operable at 2 v for iot/ioe security applications”, *Organic Electronics*, vol. 51, pp. 137–141, 2017.

[106] T.-H. Lin et al., “Wearable Inkjet Printed Energy Harvester”, in *2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, IEEE, 2017, pp. 1613–1614.

[107] I. Loshchilov, F. Hutter, et al., “Fixing weight decay regularization in adam”, *arXiv preprint arXiv:1711.05101*, vol. 5, 2017.

[108] R. Martins et al., “Bias-stress stability of amorphous IGZO-based thin-film transistors”, *Journal of Display Technology*, 2017.

[109] A. Ren et al., “Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing”, in *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2017, pp. 405–418.

[110] C. D. Schuman et al., “A survey of neuromorphic computing and neural networks in hardware”, *arXiv preprint*, vol. arXiv:1705.06963, 2017.

[111] C. D. Schuman et al., “A Survey of Neuromorphic Computing and Neural Networks in Hardware”, *arXiv preprint arXiv:1705.06963*, 2017.

[112] C. D. Schuman et al., “A survey of neuromorphic computing and neural networks in hardware”, *arXiv preprint arXiv:1705.06963*, 2017.

[113] S. Sharma, S. Sharma, and A. Athaiya, “Activation functions in neural networks”, *Towards Data Science*, 2017.

[114] A. Vaswani et al., “Attention Is All You Need”, *Advances in neural information processing systems*, vol. 30, 2017.

[115] A. T. E. et al., “Inkjet-printed egfet-based physical unclonable function—design, evaluation, and fabrication”, *IEEE TVLSI*, vol. 26, no. 12, pp. 2935–2946, 2018.

[116] A. Basu et al., “Low-Power, Adaptive Neuromorphic Systems: Recent Progress and Future Directions”, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 6–27, 2018.

[117] H. Cai et al., “Efficient Architecture Search by Network Transformation”, in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[118] J. A. Cardenas, M. J. Catenacci, J. B. Andrews, N. X. Williams, B. J. Wiley, and A. D. Franklin, “In-place printing of carbon nanotube transistors at low temperature”, *ACS Applied Nano Materials*, vol. 1, no. 4, pp. 1863–1869, 2018.

[119] H. A. Dau et al., *The UCR Time Series Classification Archive*, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/, Oct. 2018.

[120] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[121] P. He et al., "Fully printed high performance humidity sensors based on two-dimensional materials", *Nanoscale*, vol. 10, no. 12, pp. 5599–5606, 2018.

[122] Y. He et al., "Flexible printed humidity sensors for low-cost environmental monitoring", *Sensors and Actuators B: Chemical*, 2018.

[123] M. Jaworski, P. Duda, and L. Rutkowski, "New splitting criteria for decision trees in stationary data streams", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2516–2529, 2018. DOI: 10.1109/TNNLS.2017.2698204.

[124] W. L. et al., *Systems and methods for built-in self test of low dropout regulators*, US Patent 9933802, 64 2018.

[125] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. Gonzalez, and I. Stoica, *Tune: A research platform for distributed model selection and training*, Accessed: 2024-07-11, 2018. [Online]. Available: https://docs.ray.io/en/latest/tune.html.

[126] H. Liu et al., "DARTS: Differentiable Architecture Search", in *International Conference on Learning Representations (ICLR)*, 2018.

[127] K. Myny, "The development of flexible integrated circuits based on thin-film transistors", *Nature Electronics*, vol. 1, no. 1, pp. 30–39, 2018.

[128] C. W. Park, J. B. Koo, C. S. Hwang, H. Park, S. G. Im, and S. Y. Lee, "Stretchable active matrix of oxide thin-film transistors with monolithic liquid metal interconnects", *Applied Physics Express*, vol. 11, no. 12, p. 126 501, 2018.

[129] F. Rasheed et al., "Variability Modeling for Printed Inorganic Electrolyte-gated Transistors and Circuits", *IEEE transactions on electron devices*, vol. 66, no. 1, pp. 146–152, 2018.

[130] F. Rasheed et al., "Variability modeling for printed inorganic electrolyte-gated transistors and circuits", *IEEE TED*, vol. 66, no. 1, pp. 146–152, 2018.

[131] D. Weller et al., "An inkjet-printed low-voltage latch based on inorganic electrolyte-gated transistors", *IEEE EDL*, vol. 39, no. 6, pp. 831–834, 2018.

[132] Z. Zhong et al., "Practical Block-wise Neural Network Architecture Generation", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2423–2432.

[133] B. Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[134] G. Cadilha Marques et al., "Progress Report on "From Printed Electrolyte-Gated Metal-Oxide Devices to Circuits"", *Advanced Materials*, vol. 31, no. 26, p. 1 806 483, 2019.

[135] T. Elsken et al., "Neural Architecture Search: A Survey", *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[136] H. M. Fahad, M. A. Alam, A. Dodabalapur, R. S. Williams, and A. Javey, "Flexible and transparent thin-film transistors: From materials to applications", *Advanced Materials*, vol. 31, no. 25, p. 1 807 879, 2019. DOI: 10.1002/adma.201807879.

[137] S. Jain et al., "Neural network accelerator design with resistive crossbars", *IBM Journal of Research and Development*, 2019.

[138] A. Kaidarova et al., "Wearable multifunctional printed graphene sensors", *NPJ Flexible Electronics*, vol. 3, no. 1, pp. 1–10, 2019.

[139] B. Li et al., "Build Reliable and Efficient Neuromorphic Design with Memristor Technology", in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 2019, pp. 224–229.

[140] H. Li et al., "Reliable and energy-efficient design of memristor-based neuromorphic computing systems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019, Cross-layer reliability paper for memristor-based neuromorphic systems; fill in exact volume/number/pages for your chosen Li et al.

[141] S. Li and Y. Chen, "Non-invasive stability measurement technique for linear voltage regulators", *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 6, pp. 2686–2691, 2019. DOI: 10.30534/ijatcse/2019/26862019.

[142] R. Ma et al., "Transformed ℓ_1 Regularization for Learning Sparse Deep Neural Networks", *Neural Networks*, vol. 119, pp. 286–298, 2019.

[143] M. Mikolajek et al., "Fabrication and characterization of fully inkjet printed capacitors based on ceramic/polymer composite dielectrics on flexible substrates", *Scientific Reports*, vol. 9, no. 1, p. 13 324, 2019.

[144] M. Mikolajek et al., "Fabrication and Characterization of Fully Inkjet Printed Capacitors Based on Ceramic/Polymer Composite Dielectrics on Flexible Substrates", *Scientific reports*, vol. 9, no. 1, p. 13 324, 2019.

[145] S. G. Noyce, J. L. Doherty, Z. Cheng, H. Han, S. Bowen, and A. D. Franklin, "Electronic stability of carbon nanotube transistors under long-term bias stress", *Nano Letters*, vol. 19, no. 3, pp. 1460–1466, 2019.

[146] E. Ozer et al., "A bespoke machine-learning processor on flexible substrates", in *Proc. IEEE Int. Conf. on Flexible Electronics and Sensors (FLEPS)*, 2019.

[147] A. Paszke et al., "Pytorch", *NeurIPS*, 2019.

[148] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library", *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[149] A. Paszke et al., "Pytorch: An Imperative Style, High-performance Deep Learning Library", in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[150] F. Rasheed et al., "Predictive modeling and design automation of inorganic printed electronics", in *2019 IEEE DATE*, 2019, pp. 30–35.

[151] F. Rasheed, M. Hefenbrock, M. Beigl, M. B. Tahoori, and J. Aghassi-Hagmann, "Variability modeling for printed inorganic electrolyte-gated transistors and circuits", *IEEE Transactions on Electron Devices*, vol. 66, no. 1, pp. 146–152, 2019. DOI: 10.1109/TED.2018.2867461.

[152] F. Rasheed, M. Hefenbrock, R. Bishnoi, M. BeigI, J. Aghassi-Hagmann, and M. B. Tahoori, "Predictive modeling and design automation of inorganic printed electronics", in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 30–35. DOI: 10.23919/DATE.2019.8715159.

[153] G. Tzimpragos et al., "Boosted race trees for low energy classification", in *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019, pp. 215–228.

[154] E.-I. Vatajelu, G. Di Natale, and L. Anghel, "Special session: Reliability of hardware-implemented spiking neural networks (snn)", in *Proc. IEEE VLSI Test Symposium (VTS)*, 2019, pp. 1–8.

[155] E.-I. Vatajelu, G. Di Natale, and L. Anghel, "Special session: Reliability of hardware-implemented spiking neural networks (snn)", in *2019 IEEE 37th VLSI Test Symposium (VTS)*, 2019, pp. 1–8. DOI: 10.1109/VTS.2019.8758653.

[156] X. Zou, Y. Hu, Z. Tian, and K. Shen, "Logistic regression model optimization and case analysis", in *2019 IEEE 7th international conference on computer science and network technology (ICCSNT)*, IEEE, 2019, pp. 135–139.

[157] D. Baran, D. Corzo, and G. Blazquez, "Flexible electronics: Status, challenges and opportunities", *Frontiers in Electronics*, vol. 1, 2020.

[158] N. Bleier, M. Mubarik, F. Rasheed, J. Aghassi-Hagmann, M. B. Tahoori, and R. Kumar, "Printed microprocessors", in *Annu. Int. Symp. Computer Architecture (ISCA)*, Jun. 2020, pp. 213–226.

[159] N. Bleier, M. H. Mubarik, F. Rasheed, J. Aghassi-Hagmann, M. B. Tahoori, and R. Kumar, "Printed microprocessors", in *Proc. Int. Symp. Computer Architecture (ISCA)*, 2020, pp. 213–226.

[160] N. Bleier et al., "Printed microprocessors", in *2020 ACM/IEEE 47th Annual ISCA*, IEEE, 2020.

[161] A. T. Erozan et al., "A novel printed look-up table based programmable printed digital circuit", *IEEE TVLSI Systems*, 2020.

[162] A. T. Erozan, "Security aspects of printed electronics applications", 2020.

[163] N. R. Hosseini et al., "Organic electronics-based axon-hillock neuromorphic circuit on flexible substrates", *Organic Electronics*, 2020.

[164] M. Ince and S. Ozev, "Digital defect-based built-in self-test for low dropout voltage regulators", in *IEEE European Test Symposium (ETS)*, 2020, pp. 1–2. DOI: 10.1109/ETS48528.2020.9131572.

[165] T. Ince and S. Ozev, "A digital bist approach for low-dropout regulators", in *IEEE International Test Conference (ITC)*, 2020, pp. 1–10. DOI: 10.1109/ITC44778.2020.9325220.

[166] S. Kim, "Inkjet-Printed Electronics on Paper for RF Identification (RFID) and Sensing", *Electronics*, vol. 9, no. 10, p. 1636, 2020.

[167] H. Kleemann, K. Krechan, A. Fischer, and K. Leo, "A review of vertical organic transistors", *Advanced Functional Materials*, vol. 30, no. 5, p. 1 907 113, 2020.

[168] M. H. Mubarik et al., "Printed machine learning classifiers", in *Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2020, pp. 73–87.

[169] M. H. Mubarik et al., "Printed machine learning classifiers", in *2020 53rd Annual IEEE/ACM MICRO*, 2020, pp. 73–87. DOI: 10.1109/MICRO50266.2020.00019.

[170] M. H. Mubarik et al., "Printed machine learning classifiers", in *Annu. Int. Symp. Microarchitecture (MICRO)*, 2020, pp. 73–87. DOI: 10.1109/MICRO50266.2020.00019.

[171] E. Ozer et al., "A hardwired machine-learning processing engine fabricated with submicron metal-oxide thin-film transistors on flexible substrates", *Nature Electronics*, 2020.

[172] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function", 2020.

[173] A. Scholz et al., "Hybrid low-voltage physical unclonable function based on inkjet-printed metal-oxide transistors", *Nature Communications*, vol. 11, no. 1, p. 5543, 2020. DOI: 10.1038/s41467-020-19324-5.

[174] Tingting et al., "Improved convolutional neural network fault diagnosis method based on dropout", in *2020 7th IFEEA*, 2020, pp. 753–758. DOI: 10.1109/IFEEA51475.2020.00160.

[175] D. D. Weller et al., "Programmable neuromorphic circuit based on printed electrolyte-gated transistors", in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 446–451.

[176] D. D. Weller et al., "Programmable neuromorphic circuit based on printed electrolyte-gated transistors", 2020.

[177] D. D. Weller, M. Hefenbrock, M. B. Tahoori, J. Aghassi-Hagmann, and M. Beigl, "Programmable neuromorphic circuit based on printed electrolyte-gated transistors", in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 446–451.

[178] D. Wu et al., "Ugemm: Unary computing architecture for gemm applications", in *Proc. Int. Symp. Computer Architecture (ISCA)*, 2020, pp. 377–390.

[179] J. Xu et al., "Reluplex made more practical: Leaky ReLU", in *2020 IEEE Symposium on Computers and communications (ISCC)*, IEEE, 2020, pp. 1–7.

[180] J. Zhao et al., "Ambipolar deep-subthreshold printed-carbon-nanotube transistors for ultralow-voltage and ultralow-power electronics", *ACS Nano*, vol. 14, no. 10, pp. 14 036–14 046, 2020.

[181] H. Abdolmaleki, P. Kidmose, and S. Agarwala, "Droplet-based techniques for printing of functional inks for flexible physical sensors", *Advanced Materials*, vol. 33, no. 20, p. 2 006 792, 2021.

[182] A. Ahmed et al., "Additively manufactured nano-mechanical energy harvesting systems: Advancements, potential applications, challenges and future perspectives", *Nano Convergence*, vol. 8, no. 37, 2021. DOI: 10.1186/s40580-021-00289-0.

[183] A. U. Alam et al., "Fruit quality monitoring with smart packaging", *Sensors*, vol. 21, no. 4, p. 1509, 2021.

[184] A. U. Alam et al., "Fruit Quality Monitoring with Smart Packaging", *Sensors*, vol. 21, no. 4, p. 1509, 2021.

[185] M. S. Asghar, S. Arslan, and H. Kim, "A low-power spiking neural network chip based on a compact lif neuron and binary exponential charge injector synapse circuits", *Sensors*, vol. 21, no. 13, p. 4462, 2021.

[186] tsaug Contributors, *Tsaug: A python package for time series data augmentation*, Accessed: 2024-07-11, 2021. [Online]. Available: https://github.com/arundo/tsaug.

[187] A. T. Erozan et al., "Defect detection in transparent printed electronics using learning-based optical inspection", *IEEE TVLSI Systems*, 2021.

[188] J. K. Eshraghian et al., "Training Spiking Neural Networks Using Lessons from Deep Learning", *arXiv preprint arXiv:2109.12894*, 2021.

[189] E. Guo et al., "Integrated complementary inverters and ring oscillators based on vertical-channel dual-base organic thin-film transistors", *Nature Electronics*, vol. 4, pp. 588–594, 2021.

[190] M. J. M. Hosseini and R. A. Nawrocki, "A review of the progress of thin-film transistors and their technologies for flexible electronics", *Micromachines*, vol. 12, p. 655, 2021.

[191] M. Ince et al., "Fault-based built-in self-test and evaluation of phase locked loops", *ACM Trans. Des. Autom. Electron. Syst.*, vol. 26, no. 3, Jan. 2021, ISSN: 1084-4309. DOI: 10.1145/3427911. [Online]. Available: https://doi.org/10.1145/3427911.

[192] I. I. Labiano et al., "Flexible inkjet-printed graphene antenna on kapton", *Flexible and Printed Electronics*, vol. 6, no. 2, p. 025 010, 2021.

[193] T. Liang et al., "Pruning and Quantization for Deep Neural Network Acceleration: A Survey", *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[194] Y. Liu et al., "A survey of stochastic computing neural networks for machine learning applications", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2809–2824, 2021.

[195] F. Rasheed et al., "Channel geometry scaling effect in printed inorganic electrolyte-gated transistors", *IEEE TED*, vol. 68, no. 4, pp. 1866–1871, 2021.

[196] J. M. Real, W. Zhao, and L. Bossuet, "Physical side-channel attacks on deep neural networks: A survey", *Applied Sciences*, vol. 11, no. 16, p. 7369, 2021. DOI: 10.3390/app11167369.

[197] A. Scholz, "Components and systems based on printed metal oxide electronics and silicon devices", PhD dissertation, Karlsruhe Institute of Technology (KIT), 2021. DOI: 10.5445/IR/1000140819.

[198] A. Tulsiram and W. R. Eisenstadt, " Design for Testability of Low Dropout Regulators ", in *2021 IEEE 39th VLSI Test Symposium (VTS)*, Los Alamitos, CA, USA: IEEE Computer Society, Apr. 2021, pp. 1–7. DOI: 10.1109/VTS50974.2021.9441007. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/VTS50974.2021.9441007.

[199] D. Weller, "Digital and analog computing paradigms in printed electronics", Ph.D. dissertation, Karlsruhe Institute of Technology, 2021.

[200] D. D. Weller et al., "Printed stochastic computing neural networks", in *2021 IEEE DATE*, 2021, pp. 914–919.

[201] D. D. Weller et al., "Realization and training of an inverter-based printed neuromorphic computing system", *Scientific Reports*, 2021.

[202] D. D. Weller, M. Hefenbrock, M. Beigl, J. Aghassi-Hagmann, and M. B. Tahoori, "Realization and training of an inverter-based printed neuromorphic computing system", *Scientific Reports*, vol. 11, p. 9554, 2021.

[203] D. D. Weller et al., "Printed stochastic computing neural networks", in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 914–919.

[204] H. Zhao, T. Röddiger, and M. Beigl, "Aircase: Earable Charging Case with Air Quality Monitoring and Soundscape Sonification", in *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*, 2021, pp. 180–184.

[205] G. Armeniakos et al., "Cross-layer approximation for printed machine learning circuits", in *Proc. Design, Automation & Test in Europe (DATE)*, 2022, pp. 190–195.

[206] G. Armeniakos et al., "Cross-layer approximation for printed machine learning circuits", in *2022 DATE*, 2022, pp. 190–195. DOI: 10.23919/DATE54114.2022.9774689.

[207] K. Balaskas et al., "Approximate decision trees for machine learning classification on tiny printed circuits", in *Proc. Int. Symp. Quality Electronic Design*, 2022, pp. 1–6.

[208] K. Balaskas et al., "Approximate decision trees for machine learning classification on tiny printed circuits", in *Int. Symp. Quality Electronic Design*, 2022, pp. 1–6. DOI: 10.1109/ISQED54688.2022.9806213.

[209] A. Banerjee, A. Raychowdhury, and S. S. Mukherjee, "A survey of techniques for designing and optimizing low drop-out regulators", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 1, pp. 1–17, Jan. 2022. DOI: 10.1109/TCSII.2021.3102011.

[210] S. D. Gardner et al., "An inkjet-printed artificial neuron for physical reservoir computing", *IEEE Journal on Flexible Electronics*, vol. 1, no. 3, pp. 185–193, 2022.

[211] M. Hefenbrock et al., "In-situ training of printed neural networks to compensate process variations", *Flexible and Printed Electronics*, 2022, Based on the in-situ tuning work on printed NNs; complete bibliographic details once the final publication info is fixed.

[212] M. Hefenbrock et al., "In-situ tuning of printed neural networks for variation tolerance", in *Proceedings of DATE*, 2022.

[213] M. Hefenbrock, "Modelling and training printed neuromorphic circuits", Ph.D. dissertation, KIT, 2022.

[214] I. Hendy, J. Brewer, and S. Muir, "Development of high-performance igzo backplanes for displays the change in availability of gen 8 front-plane processing techniques for oled provides the additional impetus behind demand for a metal-oxide semiconductor tft", *Information Display*, vol. 38, no. 9, pp. 60–67, 2022.

[215] J. Li et al., "Micro and nano materials and processing techniques for printed biodegradable electronics", *Materials Today Nano*, vol. 18, 2022.

[216] X. Lin, "Thin film transistor - basic principles and commercial status", *Highlights in Science, Engineering and Technology*, vol. 27, pp. 428–435, 2022.

[217] L. Ouyang et al., "Training Language Models to Follow Instructions with Human Feedback", *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[218] S. A. Singaraju et al., "Artificial Neurons on Flexible Substrates: A Fully Printed Approach for Neuromorphic Sensing", *Sensors*, vol. 22, no. 11, p. 4000, 2022.

[219] Q. Sun et al., "Smart Band-Aid: Multifunctional and Wearable Electronic Device for Self-Powered Motion Monitoring and Human-Machine Interaction", *Nano Energy*, vol. 92, p. 106 840, 2022.

[220] H. Zhao et al., "Aging-Aware Training for Printed Neuromorphic Circuits", in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22)*, San Diego, CA, USA, 2022. DOI: 10.1145/3508352.3549411.

[221] L. B. Zilch, A. G. Ferreira, M. S. Lubaszewski, and T. R. Balen, "Evaluating fault coverage of structural and specification-based tests obtained with a low-cost analog tpg tool", in *2022 IEEE 23rd Latin American Test Symposium (LATS)*, 2022, pp. 1–6. DOI: 10.1109/LATS57337.2022.9936922.

[222] F. Afentaki, G. Saglam, A. Kokkinis, K. Siozios, G. Zervakis, and M. B. Tahoori, "Bespoke approximation of multiplication-accumulation and activation targeting printed multilayer perceptrons", in *2023 IEEE/ACM ICCAD*, IEEE, Oct. 2023. DOI: 10.1109/iccad57390.2023.10323613. [Online]. Available: http://dx.doi.org/10.1109/ICCAD57390.2023.10323613.

[223] M. Ahmadilivani et al., "Special session: Approximation and fault resiliency of dnn accelerators", in *2023 IEEE 41st (VTS)*, 2023, pp. 1–10.

[224] G. Armeniakos, G. Zervakis, D. Soudris, M. B. Tahoori, and J. Henkel, "Co-design of approximate multilayer perceptron for ultra-resource constrained printed circuits", *IEEE Trans. Comput.*, pp. 1–8, 2023.

[225] G. Armeniakos, G. Zervakis, D. Soudris, M. B. Tahoori, and J. Henkel, "Model-to-circuit cross-approximation for printed machine learning classifiers", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2023.

[226] K. Asifuzzaman, N. R. Miniskar, A. R. Young, F. Liu, and J. S. Vetter, "A survey on processing-in-memory techniques: Advances and challenges", *Memories–Materials, Devices, Circuits and Systems*, vol. 4, p. 100 022, 2023.

[227] G. Chen et al., "Temperature-controlled multisensory neuromorphic devices for artificial visual dynamic capture enhancement", *Nano Research*, vol. 16, no. 5, pp. 7661–7670, 2023.

[228] W. R. Eisenstadt and A. Tulsiram, *Self-test for low dropout regulator measurement*, US Patent 11,687,109, Jun. 2023.

[229] A. Khataei, G. Singh, and K. Bazargan, "Approximate hybrid binary-unary computing with applications in bert language model and image processing", in *Int. Symp. Field Programmable Gate Arrays (FPGA)*, 2023, pp. 165–175.

[230] H. Kim et al., "Temporal information dynamics in deep spiking neural networks", *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[231] A. Kokkinis et al., "Hardware-aware automated neural minimization for printed multilayer perceptrons", in *Proc. Design, Automation & Test in Europe (DATE)*, 2023, pp. 1–2.

[232] A. Mao et al., "Cross-entropy Loss Functions: Theoretical Analysis and Applications", in *ICML*, PMLR, 2023, pp. 23 803–23 828.

[233] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions", 2023.

[234] R. Nagarajan, S. Kumar, S. Banerjee, and D. Mukhopadhyay, "Scann: Side-channel analysis of spiking neural networks", *Chips*, vol. 3, no. 3, pp. 335–349, 2023. DOI: 10.3390/chips3030021.

[235] E. Shirzaei Sani et al., "A Stretchable Wireless Wearable Bioelectronic System for Multiplexed Monitoring and Combination Treatment of Infected Chronic Wounds", *Science Advances*, vol. 9, no. 12, p. 7388, 2023.

[236] F. Su, C. Liu, and H.-G. Stratigopoulos, "Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives", *IEEE Design & Test*, 2023.

[237] C. Tang and J. Han, "Hardware Efficient Weight-Binarized Spiking Neural Networks", in *2023,(DATE)*, IEEE, 2023, pp. 1–6.

[238] V. Tischler et al., "An integrate-and-fire neuron circuit made from printed organic field-effect transistors", *Organic Electronics*, vol. 113, p. 106 685, 2023.

[239] F. Torres, A. C. Basaran, and I. K. Schuller, "Thermal management in neuromorphic materials, devices, and networks", *Advanced Materials*, vol. 35, no. 37, p. 2 205 098, 2023. DOI: https://doi.org/10.1002/adma.202205098.

[240] H. Zhao, P. Pal, et al., "Towards temporal information processing – printed neuromorphic circuits with learnable filters", in *Proc. Int. Symposium on Nanoscale Architectures (NANOARCH)*, 2023.

[241] H. Zhao et al., "Highly-Bespoke Robust Printed Neuromorphic Circuits", in *Design, Automation and Test in Europe (DATE)*, IEEE, 2023.

[242] H. Zhao et al., "Highly-Dependable Printed Neuromorphic Circuits Based on Additive Manufacturing", *Flexible and Printed Electronics*, vol. 8, no. 2, p. 025 018, 2023.

[243] H. Zhao et al., "Power-Aware Training for Energy-Efficient Printed Neuromorphic Circuits", in *42nd IEEE/ACM International Conference on Computer-Aided Design*, 2023.

[244] G. Armeniakos, P. Duarte, P. Pal, et al., "On-sensor printed machine learning classification via bespoke adc and decision tree co-design", in *Proc. Design, Automation & Test in Europe (DATE)*, 2024.

[245] J. K. H. Franke, M. Hefenbrock, G. Koehler, and F. Hutter, *Improving deep learning optimization through constrained parameter regularization*, 2024. arXiv: 2311.09058 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2311.09058.

[246] G. Giustolisi et al., "A three-stage operational transconductance amplifier in tft flexible-substrate process", in *2024 39th DCIS*, 2024, pp. 1–6. DOI: 10.1109/DCIS62603.2024.10769146.

[247] A. B. Göğebakan, E. Magliano, A. Carpegna, A. Ruospo, A. Savino, and S. D. Carlo, "Spikingjet: Enhancing fault injection for fully and convolutional spiking neural networks", in *2024 IEEE 30th IOLTS*, 2024, pp. 1–7. DOI: 10.1109/IOLTS60994.2024.10616060.

[248] C. Iordanou et al., "Low-cost and efficient prediction hardware for tabular data using tiny classifier circuits", *Nature Electronics*, 2024, Add volume/issue/pages once finalized; this matches the "Tiny classifier" Nature Electronics paper.

[249] S. K. Kashyap, C. Raghavendra, S. Natarajan, and S. Ozev, "Structural built-in self test of analog circuits using on/off keying and delay monitors", in *IEEE VLSI Test Symposium (VTS)*, 2024, pp. 1–6. DOI: 10.1109/VTS60656.2024.10538672.

[250] A. Lebanov et al., "Flexible unipolar igzo transistor-based integrate and fire neurons for spiking neuromorphic applications", *IEEE TBioCAS*, vol. 18, no. 1, pp. 200–214, 2024. DOI: 10.1109/TBCAS.2023.3321506.

[251] M. Lopez et al., "A tunable multi-timescale indium-gallium-zinc-oxide thin-film transistor neuron towards hybrid solutions for spiking neuromorphic applications", *Communications Engineering*, vol. 3, Jul. 2024. DOI: 10.1038/s44172-024-00248-7.

[252] P. Pal et al., “Neural architecture search for highly bespoke robust printed neuromorphic circuits”, in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, 2024.

[253] P. Pal et al., “Analog printed spiking neuromorphic circuit”, in *IEEE DATE*, (Valencia, Spanien, Mar. 25–27, 2024), 2024, 6 S.

[254] P. Pal et al., “Fault sensitivity analysis of printed bespoke multilayer perceptron classifiers”, in *2024 IEEE European Test Symposium (ETS)*, IEEE, 2024, pp. 1–6.

[255] G. Saikiran and R. Srinivas, “Digital defect-oriented test methodology for flipped voltage follower low dropout regulators”, in *Advances in VLSI Testing*, Springer, 2024, pp. 55–70. DOI: 10.1007/978-981-99-7004-5_4.

[256] B. Sapui and M. B. Tahoori, “Power side-channel analysis and mitigation for neural network accelerators based on memristive crossbars”, in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE, 2024, pp. 612–617.

[257] A. Schmid, R. Gupta, and M. B. Tahoori, “Analoggym: A benchmark suite for analog and mixed-signal circuit validation”, *arXiv preprint arXiv:2409.08534*, 2024. [Online]. Available: https://arxiv.org/abs/2409.08534.

[258] Y. Wang et al., “Temperature-dependent electrical characteristics and reliability of a-IGZO thin-film transistors for flexible electronics”, *IEEE Transactions on Electron Devices*, 2024, Generic IGZO temperature-dependence placeholder; please update with the precise article you use.

[259] H. Zhao et al., “Neural evolutionary architecture search for compact printed analog neuromorphic circuits”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2024. DOI: 10.1109/TCAD.2024.3524357.

[260] H. Zhao et al., “Towards temporal information processing – printed neuromorphic circuits with learnable filters”, in *Proceedings of the 18th ACM International Symposium on Nanoscale Architectures*, ser. NANOARCH ’23, Dresden, Germany: Association for Computing Machinery, 2024, ISBN: 9798400703256. DOI: 10.1145/3611315.3633249. [Online]. Available: https://doi.org/10.1145/3611315.3633249.

[261] Y. Zhou et al., “Deep Neural Network Pruning with Progressive Regularizer”, in *2024 IEEE International Joint Conference on Neural Network (IJCNN 2024), Yokohama, 30th June - 05 July 2024*, (Yokohama, Japan), Institute of Electrical and Electronics Engineers (IEEE), 2024.

[262] T. Gheshlaghi, P. Pal, et al., “ADAPT-PNC: Mitigating device variability and sensor noise in printed neuromorphic circuits with adaptive learnable filters”, in *Proc. Design, Automation & Test in Europe (DATE)*, 2025.

[263] T. Gheshlaghi, P. Pal, et al., “Automatic test pattern generation for printed neuromorphic circuits”, in *Proc. IEEE European Test Symposium (ETS)*, 2025.

[264] T. Gheshlaghi, H. Zhao, P. Pal, et al., “Power-constrained printed neuromorphic hardware training”, in *Proc. Design Automation Conference (DAC)*, 2025.

[265] T. Huang et al., “A flexible low dropout regulator based on insno thin-film transistors with superior bending robustness and ultra-low quiescent current”, *IEEE Electron Device Letters*, vol. 46, no. 4, pp. 592–595, 2025. DOI: 10.1109/LED.2025.3539285.

[266] P. Pal et al., “Efficient analog error correction for printed unary-encoded computing”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[267] P. Pal et al., “PRINT-SAFE: Printed ultra-low-cost electronic x-design with scalable adaptive fault endurance”, *ACM Transactions on Embedded Computing Systems*, vol. 24, no. 5s, 2025.

[268] P. Pal et al., “Side-channel vulnerability analysis of flexible neuromorphic circuits”, in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, 2025.

[269] P. Pal et al., "Efficient analog error correction for printed unary-encoded computing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2025. DOI: 10.1109/TCAD.2025.3570162.

[270] P. Pal, A. Studt, T. Gheshlaghi, M. Hefenbrock, M. Beigl, and M. B. Tahoori, "Spikesynth: Energy-efficient adaptive analog printed spiking neural networks", in *44th ACM/IEEE International Conference on Computer Aided Design (ICCAD 2025)*, 2025.

[271] Pragmatic, *Flexic Platform Gen3*, https://www.pragmaticsemi.com/foundry/flexic-platform-gen-3, 2025.

[272] E. Services, *S9-e3_pragmatic flexics – part 3: Introducing pragmatic flexic platform gen 3*, https://www.youtube.com/watch?v=rAQlsL8fR00, Published April 14, 2025; accessed April 19, 2025, Apr. 2025.

[273] T. Gheshlaghi, A. Studt, P. Pal, et al., "Diagnostic test generation for fault localization in printed neuromorphic circuits", in *Proc. Design, Automation & Test in Europe (DATE)*, 2026.

[274] P. Pal et al., "Thermo-NAS: Thermal-resilient ultralow-cost igzo-based flexible neuromorphic circuits", in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2026.

[275] P. Pal et al., "ReFlex-LDO: Reliable Design and Delay-based Testing of Flexible Low Dropout Regulators", in *2026 VLSI Test Symposium*, IEEE, 2026.

[276] M. H. Behfar et al., "Failure mechanisms in flip-chip bonding on stretchable printed electronics", *Advanced Engineering Materials*, vol. 23, no. 12, p. 2 100 264, DOI: https://doi.org/10.1002/adem.202100264.

[277] K. N. Markelle Kelly Rachel Longjohn, "The uci machine learning repository", [Online]. Available: https://archive.ics.uci.edu.

[278] H. Zhao et al., "Printed Electrodermal Activity Sensor with Optimized Filter for Stress Detection", in *International Symposium on Wearable Computers (ISWC'22), Atlanta, GA and Cambridge, UK, September 11-15, 2022*.

Publication List

Priyanjana Pal
Karlsruhe Institute of Technology (KIT), Germany

Publications included in thesis:

1. B. Sapui*, P. Pal*, M. B. Tahoori, "FlexSpy: Side-Channel Spy Framework for Flexible Spiking Neuromorphic Hardware," [Accepted: IEEE ISVLSI 2026].
2. P. Pal, S.K. Kashyap, S. Ozev, M. B. Tahoori, "Reliable Emerging Electronics in Wearable and Implantable Healthcare Applications," 44th IEEE VLSI Test Symposium (VTS) 2026.
3. P. Pal, T. Gheshlaghi, S. Balaji, E. Ozer, M. B. Tahoori, "Thermo-NAS: Thermal-Resilient Ultralow-Cost IGZO-Based Flexible Neuromorphic Circuits," 31st IEEE Asian South Pacific Design Automation Conference (ASP-DAC) 2026.
4. P. Pal*, T. Gheshlaghi*, H. Zhao, M. Hefenbrock, M. Beigl, M. B. Tahoori, "PRINT-SAFE: Printed Ultra-Low-Cost Electronic X-Design with Scalable Adaptive Fault Endurance," ACM Transactions on Embedded Computing Systems (TECS), vol. 24, no. 5s, pp. 1–22, 2025.
5. P. Pal, B. Sapui, D. Weller, M. B. Tahoori, "Efficient Analog Error Correction for Printed Unary-Encoded Computing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2025.
6. T. Gheshlaghi, H. Zhao, P. Pal, M. Hefenbrock, M. Beigl, M. B. Tahoori, "Power-Constrained Printed Neuromorphic Hardware Training," 62nd ACM/IEEE Design Automation Conference (DAC), 2025, pp. 1–7.
7. P. Pal, A. Studt, T. Gheshlaghi, M. Hefenbrock, M. Beigl, M. B. Tahoori, "SpikeSynth: Energy-Efficient Adaptive Analog Printed Spiking Neural Networks," 44th ACM/IEEE International Conference on Computer-Aided Design (ICCAD), 2025.
8. P. Pal, B. Sapui, M. B. Tahoori, "Side-Channel Vulnerability Analysis of Flexible Neuromorphic Circuits," 44th ACM/IEEE International Conference on Computer-Aided Design (ICCAD), 2025.
9. T. Gheshlaghi*, P. Pal*, H. Zhao, M. Hefenbrock, M. Beigl, M. B. Tahoori, "Adapt-PNC: Mitigating Device Variability and Sensor Noise in Printed Neuromorphic Circuits with Adaptive Learnable Filters," Design, Automation & Test in Europe Conference (DATE), 2025.
10. H. Zhao*, P. Pal*, M. Hefenbrock, Y. Wang, M. Beigl, M. B. Tahoori, "Neural Evolutionary Architecture Search for Compact Printed Analog Neuromorphic Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2024.
11. P. Pal, H. Zhao, T. Gheshlaghi, M. Hefenbrock, M. Beigl, M. B. Tahoori, "Neural Architecture Search for Highly Bespoke Robust Printed Neuromorphic Circuits," 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2024.
12. P. Pal*, H. Zhao*, M. Shatta*, M. Hefenbrock, S. B. Mamaghani, S. Nassif, M. Beigl, M. B. Tahoori, "Analog Printed Spiking Neuromorphic Circuit," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2024.
13. G. Armeniakos, P. L. Duarte, P. Pal, G. Zervakis, M B. Tahoori, D. Soudris, "On-sensor Printed Machine Learning Classification via Bespoke ADC and Decision Tree Co-Design," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2024.
14. P. Pal, F. Afentaki, H. Zhao, G. Saglam, M. Hefenbrock, G. Zervakis, M. Beigl, M. B. Tahoori, "Fault Sensitivity Analysis of Printed Bespoke Multilayer Perceptron Classifiers," IEEE European Test Symposium, 2024.
15. H. Zhao*, P. Pal*, M. Hefenbrock, M. Beigl, M. B. Tahoori, "Towards Temporal Information Processing—Printed Neuromorphic Circuits with Learnable Filters," 18th ACM International Symposium on Nanoscale Architectures (NANOARCH), 2023.

16. H. Zhao*, P. Pal*, M. Hefenbrock, M. Beigl, M. B. Tahoori, "Power-Aware Training for Energy-Efficient Printed Neuromorphic Circuits," 42nd ACM/IEEE International Conference on Computer-Aided Design (ICCAD), 2023.

* Authors contributed equally to this work.

Publications not included in thesis:

1. M. Shatta, T. Gheshlaghi, P. Pal, G. Panagopoulos, G. Zervakis, M. B. Tahoori, "RBF-Driven Analog Neural Networks for Flexible Wearable Near-Sensor at the Extreme Edge," [DAC 2026].
2. S. Schupp, T. Gheshlaghi, P. Pal, M. B. Tahoori, "FAT-SNN: Fault-Aware Training of Flexible Analog Spiking Neural Networks Using Robust Surrogates," [Accepted ETS 2026].
3. B. Sapui, P. Pal, M. B. Tahoori, "When Faults Don't Vanish: Persistent Fault Injection and Key Recovery on MRAM-Backed AES," [Accepted Design, Automation & Test in Europe Conference & Exhibition (DATE), 2026].
4. T. Gheshlaghi, A. Studt, P. Pal, M. Hefenbrock, M. Beigl, M. B. Tahoori, "Diagnostic Test Generation for Fault Localization in Printed Neuromorphic Circuits," [Accepted Design, Automation & Test in Europe Conference & Exhibition (DATE), 2026].
5. M. B. Tahoori, E. Ozer, G. Zervakis, K. Balaskas, P. Pal, "Computing with Printed and Flexible Electronics," IEEE European Test Symposium, 2025.
6. T. Gheshlaghi, P. Pal, A. Studt, M. Hefenbrock, M. Beigl, M. B. Tahoori, "Automatic Test Pattern Generation for Printed Neuromorphic Circuits," IEEE European Test Symposium, 2025.
7. S. B. Mamaghani, P. Pal, M. B. Tahoori, "A Dynamic Testing Scheme for Resistive-Based Computation-In-Memory Architectures," 29th Asia and South Pacific Design Automation Conference (ASP-DAC), 2024.