Algorithm-Driven Design and Optimization of Printed Analog Neuromorphic Circuits

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

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

genehmigte

Dissertation

von

Haibin Zhao, M.Sc.
aus Shandong, China

Tag der mündlichen Prüfung: 23.10.2024

1. Referent: Prof. Dr. Michael Beigl
2. Referent: Prof. Dr. Mehdi Baradaran Tahoori

Acknowledgements

I would like to appreciate my primary advisor, Prof. Dr. Michael Beigl, for not only offering me the PhD position, the valuable feedback on my research, but also indispensable helps during these years. I extend my heartfelt thanks to Prof. Dr. Mehdi B. Tahoori, my co-advisor from the chair of dependable nano computing (CDNC). His comprehensive guidance and invaluable advice have profoundly impacted my academic growth. Meanwhile, I would also thank Dr. Till Riedel, who has enriched my understanding of computer science.

Furthermore, I give my special thanks to Dr. Tobias Röddiger for his scientific and cultural assist during my PhD time. I am deeply grateful to Dr. Michael Hefenbrock for his meticulous discussions on the details and methodologies described in this dissertation. His attention on the details is crucial for ensuring the quality of this dissertation. In addition, I appreciated the meaningful and in-depth talk with him on optimization and machine learning theory.

I enjoy the insightful discussions with Yexu Zhou and Yiran Huang, which offer my tremendous inspiration in AutoML and EdgeAI. Meanwhile, I appreciate the collaboration with Priyanjana Pal and Brojogopal Sapui from CDNC in the design and simulation of printed neuromorphic circuits, which build a solid foundation for this dissertation.

I want to express my thanks to all the colleagues and students I have worked with. Their kindness and motivation have been a constant source of encouragement, pushing me to exceed my own expectations. My gratitude also extends to Melissa Alpman and Zinoula Tsiouma for their supportive assistance with all the administrative matters.

The completion of this dissertation is not only built upon the experience during my PhD time, but also contributed by the persistent endeavor that has made thus far. I am therefore greatly thankful to my parents and family for teaching me the values of kindness, humility, and esteem. Their unwavering

support has been the cornerstone of my life.

I express my gratitude to my friends who have made me a significant impression and growth, Yunxiang Zhang, Wendi Huang, Guantao Dong, Mengfan Wu, and Junrui Chen. Thanks for the companionship and positive influence in my life.

My deepest gratitude goes to my girlfriend, Si Ni, for her consistent, unconditional support and love, whose presence has been a source of strength and happiness.

Lastly, I would thank myself for not giving up and persevering to this point.

Stay healthy and keep learning!

Abstract

Printed electronics is a novel technology to fabricate electronic devices based on additive manufacturing. Comparing to traditional photolithography-based silicon technologies, printed electronics is not intended to surpass silicon-based electronics in terms of computational power or integration density in very-large-scale integrated circuit. Instead, it aims to complement silicon electronics in edge scenarios, such as smart packaging in fast-moving consumer goods or smart bandages in advanced medical applications. In these domains, the requirement on computational intensity and complexity are typically moderate, however, there is a critical demand on mechanical flexibility, non-toxicity, bio-degradability, high customizability, and ultra-low fabrication cost. These features can hardly be matched by silicon-based electronics due to the subtractive manufacturing process. In contrast, printed electronics can provide these unique features because of its additive manufacturing nature and abundant functional materials, and thus, becomes a prominent facilitator of those next-generation electronics.

In the realm of printed devices, printed analog neuromorphic circuits has drawn increasing interest. These circuits not only inherit the benefits of printed electronics but also leverage the advancement of neuromorphic computing. Neuromorphic computing refers to brain-inspired computing paradigms, that has been proven to have powerful and bespoke computational functionalities through a series of elemental operations, namely weighted-sums and nonlinear activations. Therefore, printed analog neuromorphic circuits only consist of the interconnection of a series of streamlined circuit primitives, making their design and optimization highly accessible. Additionally, the analog approach allows processing signals directly in the analog domain, evading complicated devices for analog-digital conversion and thus facilitating the compactness and lightweight of the circuits. All these unique features enable printed analog

neuromorphic circuits with a broad and promising outlook.

However, existing studies on printed analog neuromorphic circuits stay primarily in the conceptual stage. They have outlined the principle of printed circuits to emulate neuromorphic computing, yet several practical factors have not been included. This dissertation explores a series of practical issues for printed analog neuromorphic circuits: (i) Regarding the **modeling and training framework**, this work summarizes two effective modeling approaches that allow precise modeling of electronic systems, namely physics-informed modeling and approximation-based modeling. For training, an existing machine learning-based training approach is improved through heuristics to include non-differentiable physical and technical constraints into the training process. Moreover, an evolutionary training approach is proposed to enable the optimization of the circuit architecture alongside its parameters. (ii) For improving the **circuit reliability**, the impact of device aging is examined and a targeted aging-aware training strategy is proposed to improve the circuit robustness against aging. Similarly, this thesis models and analyzes the collaborative influence of three primary factors affecting circuit reliability. Furthermore, circuit architecture search is employed to enable even higher circuit robustness against variations. Lastly, the impact of catastrophic faults in the printed neuromorphic circuit is also studied. (iii) In terms of **practicality**, by leveraging the advantage of additive manufacturing of printed electronics, a split manufacturing method is proposed to significantly reduce the fabrication costs of the printed neuromorphic circuits. Moreover, the power consumption of the printed neuromorphic circuits is enhanced through the proposed power-aware training, enabling the Pareto-optimal circuit performance within a prescribed power budget. Besides, improvement in circuit compactness is suggested through an area-aware training, which reduces the footprint of printed neuromorphic circuits and thus expands their application in area-scarce scenarios. (iv) Finally, the **computing paradigm** of existing printed neuromorphic circuits is extended by introducing circuit components with time dependencies such as printed capacitors. With these components, novel computing functionalities such as recurrent or spiking neural network can be implemented by printed electronics, adapting printed neuromorphic circuits to scenarios where

temporal data processing are envisioned.

In sum, this dissertation conducts a comprehensive investigation of printed analog neuromorphic circuits. It significantly accelerates the transition of these technologies from laboratory-based study to real-world deployments and therefore facilitates the electronification and intellecturalization within edge computing scenarios in the context of the Internet of Things.

Zusammenfassung

Gedruckte Elektronik erweist sich als eine innovative Technologie zur Herstellung elektronischer Geräte, die auf den Prinzipien der additiven Fertigung basiert. Im Gegensatz zu den traditionellen, auf Photolithographie basierenden Siliziumtechnologien, zielt die gedruckte Elektronik nicht darauf ab, die siliziumbasierte Elektronik in Bezug auf Rechenleistung oder Integrationsdichte in sehr großen integrierten Schaltkreisen zu übertreffen. Stattdessen ist es ihr Ziel, die Siliziumelektronik in speziellen Anwendungsfällen zu ergänzen, wie etwa bei Smartverpackung in schnelllebige Konsumgüter oder bei Smartverbänden in fortschrittliche medizinische Anwendungen. In diesen Bereichen sind die Anforderungen an die Rechenintensität und Komplexität typischerweise moderat, jedoch besteht eine kritische Nachfrage nach mechanischer Flexibilität, Nicht-Toxizität, biologischer Abbaubarkeit, hoher Anpassungsfähigkeit und extrem niedrigen Herstellungskosten. Diese Merkmale können von der siliziumbasierten Elektronik aufgrund des subtraktiven Fertigungsprozesses kaum erreicht werden. Im Gegensatz dazu wird die gedruckte Elektronik zu einem herausragenden Förderer dieser nächsten Generation von Elektronik, begünstigt durch ihre additive Fertigungsnatur und die Verfügbarkeit zahlreicher funktionaler Materialien.

Im Bereich der gedruckten Geräte hat die gedruckte analoge neuromorphe Schaltung zunehmend Interesse geweckt. Diese Schaltungen erben nicht nur die Vorteile der gedruckten Elektronik, sondern nutzen auch den Fortschritt des neuromorphen Computings. Neuromorphes Computing bezieht sich auf ein vom Gehirn inspiriertes Rechenparadigma, das durch eine Reihe von elementaren Operationen, nämlich gewichtete Summen und nichtlineare Aktivierungen, leistungsfähige und maßgeschneiderte Rechenfunktionen bewiesen hat. Daher bestehen gedruckte analoge neuromorphe Schaltungen nur aus der Verbindung mehreren einfachen Schaltkreisprimitiven, was ihr Design und ihre

Optimierung hochgradig zugänglich macht. Zusätzlich ermöglicht der analoge Ansatz die direkte Verarbeitung von Signalen im analogen Bereich, vermeidet komplizierte Geräte für die Analog-Digital-Umwandlung und fördert somit die Kompaktheit und Leichtigkeit der Schaltungen. All diese einzigartigen Merkmale verleihen gedruckten analogen neuromorphen Schaltungen einen breiten und vielversprechenden Ausblick.

Jedoch befinden sich bestehende Studien zu gedruckten analogen neuromorphen Schaltungen hauptsächlich in der konzeptionellen Phase. Diese Studien haben das Prinzip der gedruckten Schaltungen zur Nachahmung des neuromorphen Computings umrissen, doch wurden mehrere praktische Faktoren nicht einbezogen. Diese Dissertation erforscht eine Reihe praktischer Probleme für gedruckte analoge neuromorphe Schaltungen: (i) In Hinsicht auf **Modellierungs- und Trainingsrahmens** fasst diese Arbeit zwei effektive Modellierungsansätze zusammen, die eine präzise Modellierung elektronischer Systeme ermöglichen, nämlich physikbasierte Modellierung und approximationsbasierte Modellierung. Für Training wird ein bestehender, auf maschinellem Lernen basierender Trainingsansatz durch Heuristiken verbessert, um nichtdifferenzierbare physische und technische Einschränkungen in den Trainingsprozess einzubeziehen. Zudem wird ein evolutionärer Ansatz vorgeschlagen, der die Optimierung der Schaltungsarchitektur neben ihren Parametern erlaubt. (ii) Zur Verbesserung der **Zuverlässigkeit der Schaltung** wird die Auswirkung der Alterung von Bauteilen untersucht und eine gezielte altersbewusste Trainingsstrategie vorgeschlagen, um die Robustheit der Schaltung gegen Alterung zu erhöhen. Außerdem modelliert und analysiert diese Dissertation den gemeinsamen Einfluss von drei Hauptfaktoren, die die Zuverlässigkeit der Schaltung beeinflussen. Darüber hinaus wird eine Schaltungsarchitektursuche eingesetzt, um eine noch höhere Schaltungsrobustheit gegenüber Variationen anzubieten. Schließlich wird auch die Auswirkung katastrophaler Fehler in der gedruckten neuromorphen Schaltung untersucht. (iii) In Bezug auf die **Praktikabilität** wird durch Nutzung des Vorteils der additiven Fertigung gedruckter Elektronik eine geteilte Fertigungsmethode vorgeschlagen, um die Herstellungskosten der gedruckten neuromorphen Schaltungen erheblich zu senken. Zudem wird der Energieverbrauch der gedruckten neuromorphen Schaltun-

gen durch das vorgeschlagene energiebewusste Training optimiert, was eine Pareto-optimale Schaltungsleistung innerhalb eines vorgeschriebenen Energie-Budgets ermöglicht. Darüber hinaus wird eine Verbesserung der Schaltungskompaktheit durch ein flächenbewusstes Training vorgeschlagen, welches die Größe der gedruckten neuromorphen Schaltungen verringert und somit ihre Anwendung in flächenknappen Szenarien erweitert. (iv) Zuletzt fokusiert diese Arbeit auf das **Rechenparadigma** bestehender gedruckter neuromorpher Schaltungen. Durch die elektronische Komponenten mit Zeitabhängigkeiten, wie trainierbare Kondensatoren, können gedruckte Elektronik neuartige Rechenfunktionalitäten wie rekurrente neuronale Netzwerke und Spiking neuronale Netzwerke implementieren, wodurch gedruckte analoge neuromorphe Schaltungen an Szenarien angepasst werden, in denen die Verarbeitung zeitliche Daten vorgesehen ist.

Zusammenfassend führt diese Dissertation eine umfassende Untersuchung gedruckter analoger neuromorpher Schaltungen durch. Sie beschleunigt erheblich den Übergang dieser Technologien von laborbasierten Studien zu realen Anwendungen und erleichtert somit die Elektronifizierung und Intellektualisierung in Edge-Computing-Szenarien im Kontext des Internets der Dinge.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ADC analog-digital-converter. pp. 2, 21

ANN artificial neural network. pp. 15, 23, 39, 41, 42, 46, 87, 125, 126, 138, 157, 159, 160, 171, 188

CPU central processing unit. p. 15

CSPE composite solid polymer electrolyte. p. 27

DARTS differentiable architecture search. p. 141

DMSO dimethyl sulfoxide. p. 137

DNN deep neural network. pp. 141, 142

EA evolutionary algorithm. pp. 3–5, 28, 46, 48, 54, 57–59, 87, 91–94, 104, 136, 140, 141, 145, 146, 148–150, 185–187

EDA electronic design automation. p. 136

EGT electrolyte-gated transistor. pp. 27, 87

FET field effect transistor. p. 14

FPGA field programmable gate array. p. 20

I&F integrate-and-fire. pp. 15, 169, 171, 172

In_2O_3 indium oxide. p. 27

IoT Internet of Things. pp. 1, 14, 185

ITO indium tin oxide. pp. 27, 64

KKT Karush–Kuhn–Tucker. pp. 56, 135

LIF leaky-integrate-and-fire. pp. 169, 171, 178, 187

LSTM long-short-term-memory. p. 141

MC Monte-Carlo. pp. 40, 68, 80, 104, 164, 188

ML machine learning. pp. 3, 5, 15, 28, 42, 101, 142, 185

MLP multilayer perceptron. pp. 2, 4, 15–20, 23, 24, 37, 43, 142, 157–159, 167, 182, 187

MSE mean squared error. pp. 18, 175

NAS neural architecture search. pp. 87, 96, 141

NRE non-recurring engineering. pp. 59, 149

PCB printed circuit board. p. 137

PE printed electronics. pp. 1, 2, 5, 11–14, 20, 21, 37, 48, 59, 63, 69, 74, 75, 96, 97, 102, 111, 113, 117, 121, 122, 126, 135, 137, 142, 149, 150, 157, 160, 164, 169, 180–182, 185–187

PEDOT:PSS poly(3,4-ethylenedioxythiophene):polystyrene sulfonate. pp. 13, 27, 63, 64, 74, 186

PIM processing in memory. pp. 15, 23

pNC printed analog neuromorphic circuit. pp. 2–5, 7, 8, 11, 21, 23, 25, 26, 28, 37, 39, 42–44, 46, 48, 50, 54, 58, 59, 63–66, 68, 69, 72, 74, 75, 77, 79, 80, 82, 83, 85–87, 90–92, 94, 96, 97, 100–102, 104, 111–117, 119–122, 126, 129–132, 135–138, 140–143, 148, 150, 157, 158, 160, 163, 165, 167, 169, 171, 172, 174, 175, 177, 178, 180–182, 185–188

pPDK printed Process Design Kit. pp. 27, 40, 68, 123, 162, 175

pReLU printed ReLU-like. p. 20

pRNC printed recurrent neuromorphic circuit. pp. 4, 5, 38, 39, 157, 158, 160, 163–165, 167, 169, 182, 187

pSNC printed spiking neuromorphic circuit. pp. 5, 42, 157, 171, 172, 174, 175, 177, 178, 180–182, 187

ptanh printed tanh-like. pp. 20, 23–25, 37, 39, 50, 76, 79, 90, 94, 96, 98–101, 125–127, 130, 131, 160, 162, 173, 174

pTPB printed temporal processing block. pp. 158, 160, 162, 163, 165, 167, 169, 182

QMC Quasi Monte-Carlo. pp. 39, 40

ReLU rectified linear unit. pp. 17, 87–89

RNN recurrent neural network. pp. 4, 7, 157–160, 162, 163, 165, 167, 169, 182, 187

SNN spiking neural network. pp. 4, 7, 15, 157, 169, 171, 172, 174, 177, 178, 180–182, 187

SOTA state-of-the-art. pp. 13, 14, 69, 141, 142, 146, 174, 177

STE straight through estimator. p. 46

tanh hyperbolic tangent. pp. 17, 24, 25

VAE variational autoencoder. p. 138

Notation

Symbol	Description
$\boldsymbol{a}^{(l)}$	Vector of activations in the l-th layer in an MLP
$\boldsymbol{A}^{(l)}$	Activation matrix collecting all $\boldsymbol{a}^{(l)}$ in a layer
$(\cdot)^{\mathrm{A}}$	Variables for the activation circuit
$\mathcal{A}_{\boldsymbol{\omega}}(t)$	Aging behavior of printed resistors
A	Circuit area footprint of pNC
$\boldsymbol{b}^{(l)}$	Vector of biases in the l-th layer of an MLP
C	Capacitance
$c(\cdot)$	Constraint function
$\mathcal{C}$	Constraint value
$(\cdot)^{\mathrm{C}}$	Variables for the resistor crossbar
$\mathcal{D}$	Dataset containing input $\mathbf{X}$ and target output $\mathbf{Y}$
e	Magnitude of printing variation
$\mathbb{E}\{\cdot\}$	Mathematical expectation
$f(\cdot)$	Activation functions in an MLP
g	Conductance of a resistor
$\boldsymbol{g}$	Vector of conductances within a resistor crossbar
$\mathcal{L}(\cdot)$	Loss function
$(\cdot)^{\mathrm{N}}$	Variables for negation circuit

Continued on next page

Continued from previous page

$\mathcal{N}(\cdot)$	Gaussian distribution
$\mathcal{O}(\cdot)$	Objective function
$p(\cdot)$	Probabilistic distribution
$\mathcal{P}$	Power consumption of the circuit
$\boldsymbol{q}$	Physical quantities in nonlinear circuits
R	Resistance
t	Time
$\mathcal{U}[\cdot]$	Uniform distribution
V	Voltage
$\boldsymbol{V}$	Voltages within a batch
$\boldsymbol{W}^{(l)}$	Weight matrix in the l-th layer of an MLP
$\mathbf{X}$	Batch of input data from a given dataset
$\mathbf{Y}$	Batch of target output data from a given dataset
$\hat{\boldsymbol{Y}}$	Network output predicting its target $\mathbf{Y}$
$\boldsymbol{z}^{(l)}$	Vector of weighted-summed values in the l-th layer of an MLP
$\boldsymbol{Z}^{(l)}$	Matrix of weighted-summed values including all $\boldsymbol{z}^{(l)}$ in a batch
λ	Lagrangian multiplier
μ	Balance factor of the penalty term
$\varphi(\cdot)$	Property function that will be constrained
$\boldsymbol{\eta}$	Auxiliary parameters for nonlinear circuits
θ	Surrogate conductance of a resistor
$\boldsymbol{\theta}$	Vector of surrogate conductances within a resistor crossbar
$\boldsymbol{\omega}$	Fitting parameters of resistor aging

1 Introduction

As the Internet of Things (IoT) [11] continues evolving, the progression of informatization and electronification becomes ubiquitous in daily life, including the most edge scenarios. For instance, smart packaging [1] and smart labels [3] enable life-long quality monitoring of fast-moving consumer goods like meat [15] and dairy products [16]. In medical cares, smart bandages [12] and smart clothes [13] provide unobtrusive and continuous health monitoring. Additionally, smart household items, such as smart cups [2] and tableware [17], fosters the adoption of regular and healthy lifestyle habits. Within these emerging edge products, there is only a moderate acceptance for the expense of extra electronics, necessitating an extremely low-cost device production. Since these edge devices frequently serve as personal belongings, a highly flexible manufacturing process is required to support bespoke fabrication of personalized electronics. Also, some of the devices need to be featured with softness, stretchability, porosity, non-toxicity, and bio-compatibility for safety and comfort reasons. Moreover, given that some of them are envisioned to be disposable, bio-degradability is expected for environmental sustainability.

In this regard, traditional lithography-based silicon integrated circuits [6] face challenges in addressing these requirements. Either the high costs associated with their manufacturing infrastructure, or the complexity of their subtractive manufacturing processes, or the limitations imposed by their material choices, render silicon-based technology less ideal for the production of those emerging edge electronics. As an alternative, printed electronics (PE) [5] emerges as one of the most promising enabler of those next-generation electronics. Characterized by the additive manufacturing approach, PE produces circuits by depositing functional inks directly onto the substrates. Thus, PE allows for significant low fabrication costs compared to silicon-based subtractive technologies. In addition, PE supports a wide range of functional material

choices, including those, that are flexible [4], stretchable [10], porous [8], biocompatible [7] and bio-degradable [9]. All these unique advantages render PE a pivotal role in the development of the next-generation electronic devices. To equip printed devices with necessary computational functionalities, printed analog neuromorphic circuits (pNCs) emerge as a focal area of interest [14]. By connecting multiple simple-structured circuit primitives, i.e., resistor crossbars and inverter-based nonlinear circuits, pNCs are adept at emulating the neuromorphic computing paradigms, particularly, multilayer perceptrons (MLPs). This streamlined yet effective circuit schema not only enables high computational capabilities but also facilitates the design and optimization processes associated with these circuits.

Unfortunately, additive manufacturing also introduces several drawbacks to PE. Firstly, the high printing variation may perturb printed components from their designed values and thus reduce the circuit reliability. Moreover, PE has large feature sizes (in scope of μm) and therefore allows only low device counts. Although a significant number of transistors needed for analog-digital-converters (ADCs) can be saved by processing signals directly in the analog domain, it renders the pNCs more sensitive to printing variations. Secondly, many practical issues for pNCs have not been studied. For instance, circuits should exhibit extremely low manufacturing costs (in smart packaging), ultra-low power consumption (for disposable electronics), and a smaller footprint in area-limited applications, such as smart bandages. Additionally, as the components in existing pNCs possess no time-dependency, they are incapable of storing historical input information and processing temporal signals, which significantly narrows the application scope of the pNCs.

To address these issues, this work begins with modeling approaches, with which we can establish the corresponding optimization models for the aforementioned challenges. Based on these models, targeted training methods are then developed to effectively mitigate those problems. Experiment results have proven that this workflow can effectively address the problems and bring the deployment of pNCs in target scenarios one step closer to reality.

1.1 Objective and Contribution

The objective of this dissertation is to implement a comprehensive advancement to pNCs, spanning from the circuit design through to the practical issues. The contributions of this work are categorized into the following aspects.

Modeling and Training of Printed Neuromorphic Circuits

To enable the training of circuit parameters, i.e., component values, within a pNC for a specific target task, it is essential to establish a model that describes the behavior of the pNC. This work summarizes a **physics-informed modeling** approach and an **approximation-based modeling** approach. Both can accurately describe the behavior of the circuit primitives. Moreover, due to several technical limitations, constraints exist also in the modeling stage. For instance, the printing process usually imposes a certain printable range on each component, which must be considered into the modeling of the pNCs to ensure the practical manufacturability of the circuits. Consequently, the proposed modeling methods are also capable to take the constraints into account.

Additionally, this work enhances an existing **machine learning (ML)-based training** method by introducing heuristics to enable the training with tightly guaranteed non-differentiable technological constraints. Moreover, this work introduces a training strategy leveraging **evolutionary algorithms (EAs)**. Compared to the ML method, this approach allows training circuit parameters, circuit architecture, as well as other discrete decision variables simultaneously, which substantially expands the search space and thus enhances the potential capability of pNCs.

Reliability Design of Printed Neuromorphic Circuits

Compared to silicon-based electronics, additive manufacturing introduces a more significant variability in printed devices, which can notably affect the output of pNCs. Additionally, unlike silicon chips, printed components often lack adequate encapsulation, rendering them more susceptible to aging effects.

In response to these challenges, this work introduces **aging-aware training** to mitigate the impacts of device aging on circuit reliability. Afterwards, this

work investigates and addresses the collaborative effects of three primary factors (printing variation, aging effect, and sensing error) to encourage highly **dependable pNCs**. Moreover, this work proposes to utilize EA to enhance the circuit **architecture search** for further improving the circuit robustness. Lastly, this work studies the impact of **catastrophic faults** within circuit components, revealing the importance of circuit testing for pNCs.

Practicality Design of Printed Neuromorphic Circuits

To augment the practical applicability of pNCs, this dissertation focuses on three main aspects. Firstly, the **cost-effectiveness** of pNCs is enhanced. By leveraging the unique capabilities of additive manufacturing, a novel split manufacturing approach is proposed to combine the advantages of both high- and low-volume printing technologies. Furthermore, this work introduces a **power-aware training** framework that facilitates Pareto-optimal trade-offs between circuit performance and power consumption. This approach ensures the sustainability and power efficiency of pNCs in diverse applications. Lastly, considering area-limited applications, this work presents an **area-aware training** strategy. Utilizing an EA that allows topology optimization, a significant reduction in the footprint of pNCs can be achieved without performance loss.

Extension of Printed Neuromorphic Circuits

Existing pNCs predominantly follow the computing paradigm of MLP, which lacks the capability to process temporal information. Because it does not possess component with time-dependencies. To address this limitation, a printable hardware implementation of **recurrent neural networks (RNNs)** is proposed. By integrating learnable printed filters to existing pNCs and restructuring the circuit architecture, the circuit is capable to process temporal signals. Concurrently, a training framework is also proposed to enable the training of bespoke printed recurrent neuromorphic circuits (pRNCs) for target tasks.

As one of the most popular computing paradigms in neuromorphic computing, spiking neural networks (SNNs) is one of the most closely to the behavior of biological neurons. It stands out for their low power consumption and ro-

bustness against small perturbation. In this work, a printable spike-generator is proposed and integrated into the existing pNCs to emulate **printed spiking neuromorphic circuits (pSNCs)**. Analogous to pRNCs, we propose a Transformer-based model to enable the training of the pSNCs for specific scenarios in a bespoke manner.

1.2 Structure

The dissertation is structured into five main sections that provide a comprehensive research around the pNCs. This structure is depicted in Figure 1.1. It aims to facilitate an immediate grasp of the relationships among sections and their collective contribution to the improvement of pNCs.

- Chapter 2 provides the background for this dissertation, including PE, neuromorphic computing, and pNCs. In addition, this chapter introduces the preliminary of mathematics, optimization, and ML algorithms utilized in this work. These preliminaries serve to facilitate the bespoke circuit designs that are proposed in following chapters.

- Chapter 3 focuses on the modeling and training of pNCs that offers the capability to consider the physical and technological constraints. Built upon the modeling, this chapter introduces the corresponding circuit optimization strategies, leveraging ML and EAs. These modeling and training approaches are then utilized in the following chapters as fundamental methodologies to model the electronic problems and thus address them.

- Chapter 4 addresses the reliability of the circuits by analyzing a series of factors that could emerge during circuit manufacturing and operation. It proposes viable solutions to these challenges, ensuring the reliability and longevity of the circuits.

- Chapter 5 discusses the practicality issues, considering more practical factors in the real-world application. This includes the focus on ultra-low manufacturing cost, energy-efficient power consumption, and com-

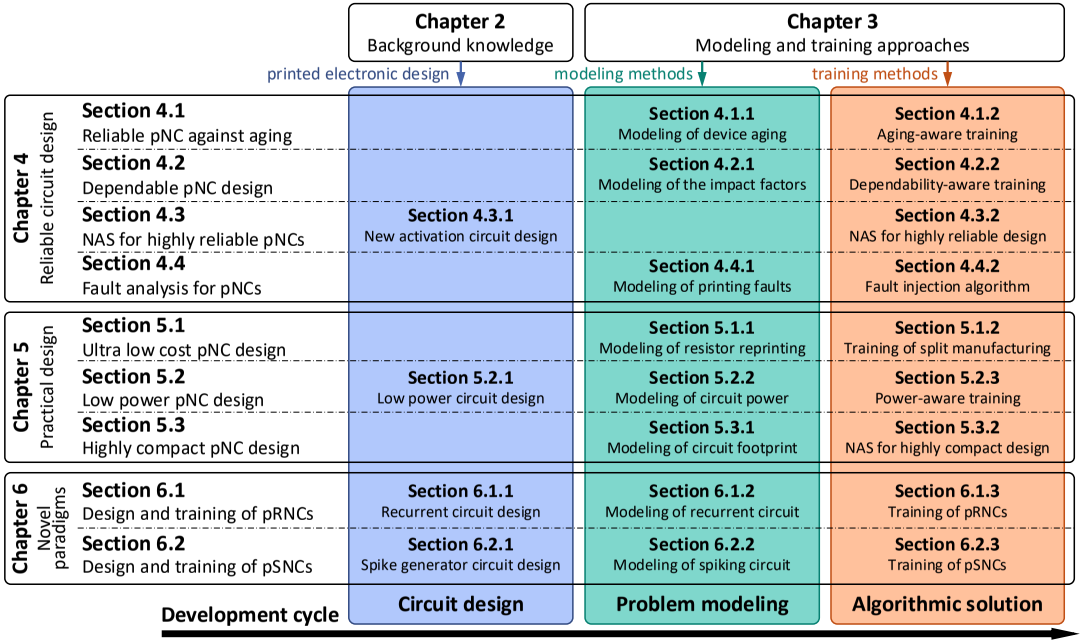


Figure 1.1: Structure of this dissertation for pNCs. Chapter 1 (Introduction) and Chapter 7 (Conclusion) are excluded, as they do not pertain scientific contribution. (pRNC: printed recurrent neuromorphic circuit, pSNC: printed spiking neuromorphic circuit, NAS: neural architecture search.)

pact circuit footprints, paving the way to move from the laboratory environment to practical deployment.

- Chapter 6 ventures into the potential extensions of the existing pNCs. By introducing electronic components with temporal dependencies, the computing paradigms of the existing pNCs can be extended to RNNs and SNNs. These extensions significantly broadens the application domains of the circuits.

- Chapter 7 concludes the work and outlines future directions of pNCs.

1.3 List of Publications

The following list gives a comprehensive overview of all scientific papers published by the author that are relevant for this dissertation. Significant parts of this dissertation (across all chapters) were partly copied from the relevant papers listed below and assembled into a coherent monograph structure.

Haibin Zhao, Brojogopal Sapui, Michael Hefenbrock, Zhidong Yang, Michael Beigl, and Mehdi B Tahoori. "Highly-Bespoke Robust Printed Neuromorphic Circuits". In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2023, pp. 1–6.

Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. "Aging-Aware Training for Printed Neuromorphic Circuits". In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–9.

Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. "Highly-dependable printed neuromorphic circuits based on additive manufacturing". In: *Flexible and Printed Electronics* 8.2 (2023), p. 025018.

Priyanjana Pal, **Haibin Zhao**, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. "Neural Architecture Search for Highly Robust Printed Neuromorphic Circuits". In: *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 2024, pp. 1–9.

Priyanjana Pal, Florentia Afentaki, **Haibin Zhao**, Gürol Saglam, Michael Hefenbrock, Georgios Zervakis, Michael Beigl, and Mehdi B Tahoori. "Fault Sensitivity Analysis of Printed Bespoke Multilayer Perceptron Classifiers". In: *2024 IEEE European Test*

Symposium (ETS). IEEE. 2024, pp. 1–6.

Haibin Zhao, Priyanjana Pal, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. "Power-Aware Training for Energy-Efficient Printed Neuromorphic Circuits". In: *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE. 2023, pp. 1–9.

Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. "Split Additive Manufacturing for Printed Neuromorphic Circuits". In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2023, pp. 1–6.

Haibin Zhao, Alexander Scholz, Michael Beigl, Si Ni, Surya Abhishek Singaraju, and Jasmin Aghassi-Hagmann. "Printed Electrodermal Activity Sensor with Optimized Filter for Stress Detection". In: *Proceedings of the 2022 ACM International Symposium on Wearable Computers*. 2022, pp. 112–114.

Haibin Zhao, Priyanjana Pal, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. "Towards Temporal Information Processing – Printed Neuromorphic Circuits with Learnable Filters". In: *Proceedings of the 18th ACM International Symposium on Nanoscale Architectures*. 2023, pp. 1–6.

Priyanjana Pal, **Haibin Zhao**, Maha Shatta, Michael Hefenbrock, Sina B Mamaghani, Sani Nassif, Michael Beigl, and Mehdi B Tahoori. "Analog Printed Spiking Neuromorphic Circuit". In: *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2024, pp. 1–6.

1.4 Statement of Reproducibility

The methodologies and experimental results reported in this dissertation are accessible and reproducible through the source code available in the associated GitHub repositories[1]. It is imperative to emphasize that, the reproducibility of the experimental results is conditional under the corresponding experiment setups described in the papers listed above. As this dissertation involves multifaceted enhancements to pNCs, which are not fully orthogonal to each other, the effectiveness of individual methodologies may yield diminishing returns when these methods are applied in conjunction.

[1]The repositories are available at `https://github.com/Neuromophic`.

Bibliography

[1] Arif U Alam, Pranali Rathi, Heba Beshai, Gursimran K Sarabha, and M Jamal Deen. “Fruit quality monitoring with smart packaging”. In: *Sensors* 21.4 (2021), p. 1509.

[2] Michael Beigl, Hans-W Gellersen, and Albrecht Schmidt. “Mediacups: experience with design and use of computer-augmented everyday artefacts”. In: *Computer Networks* 35.4 (2001), pp. 401–409.

[3] Leonardo Weiss Ferreira Chaves and Christian Decker. “A survey on organic smart labels for the Internet-of-Things”. In: *2010 Seventh International Conference on Networked Sensing Systems (INSS)*. IEEE. 2010, pp. 161–164.

[4] Sílvia Manuela Ferreira Cruz, Luís A Rocha, and Júlio C Viana. “Printing technologies on flexible substrates for printed electronics”. In: *Flexible electronics*. IntechOpen, 2018.

[5] Zheng Cui. *Printed electronics: materials, technologies and applications*. John Wiley & Sons, 2016.

[6] XiaoMing Hu. “Photolithography technology in electronic fabrication”. In: *2015 International Power, Electronics and Materials Engineering Conference*. Atlantis Press. 2015, pp. 843–850.

[7] Altynay Kaidarova, Mohammed Asadullah Khan, Marco Marengo, Liam Swanepoel, Alexander Przybysz, Cobus Muller, Andreas Fahlman, Ulrich Buttner, Nathan R Geraldi, Rory P Wilson, et al. “Wearable multifunctional printed graphene sensors”. In: *NPJ Flexible Electronics* 3.1 (2019), p. 15.

[8] Dong Jin Kang, Lola Gonzaléz-García, and Tobias Kraus. “Soft electronics by inkjet printing metal inks on porous substrates”. In: *Flexible and Printed Electronics* 7.3 (2022), p. 033001.

[9] Jiameng Li, Jiayin Liu, Wenxing Huo, Jingxian Yu, Xinyu Liu, Michael J Haslinger, Michael Muehlberger, Pavel Kulha, and Xian Huang. “Micro and nano materials and processing techniques for printed biodegradable electronics”. In: *Materials Today Nano* 18 (2022), p. 100201.

[10] Mohammed G Mohammed and Rebecca Kramer. “All-printed flexible and stretchable electronics”. In: *Advanced Materials* 29.19 (2017), p. 1604965.

[11] Karen Rose, Scott Eldridge, and Lyman Chapin. “The internet of things: An overview”. In: *The internet society (ISOC)* 80 (2015), pp. 1–50.

[12] Ehsan Shirzaei Sani, Changhao Xu, Canran Wang, Yu Song, Jihong Min, Jiaobing Tu, Samuel A Solomon, Jiahong Li, Jaminelli L Banks, David G Armstrong, et al. "A stretchable wireless wearable bioelectronic system for multiplexed monitoring and combination treatment of infected chronic wounds". In: *Science Advances* 9.12 (2023), eadf7388.

[13] Yu-Han Wang, Chyan-Goei Chung, Chung-Chih Lin, and Chih-Ming Lin. "The study of the electrocardiography monitoring for the elderly based on smart clothes". In: *2018 Eighth International Conference on Information Science and Technology (ICIST)*. IEEE. 2018, pp. 478–482.

[14] Dennis D Weller, Michael Hefenbrock, Michael Beigl, Jasmin Aghassi-Hagmann, and Mehdi B Tahoori. "Realization and training of an inverter-based printed neuromorphic computing system". In: *Scientific reports* 11.1 (2021), p. 9554.

[15] Zhilong Yu, Dongyun Jung, Soyoun Park, Yaxi Hu, Kang Huang, Barbara A Rasco, Shuo Wang, Jennifer Ronholm, Xiaonan Lu, and Juhong Chen. "Smart traceability for food safety". In: *Critical Reviews in Food Science and Nutrition* 62.4 (2022), pp. 905–916.

[16] Chao Zhang, An-Xiang Yin, Ruibin Jiang, Jie Rong, Lu Dong, Tian Zhao, Ling-Dong Sun, Jianfang Wang, Xing Chen, and Chun-Hua Yan. "Time–Temperature indicator for perishable products based on kinetically programmable Ag overgrowth on Au nanorods". In: *ACS nano* 7.5 (2013), pp. 4561–4568.

[17] Liyang Zhang, Kohei Kaiya, Hiroyuki Suzuki, and Akio Koyama. "Meal information recognition based on smart tableware using multiple instance learning". In: *Advances in Networked-based Information Systems: The 22nd International Conference on Network-Based Information Systems (NBiS-2019)*. Springer. 2020, pp. 189–199.

2 Background

To provide a comprehensive context for this dissertation, this chapter briefly introduces the preliminary knowledge of printed electronics (PE) regarding printing technologies and materials. Subsequently, the concept of neuromorphic computing, which refers to a computing paradigm inspired by biological brains, is described. Finally, the primitives of printed analog neuromorphic circuits (pNCs) are presented, along with an overview of the technology specifications of the pNCs employed in this work.

2.1 Printed Electronics

Printed electronics (PE) is an emerging technology that manufactures electronics in an additive way [13]. Analogous to color printing, PE enables direct deposition of functional inks onto substrates to fabricate electronic products, as illustrated by the lower part in Figure 2.1. Evidently, this additive strategy significantly simplifies the complicated manufacturing process and reduces the demand for expensive infrastructures in traditional photolithography-based subtractive processes, as shown in the upper part in Figure 2.1.

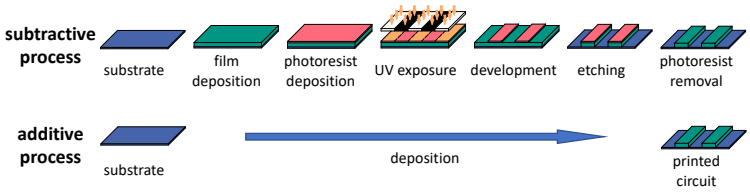


Figure 2.1: Comparison of photolithography-based subtractive process and additive printed manufacturing. Sourced from [80].

2.1.1 Printing Technologies

PE encompasses several processes, which can be broadly categorized into high- and low-volume approaches (see Figure 2.2). High-volume manufacturing approaches, e.g., screen printing [68] and gravure printing [67], typically necessitate supplementary masks, which may increase the fabrication complexity and cost. Nevertheless, once the masks have been produced, they can facilitate the efficient mass replication of electronics. In contrast, low-volume methods, such as inkjet printing [64] and aerosol jet printing [19], eliminate the requirement for masks. They print target electronics through precise control over the trajectory of nozzle movement. Although these techniques do not match the scalability of mask-based methods and exhibit slower production speed, they provide significant technical flexibility and support extremely low manufacturing cost for bespoke electronic fabrication. Some technique specifications of typical printing technologies are summarizes in Table 2.1.

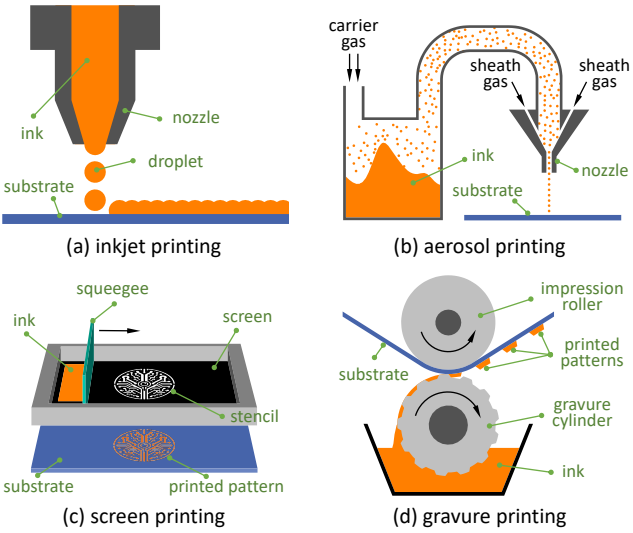


Figure 2.2: Exemplary printing technologies in PE for high and low throughput: (a) inkjet printing, (b) aerosol printing, (c) screen printing, and (d) gravure printing.

With the diversity of printing technologies, PE can be effectively deployed across a wide range of applications. Specifically, low-volume manufacturing techniques can not only facilitate the rapid prototyping thus accelerate product development, but also enable the personalization of electronic products, which may even allow individuals to "print their own functional electronic devices anywhere" [9]. On the other hand, high-volume production methods can significantly reduce the manufacturing time and cost per circuit, promoting the electronification and intellectualization of commodities.

Table 2.1: Comparison of technology specifications of typical printing technologies. Sourced from [32, 40, 49].

Parameter	**Inkjet**	**Aerosol**	**Screen**	**Gravure**
Resolution (μm)	$15-100$	$10-100$	$30-100$	$50-200$
Speed (m/min)	$0.02-5$	$0.03-12$	$0.6-100$	$8-100$
Print Size	Large	Large	Medium	Large
Contact mode	Contactless	Contactless	Contact	Contact
Mask requirement	No	No	Yes	Yes

2.1.2 Functional Inks

The existence of conductive, semiconducting or dielectric inks is the key enabler that printing technologies can be used for producing electronic products. The ink materials can be broadly classified into two categories, namely organic and inorganic materials. Compared to inorganic inks, organic materials are distinguished by their versatile molecular structures, lower fabrication costs, and compatibility with flexible polymer substrates [13]. These features foster their wide applications as conductive materials, such as poly(3,4-ethylenedioxythiophene):polystyrene sulfonate (PEDOT:PSS) [15, 66, 71].

However, the state-of-the-art (SOTA) carrier mobilities of organic materials are 3 to 5 orders of magnitude lower than their inorganic counterparts, for instance, $10^1\,\mathrm{cm\,V^{-1}\,s^{-1}}$ for organic inks [36, 78] versus $10^5\,\mathrm{cm\,V^{-1}\,s^{-1}}$ for inorganic inks [7]. This disadvantage forces organic semiconductors operating at high voltages ($\geq 25\mathrm{V}$), rendering them less suitable for field effect transistors

(FETs) applied in low-energy scenarios in IoT context. In contrast, inorganic semiconductors composed by oxides are capable to provide substantially low operating voltages ($\leq$ 1V), which makes them well-suited for devices powered by on low-capacity batteries [12, 42] or energy harvesters [41]. Therefore, inorganic semiconductors become more favored in the target domain of PE.

Apart from carrier mobility, several additional outstanding properties of printing materials are also critical factors in material selection. These include stretchability [31], transparency [2], non-toxicity [50], bio-compatibility [30, 39], and bio-degradability [37, 54], which can be effectively leveraged to accommodate requirements in diverse target applications of PE.

2.1.3 Substrates

As the foundational and the largest component of electronic devices, the properties of substrates greatly impact the characteristics of the entire device. Fortunately, PE offers the adaptability to be applied onto a diverse array of substrates including glass [25], papers [28], plastics [33], textiles [6], and metallic foils [56], among others. This versatility enables PE to possess unique features such as porosity [28], stretchability [6], mechanical flexibility [6, 28, 33, 56], comfort [6], and transparency [33, 25]. Additionally, the SOTA innovations extend PE to directly printing onto unconventional surfaces such as fruits [8] or even human skins [63], which significantly broadening the utility of PE.

2.1.4 Discussion

The distinctive advantages facilitate PE for a wide range of emerging applications where traditional silicon-based solution may be either too expensive or unable to meet specific requirements. Such applications include smart packaging [1] and smart labels [10] that should be cheap and disposable, or soft sensors [11, 44] and soft robotics [38] which requires mechanical flexibility. However, it must be emphasized that, despite its incomparable benefit, PE does not cause conflict with traditional silicon chips. Because the aim of PE is not to compete with chips in computationally intensive scenarios, but rather to complement silicon chips in edge scenarios by leveraging their own properties.

2.2 Neuromorphic Computing

The concept of neuromorphic computer has been envisioned since the era of Alan Turing [72] and John von Neumann [74], referring to the hardware implementation of neurally inspired computing paradigms [60]. Distinct from the classic von Neumann architecture [29] with segregated central processing unit (CPU) and memory unit, neuromorphic computing systems possess collocated memory and processing, namely processing in memory (PIM) [4]. This collocation not only addresses the bottleneck of limited bandwidth between the CPU and memory [48], but also facilitates highly parallel processing and energy-efficiency [60] of the computing system. Consequently, these systems have shown superior performance over von Neumann architectures in various domains, including neuroscience [59] and machine learning (ML) [52].

2.2.1 Computing Models

There are several neuromorphic computing models. According to their biological plausibility, they can be (decreasingly) ranked as the Hodgkin Huxley model [23], Fitzhugh Nagumo model [17], membrane dynamics model [3], integrate-and-fire (I&F) model [18], and the McCulloch Pitts model [46]. The last two models are also widely employed in the research of ML, specifically, the I&F model forms the foundation of SNNs, whereas the McCulloch Pitts model serves as the basic of feed-forward MLPs in artificial neural networks (ANNs) [55].

Among the aforementioned models of neuromorphic computing, the computational scheme of feed-forward MLP has been most explored [60]. This is due to the inherent commonalities between neuromorphic computing and ANNs, coupled with the streamlined operations and exceptional computing efficiency of ANNs, especially MLPs [55]. Concurrently, neuromorphic devices become also ideal platforms for executing ANNs, effectively acting as hardware ANN accelerators [27]. Therefore, the neuromorphic systems that emulate MLPs are also referred to as *hardware MLPs* [51]. The following part describes the preliminary of MLPs and their adaption to neuromorphic hardware.

2.2.2 Multilayer Perceptrons

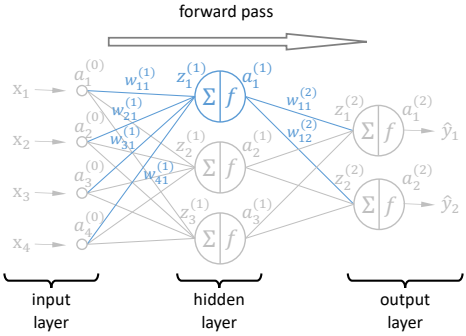


Figure 2.3: An example of the forward pass in a 4-3-2 multilayer perceptron, receiving four input data $\mathbf{x}_1, \cdots, \mathbf{x}_4$ and producing two output data $\hat{\boldsymbol{y}}_1$ and $\hat{\boldsymbol{y}}_2$. The circles refer to the neuron that performs weighted-sum operations followed by nonlinear activations, whereas the edges indicate the weights $w_{i,j}^{(l)}$ to their corresponding inputs. The blue color highlights the forward pass of one single neuron.

A modern MLP typically comprises an input layer, multiple hidden layers, and an output layer. Each hidden and output layer contains multiple neurons. These neurons process data from the preceding layer by applying a weighted-sum operation and nonlinear activation. Afterwards, the data will be forwarded to the subsequent layer as input data. Specifically, the behavior of the l-th layer can be described by

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

$$\boldsymbol{a}^{(l)} = f(\boldsymbol{z}^{(l)}), \tag{2.2}$$

where $\boldsymbol{a}^{(l-1)} = [a_1^{(l-1)}, \cdots, a_{N_{l-1}}^{(l-1)}] \in \mathbb{R}^{1 \times N_{l-1}}$ summarizes output from the preceding layer containing N_{l-1} neurons, $\boldsymbol{W}^{(l)} \in \mathbb{R}^{N_{l-1} \times N_l}$ refers to the learnable weight matrix that maps $\boldsymbol{a}^{(l-1)}$ into N_l values in the l-th layer, and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{1 \times N_l}$ indicates the learnable bias term added to each neuron in the l-th layer, finally, $\boldsymbol{z}^{(l)} \in \mathbb{R}^{1 \times N_l}$ describes the result after weighted-sum operation. Sub-

sequently, $\boldsymbol{z}^{(l)}$ is activated by a nonlinear function $f(\cdot)$, e.g., rectified linear unit (ReLU), sigmoid, or hyperbolic tangent (tanh) function [62], and yields $\boldsymbol{a}^{(l)} \in \mathbb{R}^{1\times N_l}$. Figure 2.3 exemplifies a 3-layer MLP.

To simplify Equation (2.1), the learnable parameters $\boldsymbol{b}^{(l)}$ can be fused into the weight matrix $\boldsymbol{W}^{(l)}$ by

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

In this way, the learnable parameters in a layer can be simplified in one matrix $\tilde{\boldsymbol{W}}^{(l)} \in \mathbb{R}^{(N_{l-1}+1)\times N_l}$ while padding the input vector $\boldsymbol{a}^{(l-1)}$ by a 1. Here, the overscript $\tilde{(\cdot)}$ denotes the extended variables for this simplification. Moreover, to facilitate batch data processing, i.e., processing several data simultaneously, multiple input data $\tilde{\boldsymbol{a}}_b^{(l-1)}$ can be assembled into a matrix, i.e.,

$$\tilde{\boldsymbol{A}}^{(l-1)} = \begin{bmatrix}\tilde{\boldsymbol{a}}_1^{(l-1)} \\ \vdots \\ \tilde{\boldsymbol{a}}_{\mathrm{B}}^{(l-1)}\end{bmatrix} \in \mathbb{R}^{\mathrm{B}\times(N_{l-1}+1)}.$$

Here, B denotes the batch size of the data, and $\tilde{\boldsymbol{a}}_b^{(l-1)}$ with $b \in \{1,\cdots,\mathrm{B}\}$ refers to the b-th data within the batch. Correspondingly, the weighted-summed value $\boldsymbol{z}^{(l)}$ is also extended to $\boldsymbol{Z}^{(l)} \in \mathbb{R}^{\mathrm{B}\times N_l}$. In sum, the mathematical behavior of an MLP is given by

$$\hat{\boldsymbol{Y}}_{\mathbf{X}}(\tilde{\boldsymbol{W}}) = f(\cdots f(f(\tilde{\mathbf{X}} \cdot \tilde{\boldsymbol{W}}^{(1)}) \cdot \tilde{\boldsymbol{W}}^{(2)}) \cdots \tilde{\boldsymbol{W}}^{(L)}),$$

where $\mathbf{X}$ indicates the input to the first layer of the MLP, which can be explained as $\boldsymbol{A}^{(0)}$, whereas $\hat{\boldsymbol{Y}}$ is the final output from the MLP, which is equivalent to $\boldsymbol{A}^{(L)}$ with L being the total number of layers in the MLP. Note that, both $\mathbf{X}$ and $\mathbf{Y}$ are constant matrices provided by the target dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, which is usually measured from real world scenarios. Rather, $\tilde{\boldsymbol{W}} = \tilde{\boldsymbol{W}}^{(1)} \cup \cdots \cup \tilde{\boldsymbol{W}}^{(L)}$ is the summary of all optimization variables that influence the network output $\hat{\boldsymbol{Y}}$.

Training of MLPs. In the early stage of neuromorphic computing, such as in McCulloch Pitts model, the parameters $\tilde{\boldsymbol{W}}$ are not learnable but predetermined [46] or randomized [58]. Benefit from backpropagation [21], a gradient-based approach, the network parameters are nowadays allowed to be trained efficiently. Here, training an MLP refers to optimizing the learnable parameters $\tilde{\boldsymbol{W}}$ in a way that the difference between network output $\hat{\boldsymbol{Y}}$ (with given input $\mathbf{X}$) and the target output value $\mathbf{Y}$ in the dataset is minimized.

For this purpose, an objective function, i.e., a loss function, is required to guide the update of the parameters. In regression tasks, the loss function can be simply formulated as the mean squared error (MSE) between $\hat{\boldsymbol{Y}}$ and $\mathbf{Y}$, namely,

$$\mathcal{L}(\tilde{\boldsymbol{W}}) = \frac{1}{\mathrm{B} \cdot N_L} \|\hat{\boldsymbol{Y}}_{\mathbf{X}}(\tilde{\boldsymbol{W}}) - \mathbf{Y}\|_{\mathrm{F}}^2,$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm [24], denoting the square root of the sum of the square of each element in the matrix.

However, in classification tasks, the loss function is not intuitive, because the ultimate objective is to increase classification accuracy, which is a discrete criterion (either correct or incorrect). This discrete function cannot provide useful gradient information to guide the gradient-based training process. To address this issue, a strictly convex function, i.e., the cross-entropy loss [43], is widely employed to update the learnable parameters. The cross-entropy loss is defined as

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

where $\mathbf{Y}^{\mathrm{OH}} \in \mathbb{R}^{\mathrm{B} \times \mathrm{C}}$ is the one-hot encoding [61] of the target $\mathbf{Y} \in \mathbb{R}^{B}$ from the dataset, and $\mathbf{1}$ denotes a matrix having the same dimension as $\mathbf{Y}^{\mathrm{OH}}$ with all the elements being 1. $\odot$ refers to the elementwise product. Notably, the number of neurons N_L in the final output layer of the MLP is identical to that of the number of classes C. In this case, the index of the neuron with the highest output will be regarded as the classification result. Therefore, the cross-entropy loss aims to suppress the outputs relating to the wrong classes, while the correct output will be encouraged to produce a higher value.

Subsequently, training an MLP can be expressed as minimizing the loss

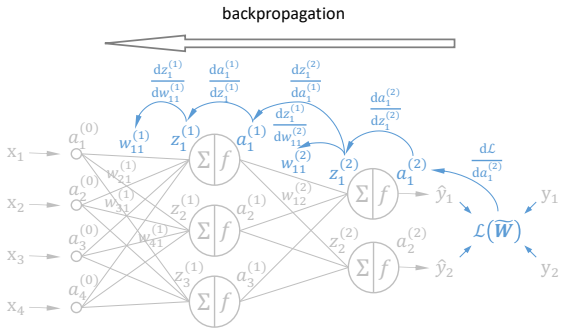


Figure 2.4: Example of the backpropagation in a multilayer perceptron, propagating from the loss $\mathcal{L}$ to several weights ($w_{11}^{(1)}$ and $w_{11}^{(2)}$). The blue color highlights one of the backpropagation chains.

function $\mathcal{L}(\tilde{\boldsymbol{W}})$ through the change of $\tilde{\boldsymbol{W}}$, namely,

$$\underset{\tilde{\boldsymbol{W}}}{\text{minimize}}\ \mathcal{L}(\tilde{\boldsymbol{W}}),$$

which is an iterative process. In each iteration, the learnable parameters are updated through gradient descent, i.e.,

$$\tilde{\boldsymbol{W}} \leftarrow \tilde{\boldsymbol{W}} - \alpha \cdot \nabla_{\tilde{\boldsymbol{W}}} \mathcal{L}(\tilde{\boldsymbol{W}}),$$

where $\alpha \in \mathbb{R}^+$ denotes the step size of the update, which is also referred to as the learning rate, and $\nabla_{\tilde{\boldsymbol{W}}} \mathcal{L}(\tilde{\boldsymbol{W}})$ denotes the derivative (gradient) of $\mathcal{L}(\tilde{\boldsymbol{W}})$ with respect to $\tilde{\boldsymbol{W}}$. Notably, owing to the structured connectivity of MLPs, the gradient of each parameter can be effectively obtained through backpropagation leveraging the chain rule, i.e.,

$$\nabla_{\tilde{\boldsymbol{W}}^{(l)}} \mathcal{L}(\tilde{\boldsymbol{W}}) = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{A}^{(L)}} \cdot \frac{\mathrm{d}\boldsymbol{A}^{(L)}}{\mathrm{d}\boldsymbol{Z}^{(L)}} \cdot \frac{\mathrm{d}\boldsymbol{Z}^{(L)}}{\mathrm{d}\boldsymbol{A}^{(L-1)}} \cdots \frac{\mathrm{d}\boldsymbol{A}^{(l+1)}}{\mathrm{d}\boldsymbol{Z}^{(l)}} \cdot \frac{\mathrm{d}\boldsymbol{Z}^{(l)}}{\mathrm{d}\tilde{\boldsymbol{W}}^{(l)}}.$$

With this approach, the complicated gradient calculation can be split into multiple simple subproblems, as shown in Figure 2.4.

Several frameworks, such as PyTorch [53] and TensorFlow [45], have al-

ready been developed to automatically perform the backpropagation and calculate the gradient of the learnable parameters. Moreover, these tool also offer many variations of gradient descent techniques, such as RMSprop [69] or Adam [34]. These techniques are proposed to adaptively modify the direction or step size of parameter updates, which may mitigate some challenges of purely gradient descent method during training, e.g., trapped in local minimum.

Adaption of MLPs to neuromorphic hardware. Three primary methodologies exist for adapting MLPs on neuromorphic hardware. The first approach refers to *hardware synthesis* that transforms existing MLPs into low-level circuit description [5]. This approach is predominantly utilized to development specialized neuromorphic hardware for specific applications. The second technique involves *mapping* trained MLPs onto given neuromorphic architectures [14]. This type of tools generally has to consider the inherent limitations of the employed architecture and the target hardware. Therefore, the MLPs usually need to be modified to adapt the limitations. Lastly, *programming tools* can offer the capability for more flexible programming and thus allow users modifying computing algorithms manually [70]. These tools are often paired with programmable or reconfigurable hardware such as field programmable gate arrays (FPGAs).

Despite the availability of aforementioned tools for adapting MLPs to neuromorphic hardware, these strategies are mainly proposed for silicon-based large-scale digital circuits and are less effective for PE. For instance, in PE, activation functions such as printed tanh-like (ptanh) function [76] or printed ReLU-like (pReLU) function [77] cannot accurately resemble the mathematical counterparts. Consequently, the design and adaption of MLPs in printed neuromorphic hardware would have to be changed to accommodate printed transistors and printed circuit components [13].

2.3 Printed Neuromorphic Circuits

Printed neuromorphic circuits (pNCs) combine the methodologies of PE and neuromorphic computing to create systems that not only leverage the inher-

ent benefits of printed devices, such as ultra-low cost, adaptable and flexible manufacturing methods, and circuit softness, but also incorporate the powerful computational capabilities of neuromorphic computing. These fused advantages make pNCs emerging as a leading technology in the evolution of the next-generation electronics.

2.3.1 Analog versus Digital Circuits

Despite the unique advantages of PE, there are still drawbacks due to the limitations of its additive manufacturing process. For instance, the feature size of printed devices is typically in the scope of micrometer to millimeter, which is three to five orders of magnitude larger than the nanoscale dimensions achievable with photolithographic silicon electronics. Consequently, analog implementation is more favorable than Boolean digital design. Figure 2.5 illustrates a 2-bit adder, which is one of the most simple component in digital circuits. It already requires 7 gates, i.e., 36 transistors, not to mention higher precision (more bits), the demand of analog-digital-converters (ADCs), and the requirement on other components in neuromorphic computing, such as multipliers and activation functions. Table 2.2 compares the hardware cost for implementing a printed neuron with three inputs and one output across different design approaches, namely, 4-bit, 8-bit digital design and analog design. It can be seen that the analog method necessitates a significantly reduced device counts (two to three orders of magnitude) compared to its digital counterpart.

Furthermore, the faults, including parametric and catastrophic faults, cannot be ignored in the additive manufacturing process, where the latter may even severely impact the performance of pNCs (see Chapter 4.4). To avoid the occurrence of catastrophic faults and enable efficient circuit testing, pNCs are also justified to have low device counts.

In sum, with the consideration of the additive fabrication method, the circuit footprint, the fabrication cost, the power consumption, and the device faults, pNCs are limited to have a low device counts. Consequently, analog approach gains significantly superior. Therefore, this thesis focuses on studying printed *analog* neuromorphic circuits and utilizes the term *printed neuromorphic circuits (pNCs)* to specifically refer to the analog ones.

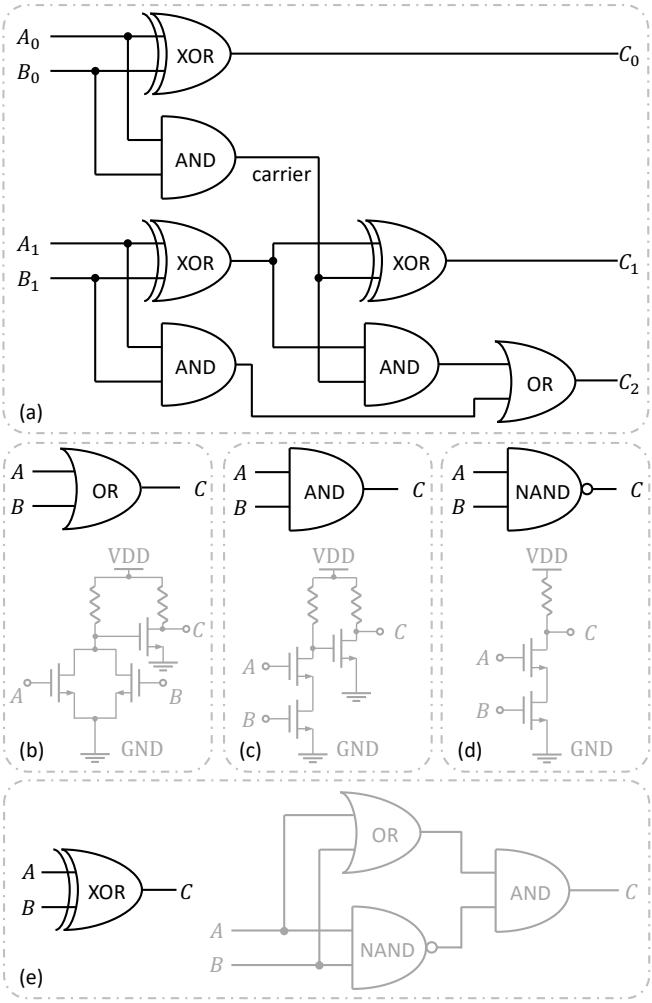


Figure 2.5: Schematic diagram of a printed (2-bit) digital adder: (a) gate-level description of the adder, (b)-(e) transistor-level implementation of the required gates in the adder, where the black part refers to the gates, while the gray part denotes their lower level design.

Table 2.2: Comparison of the hardware cost between analog and digital (4-bit and 8-bit) approaches for a 3-input neuron. (ADC: analog-digital-converter, ReLU: rectified linear unit, #T: number of transistors). Sourced from [76].

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**

2.3.2 Circuit Primitives

Circuit primitives for pNCs were originally proposed in [22, 75, 76]. They consist of resistor crossbars for implementing weighted-sum operations, ptanh circuit for emulating activation function, and printed negation circuits to resemble the expression of negative weights.

Resistor crossbar. The green structure in Figure 2.6 shows the most fundamental architecture in pNCs, i.e., the resistor crossbar, resembling the weighted-sum operations in MLPs. This structure has been widely adopted in various applications, including PIM [4] and ReRAM-based ANN accelerators [27]. According to Kirchhoff's law [35], we obtain

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

By expressing the resistance R as the corresponding conductance $g = 1/R$ and fixing $V_b = 1$V, this equation can be formulated to

$$V_z = \sum_j \frac{g_j^C}{G} V_j + \frac{g_b^C}{G}, \tag{2.4}$$

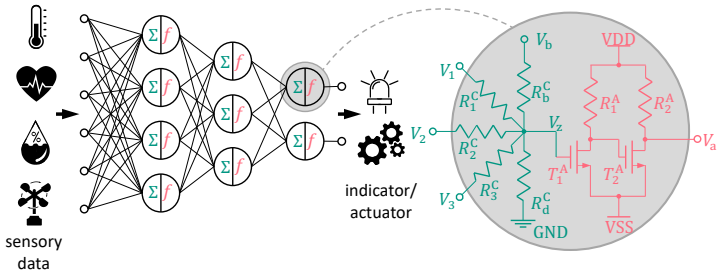


Figure 2.6: Schematic of a pNC receiving sensory data in analog domain and producing output to succeeding components. The right side is the circuit design of a printed neuron with 3 inputs, the green part refers to a resistor crossbar for weighted-sum operation, while the red part represents the ptanh circuit as the activation function.

where G refers to the summed conductance of the resistors in the crossbar, i.e., $\sum_i g_i^{\mathrm{C}} + g_{\mathrm{b}}^{\mathrm{C}} + g_{\mathrm{d}}^{\mathrm{C}}$. In this case, Equation (2.4) shares the same form as Equation (2.1), i.e., the output voltage V_{z} can be seen as the weighted-sum of the input voltages V_j. Here, the weights and bias are represented by the ratio of the conductances. In this way, by designing and printing proper conductance values, the desired weights and biases can be implemented.

Printed tanh-like circuit. Following the resistor crossbar, the signals are passed through an inverter-based ptanh circuit to resemble the activation functions in MLPs. The circuit diagram is illustrated by the red part in Figure 2.6. The characteristic curve of the circuit can be represented by a modified tanh function,

$$V_{\mathrm{a}} = \mathrm{ptanh}(V_{\mathrm{z}}) = \eta_1^{\mathrm{A}} + \eta_2^{\mathrm{A}} \cdot \tanh\left(\left(V_{\mathrm{z}} - \eta_3^{\mathrm{A}}\right) \cdot \eta_4^{\mathrm{A}}\right), \tag{2.5}$$

where $\boldsymbol{\eta}^{\mathrm{A}} = [\eta_1^{\mathrm{A}}, \eta_2^{\mathrm{A}}, \eta_3^{\mathrm{A}}, \eta_4^{\mathrm{A}}]$ are auxiliary parameters describing the scaling and translation of the tanh function. Here, $\boldsymbol{\eta}^{\mathrm{A}}$ is ultimately determined by the physical quantities $\boldsymbol{q}^{\mathrm{A}} = [R_1^{\mathrm{A}}, R_2^{\mathrm{A}}, W_1^{\mathrm{A}}, L_1^{\mathrm{A}}, W_2^{\mathrm{A}}, L_2^{\mathrm{A}}]$ in the circuit, where W_1^{A}, L_1^{A}, W_2^{A}, and L_2^{A} are the geometric features (width and length) of the transistor T_1^{A} and T_2^{A}. In previous work like [22, 75, 76, 79], $\boldsymbol{q}^{\mathrm{A}}$ was designed as a fixed value, namely $\boldsymbol{q}^{\mathrm{A}} = [180\,\mathrm{k\Omega}, 80\,\mathrm{k\Omega}, 100\,\mu\mathrm{m}, 80\,\mu\mathrm{m}, 500\,\mu\mathrm{m}, 40\,\mu\mathrm{m}]$, whereas

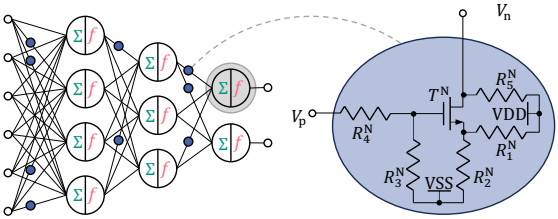


Figure 2.7: Schematic of a pNC with several negation circuits. The right side details the negation circuit proposed in [76].

the corresponding auxiliary parameter was $\boldsymbol{\eta}^{\mathrm{A}} = [0.134, 0.962, 0.183, 24.10]$. In Chapter 3.1, a parametric model of the ptanh circuit is established to enable the training of $\boldsymbol{q}^{\mathrm{A}}$ as a learnable parameter. Consequently, $\boldsymbol{q}^{\mathrm{A}}$ can be optimized alongside the weights and biases (i.e., conductances in resistor crossbars) for specific tasks.

Printed negation circuit. Since the conductance can only be positive values, the resistor crossbars are only able to represent positive weights. However, negative weights are critical in neuromorphic computing to express negative relationships. To address this problem, the inverter-based negation circuit is proposed in [76]. The left side of Figure 2.7 sketches a pNC with several negation circuits prepended to some input of the printed neurons, expressing negative weights, while the detailed schematic of the negation circuit is shown on the right side. Similar to the ptanh circuit, the transfer characteristic of the circuit can be described by a modified negative tanh function, namely

$$\mathrm{neg}(V_{\mathrm{in}}) = -\left(\eta_1^{\mathrm{N}} + \eta_2^{\mathrm{N}} \cdot \tanh\left(\left(V_{\mathrm{in}} - \eta_3^{\mathrm{N}}\right) \cdot \eta_4^{\mathrm{N}}\right)\right), \tag{2.6}$$

where the auxiliary parameter $\boldsymbol{\eta}^{\mathrm{N}} = [\eta_1^{\mathrm{N}}, \eta_2^{\mathrm{N}}, \eta_3^{\mathrm{N}}, \eta_4^{\mathrm{N}}]$ is determined by the physical quantities $\boldsymbol{q}^{\mathrm{N}} = [R_1^{\mathrm{N}}, R_2^{\mathrm{N}}, R_3^{\mathrm{N}}, R_4^{\mathrm{N}}, R_5^{\mathrm{N}}, W^{\mathrm{N}}, L^{\mathrm{N}}]$. Analogously, $\boldsymbol{q}^{\mathrm{N}}$ was a fixed design being $[160\,\Omega, 80\,\Omega, 25\,\mathrm{k}\Omega, 15\,\mathrm{k}\Omega, 80\,\mathrm{k}\Omega, 500\,\mu\mathrm{m}, 40\,\mu\mathrm{m}]$ with $\boldsymbol{\eta}^{\mathrm{N}} = [-0.104, 0.899, -0.056, 3.858]$. They will be extended as a learnable parameter through the parametric model described in Chapter 3.1.

With negation circuit, whenever a negative weight is required, i.e.,

$$(-|w|) \cdot V_{\text{in}},$$

the respective input will be negated to emulate the negative weight through

$$|w| \cdot (-V_{\text{in}}) \leftarrow |w| \cdot \text{neg}(V_{\text{in}}).$$

Printed neuron. Combining Equation (2.4), Equation (2.5), and Equation (2.6), the overall behavior of a printed neuron is given by

$$V_{\text{a}} = \text{ptanh}\left(\sum_j \frac{g_j^{\text{C}}}{G} V_j' + \frac{g_{\text{b}}^{\text{C}}}{G}\right), \tag{2.7}$$

where V_j' denotes the modified input (i.e., either the original or the negated input) voltage, depending on the circuit structure, namely,

$$V_j' = \begin{cases} \text{neg}(V_j), & \text{negation circuit exists,} \\ V_j, & \text{otherwise.} \end{cases} \tag{2.8}$$

Moreover, analogous to Equation (2.3), Equation (2.7) can be formulated in form of matrix multiplication with extension to simplify the bias term, i.e.,

$$V_{\text{a}} = \text{ptanh}\left(\tilde{\boldsymbol{V}}' \cdot \tilde{\boldsymbol{W}}\right), \tag{2.9}$$

with $\tilde{\boldsymbol{V}}'$ collecting the modified input voltages $V_1', V_2', \cdots$ with an additional "1 V" and a "0 V" at the end. Meanwhile, the weight matrix $\tilde{\boldsymbol{W}}$ is given by

$$\tilde{\boldsymbol{W}} = \text{diag}(\boldsymbol{g} \cdot \boldsymbol{1})^{-1} \cdot \boldsymbol{g}^{\top}, \tag{2.10}$$

where $\boldsymbol{g}$ vectorizes the conductances in the crossbar, i.e., $\boldsymbol{g} = [g_1^{\text{C}}, g_2^{\text{C}}, \cdots, g_{\text{b}}^{\text{C}}, g_{\text{d}}^{\text{C}}]$.

2.4 Employed Technology Specification

As technological advancements, particularly in hardware, continue to progress, the performance of the pNCs is expected to evolve as well. Therefore, to ensure the reproducibility of this work, this section specifies and appreciates the

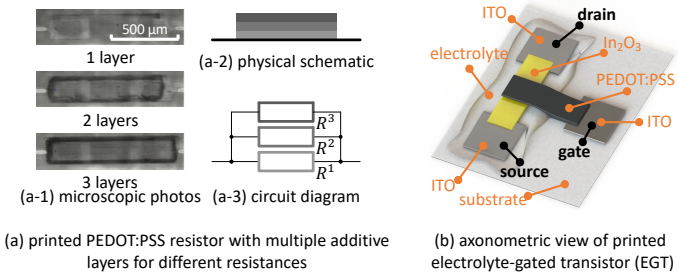


Figure 2.8: Employed printed hardware in this dissertation. (a) printed PEDOT:PSS resistors in the crossbars, sourced from [81], and (b) N-type electrolyte-gated transistor (EGT).

hardware and software technologies utilized in the completion of this thesis.

Hardware specification. In the resistor crossbar, the organic PEDOT:PSS conductive material is utilized. It can typically produce resistance ranging from 100 kΩ to 10 MΩ by adjusting the width, length, and the number of printed layers (height), as shown in Figure 2.8 (a).

For transistors, the N-type electrolyte-gated transistor (EGT) is employed, with signal routing facilitated by indium tin oxide (ITO). The semiconductor indium oxide (In_2O_3) serves as the channel material, composite solid polymer electrolyte (CSPE) is used for the gate insulator, and PEDOT:PSS forms the top gate. The EGT structure is illustrated in Figure 2.8 (b).

Moreover, the entire printing procedure is conducted using the Dimatix DMP-2850 inkjet printer. Other processing details can be found in [75, 76].

Software specification. For the SPICE simulation of the hardware, Cadence[1] along with the printed Process Design Kit (pPDK) [57] were employed.

At the algorithmic level, the implementation was fully carried out using *Python*. The reading and processing of the raw data were facilitated by the *pandas* [47] and *numpy* [20] libraries. The *auto-sklearn* library [16] and optimization tools in the *scipy* library [73] were utilized to fit the transfer char-

[1] `https://www.cadence.com/`.

acteristics and power consumption models of the nonlinear circuits. For the ML-based modeling and training of pNCs, *PyTorch* [53] was the chosen framework, whereas the *NEAT* library [65] was used as reference for the EA-based optimization of pNCs.

For the creation of plots, *matplotlib* [26] in Python was applied, while vector graphs and flowcharts were developed with *AutoDesk AutoCAD* and *Microsoft PowerPoint*. The organization and completion of the thesis were carried out using *LaTeX* and *Overleaf*[2].

[2] `https://www.overleaf.com/`.

Bibliography

[1] Arif U Alam, Pranali Rathi, Heba Beshai, Gursimran K Sarabha, and M Jamal Deen. “Fruit quality monitoring with smart packaging”. In: *Sensors* 21.4 (2021), p. 1509.

[2] Dania A Alsaid, Erika Rebrosova, Margaret Joyce, Marian Rebros, M Atashbar, and Bradley Bazuin. “Gravure printing of ITO transparent electrodes for applications in flexible electronics”. In: *Journal of Display Technology* 8.7 (2012), pp. 391–396.

[3] John V. Arthur and Kwabena A. Boahen. “Silicon-Neuron Design: A Dynamical Systems Approach”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 58.5 (2011), pp. 1034–1043. DOI: 10.1109/TCSI.2010.2089556.

[4] Kazi Asifuzzaman, Narasinga Rao Miniskar, Aaron R Young, Frank Liu, and Jeffrey S Vetter. “A survey on processing-in-memory techniques: Advances and challenges”. In: *Memories-Materials, Devices, Circuits and Systems* 4 (2023), p. 100022.

[5] İsmet Bayraktaroğlu, Arif Selçuk Öğrenci, Günhan Dündar, Sina Balkır, and Ethem Alpaydın. “ANNSyS: An analog neural network synthesis system”. In: *Neural Networks* 12.2 (1999), pp. 325–338.

[6] R Bhattacharya, Margareet M de KOK, and J Zhou. “Rechargeable electronic textile battery”. In: *Applied Physics Letters* 95.22 (2009).

[7] Kirill I Bolotin, KJ Sikes, Zhifang Jiang, M Klima, G Fudenberg, James Hone, Phaly Kim, and Horst L Stormer. “Ultrahigh electron mobility in suspended graphene”. In: *Solid state communications* 146.9-10 (2008), pp. 351–355.

[8] Giorgio E Bonacchini, Caterina Bossio, Francesco Greco, Virgilio Mattoli, Yun-Hi Kim, Guglielmo Lanzani, and Mario Caironi. “Tattoo-paper transfer as a versatile platform for all-printed organic edible electronics”. In: *Advanced Materials* 30.14 (2018), p. 1706091.

[9] Joseph S Chang, Antonio F Facchetti, and Robert Reuss. “A circuits and systems perspective of organic/printed electronics: Review, challenges, and contemporary and emerging design approaches”. In: *IEEE Journal on emerging and selected topics in circuits and systems* 7.1 (2017), pp. 7–26.

[10] Leonardo Weiss Ferreira Chaves and Christian Decker. “A survey on organic smart labels for the Internet-of-Things”. In: *2010 Seventh International Conference on Networked Sensing Systems (INSS)*. IEEE. 2010, pp. 161–164.

[11] Alex Chortos, Jia Liu, and Zhenan Bao. “Pursuing prosthetic electronic skin”. In: *Nature materials* 15.9 (2016), pp. 937–950.

[12] CM Costa, R Gonçalves, and S Lanceros-Méndez. “Recent advances and future challenges in printed batteries”. In: *Energy Storage Materials* 28 (2020), pp. 216–234.

[13] Zheng Cui. *Printed electronics: materials, technologies and applications*. John Wiley & Sons, 2016.

[14] Yiping Dong, Yang Wang, Zhen Lin, and Takahiro Watanabe. “High performance and low latency mapping for neural network into network on chip architecture”. In: *2009 IEEE 8th International Conference on ASIC*. IEEE. 2009, pp. 891–894.

[15] Xi Fan, Wanyi Nie, Hsinhan Tsai, Naixiang Wang, Huihui Huang, Yajun Cheng, Rongjiang Wen, Liujia Ma, Feng Yan, and Yonggao Xia. “PEDOT: PSS for flexible and stretchable electronics: modifications, strategies, and applications”. In: *Advanced Science* 6.19 (2019), p. 1900813.

[16] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. “Efficient and robust automated machine learning”. In: *Advances in neural information processing systems* 28 (2015).

[17] Richard FitzHugh. “Impulses and physiological states in theoretical models of nerve membrane”. In: *Biophysical journal* 1.6 (1961), pp. 445–466.

[18] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[19] Anubha A Gupta, Antoine Bolduc, Sylvain G Cloutier, and Ricardo Izquierdo. “Aerosol Jet Printing for printed electronics rapid prototyping”. In: *2016 IEEE International Symposium on Circuits and systems (ISCAS)*. IEEE. 2016, pp. 866–869.

[20] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. “Array programming with NumPy”. In: *Nature* 585.7825 (2020), pp. 357–362.

[21] Robert Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[22] Michael Hefenbrock. “Modelling and Training Printed Neuromorphic Circuits”. PhD thesis. Dissertation, Karlsruhe, Karlsruhe Institute of Technology (KIT), 2022.

[23] Alan L Hodgkin and Andrew F Huxley. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of physiology* 117.4 (1952), p. 500.

[24] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[25] Erika Hrehorova, Marian Rebros, Alexandra Pekarovicova, Bradley Bazuin, Amrith Ranganathan, Sean Garner, Gary Merz, John Tosch, and Robert Boudreau. "Gravure printing of conductive inks on glass substrates for applications in printed electronics". In: *Journal of Display Technology* 7.6 (2011), pp. 318–324.

[26] John D Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.03 (2007), pp. 90–95.

[27] S. Jain, A. Ankit, I. Chakraborty, T. Gokmen, M. Rasch, W. Haensch, K. Roy, and A. Raghunathan. "Neural network accelerator design with resistive crossbars: Opportunities and challenges". In: *IBM Journal of Research and Development* 63.6 (2019), 10:1–10:13. DOI: 10.1147/JRD.2019.2947011.

[28] Elina Jansson, Johanna Lyytikäinen, Panu Tanninen, Kim Eiroma, Ville Leminen, Kirsi Immonen, and Liisa Hakola. "Suitability of paper-based substrates for printed electronics". In: *Materials* 15.3 (2022), p. 957.

[29] Von Neumann John. "First Draft of a Report on the EDVAC". In: *Pennsylvania: University of Pennsylvania* (1945).

[30] Altynay Kaidarova, Mohammed Asadullah Khan, Marco Marengo, Liam Swanepoel, Alexander Przybysz, Cobus Muller, Andreas Fahlman, Ulrich Buttner, Nathan R Geraldi, Rory P Wilson, et al. "Wearable multifunctional printed graphene sensors". In: *NPJ Flexible Electronics* 3.1 (2019), p. 15.

[31] Laure V Kayser and Darren J Lipomi. "Stretchable conductive polymers and composites based on PEDOT and PEDOT: PSS". In: *Advanced Materials* 31.10 (2019), p. 1806133.

[32] Saleem Khan, Leandro Lorenzelli, and Ravinder S Dahiya. "Technologies for printing sensors and electronics over large flexible substrates: A review". In: *IEEE Sensors Journal* 15.6 (2014), pp. 3164–3185.

[33] Yasser Khan, Arno Thielens, Sifat Muin, Jonathan Ting, Carol Baumbauer, and Ana C Arias. "A new frontier of printed electronics: flexible hybrid electronics". In: *Advanced Materials* 32.15 (2020), p. 1905279.

[34] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[35] G Kirchhoff. "LXIV. On a Deduction of Ohm's Laws, in connexion with the Theory of Electro-statics". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 37.252 (1850), pp. 463–468.

[36] Kwang H Lee, Kimoon Lee, GyuBaek Lee, Min Suk Oh, Jeong-M Choi, Seongil Im, Sungjin Jang, and Eugene Kim. "Flexible high mobility pentacene transistor with high-k/low-k double polymer dielectric layer operating at-5V". In: *ECS Transactions* 16.9 (2008), p. 239.

[37] Jiameng Li, Jiayin Liu, Wenxing Huo, Jingxian Yu, Xinyu Liu, Michael J Haslinger, Michael Muehlberger, Pavel Kulha, and Xian Huang. “Micro and nano materials and processing techniques for printed biodegradable electronics”. In: *Materials Today Nano* 18 (2022), p. 100201.

[38] Jinhao Li, Jie Cao, Baoyang Lu, and Guoying Gu. “3D-printed PEDOT: PSS for soft robotics”. In: *Nature Reviews Materials* 8.9 (2023), pp. 604–622.

[39] Xiaoping Liang, Haifang Li, Jinxin Dou, Qi Wang, Wenya He, Chunya Wang, Donghang Li, Jin-Ming Lin, and Yingying Zhang. “Stable and biocompatible carbon nanotube ink mediated by silk protein for printed electronics”. In: *Advanced Materials* 32.31 (2020), p. 2000165.

[40] Samuel Clark Ligon, Robert Liska, Jürgen Stampfl, Matthias Gurr, and Rolf Mülhaupt. “Polymers for 3D printing and customized additive manufacturing”. In: *Chemical reviews* 117.15 (2017), pp. 10212–10290.

[41] Tong-Hong Lin, Jo Bito, and Manos M Tentzeris. “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.

[42] Li Liu, Yu Feng, and Wei Wu. “Recent progress in printed flexible solid-state supercapacitors for portable and wearable energy storage”. In: *Journal of Power Sources* 410 (2019), pp. 69–77.

[43] Anqi Mao, Mehryar Mohri, and Yutao Zhong. “Cross-entropy loss functions: Theoretical analysis and applications”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 23803–23828.

[44] Hagen Marien, Michiel Steyaert, and Paul Heremans. *Analog organic electronics: building blocks for organic smart sensor systems on foil*. Springer Science & Business Media, 2012.

[45] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[46] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[47] Wes McKinney et al. “Data structures for statistical computing in Python.” In: *SciPy*. Vol. 445. 1. 2010, pp. 51–56.

[48] Don Monroe. *Neuromorphic computing gets ready for the (really) big time*. 2014.

[49] Md Abu Mosa, Jeong Yeop Jo, and Kye-Si Kwon. “Fast on-off jet control of aerosol jet printing (AJP) using internal rotary valve”. In: *Additive Manufacturing* 67 (2023), p. 103466.

[50] Susan Mühl and Beatrice Beyer. “Bio-organic electronics—overview and prospects for the future”. In: *Electronics* 3.3 (2014), pp. 444–461.

[51] Eva M Ortigosa, Antonio Cañas, Eduardo Ros, Pilar Martínez Ortigosa, Sonia Mota, and Javier Díaz. “Hardware description of multi-layer perceptrons with different abstraction levels”. In: *Microprocessors and Microsystems* 30.7 (2006), pp. 435–444.

[52] Lucas Antón Pastur-Romay, Francisco Cedrón, Alejandro Pazos, and Ana Belén Porto-Pazos. “Deep artificial neural networks and neuromorphic chips for big data analysis: pharmaceutical and bioinformatics applications”. In: *International journal of molecular sciences* 17.8 (2016), p. 1313.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[54] Manuel Pietsch, Stefan Schlisske, Martin Held, Noah Strobel, Alexander Wieczorek, and Gerardo Hernandez-Sosa. “Biodegradable inkjet-printed electrochromic display for sustainable short-lifecycle electronics”. In: *Journal of Materials Chemistry C* 8.47 (2020), pp. 16716–16724.

[55] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. “Multilayer perceptron and neural networks”. In: *WSEAS Transactions on Circuits and Systems* 8.7 (2009), pp. 579–588.

[56] Egidio Ragonese, Marco Fattori, and Eugenio Cantatore. “Printed organic electronics on flexible foil: Circuit design and emerging applications”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 68.1 (2020), pp. 42–48.

[57] Farhan Rasheed, Michael Hefenbrock, Michael Beigl, Mehdi B Tahoori, and Jasmin Aghassi-Hagmann. “Variability modeling for printed inorganic electrolyte-gated transistors and circuits”. In: *IEEE transactions on electron devices* 66.1 (2018), pp. 146–152.

[58] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[59] Johannes Schemmel, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. “A wafer-scale neuromorphic hardware system for large-scale neural modeling”. In: *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2010, pp. 1947–1950.

[60] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. “A survey of neuromorphic computing and neural networks in hardware”. In: *arXiv preprint arXiv:1705.06963* (2017).

[61] Cedric Seger. *An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing*. 2018.

[62] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. “Activation functions in neural networks”. In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[63] Ehsan Shirzaei Sani, Changhao Xu, Canran Wang, Yu Song, Jihong Min, Jiaobing Tu, Samuel A Solomon, Jiahong Li, Jaminelli L Banks, David G Armstrong, et al. “A stretchable wireless wearable bioelectronic system for multiplexed monitoring and combination treatment of infected chronic wounds”. In: *Science Advances* 9.12 (2023), eadf7388.

[64] Madhusudan Singh, Hanna M Haverinen, Parul Dhagat, and Ghassan E Jabbour. “Inkjet printing—process and its applications”. In: *Advanced materials* 22.6 (2010), pp. 673–685.

[65] Kenneth O Stanley and Risto Miikkulainen. “Efficient evolution of neural network topologies”. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No. 02TH8600)*. Vol. 2. IEEE. 2002, pp. 1757–1762.

[66] Kuan Sun, Shupeng Zhang, Pengcheng Li, Yijie Xia, Xiang Zhang, Donghe Du, Furkan Halis Isikgor, and Jianyong Ouyang. “Review on application of PEDOTs and PEDOT: PSS in energy conversion and storage devices”. In: *Journal of Materials Science: Materials in Electronics* 26 (2015), pp. 4438–4462.

[67] Donovan Sung, Alejandro de la Fuente Vornbrock, and Vivek Subramanian. “Scaling and optimization of gravure-printed silver nanoparticle lines for printed electronics”. In: *IEEE Transactions on Components and Packaging Technologies* 33.1 (2009), pp. 105–114.

[68] Raghavv Raghavender Suresh, Muthaiyan Lakshmanakumar, JBB Arockia Jayalatha, KS Rajan, Swaminathan Sethuraman, Uma Maheswari Krishnan, and John Bosco Balaguru Rayappan. “Fabrication of screen-printed electrodes: opportunities and challenges”. In: *Journal of Materials Science* 56 (2021), pp. 8951–9006.

[69] Tijmen Tieleman. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2 (2012), p. 26.

[70] Alin Tisan and Jeannette Chin. “An end user platform for implementing Artificial Neutron Networks on FPGA”. In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE. 2015, pp. 856–859.

[71] Granch Berhe Tseghai, Desalegn Alemu Mengistie, Benny Malengier, Kinde Anlay Fante, and Lieva Van Langenhove. “PEDOT: PSS-based conductive textiles and their applications”. In: *Sensors* 20.7 (2020), p. 1881.

[72] Alan M Turing. *Computing machinery and intelligence*. Springer, 1950.

[73] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. In: *Nature methods* 17.3 (2020), pp. 261–272.

[74] John Von Neumann and Ray Kurzweil. *The computer and the brain*. Yale university press, 1958.

[75] Dennis Weller. “Digital and analog computing paradigms in printed electronics”. PhD thesis. Dissertation, Karlsruhe, Karlsruhe Institute of Technology (KIT), 2020, 2021.

[76] Dennis D Weller, Michael Hefenbrock, Michael Beigl, Jasmin Aghassi-Hagmann, and Mehdi B Tahoori. “Realization and training of an inverter-based printed neuromorphic computing system”. In: *Scientific reports* 11.1 (2021), p. 9554.

[77] Dennis D Weller, Michael Hefenbrock, Mehdi B Tahoori, Jasmin Aghassi-Hagmann, and Michael Beigl. “Programmable neuromorphic circuit based on printed electrolyte-gated transistors”. In: *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2020, pp. 446–451.

[78] He Yan, Zhihua Chen, Yan Zheng, Christopher Newman, Jordan R Quinn, Florian Dötz, Marcel Kastler, and Antonio Facchetti. “A high-mobility electron-transporting polymer for printed transistors”. In: *Nature* 457.7230 (2009), pp. 679–686.

[79] Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Aging-Aware Training for Printed Neuromorphic Circuits”. In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–9.

[80] Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Highly-dependable printed neuromorphic circuits based on additive manufacturing”. In: *Flexible and Printed Electronics* 8.2 (2023), p. 025018.

[81] Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Split Additive Manufacturing for Printed Neuromorphic Circuits”. In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2023, pp. 1–6.

3 Modeling and Training

Printed electronics (PE) enables highly flexible and agile fabrication process that can easily adapt any circuit component values through a simple modification of printing trajectory. Therefore, PE enables bespoke design of the pNCs for target tasks. To leverage this advantage, it is imperative to optimize the circuit parameters, e.g., the crossbar resistances corresponding to the weights in MLPs, in a bespoke manner. For this purpose, a parametric circuit model is required to be established. With the established circuit model, the pNCs can be trained to fulfill the desired functionalities by adopting appropriate training objectives. This chapter elucidates the modeling of the pNCs and handling the parametric constraints introduced by physical and technological limitations. Lastly, this chapter introduces the strategies for the training of these pNCs and consider holistic constraints during training.

3.1 Modeling of Printed Neuromorphic Circuits

The modeling methods for pNCs encompass two primary strategies: *physics-informed modeling* and *approximation-based modeling*. The former is suitable for systems that are streamlined and have analytic expression, whereas the latter is preferable for handling more complicated objects. For instance, the resistor crossbar, as discussed in Chapter 2.3.2, can be analytically modeled through Kirchhoff's law. Conversely, for circuits with complex nonlinearities and structures, such as ptanh circuits and negation circuits, to ascertain their transfer characteristic curve with respect to circuit parameters poses substantial challenge. Consequently, it is more practical to employ data-driven approximation methods for the parametric modeling.

3.1.1 Physics-Informed Modeling

Physics-informed modeling involves incorporating the physic laws directly into the modeling process [3]. This integration can substantially reduce (sometimes even avoid) the demand for extensive dataset to develop accurate models. Furthermore, by embedding such a priori knowledge into the model, the generalizability can be well guaranteed.

For instance, in Equation (2.4), the input-output relationship of the resistor crossbar can already be accurately described by introducing the Kirchhoff's law. Certainly, the model should still be further improved to enable the optimization of the existence of the negative weights. In Equation (2.8), the expression of negative weight, i.e., V_j or $\mathrm{neg}(V_j)$, depends on the presence or absence of the negation circuit. However, this is a non-causal relationship. Because the inclusion of a negation circuit should contingent upon the necessity for negative weights. In other word, the presence of the negation circuit should be determined by the requirement for negative weights (which should be the result of the circuit training), rather than determining the sign of the weights through the existence of the negation circuit. To address this problem, the concept of *surrogate conductance* was proposed [20]. A surrogate conductance θ encodes the physical conductance g by its absolute value $|\theta|$, and denote the presence of the negation circuit through $\mathrm{sign}(\theta)$. With this approach, Equation (2.9) and Equation (2.8) can be reformulated as

$$V_\mathrm{a} = \mathrm{ptanh}\left(\tilde{\boldsymbol{V}} \cdot \left(\tilde{\boldsymbol{W}} \odot \mathbb{1}_{\{\boldsymbol{\theta} \geq 0\}}\right) + \mathrm{neg}(\tilde{\boldsymbol{V}}) \cdot \left(\tilde{\boldsymbol{W}} \odot \mathbb{1}_{\{\boldsymbol{\theta} < 0\}}\right)\right). \tag{3.1}$$

Here $\mathbb{1}_{\{\cdot\}}$ is an elementwise indicator function that returns 1 if the respective condition is true, else 0. Consequently, the well-trained surrogate conductances $\boldsymbol{\theta}$ can be converted to the printed conductances through $\boldsymbol{g} = |\boldsymbol{\theta}|$, and the presence of the negation circuits by the sign of each element in $\boldsymbol{\theta}$.

Physics-informed modeling is also employed to model the printed recurrent neuromorphic circuit (pRNC) to describe the temporal behavior of printed capacitors through

$$I = C\frac{\mathrm{d}V}{\mathrm{d}t}.$$

Consequently, the sequential input-output behavior of the whole pRNCs can be precisely modeled. More details can be found in Chapter 6.1.

3.1.2 Approximation-Based Modeling

Approximation-based modeling refers to capturing the system inputs with corresponding outputs through measurement or simulation. Subsequently, the relationship between these inputs and outputs are modeled utilizing approximation methods. In this regard, ANN is recognized as one of the most promising candidates as they have been proven to be universal approximators [7], and are thus particularly suited for developing the black-box surrogate system models. Moreover, ANN-based system models natively allow gradient-based training, because the operations in ANN are fully differentiable. It is important to clarify that the ANN employed in system modeling are not the ones intended to be printed. Instead, these ANNs are the surrogate circuit parametric models that are created to assist the algorithmic training of the circuit parameters.

Algorithm 1 demonstrates an overview of the workflow of the approximation-based modeling employed in this work. With this process, the nonlinear circuits in the pNCs, e.g., ptanh circuit and negation circuit, can be precisely modeled. Subsequently, their circuit parameters can be trained to yield optimal transfer characteristic curves for the training objective like classification accuracy.

Taking [23] as an example, we first define a rough search space $\mathcal{Q}$ for the parameters in the nonlinear circuit based on the e.g., printing technologies. Here, the space of the physical quantities is formulated as $\mathcal{Q} = \{\boldsymbol{q} \,|\, \boldsymbol{q} \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}]\}$. In this initial search space, we sampled $\mathrm{B} = 10\,000$ points through Quasi Monte-Carlo (QMC) strategy [16], which are denoted by $\boldsymbol{q}_i$, $i = 1, ..., 10\,000$. QMC is a pseudo-random sampling strategy, which guarantees the estimation error of to converge with

$$\mathcal{O}\left(\frac{(\log \mathrm{B})^d}{\mathrm{B}}\right),$$

where d is the dimension of the approximation problem, whereas the conver-

Algorithm 1: Approximation-based modeling

Input: Number of samples B
Init : Initial feasible search space of circuit parameters $\boldsymbol{Q}$
while *sampling not finished* **do**
  Sample B points $\{\boldsymbol{q}_1, \cdots, \boldsymbol{q}_B\}$ from the feasible space $\boldsymbol{Q}$
  Conduct B SPICE simulations with sampled circuit parameters $\boldsymbol{q}_b$
  Obtaining outputs $\{\boldsymbol{\eta}_1, \cdots, \boldsymbol{\eta}_B\}$
  Check feasibility of $\boldsymbol{Q}$ through simulation results
  if $\boldsymbol{Q}$ *is feasible* **then**
    sampling finished
  else
    update $\boldsymbol{Q}$
  end
end
Train ANN to fit $\boldsymbol{\eta}_b$ from $\boldsymbol{q}_b$, namely, $\mathrm{ANN}(\boldsymbol{q}_b) = \boldsymbol{\eta}_b, \forall b = 1, 2, \cdots, B$

gence of random Monte-Carlo (MC) follows

$$\mathcal{O}\left(\frac{1}{\mathrm{B}}\right).$$

In other word, for sufficiently large samplings B, the approximation precision of QMC will always outperform that of the random MC. Meanwhile, compared to grid-based sampling, QMC can provide more marginal information. Afterwards, we use Cadence Virtuoso[1] for SPICE simulation based on a prior developed pPDK [15] to simulate the input and output voltages $(\boldsymbol{V}_{\mathrm{in}}, \boldsymbol{V}_{\mathrm{out}})_i$ for each sampled circuit (parameterized by $\boldsymbol{q}_i$). The green points in Figure 3.1 (left) exemplify a simulation result with a certain $\boldsymbol{q}_i$. Note that the number of points plotted in the figure has been reduced for clarity of visualization.

Afterwards, we fit the discrete simulation points by tanh-like curves parameterized by $\boldsymbol{\eta}$, specifically, we fit the simulated data $(\boldsymbol{V}_{\mathrm{in}}, \boldsymbol{V}_{\mathrm{out}})_i$ by Equation (2.5) or Equation (2.6) (depending on the circuit) with minimal Euclidean distance, e.g.,

$$\boldsymbol{\eta}^* = \arg\min_{\boldsymbol{\eta}} \left\| \mathrm{neg}_{\boldsymbol{\eta}}(\boldsymbol{V}_{\mathrm{in}}) - \boldsymbol{V}_{\mathrm{out}} \right\|_2.$$

This optimization problem is solved by the optimization tools in the *scipy* li-

[1] `https://www.cadence.com`

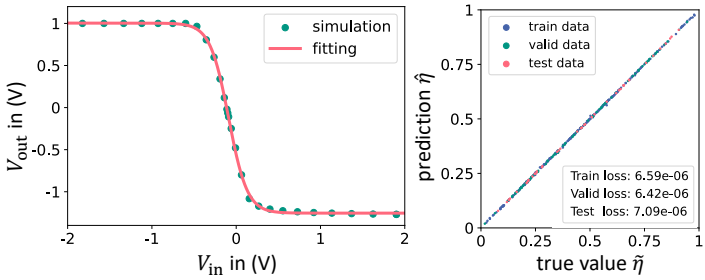


Figure 3.1: Left: parameter fitting from $(\boldsymbol{V}_{\text{in}}, \boldsymbol{V}_{\text{out}})$ to $\boldsymbol{\eta}^{\text{N}}$. Green points show the simulated output voltages with SPICE, and the red curve indicates the fitted negation function parameterized by $\boldsymbol{\eta}^{\text{N}}$. Right: visualization of the results from the surrogate model. The x-axis and the y-axis refer to the true value $\tilde{\boldsymbol{\eta}}^{\text{N}}$ and predicted value $\hat{\boldsymbol{\eta}}^{\text{N}}(\boldsymbol{q})$. Blue, green, and red colors denotes the data from training, validation, and test sets. Sourced from [23].

brary [19]. In this process, we also monitor the fitting error

$$\varepsilon = \left\| \text{neg}_{\boldsymbol{\eta}}(\boldsymbol{V}_{\text{in}}) - \boldsymbol{V}_{\text{out}} \right\|_2 .$$

If the fitting error ε_i exceeds a certain threshold, i.e., if the transfer characteristic does not follow a tanh-like curve, we marked the corresponding $\boldsymbol{q}_i$ as *infeasible*. Consequently, we can update the search space $\mathcal{Q}$ to exclude the infeasible space that do not yield tanh-like transfer characteristics. It is worth noting that, it is usually an iterative process to determine the feasible search space $\mathcal{Q}$. Taking the negation circuit as an example, the feasible search space is finally defined in Table 3.1. We also observe that, the resistances R_1^{N} and R_3^{N} must be larger than R_2^{N} and R_4^{N}, respectively. Otherwise, the voltage divider cannot meet the assumption of a constant ratio due to the connections with surrounding circuit elements.

Finally, we collected feasible physical design parameters $\boldsymbol{q}_i$ and their corresponding auxiliary parameters $\boldsymbol{\eta}_i$. Since the relationship between $\boldsymbol{q}_i$ and $\boldsymbol{\eta}_i$ is complicated, we propose to approximate it by surrogate models based on ANNs. For this, we build the dataset $\mathcal{D} = \{\boldsymbol{q}, \boldsymbol{\eta}\}$ for training the ANN to

Table 3.1: Feasible design space of negation circuit. Sourced from [23].

	R_1^N (Ω)	R_2^N (Ω)	R_3^N (kΩ)	R_4^N (kΩ)	R_5^N (kΩ)	W^N (μm)	L^N (μm)
minimal	10	5	10	8	10	200	10
maximal	500	250	500	400	500	800	70
inequality	$R_1^N > R_2^N$		$R_3^N > R_4^N$		-	-	-

describe the transformation from $\boldsymbol{q}$ to $\boldsymbol{\eta}$.

To train an effective surrogate model, several ML techniques should be considered. *Feature engineering* refers to generate significant more features from existing datasets [24]. For this, the ratios of voltage dividers, i.e., R_2^N/R_1^N and R_4^N/R_3^N, and the ratio between W^N and L^N can be seen as critical features of the circuits. We can therefore extend the design parameters manually with these three ratios, i.e.,

$$\boldsymbol{q} \mapsto [R_1^N, R_2^N, R_3^N, R_4^N, R_5^N, W, L, k_1, k_2, k_3],$$

where k_1, k_2, and k_3 denote the aforementioned ratios. In addition, techniques for automated ML [5] such as data normalization [13], data split [11], weight decay [9], early-stopping [14], hyperparameter tuning [8], and neural architecture search [4] can also be used to produce better performing surrogate models. After employing the training techniques, a 13-layer ANN is obtained as the final surrogate negation circuit model. The plot on the right side of Figure 3.1 visualizes the results of all three sets from a surrogate model. We can thus conclude that, the surrogate model provides acceptable predictions from the component values $\boldsymbol{q}$ to the characteristic curves $\boldsymbol{\eta}$ and there is no overfitting on the training data.

In conclusion, approximation-based modeling is more versatile than physics-informed modeling methods, as it can be applied without any/fewer prior knowledge about the system. In this thesis, we utilize this approach to model the power consumption of pNCs (Chapter 5.2) and the printed spiking neuromorphic circuits (pSNCs) (Chapter 6.2). However, this approach requires expensive data collection (either through experimental measurements or simu-

lations), and necessitates significant effort to train effective and generalizable surrogate system models.

3.1.3 Constraints in Modeling

Although pNCs emulate the computational paradigm of MLPs, as hardware, pNCs suffer more limitations than MLPs. These limitations must be considered as constraints in the design and optimization of pNCs. The constraints are primarily categorized into two types: *parametric constrains* and *holistic constraints*. The former relates to restrictions imposed on single parameters, usually due to physical, technological, and electrical limitations. The latter pertains to performance indicators of circuit design, such as power consumption and circuit footprint. These holistic constraints cannot be split into parametric constraints, posing more challenging to resolve them. Thus, holistic constrains are generally considered during training instead of during circuit modeling. Therefore, this section primarily introduces methods for parametric constrains that convert the constrained parameters into unconstrained ones. The solutions for holistic constraints are introduced in Chapter 3.2.3.

A typical parametric constraint can be observed in Table 3.1, e.g., W^{N} is limited to a certain interval $[W^{\mathrm{N}}_{\mathrm{min}}, W^{\mathrm{N}}_{\mathrm{max}}]$. Drawing inspiration from data normalization [13], we can convert the constrained problem into an unconstrained problem by introducing a function with a finite range of values, e.g., the sigmoid function. Specifically, we introduce an unconstrained optimization variable $\mathfrak{w} \in \mathbb{R}$ and map it through

$$W^{\mathrm{N}} = W^{\mathrm{N}}_{\mathrm{min}} + \mathrm{sigmoid}(\mathfrak{w}) \cdot (W^{\mathrm{N}}_{\mathrm{max}} - W^{\mathrm{N}}_{\mathrm{min}}).$$

In this way, the range of W^{N} is automatically limited in its feasible range. Moreover, we can optimize $\mathfrak{w}$ without any constraints and use the intermediate W^{N} as the feasible width (a geometric feature of the transistors) for further calculation.

A more complex example is an inequality constraint between two parameters, such as $R^{\mathrm{N}}_1 > R^{\mathrm{N}}_2$ in Table 3.1. This type of constraint can also be simpli-

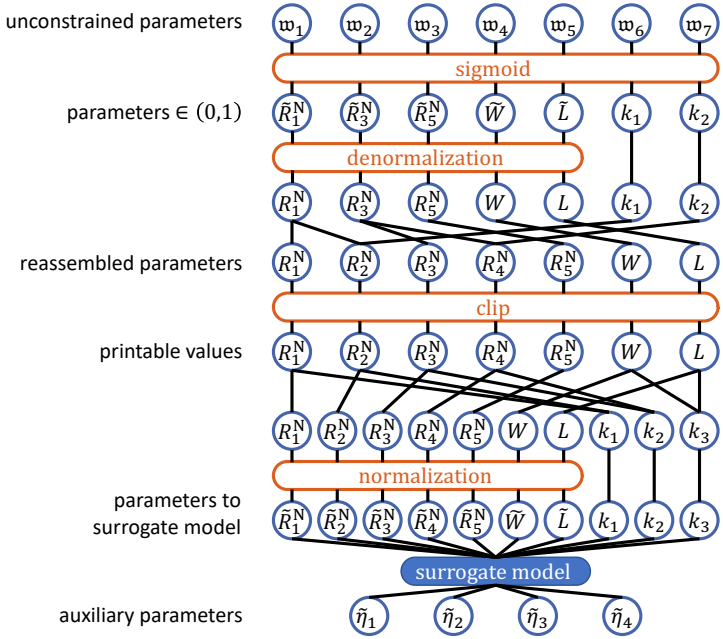


Figure 3.2: Flowchart for processing the unconstrained learnable parameters to satisfy circuit constraints. Subsequently, the constrained parameters are converted to match the surrogate nonlinear circuit model. Sourced from [23].

fied as unconstrained problem by introducing a slave variable $\mathfrak{w} \in \mathbb{R}$ through

$$R_1^{\mathrm{N}} > R_2^{\mathrm{N}} \mapsto R_2^{\mathrm{N}} = k_1 \cdot R^{\mathrm{N}} \text{ with } k = \mathrm{sigmoid}(\mathfrak{w}) < 1.$$

Figure 3.2 illustrates this process for converting a constrained optimization problem of Table 3.1 into an unconstrained one.

Another typical constraint in pNC is the range of printable conductances, which is defined as $\{0\} \cup [\mathrm{g_{min}}, \mathrm{g_{max}}]$ (zero refers to not printing). Unlike the aforementioned single interval-based constraints, the conductance is learned

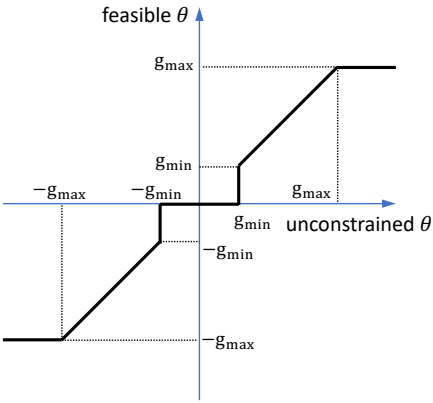


Figure 3.3: Mapping of an unconstrained surrogate conductance to the printable range. Sourced from [21].

through surrogate conductance θ, which encodes the printed conductance by its absolute value, i.e., $g = |\theta|$. Therefore, the constraint on θ is given by

$$\theta \in [-g_{max}, -g_{min}] \cup \{0\} \cup [g_{min}, g_{max}].$$

To respect this constraint, each learnable surrogate conductance θ is projected to the printable range before performing the weighted-sum operation in Equation (2.10), i.e.,

$$\theta \leftarrow \begin{cases} 0, & |\theta| < g_{min}, \\ \theta, & |\theta| \in [g_{min}, g_{max}], \\ \text{sign}(\theta) \cdot g_{max}, & |\theta| > g_{max}, \end{cases} \tag{3.2}$$

as shown in Figure 3.3. Obviously, there are intervals of the projection with zero-gradient. Once θ falls into these intervals, it can no longer be updated by gradient-based training methods. In response, we introduce a gradient relaxation method to still allow gradient-based training, details are described in Chapter 3.2.1.

3.2 Training of Printed Neuromorphic Circuits

Upon modeling the pNCs with consideration of their constraints, a training process is initiated to optimize the learnable parameters in pNCs. Benefiting from the efficiency of backpropagation, gradient-based optimization emerges as the primary strategy. Nonetheless, the presence of non-differentiable operations within circuit design and the circuit constraints, such as those seen in Equation (3.2), can impede gradient-driven training. This thesis proposes a solution by introducing the relaxed gradients as heuristics to facilitate gradient-based training despite these hindrances. In addition, this thesis proposes an evolutionary algorithm (EA). This algorithm skillfully encodes the pNCs and designs their crossover and mutation processes, enabling the pNCs to evolve over generations. The superiority of EA over the gradient methods is its versatility: it does not necessitate the problems to be differentiable. Furthermore, it can simply incorporate constraints by extinction of infeasible individuals. Moreover, EA can even support the topological optimization of pNCs, offering a more flexible and comprehensive optimization solution within a larger search space.

3.2.1 Machine Learning-Based Training

Gradient-based learning with backpropagation [6] forms the backbone of training modern ANNs. The essential idea of gradient-based training through backpropagation has been introduced in Chapter 2.2.2. However, straightforward gradient-based optimization is unable to handle problems that encompass operations which are non-differentiable, e.g., hyperparameters related to neural architecture. Additionally, some functions do not provide useful update information through their gradient, e.g., piece-wise constant functions described in Equation (3.2). To still allow gradient-based training, this thesis proposes to introduce heuristics, i.e., relaxed gradients, as surrogate gradients to tackle this issue, such as straight through estimator (STE) [2].

As exemplified in Figure 3.4, the function produces 0 gradient in the interval $\theta \in [-\infty, -g_{max}] \cup [-g_{min}, g_{min}] \cup [g_{max}, \infty]$. Once θ falls into this interval, θ will be no longer updated. To mitigate this issue, we introduce a relaxed

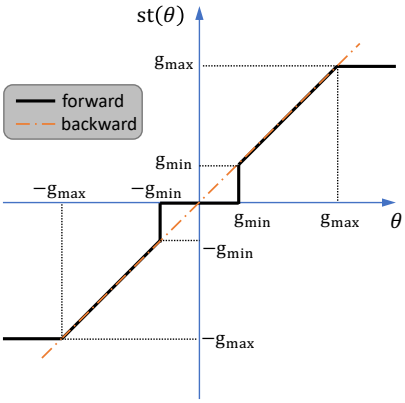


Figure 3.4: Straight-through gradient estimator for feasible θ. The black curve indicates the forward pass and the orange dash-dot line denotes the backward pass for gradient estimation. Sourced from [21].

function, i.e., the orange function, as a heuristic to further enable training of θ. Although the relaxed function does not produce gradient information that is identical to the original function, it nonetheless offers useful and heuristic gradient information that can guide the training process toward the correct direction.

This approach is extensively utilized throughout this thesis, because physical systems frequently suffer from such limitations. Moreover, circuit design often contains discrete parameters, such as the count of devices, with Figure 3.5 depicting an example for the count a resistor. As shown in Equation (3.2), if a conductance g falls below g_{min}, the component will not be printed; otherwise, the count of this resistor will be 1. This is represented by the black function in Figure 3.5. Notably, the gradient of this function is almost 0 everywhere, making it impossible for providing meaningful gradient information for the training of parameters with respect to its count. In this case, the gradient relaxation method can also be employed to heuristically enable the training through gradient-based approach.

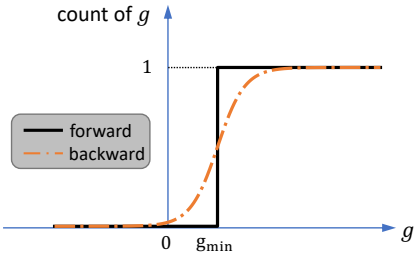


Figure 3.5: Forward and backward pass of the count of a printed resistor, featured by its conductance g. The black curve counts the number of g by 1 when $g > g_{min}$, otherwise 0. In backpropagation, the orange function is employed to derive the gradient information for the backpropagation. Sourced from [22].

3.2.2 Evolutionary Algorithm-Based Training

As a counterpart of gradient-based optimization process, evolutionary algorithm (EA) are inspired by the natural selection and biological evolution [1]. By leveraging operations like crossover and mutation, the solutions are optimized incrementally during evolution. The distinct advantage of EA over gradient-based methods lies in their versatility and adaptability, e.g., it does not necessitate a problem to be differentiable. Thus, through strategic encoding, EA allows larger search space, including parameter (weight) optimization and topology (architecture) optimization. Although EAs are generally less efficient than gradient-based methods, especially for large scale problems, this drawback is mitigated by the fact that PE generally targets small-scale circuits in edge scenarios. Inspired by *NeuroEvolution of Augmenting Topologies (NEAT)* [17], this thesis proposes an EA, capable of both training network parameters (conductances) and searching for optimal neural architecture (circuit topologies).

The core parts of the proposed approach for training pNCs is shown in Figure 3.6, involving *genes* that encode the circuit parameters and *genomes* that compose of multiple structured genes to represent the structure of pNCs. They are optimized through the crossover and mutation during their evolution.

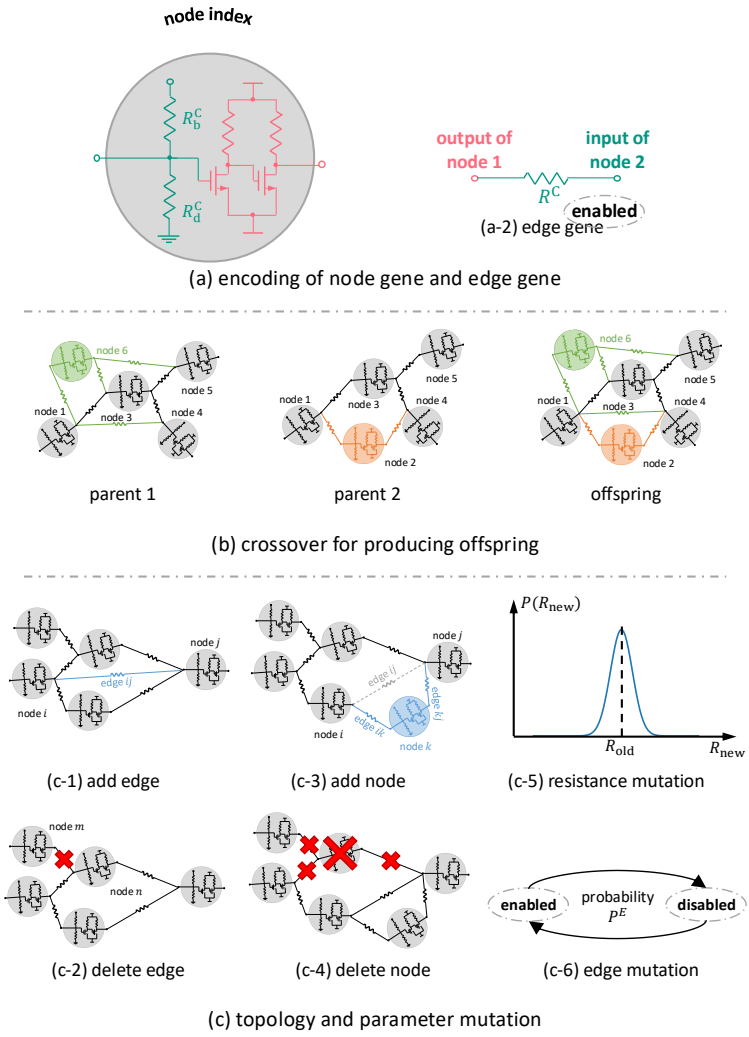


Figure 3.6: Overview of the EA-based training of pNCs: (a) genes that encode nodes and edges, (b) crossover from the parent genomes to offsprings, and (c) mutation of the topology and learnable parameters.

Encoding. The algorithm involves two gene types, namely *node genes* indicating neurons and *edge genes* denoting connections between neurons. As shown in Figure 3.6(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 an activation circuit as shown in Figure 2.6. Note that, here the figure illustrates a ptanh circuit as the activation circuit, however, through specific encoding, the multiple activation circuits can be optimized and selected during evolution. More details can be found in Chapter 4.3. 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 3.6(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 (topology). Similarly, these learnable parameters mutate during evolution.

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

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 Algorithm 2.

Speciation. To protect novel genomes from immediate extinction without being evolved, the genomes within the population 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 3.6(b) in parent

1 and 2) and distinctive structures (exemplified as the green and orange parts in Figure 3.6(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 $\mathcal{F}$ of each genome is evaluated by the objective function $\mathcal{O}(\cdot)$. Meanwhile, the average fitness within a species is calculated to represent the fitness of each species F_s and is summarized in the set $\mathcal{F}_S$ for all species. If the fitness F_s of a certain species does not show improvement over $\mathcal{K}_1$ generations, it will be 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 $P_s = \{p_1, \cdots, p_{\mathcal{K}}\}$ 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 3.6(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 3.6(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 3.6(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

Algorithm 2: Evolutionary algorithm-based training

Input: $\mathcal{D}$: dataset,
N: population size,
$\mathcal{O}(\cdot)$: objective function,
selection$(\cdot)$: parents selection function,
adjust$(\cdot)$: species population size calculator,
$\mathcal{K}$: number of candidate parents for crossover,
$\mathcal{K}_1$: patience for species improvement,
$\mathcal{K}_2$: number of protected species,
$\mathcal{K}_3$: patience for evolution

Init : population $P \leftarrow N$ genomes,
stop $\leftarrow$ *False*,
set of species $\mathcal{S} \leftarrow \varnothing$,
set of species population size $\mathcal{N}_S \leftarrow \varnothing$,
set of genome fitness $\mathcal{F} \leftarrow \varnothing$,
set of species fitness $\mathcal{F}_S \leftarrow \varnothing$

while *not* stop **do**
  $\mathcal{S} \leftarrow$ speciation$(\mathcal{P})$
  $\mathcal{F}, \mathcal{F}_S \leftarrow \mathcal{O}(\mathcal{S}, \mathcal{D})$
  $\mathcal{N}_S \leftarrow$ adjust$(\mathcal{F}_s)$
  for *s in* $\mathcal{S}$ **do**
    if F_s *not* improved for $\mathcal{K}_1$ generations **then**
      if *s not* the best $\mathcal{K}_2$ species **then**
        s extinct
      end
    else
      for *n in* $\{1, 2, \cdots, N_s\}$ **do**
        $p_1, p_2 \leftarrow$ selection$(s, \mathcal{K})$
        $o_n \leftarrow$ crossover(p_1, p_2)
        $o_n \leftarrow$ mutation(o_n)
      end
    end
    $s \leftarrow \{o_1, o_2, \cdots, o_{N_s}\}$
  end
  if F *not* improved for $\mathcal{K}_3$ generations **then**
    stop $\leftarrow$ *True*
  end
end

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 3.6(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 3.6(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.,

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

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

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

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

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

with

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

Note that, according to Equation (2.5), M is always real-valued. Furthermore, the algorithm is only negligible affected even if $M\approx 1$ or $V_{\text{a}}^{i}\approx 0$, 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 3.6(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 3.6(c-6).

Objective. The training objective, i.e., the fitness function, for classification accuracy 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 [10]. Although EA enables to directly employ accuracy (i.e., the actual classification metric) as the training target, integrating cross-entropy can offer a 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

$$\mathcal{O}(\mathcal{D}, \text{🧬}) = \text{CE}\left(\mathcal{D}, \text{🧬}\right) - \text{ACC}\left(\mathcal{D}, \text{🧬}\right). \tag{3.3}$$

During the evolution, the algorithm will progressively increase 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.

3.2.3 Constraints during Training

As mentioned in Chapter 3.1.3, apart from parametric constraints, there are some sophisticated constraints need to be considered during training, such as the power consumption or the circuit footprint of the pNCs. These constraints typically cannot be simplified into simple parametric forms but are instead represented as a comprehensive functional form, i.e., $\varphi(\boldsymbol{\theta}, \boldsymbol{q})$. Here, $\varphi(\cdot)$ can be either analytical expression or black-box models that acquired through e.g. approximation-based modeling. In this work, we refer to this kind of constraints as *holistic constraints*. In addition, holistic constraints may be either equality, i.e., $\varphi(\boldsymbol{\theta}, \boldsymbol{q}) = \mathcal{C}$ or inequality $\varphi(\boldsymbol{\theta}, \boldsymbol{q}) \leq \mathcal{C}$ constraint, where $\mathcal{C}$ denotes the design requirement of circuit performance $\varphi(\boldsymbol{\theta}, \boldsymbol{q})$.

Holistic constraints are easy to be tackled by EA-based training, because it can simply remove the genomes that do not fit the constraints. However,

Algorithm 3: Augmented Lagrangian for equality constraint

Input: dataset $\mathcal{D}$,
loss function $\mathcal{L}(\cdot)$,
constraint $c(\cdot)$,
learning rate α,
incremental step size $\Delta\mu$,
stop-criterion,
early-stopping criterion,
Init : $\lambda \leftarrow 0$,
$\mu \leftarrow 0$
while *stop-criterion not satisfied* **do**
  while *not early-stopping* **do**
    $\mathcal{O}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) \leftarrow \mathcal{L}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) + \lambda \cdot c(\boldsymbol{\theta}, \boldsymbol{q}) + \frac{\mu}{2} \cdot c^2(\boldsymbol{\theta}, \boldsymbol{q})$
    Backpropagation
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \nabla_{\boldsymbol{\theta}} \mathcal{O}$
    $\boldsymbol{q} \leftarrow \boldsymbol{q} - \alpha \cdot \nabla_{\boldsymbol{q}} \mathcal{O}$
  end
  $\lambda \leftarrow \lambda + \mu \cdot c(\boldsymbol{\theta}, \boldsymbol{q})$
  $\mu \leftarrow \mu + \Delta\mu$
end

in gradient-based training, the constraints are hard to be guaranteed, as the training dynamic is only aware of the local gradient information. To this end, a naive approach to handle holistic constraints would be adding the constraint $\varphi(\cdot)$ as a weighted regularizer (penalty term) to the training objective, e.g.,

$$\mathcal{O}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) = (1-\mu) \cdot \mathcal{L}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) + \mu \cdot \varphi(\boldsymbol{\theta}, \boldsymbol{q}), \tag{3.4}$$

where $\mu \in [0,1]$ is a balance factor that weights the loss function and the constraint (e.g., circuit power consumption). However, it is almost impossible to select a suitable μ to meet the constraints with only few trials. Rather, it necessitates to tune μ for numerous times to draw a Pareto-optimal [18] trade-off for satisfying the constraint while preserving optimal classification accuracy. Thus, this approach is mainly used to investigate the relationships between multiple objectives like power consumption versus classification accuracy.

To match the constraints with fewer training trials, we propose to employ the *augmented Lagrangian* algorithm [12] to guarantee the holistic constraints

during training. Different from the penalty method, augmented Lagrangian introduces an additional term to emulate the Lagrange multiplier. Consequently, it integrates the advantages of both the penalty method and the normal Lagrangian method. This integration not only guarantees the strict satisfaction of constraints, but also possesses a robust and fast convergence. Specifically, in augmented Lagrangian, the objective function is formulated as

$$\mathcal{O}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q}) = \mathcal{L}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q}) + \lambda \cdot c(\boldsymbol{\theta},\boldsymbol{q}) + \frac{\mu}{2} \cdot c^2(\boldsymbol{\theta},\boldsymbol{q}),$$

where λ is the Lagrangian multiplier of the constraint $c(\boldsymbol{\theta},\boldsymbol{q}) = \varphi(\boldsymbol{\theta},\boldsymbol{q}) - \mathcal{C}$, and μ serves as the weight of the quadratic penalty term. Unlike Equation (3.4), it is no longer necessary to tune the value of the penalty term μ. Rather, following Karush–Kuhn–Tucker (KKT) condition, the optimal condition (while satisfying the constraint) is given by

$$\nabla\mathcal{O}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q}) = \nabla\mathcal{L}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q}) + (\lambda + \mu \cdot c(\boldsymbol{\theta},\boldsymbol{q})) \cdot \nabla c(\boldsymbol{\theta},\boldsymbol{q}) = 0.$$

Subsequently, the Lagrangian multiplier λ should be updated through

$$\lambda \leftarrow \lambda + \mu \cdot c(\boldsymbol{\theta},\boldsymbol{q})$$

to converge to the optimal value that satisfies the KKT condition [12]. Algorithm 3 describes an augmented Lagrangian for equality constraints based on gradient optimization approach.

In practical applications, inequality constraints tend to be more of interest. Because in circuit design, there is generally a prescribed upper bound of the budget, e.g., maximal power consumption, rather than a fixed value that must be reached. Therefore, following the motivation of [12], we formulate an auxiliary objective function

$$\mathcal{O}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q}) = \max_{\lambda\geq 0}\Big\{\mathcal{L}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q}) + \lambda \cdot c(\boldsymbol{\theta},\boldsymbol{q})\Big\} = \begin{cases} \mathcal{L}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q}), & \text{if feasible,} \\ \infty, & \text{otherwise.} \end{cases} \tag{3.5}$$

In this way, the minimizer of $\mathcal{O}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})$ is identical to that of $\mathcal{L}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})$ while satisfying the inequality constraint. However, Equation (3.5) poses another

challenge: When the constraint is not satisfied, i.e., $c(\boldsymbol{\theta}, \boldsymbol{q}) > 0$, the multiplier λ tends to ∞, thus, the function cannot provide any gradient information to update the parameters $\boldsymbol{\theta}$ and $\boldsymbol{q}$. To mitigate this issue, we introduced a penalty term on λ to prevent from a very large value, namely,

$$\mathcal{O}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) = \max_{\lambda \geq 0} \left\{ \mathcal{L}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) + \lambda \cdot c(\boldsymbol{\theta}, \boldsymbol{q}) - \frac{1}{2\mu} \cdot (\lambda - \lambda')^2 \right\}, \tag{3.6}$$

where λ' denotes the λ value from the last update. In this way, each update to λ is suppressed to the neighborhood of previous location, preventing from infinity. Consequently, Equation (3.6) serves as a smoothed approximation of Equation (3.5) and is utilized as the objective function. Fortunately, Equation (3.6) presents a quadratic optimization problem with respect to λ, which is has an analytical solution, namely,

$$\lambda = \begin{cases} 0, & \lambda' + \mu \cdot c(\boldsymbol{\theta}, \boldsymbol{q}) < 0, \\ \lambda' + \mu \cdot c(\boldsymbol{\theta}, \boldsymbol{q}), & \text{otherwise.} \end{cases}$$

Replacing the λ value in Equation (3.6) by this analytical solution, we simplify the objective function as

$$\mathcal{O}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) = \mathcal{L}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) + \begin{cases} -\frac{1}{2\mu} \cdot (\lambda')^2, & \lambda' + \mu \cdot c(\boldsymbol{\theta}, \boldsymbol{q}) < 0, \\ c(\boldsymbol{\theta}, \boldsymbol{q}) \cdot \left(\lambda' + \frac{\mu}{2} c(\boldsymbol{\theta}, \boldsymbol{q}) \right), & \text{otherwise.} \end{cases}$$

Algorithm 4 describes an augmented Lagrangian for inequality constraints based on gradient optimization approach.

3.2.4 Discussion

Gradient-based approaches surpass EA in efficiency, however, they require the problem to be differentiable. Consequently, the efforts in gradient methods largely lies in designing strategies to convert or approximate non-differentiable problems into differentiable. Additionally, gradient methods have inadequate capabilities for constrained optimization because the constraints are essentially also non-differentiable (either satisfied or unsatisfied). Therefore, in

Algorithm 4: Augmented Lagrangian for inequality constraint

Input: dataset $\mathcal{D}$,
loss function $\mathcal{L}(\cdot)$,
constraint $c(\cdot)$,
learning rate α,
incremental step size $\Delta\mu$,
stop-criterion,
early-stopping criterion,

Init : $\lambda \leftarrow 0$,
$\mu \leftarrow 0$

while *stop-criterion not satisfied* **do**

  while *not early-stopping* **do**

    $$\mathcal{O}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) \leftarrow \mathcal{L}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) + \begin{cases} -\frac{1}{2\mu} \cdot \lambda^2, & \lambda + \mu \cdot c(\boldsymbol{\theta}, \boldsymbol{q}) < 0, \\ c(\boldsymbol{\theta}, \boldsymbol{q}) \cdot \left(\lambda + \frac{\mu}{2} c(\boldsymbol{\theta}, \boldsymbol{q})\right), & \text{otherwise.} \end{cases}$$

    Backpropagation
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \nabla_{\boldsymbol{\theta}} \mathcal{O}$
    $\boldsymbol{q} \leftarrow \boldsymbol{q} - \alpha \cdot \nabla_{\boldsymbol{q}} \mathcal{O}$

  end

  $$\lambda \leftarrow \begin{cases} 0, & \lambda + \mu \cdot c(\boldsymbol{\theta}, \boldsymbol{q}) < 0, \\ \lambda + \mu \cdot c(\boldsymbol{\theta}, \boldsymbol{q}), & \text{otherwise.} \end{cases}$$

  $\mu \leftarrow \mu + \Delta\mu$

end

constrained problems, it is necessary to employ additional techniques to convert the constrained optimization problem into an unconstrained one, such as penalty terms or augmented Lagrangian methods.

On the other hand, EAs are favored for their capacity to explore larger optimization spaces and to handle constraints during evolution. However, they possess higher algorithmic complexity compared to the gradient counterpart. However, the design of EA methods must consider the encodings that render the desired parameters to be learnable during evolution. Meanwhile, mechanisms need to be employed to protect novel genomes from immediate extinction without being sufficiently evolved.

Although EAs are less efficient than gradient-based training, it is still favored in the design and optimization of pNCs, because

1. The most time-consuming steps in EA are evaluating genomes and producing offsprings, which are strongly related (almost proportional to) the population size N. However, the evaluation and offspring production in EA are highly suitable to parallelization, through which the training can be accelerated.
2. Even though EA takes longer training time than gradient methods, the duration 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.

In summary, both gradient-based and EA-based training methodologies offer unique advantages. Based on specific characteristics of different training objectives, this dissertation will employ appropriate methods to train the pNCs. For instance, when we incorporate circuit aging into the training objective (Chapter 4.1), an additional temporal dimension is introduced, which can significantly increase the training complexity. As a result, we choose the gradient-based method for circuit training to guarantee the training efficiency. Meanwhile, as we focus on optimizing circuit compactness (Chapter 5.3), EA method is employed because of its superior capability in searching circuit architecture.

Bibliography

[1] Thomas Bäck and Hans-Paul Schwefel. “An overview of evolutionary algorithms for parameter optimization”. In: *Evolutionary computation* 1.1 (1993), pp. 1–23.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. “Estimating or propagating gradients through stochastic neurons for conditional computation”. In: *arXiv preprint arXiv:1308.3432* (2013).

[3] Wenqian Chen, Qian Wang, Jan S Hesthaven, and Chuhua Zhang. “Physics-informed machine learning for reduced-order modeling of nonlinear problems”. In: *Journal of computational physics* 446 (2021), p. 110666.

[4] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. “Neural architecture search: A survey”. In: *Journal of Machine Learning Research* 20.55 (2019), pp. 1–21.

[5] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. “Efficient and robust automated machine learning”. In: *Advances in neural information processing systems* 28 (2015).

[6] Robert Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[7] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[8] Md Riyad Hossain and Douglas Timmer. “Machine learning model optimization with hyper parameter tuning approach”. In: *Global Journal of Computer Science and Technology* 21.D2 (2021), pp. 7–13.

[9] Anders Krogh and John Hertz. “A simple weight decay can improve generalization”. In: *Advances in neural information processing systems* 4 (1991).

[10] Anqi Mao, Mehryar Mohri, and Yutao Zhong. “Cross-entropy loss functions: Theoretical analysis and applications”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 23803–23828.

[11] Ismail Muraina. “Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts”. In: *7th International Mardin Artuklu Scientific Research Conference*. 2022, pp. 496–504.

[12] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[13] SGOPAL Patro and Kishore Kumar Sahu. “Normalization: A preprocessing stage”. In: *arXiv preprint arXiv:1503.06462* (2015).

[14] Lutz Prechelt. “Automatic early stopping using cross validation: quantifying the criteria”. In: *Neural networks* 11.4 (1998), pp. 761–767.

[15] Farhan Rasheed, Michael Hefenbrock, Michael Beigl, Mehdi B Tahoori, and Jasmin Aghassi-Hagmann. “Variability modeling for printed inorganic electrolyte-gated transistors and circuits”. In: *IEEE transactions on electron devices* 66.1 (2018), pp. 146–152.

[16] IM SobolÍ. “Quasi-Monte Carlo Methods”. In: *Progress in Nuclear Energy* 24.1-3 (1990), pp. 55–61.

[17] Kenneth O Stanley and Risto Miikkulainen. “Efficient evolution of neural network topologies”. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*. Vol. 2. IEEE. 2002, pp. 1757–1762.

[18] Joseph E Stiglitz. “Pareto optimality and competition”. In: *The Journal of Finance* 36.2 (1981), pp. 235–251.

[19] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. In: *Nature methods* 17.3 (2020), pp. 261–272.

[20] Dennis D Weller, Michael Hefenbrock, Michael Beigl, Jasmin Aghassi-Hagmann, and Mehdi B Tahoori. “Realization and training of an inverter-based printed neuromorphic computing system”. In: *Scientific reports* 11.1 (2021), p. 9554.

[21] Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Highly-dependable printed neuromorphic circuits based on additive manufacturing”. In: *Flexible and Printed Electronics* 8.2 (2023), p. 025018.

[22] Haibin Zhao, Priyanjana Pal, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Power-Aware Training for Energy-Efficient Printed Neuromorphic Circuits”. In: *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE. 2023, pp. 1–9.

[23] Haibin Zhao, Brojogopal Sapui, Michael Hefenbrock, Zhidong Yang, Michael Beigl, and Mehdi B Tahoori. “Highly-Bespoke Robust Printed Neuromorphic Circuits”. In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2023, pp. 1–6.

[24] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018.

4 Reliability Design

The additive manufacturing process of printed electronics (PE) offers significant advantages, including exceptional fabrication flexibility, abundant choices of functional inks, and extremely low production costs. However, this maskless approach also presents drawbacks in comparison to subtractive manufacturing, such as the reduced printing precision and large feature sizes [8, 22, 49]. The former will directly influence the geometric features of the printed devices, whereas the latter hinders the packaging of the circuits, which leads to aging problem of circuit components. Unfortunately, as pNCs employ analog computing scheme, these circuits are more susceptible to those variations compared to digital circuits. In this regard, the design of pNCs have to take circuit robustness into account. This work emphasizes the importance of algorithmic level solutions through its ability to substantially enhance the reliability and robustness of pNCs. Note that, this purely algorithmic level optimization is independent of the improvements in printing techniques [25] or materials [44], allowing for enhanced circuit reliability even if the influences could not be controlled or reduced from those standpoints.

4.1 Robustness against Device Aging

Due to environmental influences, e.g., thermal stress in the field, the thin-film printed devices exhibit run-time degradation through usage, i.e., **aging**) [5, 17, 27]. Compared to inorganic materials, the properties of organic materials make them susceptible to environmental influences such as water, oxygen or photon irradiation, resulting in poor stability and repeatability of organic electronic devices under normal operating conditions [8]. Consequently, accounting for the aging of organic materials is essential in the design of pNCs to enhance circuit robustness. Given that resistors containing PEDOT:PSS are the primary

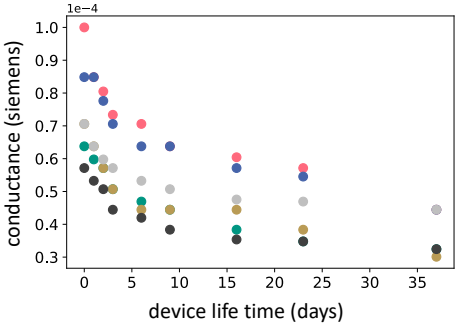


Figure 4.1: The conductance values of six printed PEDOT:PSS resistors measured over 37 days. Sourced from [51].

organic elements in the pNCs discussed in this study, our attention is centered on the aging of printed resistors.

4.1.1 Modeling of Resistor Aging

To study the aging of printed resistors, six printed PEDOT:PSS resistors were fabricated, and their conductances were measured over 37 days. Five of them have different initial conductances, while two of them have the same initial conductance. Their conductance values over time are displayed in Figure 4.1.

We first process the measurement data by normalizing the time to an interval of $[0,1]$. Furthermore, we divided the measured conductance values by the initial value $g_0 = g(0)$ to assess the relative conductance degradation, see Figure 4.2. Similar to the aging behaviors of ITO resistors described in [18], all resistors display the aging behavior in the two regions: First, a relatively fast degradation followed by a more gradual phase. Hence, we model a multiplicative change of the initial conductance, i.e.,

$$g(t) = g_0 \cdot \mathcal{A}_{\boldsymbol{\omega}}(t), \tag{4.1}$$

and refer to the function $\mathcal{A}_{\boldsymbol{\omega}}(t)$ as the aging curve parameterized by the vector $\boldsymbol{\omega}$. Several functional forms can describe $\mathcal{A}_{\boldsymbol{\omega}}(t)$ such as a double linear model suggested in [43] or an exponential behavior which we employ in this

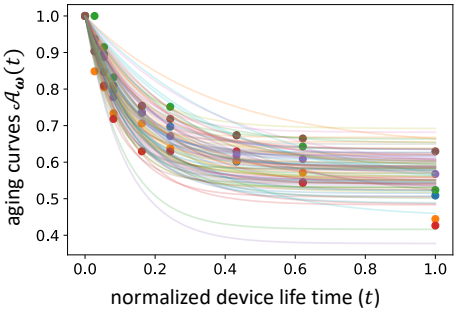


Figure 4.2: Curves from the aging model with sampled $\boldsymbol{\omega} \sim p_\omega(\boldsymbol{\omega})$. The dots denote conductance measurements of printed resistors normalized by their initial conductance g_0. Sourced from [51].

work. We thus choose the following functional form

$$\mathcal{A}_{\boldsymbol{\omega}}(t) = \omega_1 \cdot \mathrm{e}^{-\omega_2 \cdot t} - \omega_1 + 1,$$

with the fitting coefficients $\boldsymbol{\omega} = [\omega_1, \omega_2]^\top$. Note that $\mathcal{A}_{\boldsymbol{\omega}}(0) = 1$, such that $g(0) = g_0$ at $t = 0$.

Variational aging model. Since different resistors, **even with the same initial conductance**, display **different aging behaviors** over time, a variational model of the aging behaviors is required. For this purpose, we model the distributions of the fitting coefficients $p_\omega(\boldsymbol{\omega})$. To ensure a plausible functional form (i.e., monotonically decreasing) with respect to the observed behavior, ω_1 and ω_2 need to be positive. This can be achieved by modeling their distribution (either jointly or independently) using log-normal distributions. In Figure 4.2, we visualized some generated aging curves by drawing samples $\boldsymbol{\omega} \sim p_\omega(\boldsymbol{\omega})$. Note that also other distributions for positive random variables (e.g., gamma distribution) could be used, depending on the quality of the fit.

In the context of pNCs, the resistor crossbar is significantly impacted by the aging of the printed resistors. Consequently, we further investigate the influence of resistor aging on the outcomes of crossbar. As described in Equation (2.4), the weights w embodied by resistor crossbars are determined by the

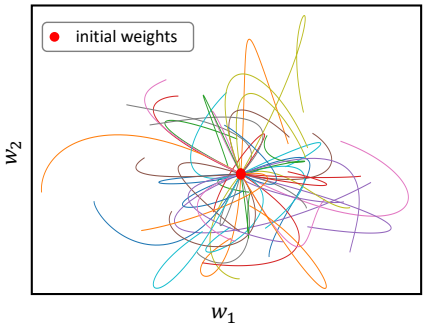


Figure 4.3: Exemplary aging trajectories of the weights with aging conductances. Sourced from [52].

ratios of the conductances g. Thus, non-proportional changes in conductance result in deviations of the weights from their initial values. Figure 4.3 depicts multiple trajectories of weight changes due to aging in a two-dimensional scenario. Since the aging behavior is stochastic, the aging trajectory of the weights are also stochastic. To consider the aging effects of weights into the design process of pNCs and thus achieve robust circuits, we subsequently proposed an aging-aware training approach.

4.1.2 Aging-Aware Training

As introduced in Chapter 2.2.2, the training objective of pNCs without considering aging is generally defined as the cross-entropy loss to improve the classification accuracy, denoted by

$$\underset{\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}}\ \mathcal{L}\left(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q}\right).$$

We refer to the trainings of pNCs with this objective as the *nominal training*. However, as mentioned before, this formulation only optimizes for the (surrogate) conductance values immediately after fabrication, i.e., $\boldsymbol{\theta}_0$. To account for the changes of the (surrogate) conductances over time, the whole trajectory of $\boldsymbol{\theta}(t)$ over the lifetime has to be considered. To achieve this, we integrate the stochastic aging behaviors into the loss function over the lifetime, leading to

the aging-aware training objective

$$\underset{\boldsymbol{\theta}(t)}{\text{minimize}} \int_{t=0}^{1} \mathcal{L}\left(\mathcal{D}, \boldsymbol{\theta}(t), \boldsymbol{q}\right) \mathrm{d}t,$$

where $\boldsymbol{\theta}(t) = \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t)$ represents the element-wise product of surrogate conductances $\boldsymbol{\theta}_0$ with their aging curves $\boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t) = [\mathcal{A}_{\boldsymbol{\omega}_1}(t),\ \mathcal{A}_{\boldsymbol{\omega}_2}(t), \cdots]^\top$. Here, $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots$ are sampled values from $p_\omega(\boldsymbol{\omega})$. Then, for the sampled $\boldsymbol{\omega}$, the aging-aware training objective is given by

$$\underset{\boldsymbol{\theta}_0}{\text{minimize}} \int_{t=0}^{1} \mathcal{L}\left(\mathcal{D}, \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t), \boldsymbol{q}\right) \mathrm{d}t. \tag{4.2}$$

To additionally account for the variations in the aging curves due to $p_\omega(\boldsymbol{\omega})$, we also have to minimize for the expected loss with respect to $p_\omega(\boldsymbol{\omega})$, i.e.,

$$\underset{\boldsymbol{\theta}_0}{\text{minimize}}\ \mathbb{E}_{p_\omega(\boldsymbol{\omega})} \left\{ \int_{t=0}^{1} \mathcal{L}\left(\mathcal{D}, \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t), \boldsymbol{q}\right) \mathrm{d}t \right\}. \tag{4.3}$$

In the following, we refer to *aging-aware training* when using this training objective. To apply gradient-based optimization for this objective requires the calculation of Equation (4.3). For this, we first reformulate the gradient of Equation (4.3) using the definition of the expected value and Leibniz rule [16]:

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}_0} \int_{\boldsymbol{\omega}} \int_{t=0}^{1} \mathcal{L}\left\{\mathcal{D}, \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t), \boldsymbol{q}\right\} \mathrm{d}t\, p_\omega(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} \\
=&\int_{\boldsymbol{\omega}} \int_{t=0}^{1} \nabla_{\boldsymbol{\theta}_0} \left(\mathcal{L}\left\{\mathcal{D}, \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t), \boldsymbol{q}\right\} p_\omega(\boldsymbol{\omega}) \right) \mathrm{d}t\, \mathrm{d}\boldsymbol{\omega}.
\end{aligned}$$

Due to the independence of $\boldsymbol{\theta}_0$ and $p_\omega(\boldsymbol{\omega})$, we can simplify the expression to

$$\begin{aligned}
&\int_{\boldsymbol{\omega}} \int_{t=0}^{1} \nabla_{\boldsymbol{\theta}_0} \mathcal{L}\left(\mathcal{D}, \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t), \boldsymbol{q}\right) \mathrm{d}t\, p_\omega(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} \\
=&\int_{\boldsymbol{\omega}} \int_{t=0}^{1} \nabla_{\boldsymbol{\theta}_0} \mathcal{L}\left(\mathcal{D}, \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t), \boldsymbol{q}\right) \mathrm{d}t\, p_\omega(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega} \\
=&\mathbb{E}_{p_\omega(\boldsymbol{\omega})} \left\{ \int_{t=0}^{1} \nabla_{\boldsymbol{\theta}_0} \mathcal{L}\left(\mathcal{D}, \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t), \boldsymbol{q}\right) \mathrm{d}t \right\}.
\end{aligned}$$

Unfortunately, the integration of the loss function $\mathcal{L}(\cdot)$ cannot be calculated in closed form, and thus the gradient can not be back propagated from the loss

function to learnable parameters. According to the law of large numbers [10], the expected value of a function can be estimated by the average of the results obtained from multiple samples. Thus, we aim to employ MC estimation for the integration. We first express the integral over t as an expected value with respect to a uniform distribution $t \sim \mathcal{U}[0,1]$ with $p(t) = 1$, then, we can formulate Equation (4.3) as

$$\mathbb{E}_{p_\omega(\boldsymbol{\omega})} \left\{ \int_{t=0}^{1} \nabla_{\boldsymbol{\theta}_0} \mathcal{L}\left(\mathcal{D}, \boldsymbol{\theta}_0 \odot \boldsymbol{\mathcal{A}}_{\boldsymbol{\omega}}(t), \boldsymbol{q}\right) p(t) \, \mathrm{d}t \right\}$$

which can be seen as

$$\mathbb{E}_{p_\omega(\boldsymbol{\omega})} \left\{ \mathbb{E}_{p_t(t)} \left\{ \nabla_{\boldsymbol{\theta}_0} \mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\omega}, t) \right\} \right\}.$$

We then draw multiple samples $\boldsymbol{\omega} \sim p_\omega(\boldsymbol{\omega})$ and $t \sim p_t(t)$ using MC to estimate Equation (4.3) through

$$\frac{1}{N^{\boldsymbol{\omega}}} \frac{1}{N^t} \sum_{\boldsymbol{\omega}'} \sum_{t'} \nabla_{\boldsymbol{\theta}_0} \mathcal{L}(\boldsymbol{\theta}_0, \boldsymbol{\omega}', t') \quad \text{with} \quad \begin{array}{l} t' \sim \mathcal{U}[0,1], \\ \boldsymbol{\omega}' \sim p_\omega(\boldsymbol{\omega}), \end{array}$$

where $N^{\boldsymbol{\omega}}$ is the number of samples $\boldsymbol{\omega}'$ drawn from $p(\boldsymbol{\omega})$ and N^t is the number of samples t' drawn from $\mathcal{U}[0,1]$ to approximate the integral over t.

Figure 4.4 visualizes the idea of aging-aware training. It shows an exemplary aging trajectory of weight $\boldsymbol{w}(t)$ corresponding to the printed resistors in time t. We can see from the right side that, compared to the blue curve from nominal training, the red curve from aging-aware training tries to locate the whole curve within a lower loss area, even though it may have a higher initial loss.

4.1.3 Experiment

To evaluate the effectiveness of the aging-aware training of pNCs, we implemented the proposed training approach with PyTorch [35]. As the functionality of the printed neuromorphic hardware has been validated in [42, 49] and the contribution of this work is primarily at algorithmic level, the experiment is conducted at simulation level based on the pPDKs [37].

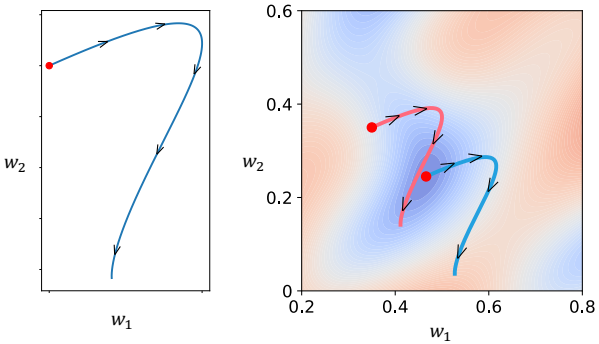


Figure 4.4: Exemplary aging trajectory of given weight $\boldsymbol{w}(t)$ (left) and the optima of different objective functions (right). The red dots indicate the initial (non-aged) weights and the arrows represent the change in weights due to aging. The background contour in the right figure exemplifies a loss function $\mathcal{L}(\cdot)$, where red and blue denote regions of higher and lower loss, respectively. The blue curve is the result of nominal training, while the red curve is the result of aging-aware training. Sourced from [51].

Datasets. We conduct experiments on the 13 benchmark datasets, which are recommended by related surveys [15, 38]. They are also employed in other SOTA studies on pNCs [19, 49] and aligned with the complexity of the application domains of PE. Specifically, datasets with a modest number of inputs and outputs (generally fewer than ten) are more appropriate, due to the large feature size and low integration density of PE. The detailed information of the datasets are listed in Table 4.1. Additionally, we normalized the inputs to $[0, 1]$ to simulate the electrical signals from sensors. Then, we split each dataset into training (60%), validation (20%), and test (20%) sets.

Experiment setup. We use a consistent topology (#*input*-3-#*output*) for all pNCs on each dataset. In addition, we selected $N^{\omega} = N^{T} = 50$ samples for Monte-Carlo integration. During training, we employ full-batch training with the Adam [23] optimizer (in default parameterization) to update parameters in pNCs. To prevent overfitting, we calculated the loss on validation set for early-stopping [36] after each parameter update. We start with an initial learning

Table 4.1: Information of the datasets employed in this thesis.

Index	Dataset	#Input	#Output	#Instance	Description	Domain	Source
1	Acute Inflammation	6	2	120	Classifying acute inflammations of the urinary bladder and acute nephritises from patient symptoms and temperature data	medical care	[9]
2	Balance Scale	4	3	625	Classifying balance scale tipping direction (right, left, or balanced) from weight and distance measurements on both sides of the scale	psychology	[41]
3	Breast Cancer Wisconsin	9	2	699	Classifying clinical case outcomes from periodic samples grouped chronologically	medical care	[28]
4	Cardiotocography	21	3	2126	Classifying fetal state and morphologic patterns from cardiotocogram data	obstetrics	[3]
5	Energy Efficiency (y_1)	8	3	768	Classifying heating load from building characteristics using simulations in Ecotect	energy analysis	[47]
6	Energy Efficiency (y_2)	8	3	768	Classifying cooling load from building characteristics using simulations in Ecotect	energy analysis	[47]

Continued on next page

Continued from previous page

7	Iris	4	3	150	Classifying iris plant species from morphological measurements	botanic	[2]
8	Mammographic Mass	5	2	961	Classifying the severity (benign or malignant) of mammographic mass lesions from BI-RADS attributes and patient age	medical care	[12]
9	Pendigits	16	10	7494	Classifying handwritten digits from writing pressure	vision	[1]
10	Seeds	7	3	210	Classifying wheat varieties (Kama, Rosa, and Canadian) from kernels using X-ray imaging data	botanic	[7]
11	Tic-Tac-Toe Endgame	9	2	958	Classifying the game results from the state of the game board configuration	gaming	[30]
12	Vertebral Column (2 cl.)	6	2	310	Classifying healthiness of orthopedic patients from biomechanical information of bones	medical care	[4]
13	Vertebral Column (3 cl.)	6	3	310	Classifying orthopedic patients from biomechanical information of bones	medical care	[4]

rate of 0.1 and halve it after a patience (updates without improvement) of 100-epochs on the validation set. Additionally, the training process is stopped, when the learning rate decreases below 10^{-4}.

Regarding the nonlinear circuits, we employ the shared $\boldsymbol{q}^{\mathrm{N}}$ and shared $\boldsymbol{q}^{\mathrm{A}}$ across an entire pNC, rather than allowing each neuron to have independent $\boldsymbol{q}^{\mathrm{N}}$ and $\boldsymbol{q}^{\mathrm{A}}$. Although the latter strategy offers larger search space for optimization, it empirically yields worse results [46, 53].

Note that, more details about the implementation can be found in the GitHub repository[1]. Moreover, this experimental setup will be the *default setup* for all experiments included in this thesis. The description of experiment setups in following sections will only emphasize the differences to this setup.

Baseline. As the baseline, we report the performance of *random guess*, i.e., to always predict the most frequent class from the combined training and validation set. Hence, if the pNC gets worse than this baseline, there is no benefit to considering the output of the pNC anymore.

Result. After training, we choose pNCs based on the best validation loss, as it would be the one selected for fabrication. We evaluate the results of the test set. The pNCs are not only evaluated within the normalized time interval $[0,1]$ (training interval of 37 days), but also extended to $[0,10]$, which represents an extrapolation to approximately one year. As evaluation metrics, we report the mean and standard deviation of the classification accuracy with respect to the stochastic aging behaviors. It can be seen from Figure 4.5, nominal training produces pNCs that may perform better at $t = 0$, while aging-aware training has a higher expectation of accuracy throughout the lifetime, and it is more robust against the expected aging behaviors based on our aging model. Moreover, since the conductance decay exponentially over time, only slight changes in the conductances in the interval $[1,10]$ can be observed. To summarize the overall improvement, we average the accuracy cross all datasets and calculate the improvement of averaged aging-aware training (relative to nominal training) across all datasets. The result reflects an overall 35.8% improvement in the expected accuracy of aging-aware training over nominal training.

[1] `https://github.com/Neuromophic/Aging-aware-training`.

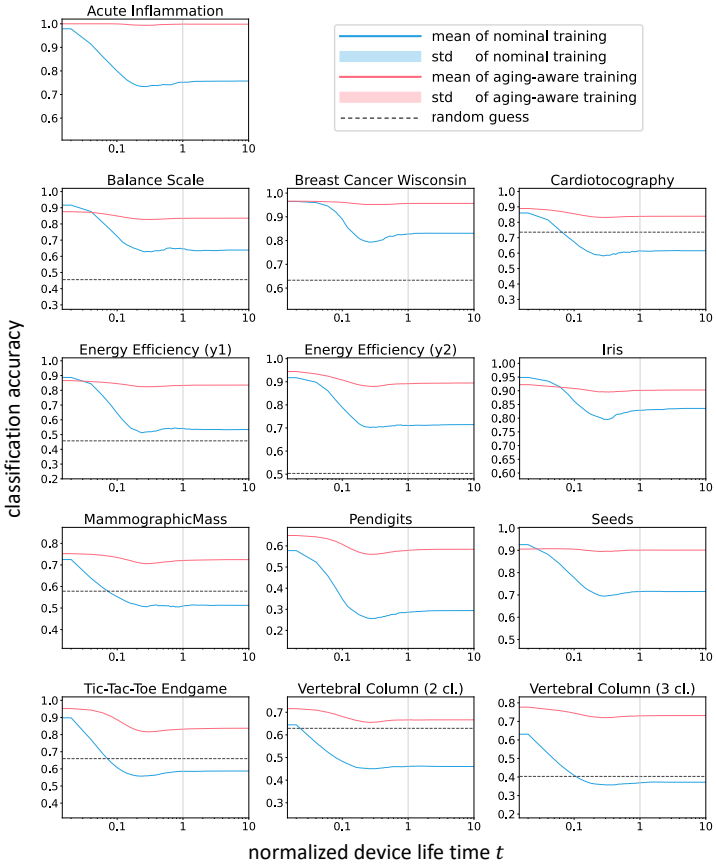


Figure 4.5: Classification accuracy of pNCs from nominal and aging-aware training on test set. The red lines and areas represent the accuracy and standard deviation of aging-aware training, while the blues represent that of nominal training. The horizontal black dot lines indicate the random guess. Charts without black dot line mean that the accuracy of random guesses are lower than the range of charts. The gray vertical lines separate the extrapolation region from the training region in terms of time. Sourced from [51].

Notably, for some datasets, the results of aging-aware training exceed those of nominal training even at t_0, e.g., on *Cardiotocography*, *Energy Efficiency (y_2)*, *Pendigits*, and *Vertebral Column*. This is possibly due to favorable optimization dynamics through sampling. For example, since aging-aware training samples and calculates gradients for several sets of parameters in the vicinity of the current solution, it may have an easier time escaping local minima.

4.1.4 Discussion

Due to the vulnerability of organic materials to environmental factors, enhancing the robustness of pNCs against device aging, particularly resistor aging within crossbars, is critical. To this end, we experimentally measured the aging characteristics of PEDOT:PSS and developed a stochastic aging model for printed PEDOT:PSS resistors. Subsequently, we proposed a framework capable of considering resistor aging into the training framework of pNCs and demonstrated its efficacy through experiments. The training framework is independent of the specific choice of the functional form of the aging behaviors described in Chapter 4.1.1. Thus, one can easily modify the functional form $\mathcal{A}_{\boldsymbol{\omega}}(t)$ without any changing the aging-aware training. Apart from algorithmic level optimization, another aspect for mitigating aging effect is to develop effective and low cost encapsulation and packaging technologies for PE.

4.2 Highly Dependable Circuit Design

In addition to aging, pNCs also subject to other influence factors, which can also substantially impact the functionality of pNCs: Firstly, input variations caused by uncertainty in the sensing process [14] may cause faulty processing. Secondly, variations in the printing process due to non-uniformly printed material (device geometry) as well as variations in ink compositions and substrates can perturb the fabricated component values from the design ones [22]. Therefore, to ensure the highly reliable functioning of pNCs, it is essential to include all these factors during into training. Consequently, this section introduces a dependability-aware training to improve the circuit reliability. Finally, an ablation study is conducted to analyze the mechanisms of factors affecting

the pNCs and their joint effect.

4.2.1 Modeling of Impact Factors

Sensing uncertainty. Measurement uncertainty is a quantitative assessment that provides an estimate of the potential range of the true value of a physical quantity with a specific level of confidence [11]. The uncertainty arises due to various processes during the measurement, including the intrinsic error of the measuring instrument, the coupling between the measuring instrument and the system being measured, changes in measurement conditions, and the imperfections in the calibration procedure. Therefore, it is imperative to respect the measurement uncertainty during the design of a robust and reliable printed neuromorphic circuit. Moreover, since the pNC works directly with sensors in analog domain instead of digital, it is more sensitive to sensing uncertainties.

As measurement uncertainty is the cumulative outcome of various stochastic processes mentioned above, it is often modeled by a Gaussian distribution in the signal processing community, in accordance with the central limit theorem [31] and the principle of maximum entropy [21]. In this work, we model the noisy input signals by a Gaussian distribution centered around the original input $\mathbf{X}$ with standard deviation being σ. Formally, this is expressed as

$$\mathbf{X}^{\text{noisy}} \sim \mathcal{N}(\mathbf{X}, \sigma).$$

Here, σ can also be explained as the uncertainty of the measurement. Essentially, this is an approach of data augmentation [48].

Printing variation. In the manufacturing of PE, the desired component values, like conductances, can generally not be printed exactly. This variation primarily arises from the constrained print resolution, which stems from the physical properties of the functional inks and limitations of the printing technology. The printing resolution is principally determined by, e.g., the volume of the smallest printable volume of the droplets [32]. Consequently, by assuming that, the printing variation is determined by the geometric variation of the printing shape which varies within one printing pixel, the printing variation is often modeled as a uniformly distributed stochastic variable within the

minimum resolution, i.e.,

$$\boldsymbol{\theta}^{\text{PV}} \sim \mathcal{U}[(1-\text{e})\boldsymbol{\theta}^{\text{ideal}}, (1+\text{e})\boldsymbol{\theta}^{\text{ideal}}],$$
$$\boldsymbol{q}^{\text{PV}} \sim \mathcal{U}[(1-\text{e})\boldsymbol{q}^{\text{ideal}}, (1+\text{e})\boldsymbol{q}^{\text{ideal}}].$$

Here, the value for e is selected based on the specific printing technology to accommodate printing variations. However, this modeling does not support gradient-based training approach, because to consider the expected loss with respect to the stochastic parameters $\boldsymbol{\theta}^{\text{PV}}$ and $\boldsymbol{q}^{\text{PV}}$, the loss function will integrate the parameters $\boldsymbol{\theta}^{\text{PV}}$ and $\boldsymbol{q}^{\text{PV}}$ out. Specifically, the expression of the expected loss is given by

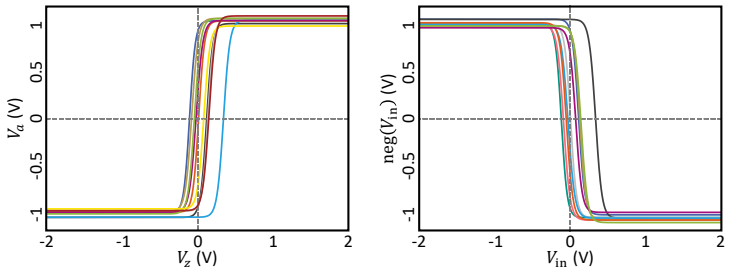
$$\int_{\boldsymbol{\theta}^{\text{PV}}} \int_{\boldsymbol{q}^{\text{PV}}} L(\boldsymbol{\theta}^{\text{PV}}, \boldsymbol{q}^{\text{PV}}, \mathcal{D}) p(\boldsymbol{\theta}^{\text{PV}}) p(\boldsymbol{q}^{\text{PV}}) \, \mathrm{d}\boldsymbol{\theta}^{\text{PV}} \, \mathrm{d}\boldsymbol{q}^{\text{PV}},$$

where the parameters $\boldsymbol{\theta}^{\text{PV}}$ and $\boldsymbol{q}^{\text{PV}}$ "*integrated out*" from the equation after the integration over them, and thus hinders the backpropagation. To facilitate the training process for the learnable parameter $\boldsymbol{\theta}^{\text{ideal}}$, we utilized the *reparameterization trick* [24], i.e., we introduce stochastic variables $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}$ and $\boldsymbol{\varepsilon}_{\boldsymbol{q}}$ to independently parameterize and extract $\boldsymbol{\theta}^{\text{ideal}}$ and $\boldsymbol{q}^{\text{ideal}}$ by

$$\boldsymbol{\theta}^{\text{PV}} = \boldsymbol{\varepsilon}_{\boldsymbol{\theta}} \odot \boldsymbol{\theta}^{\text{ideal}} \text{ and } \boldsymbol{q}^{\text{PV}} = \boldsymbol{\varepsilon}_{\boldsymbol{q}} \odot \boldsymbol{q}^{\text{ideal}},$$

where $\boldsymbol{\theta}^{\text{PV}}$ models the manufactured conductance with printing variation and $\boldsymbol{\varepsilon}$ is a stochastic variable denoting the printing variation with each element in $\boldsymbol{\varepsilon}$ following a uniform distribution $\mathcal{U}[1-\text{e}, 1+\text{e}]$.

The impact of printing variation on weights within the resistor crossbar is rather intuitive: as the conductances deviate from ideal values, their corresponding weights will also deviate. However, its impact on nonlinear circuits is intricate. We visualize the impact of the variation in nonlinear circuits in Figure 4.6. The left and right figures exemplify some varied characteristic curves of ptanh circuit and printed negation circuit under $\text{e} = 10\%$ variation. It can be seen, the printing variation perturbs the nonlinear functions from the designated ones, which justifies the consideration of printing variations in the nonlinear circuit primitives.



(a) Printed tanh-like circuit with variation (b) Negative weight circuit with variation

Figure 4.6: Printing variation perturbs the characteristic curves of nonlinear circuits from the ideal ones: (a) exemplary characteristic curves of ptanh circuit with 10% printing variation, (b) exemplary characteristic curves of printed negation circuit with 10% printing variation. Sourced from [52].

4.2.2 Dependability-Aware Training

To include the aforementioned influence into the design of dependable pNCs, we propose a framework that is capable of incorporating all relevant factors into the training of pNCs, as illustrated in Figure 4.7.

For each resistor crossbar, the learnable surrogate conductance $\boldsymbol{\theta}^{\text{ideal}}$ is processed by a straight-through estimator to maintain its printability. Once converted to a printable value, the stochastic variable $\boldsymbol{\varepsilon}$ is element-wise multiplied to simulate printing variations for each conductance. $\boldsymbol{\theta}^{\text{PV}}$ summarizes all the corresponding surrogate conductances suffering from printing variation. Subsequently, the aging decay $\mathcal{A}_{\boldsymbol{\omega}}(t)$ is multiplied to reflect the aging behaviors of conductances already affected by printing variations, denoted by $\boldsymbol{\theta}^{\text{aged}}$. Finally, the resulting conductances are transformed to the corresponding weights in the weighted-sum operation.

Regarding the nonlinear circuits, we do not consider their parameters learnable $\boldsymbol{q}^{\text{ideal}}$ but rather fixed to certain design values. Details on how these parameters can be learned can be referred to Chapter 3.1. Nevertheless, we still account for their printing variations and aging behaviors during training. The nonlinear circuits comprise two types of components, namely resistors, charac-

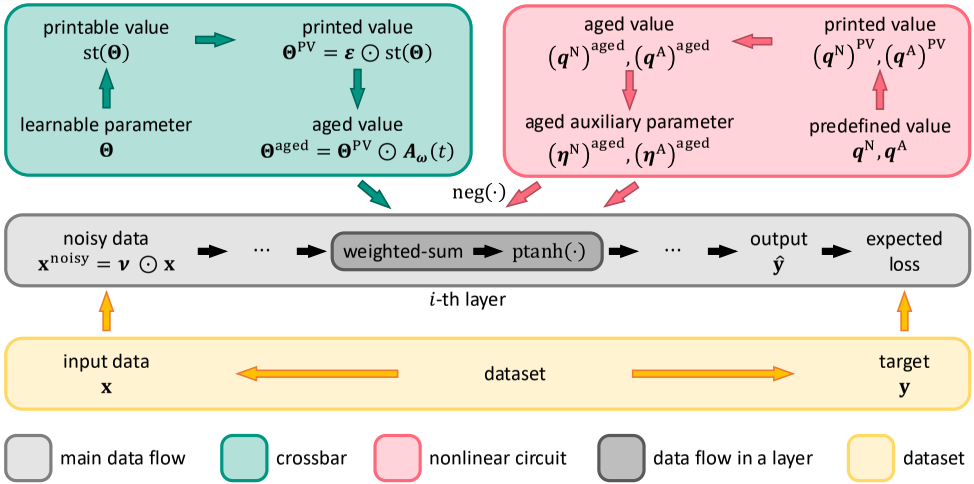


Figure 4.7: Data flow in the dependability-aware training for pNCs. The gray part indicates the main data flow, the green box contains the data processing for weighted-sum resistor crossbar, the red box indicates the processing of nonlinear circuits, while the bottom yellow box refers to the target dataset. Sourced from [51].

terized by their conductances, and transistors, characterized by their geometric features. We consider both printing variations and aging effects for the printed resistors, but for the transistors, we only consider printing variation on W and L, as aging behavior of inorganic materials not as significant as organic materials. After the aged parameters for the nonlinear circuits, i.e., $(\boldsymbol{q}^{\mathrm{A}})^{\mathrm{aged}}$ and $(\boldsymbol{q}^{\mathrm{N}})^{\mathrm{aged}}$, have been calculated, they can be mapped to the auxiliary parameters $(\boldsymbol{\eta}^{\mathrm{A}})^{\mathrm{aged}}$ as well as $(\boldsymbol{\eta}^{\mathrm{N}})^{\mathrm{aged}}$ via differentiable surrogate nonlinear circuit models. These auxiliary parameters can then be utilized to construct negation functions and ptanhs functions, which will be integrated into pNCs for weighted-sum and activation functions, respectively.

With consideration of the stochasticity in the data flow, the value of the loss function $L(\mathcal{D}^{\mathrm{noisy}}, (\boldsymbol{\theta})^{\mathrm{aged}}, (\boldsymbol{q})^{\mathrm{aged}})$ is no longer a deterministic value, but rather a stochastic distribution with respect to $\mathbf{X}^{\mathrm{noisy}}$, $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}$, $\boldsymbol{\varepsilon}_{\boldsymbol{q}}$, and $\boldsymbol{\omega}$. The loss can be explicitly denoted as

$$L(\mathbf{X}^{\mathrm{noisy}}, \mathbf{Y}, \boldsymbol{\theta}, \boldsymbol{q}, \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}, \boldsymbol{\varepsilon}_{\boldsymbol{q}}, \boldsymbol{\omega}, t).$$

In general, gradient-based numerical optimizers require the loss to be expressed as a deterministic scalar value instead of a function or distribution. To solve this problem, we adopt the expectation to assess the value of the loss and obtain the **dependability-aware training** objective function:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{q}) = \mathbb{E}_{\mathbf{X}^{\mathrm{noisy}}} \left\{ \mathbb{E}_{\boldsymbol{\varepsilon}_{\boldsymbol{q}}} \left\{ \mathbb{E}_{\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}} \left\{ \mathbb{E}_{\boldsymbol{\omega}} \left\{ \int_0^1 L(\mathbf{X}^{\mathrm{noisy}}, \mathbf{Y}, \boldsymbol{\theta}, \boldsymbol{q}, \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}, \boldsymbol{\varepsilon}_{\boldsymbol{q}}, \boldsymbol{\omega}, t) \, \mathrm{d}t \right\} \right\} \right\} \right\}.$$

For simplify, the lifetime was normalized to $t \in [0,1]$. Additionally, the integral over the lifetime can be expressed as a mathematically equivalent expected value with respect to a uniform distribution $p_t(t) = \mathcal{U}[0,1]$, and the stochastic input $\mathbf{X}^{\mathrm{noisy}}$ can be reparameterized as the multiplication of the original $\mathbf{X}$ with a stochastic variable $\boldsymbol{\nu}$, with $\boldsymbol{\nu} \sim \mathcal{N}(1, \sigma)$. With this approach, all variables can be consistently treated as the expectation of the perspective variables. Consequently, $\boldsymbol{\nu}$, $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}$, $\boldsymbol{\varepsilon}_{\boldsymbol{q}}$, $\boldsymbol{\omega}$, and t can be summarized by one stochastic variable

$$\boldsymbol{\gamma} := [\boldsymbol{\nu}, \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}, \boldsymbol{\varepsilon}_{\boldsymbol{q}}, \boldsymbol{\omega}, t]$$

with density

$$p_\gamma(\boldsymbol{\gamma}) = p_\nu(\boldsymbol{\nu}) \cdot p_\varepsilon(\boldsymbol{\varepsilon_\theta}) \cdot p_\varepsilon(\boldsymbol{\varepsilon_q}) \cdot p_\omega(\boldsymbol{\omega}) \cdot p_t(t).$$

Thus, the final objective function of the dependability-aware training can be written as

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) = \int_{\boldsymbol{\gamma}} L(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}, \boldsymbol{\gamma}) p_\gamma(\boldsymbol{\gamma}) \, \mathrm{d}\boldsymbol{\gamma}. \tag{4.4}$$

Unfortunately, due to the complexity of $\mathcal{L}(\cdot)$, usually no analytical solution for Equation (4.4) can be found. We thus employ MC method to obtain an approximation of Equation (4.4), namely,

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\gamma}}\{L(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}, \boldsymbol{\gamma})\} &= \int_{\boldsymbol{\gamma}} L(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}, \boldsymbol{\gamma}) p_\gamma(\boldsymbol{\gamma}) \, \mathrm{d}\boldsymbol{\gamma} \\ &\approx \frac{1}{N_\gamma} \sum_{\boldsymbol{\gamma'}} L(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}, \boldsymbol{\gamma'}), \end{aligned} \tag{4.5}$$

where $\boldsymbol{\gamma'} = [\boldsymbol{\nu'}, \boldsymbol{\varepsilon'_\theta}, \boldsymbol{\varepsilon'_q}, \boldsymbol{\omega'}, t']$ describes a set of N_γ samples drawn from the distribution $p_\gamma(\boldsymbol{\gamma})$. With this approach, each time the loss need to be computed, Equation (4.5) can be employed to obtain an estimate by drawing N_γ samples from $p_\gamma(\boldsymbol{\gamma})$. Similar to the objective, also gradients for Equation (4.4) can be obtained via MC gradient estimation via

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) \approx \frac{1}{N_{\boldsymbol{\gamma}}} \sum_{\gamma'} \nabla_{\boldsymbol{\theta}} L(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}, \boldsymbol{\gamma'}).$$

These estimated gradients can then be used by common gradient-based optimization algorithms such as SGD [6] or Adam [23].

4.2.3 Experiment

To evaluate the effectiveness of the dependability-aware training of pNCs, we implemented the proposed training approach with PyTorch [35] and conduct experiments on the 13 benchmark datasets as described in Chapter 4.1.3.

Experiment setup. We choose $\mathrm{e} = 10\%$ to reflect the printing variation, as typical printing resolutions range from 20 µm to 100 µm [22], whereas the

Table 4.2: The mean and standard deviation of classification accuracy with respect to stochastic variable $\boldsymbol{\gamma}$ on each dataset for each experiment. Sourced from [52].

Notation		**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**	**Exp. 5**	**Exp. 6**	**Exp. 7**	**Exp. 8**
Awareness		$\overline{\text{AG}}, \overline{\text{PV}}, \overline{\text{SU}}$	$\overline{\text{AG}}, \overline{\text{PV}}, \text{SU}$	$\overline{\text{AG}}, \text{PV}, \overline{\text{SU}}$	$\overline{\text{AG}}, \text{PV}, \text{SU}$	$\text{AG}, \overline{\text{PV}}, \overline{\text{SU}}$	$\text{AG}, \overline{\text{PV}}, \text{SU}$	$\text{AG}, \text{PV}, \overline{\text{SU}}$	$\text{AG}, \text{PV}, \text{SU}$
Dataset	1	0.92 ± 0.10	0.92 ± 0.08	0.98 ± 0.06	0.98 ± 0.06	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.00
	2	0.63 ± 0.27	0.63 ± 0.28	0.75 ± 0.17	0.76 ± 0.18	0.81 ± 0.03	0.80 ± 0.03	0.81 ± 0.03	0.81 ± 0.02
	3	0.88 ± 0.12	0.89 ± 0.13	0.97 ± 0.01	0.96 ± 0.01	0.97 ± 0.00	0.97 ± 0.00	0.96 ± 0.01	0.96 ± 0.00
	4	0.75 ± 0.12	0.70 ± 0.19	0.78 ± 0.16	0.81 ± 0.09	0.84 ± 0.03	0.84 ± 0.03	0.85 ± 0.05	0.85 ± 0.03
	5	0.69 ± 0.22	0.64 ± 0.25	0.83 ± 0.13	0.84 ± 0.11	0.91 ± 0.04	0.90 ± 0.06	0.90 ± 0.05	0.92 ± 0.03
	6	0.74 ± 0.13	0.75 ± 0.09	0.82 ± 0.09	0.83 ± 0.06	0.84 ± 0.05	0.85 ± 0.04	0.85 ± 0.04	0.86 ± 0.03
	7	0.84 ± 0.08	0.85 ± 0.11	0.87 ± 0.11	0.87 ± 0.11	0.92 ± 0.05	0.94 ± 0.03	0.93 ± 0.06	0.92 ± 0.05
	8	0.54 ± 0.18	0.56 ± 0.14	0.66 ± 0.14	0.68 ± 0.13	0.72 ± 0.10	0.74 ± 0.09	0.75 ± 0.06	0.75 ± 0.06
	9	0.10 ± 0.10	0.10 ± 0.10	0.37 ± 0.10	0.51 ± 0.07	0.44 ± 0.06	0.57 ± 0.05	0.48 ± 0.05	0.54 ± 0.05
	10	0.76 ± 0.13	0.74 ± 0.11	0.78 ± 0.12	0.82 ± 0.07	0.82 ± 0.04	0.86 ± 0.03	0.85 ± 0.05	0.86 ± 0.03
	11	0.59 ± 0.19	0.66 ± 0.15	0.80 ± 0.08	0.75 ± 0.10	0.76 ± 0.06	0.79 ± 0.09	0.73 ± 0.09	0.81 ± 0.07
	12	0.57 ± 0.13	0.65 ± 0.09	0.64 ± 0.10	0.76 ± 0.07	0.61 ± 0.10	0.73 ± 0.07	0.61 ± 0.07	0.77 ± 0.06
	13	0.50 ± 0.15	0.60 ± 0.12	0.59 ± 0.13	0.71 ± 0.08	0.63 ± 0.11	0.76 ± 0.06	0.63 ± 0.09	0.75 ± 0.07
Average		$\mathbf{0.66 \pm 0.15}$	$\mathbf{0.67 \pm 0.14}$	$\mathbf{0.76 \pm 0.11}$	$\mathbf{0.79 \pm 0.09}$	$\mathbf{0.79 \pm 0.05}$	$\mathbf{0.83 \pm 0.04}$	$\mathbf{0.80 \pm 0.05}$	$\mathbf{0.83 \pm 0.04}$

Table 4.3: Independent effects of aging (AG), printing variation (PV), and sensing uncertainty (SU) in the dependability-aware training. Sourced from [52].

Awareness	Averaged experiments	Averaged accuracy	Improvement	
			(mean) Accuracy	Robustness (std)
AG-aware	Exp. 5, 6, 7, 8	0.811 ± 0.046	13.03%	61.58%
AG-unaware	Exp. 1, 2, 3, 4	0.718 ± 0.120		
PV-aware	Exp. 3, 4, 7, 8	0.794 ± 0.071	7.99%	26.46%
PV-unaware	Exp. 1, 2, 5, 6	0.735 ± 0.096		
SU-aware	Exp. 2, 4, 6, 8	0.779 ± 0.077	3.89%	13.26%
SU-unaware	Exp. 1, 3, 5, 7	0.750 ± 0.089		

component feature sizes in printed neuromorphic circuits are on the order of 1 mm [49]. Therefore, $\pm 10\%$ can be seen as a reasonable estimate. Moreover, we take $\sigma = 0.1$ for $\boldsymbol{\nu}$ to simulate sensing uncertainty of the inputs $\mathbf{X}$. Other training configurations are also the same by default as in Chapter 4.1.3.

Ablation study. In this work, multiple factors that could potentially impact the results are considered. To investigate the effects of these factors independently and jointly, we conduct an ablation study. Specifically, we conducted experiments on all possible combinations of the three factors (8 combinations in total) to assess their combinatorial effects. To facilitate the identification of individual experiments, they are numbered from **Exp. 1** to **Exp. 8**. Furthermore, the terms "aging behavior", "printing variation", and "sensing uncertainty" are abbreviated to "AG", "PV", and "SU", respectively (see Table 4.2 for details). The abbreviation with an additional over line indicates that, the experiment is unaware of the corresponding factor, e.g., $\overline{\text{AG}}$ refers to aging-unaware training. Hence, the specific pNCs are only trained with consideration of the specific factors, while for testing, all effects, i.e., aging, printing variation, and sensing uncertainty are included.

Result. After training, we choose pNCs based on the best validation loss, as it would be the one selected for fabrication. We evaluate the resulted pNCs on the test set base on $N_\gamma = 5000$ samples.

In this work, **dependability** is conceptualized to reflect two aspects of the performance of the pNCs, i.e., **accuracy** and **robustness** against stochastic variations. Here, the accuracy and robustness are indicated by the **mean accuracy** and the **standard deviation** of accuracy with respect to the stochastic variable $\boldsymbol{\gamma}$. Thus, the metrics are calculated on each dataset and reported in Table 4.3. As a summary, the averaged values of all the dataset for each experiment are also reported.

Through the comparison of Exp. 8 and Exp. 1, we conclude that, with consideration of all three factors in the training process, a substantial 27% improvement in accuracy and a 74% improvement in robustness.

Independent analysis. To analyze the impact of a certain factor independently of other factors, we divided the eight experiments into two groups (e.g., experiments with, and without AG-aware training) and average the performance respectively. Table 4.3 summarizes the analysis of each factor.

It is evident from Table 4.3, that AG-aware training provides the most significant improvement in both accuracy and robustness, namely 13.03% and 61.58%. This is followed by PV-aware training, which achieves an improvement of 7.99% in average accuracy and 26.46% in robustness. Lastly, the SU-aware training approach delivers the lowest accuracy improvement of 3.89% and lowest robustness improvement of 13.26%.

Based on the given comparison, we conclude that the three stochastic factors exhibit different degrees of influence on the pNCs: As aging and printing variation lead to changes in every conductance (thus weight) of the pNCs, whereas sensing uncertainty only explicitly affects the first weighted-sum operation (multiplicatively), which results in the weaker impact on the performance of the pNCs. On the other hand, for the comparison of aging and printing variation, we hypothesize that aging has a more substantial impact on the conductance than printing variation (based on the input noise, variations, and aging behavior assumed in this experiment). Consequently, AG-aware training yields greater improvements compared to PV-aware training.

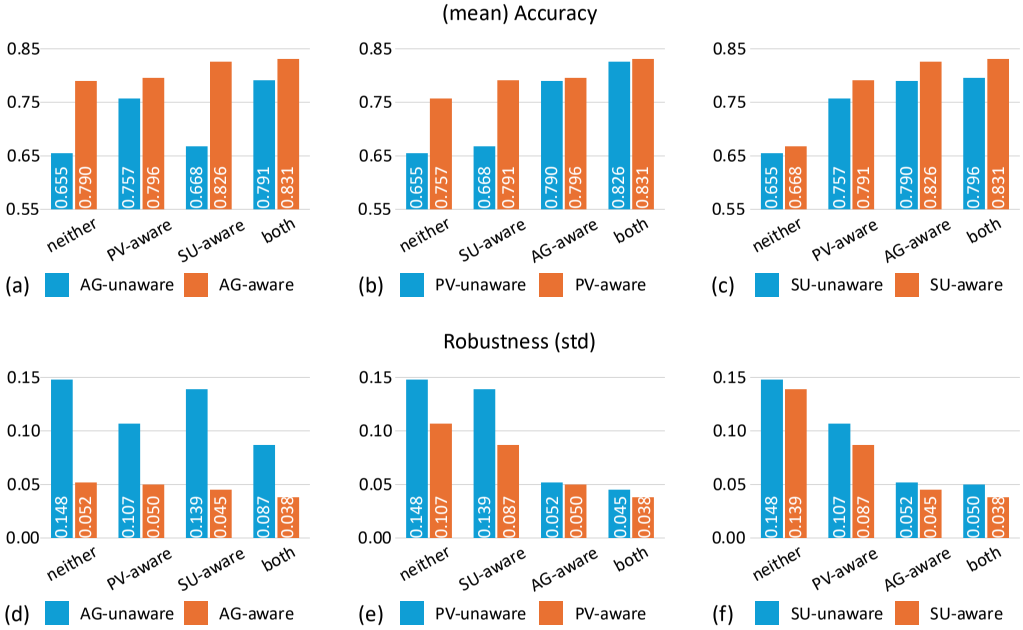


Figure 4.8: Ablation study of the combinatorial effects of aging (AG), printing variation (PV), and sensing uncertainty (SU) on the mean accuracies (top) and robustness (bottom). Each plot refers to one impact factor. Sourced from [52].

Joint analysis. Despite the independent analysis of the impact of each factor on pNCs, their actual effects are not entirely independent of each other. To assess their relationships, we conduct an ablation study to evaluate the improvement provided by each factor in different settings (see Figure 4.8).

Figure 4.8(a) demonstrates that AG-aware training can substantially improve the accuracy of pNCs from nominal training (unaware of neither aging nor printing variation nor sensing uncertainty). The same is true when only SU-aware training is used. Conversely, it offers less improvement for pNCs that have already trained with consideration of printing variations. We hypothesize that printing variation and aging effect may have similar effects in training pNCs, which renders the additional improvement of AG-aware training over PV-aware training insignificant. This hypothesis is also supported by Figure 4.8(b).

In Figure 4.8(b), it can be seen that PV-aware training can significantly increase the accuracy of pNCs from standard training and SU-aware training. However, PV-aware training offers only less improvement when AG-aware training is already employed. This suggests that the aging effects not only have a similar influence as printing variation, namely perturbing the resulting weights, but also exert a stronger impact than printing variation. Therefore, additional PV-aware training has little benefit when AG-aware training is already utilized. In contrast to PV- and AG-aware training, SU-aware training offers around 2% - 3% improvement in all cases, as shown in Figure 4.8(c). This indicates that the impact of sensing uncertainty may be orthogonal to that of printing variation or aging.

Regarding robustness, similar effects are observed. From Figure 4.8, it is evident that both AG- and PV-aware training can substantially enhance the robustness of pNCs. However, as they might have similar mechanisms of influence on pNCs, their combined effect exhibits some overlapping. Moreover, since the impact of aging is more significant than that of printing variation, the additional PV-aware training in conjunction with AG-aware training does not bring significant benefits. Orthogonal to them, SU-aware training consistently provides stable improvement in robustness.

4.2.4 Discussion

In this section, we perform experiments to confirm the effectiveness of the dependability-aware training of pNCs. Our results demonstrate that the proposed method can enhance the accuracy and robustness of pNCs by 27% and 74%, respectively. Among all effects considered in training, AG-aware training yields the most significant improvement. While PV-aware training also contributes significantly, our ablation study reveals that printing variation and aging effect may have similar potential mechanisms on the pNCs, suggesting that the contribution of AG-aware training may partly cover that of PV-aware training.

Notably, SU-aware training consistently delivers improvement in accuracy and robustness across all experiments. This suggests that, sensing uncertainty might have a distinct mode of effect on pNCs compared to the other two factors. Even though the improvement from SU-aware training is slightly lower than that from PV- and AG-aware trainings, it is possibly due to the choice of σ value. We suspect that, the effect of SU-aware training may change with different σ values. Nevertheless, due to the possibly orthogonal effects to the other two factors, it is meaningful to consider sensing uncertainty to improve the dependability of pNCs.

Although the dependability-aware training is conducted at fully algorithmic level, the outcomes from the ablation study may also indicate the impact of various factors on network performance, and thus, guide the development of the hardware technologies. For instance, aging-aware training demonstrates a substantial enhancement in network performance, implying that, the aging process may exert a more pronounced effect on circuits. Consequently, in fabrication, it is suggestible to prioritize the efforts both in the materials as well as in the process to reduce the impact of aging, including the adoption of passivation techniques or superior materials that display lower aging variation.

4.3 Architecture Search for Reliability Design

The previous section offered a comprehensive consideration and analysis of the primary factors influencing the reliability of pNCs. Nevertheless, its optimiza-

tion remains limited to the parameter space of continuous and differentiable conductance values and EGT geometrics. In this section, we will bring the optimization of pNCs to a higher level by leveraging the capability of the EA method for discrete variables [34]. This approach enables not only to investigate the circuit architecture, which is also referred to as neural architecture search (NAS), but also to automatically and optimally select the activation circuits for each printed neuron, thereby achieving higher robustness of the pNCs.

4.3.1 Design of Printed Activation Circuits

Different nonlinear circuit designs often correspond to different transfer characteristic curves. Additionally, these schemes exhibit varying levels of robustness against variations in circuit components. To investigate the impact of different activation circuits on classification accuracy and robustness, this work designs and models multiple nonlinear circuits that emulate classical activation functions in ANNs. Figure 4.9 and Figure 4.10 shows the circuit schematics and the transfer characteristic curves of the proposed circuit.

Printed sigmoid circuit. The design of printed sigmoid (pSigmoid) circuit can is shown in Figure 4.9(b) and can be modeled by a modified sigmoid function [39], i.e.,

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

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

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

Here, similar to Equation (2.5), $\boldsymbol{\eta}^{\mathrm{S}} = [\eta_1^{\mathrm{S}}, \eta_2^{\mathrm{S}}, \eta_3^{\mathrm{S}}, \eta_4^{\mathrm{S}}]$ is the auxiliary parameter determined by the physical quantities $\boldsymbol{q}^{\mathrm{S}} = [R_1^{\mathrm{S}}, R_2^{\mathrm{S}}, R_3^{\mathrm{S}}, W_1^{\mathrm{S}}, L_1^{\mathrm{S}}, W_2^{\mathrm{S}}, L_2^{\mathrm{S}}]$ in the circuit.

Printed clipped ReLU circuit. Printed clipped ReLU (pCReLU) circuit, as shown in Figure 4.9(c), emulates the clipped ReLU function. Different from ReLU function, clipped ReLU function, also called hard sigmoid, does not

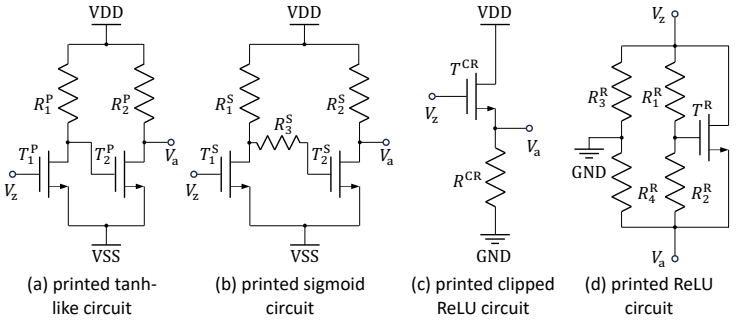


Figure 4.9: Proposed nonlinear activation circuits. The functional forms, including (a) tanh function, (b) sigmoid function, (c) clipped ReLU function, and (d) ReLU function. Sourced from [34].

increase proportionally with increasing input, but rather truncated by a certain value when the input is sufficiently large. The functional form of its transfer characteristic curve is illustrated in Figure 4.10. The mathematical model of this circuit is modeled as

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

where $\boldsymbol{\eta}^{\mathrm{CR}} = [\eta_1^{\mathrm{CR}}, \eta_2^{\mathrm{CR}}, \eta_3^{\mathrm{CR}}, \eta_4^{\mathrm{CR}}]$ are auxiliary parameters determined by the physical quantities $\boldsymbol{q}^{\mathrm{CR}} = [R_1^{\mathrm{CR}}, W_1^{\mathrm{CR}}, L_1^{\mathrm{CR}}]$.

Printed ReLU circuit. The design of printed ReLU (pReLU) circuit is shown in Figure 4.9(d). As the transfer characteristic curve has not only a slope in the negative half but also a smooth transition at $V_{\mathrm{z}} = 0$ (see Figure 4.10), neither the ideal ReLU function, nor its variation, e.g., LeakyReLU [50] function nor the softplus [26] function, is sufficient to precisely describe the printed ReLU circuit. Thus, we combine a softplus function to provide the smoothness at $V_{\mathrm{z}} = 0$ and a constant linear function to provide the slope at negative half.

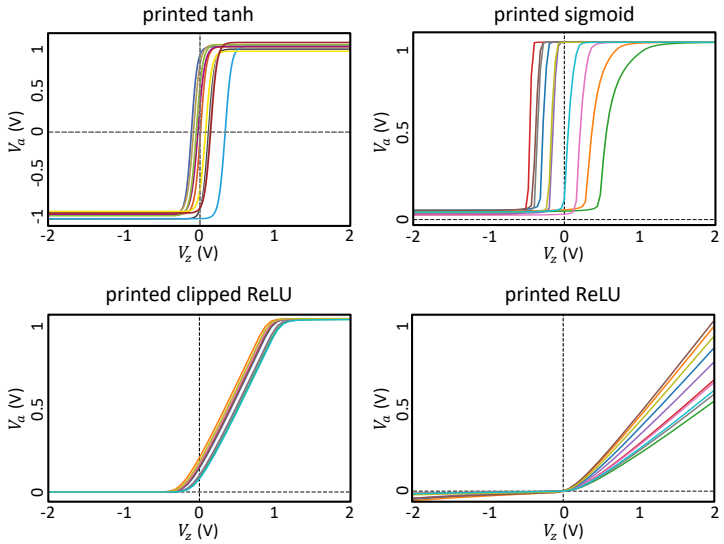


Figure 4.10: Exemplified characteristic curves of different activation circuits with different designs of physical quantities $\boldsymbol{q}$, including printed tanh circuit with different $\boldsymbol{q}^{\mathrm{T}}$, printed sigmoid circuit with different $\boldsymbol{q}^{\mathrm{S}}$, printed clipped ReLU circuit with different $\boldsymbol{q}^{\mathrm{CR}}$, and printed ReLU circuit with different $\boldsymbol{q}^{\mathrm{R}}$. Sourced from [34].

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

$$V_{\mathrm{a}} = \eta_1^{\mathrm{R}} \cdot (x - \eta_3^{\mathrm{R}}) + \eta_2^{\mathrm{R}} \cdot \mathrm{softplus}(V_{\mathrm{z}} - \eta_3^{\mathrm{R}}, \eta_5^{\mathrm{R}}) + \eta_4^{\mathrm{R}},$$

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

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

Likewise, $\boldsymbol{\eta}^{\mathrm{R}} = [\eta_1^{\mathrm{R}}, \eta_2^{\mathrm{R}}, \eta_3^{\mathrm{R}}, \eta_4^{\mathrm{R}}, \eta_5^{\mathrm{R}}]$ are auxiliary parameters determined by the physical quantities $\boldsymbol{q}^{\mathrm{R}} = [R_1^{\mathrm{R}}, R_2^{\mathrm{R}}, R_3^{\mathrm{R}}, R_4^{\mathrm{R}}, W_1^{\mathrm{R}}, L_1^{\mathrm{R}}]$.

Table 4.4 reports the empirical feasible design spaces of the proposed printed activation circuits. Out of these spaces, the transfer characteristics of the circuits do not emulate the corresponding functional forms.

Table 4.4: Feasible design space of different activation circuits. Sourced from [34].

Circuit	Range	R_1 (kΩ)	R_2 (kΩ)	R_3 (kΩ)	R_4 (kΩ)	W_1 (kΩ)	L_1 (µm)	W_2 (µm)	L_2 (µm)
ptanh	min	250	6	-	-	80	40	480	30
	max	2000	32	-	-	100	80	500	40
pSigmoid	min	350	40	-	-	80	80	500	40
	max	750	80	-	-	600	200	800	80
pCReLU	min	1000	-	-	-	40	80	-	-
	max	10000	-	-	-	100	200	-	-
pReLU	min	10	500	1	30	200	80	-	-
	max	100	2000	20	100	800	120	-	-

4.3.2 Modeling of Activation Circuits

Figure 4.10 exemplifies some specific transfer characteristics with different physical quantities $\boldsymbol{q}$. Similar to the ptanh and negation circuits, the characteristics of the proposed circuits can be modified through their physical quantities. This property enables us to do bespoke design of these nonlinear circuits for specific objectives. However, on the other hand, the printing variation of the physical quantities will also perturb the transfer curves from the designed ones. Therefore, it is crucial to consider these issues into the training of the pNCs. To this end, we establish the surrogate models for these nonlinear circuits. Analogous to Chapter 3.1.2, the modeling of the proposed printed activation circuits are implemented through approximation-based modeling. Following the algorithm introduced in Algorithm 1, we build the datasets of the transfer characteristic curves of the designed circuits and train corresponding surrogate models to calculate the auxiliary parameters $\boldsymbol{\eta}$ from their physical quantities $\boldsymbol{q}$, namely:

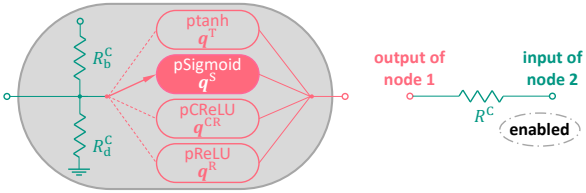
$$\boldsymbol{q}^{\mathrm{S}} \mapsto \boldsymbol{\eta}^{\mathrm{S}},\ \boldsymbol{q}^{\mathrm{R}} \mapsto \boldsymbol{\eta}^{\mathrm{R}},\ \boldsymbol{q}^{\mathrm{CR}} \mapsto \boldsymbol{\eta}^{\mathrm{CR}}$$

With these surrogate circuit models, the physical quantities $\boldsymbol{q}$ can be incorporated into the forward pass in pNCs. Meanwhile, they may also be optimized

to yield the best transfer characteristic curves parameterized by $\boldsymbol{\eta}$ for specific tasks.

4.3.3 Architecture Search for High Reliability

To enable EAs to automatically select the optimal activation circuit for each neuron, along with the optimization of physical quantities within the circuits, we modify the encoding and mutation strategy based on the one introduced in Chapter 3.2.2.

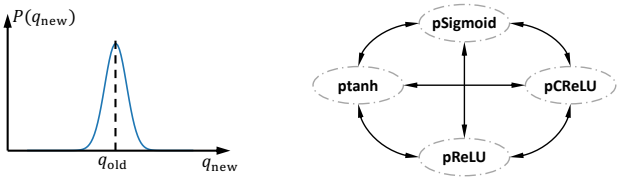


(a) node gene for selectable activation (b) edge gene

Figure 4.11: Encoding of the genes that can enable both selection of activation circuit types and physical quantities of the selected activation circuit. Sourced from [34].

Modification of the EA. Figure 4.11(a) depicts the modified encoding of a node gene. Different from the fixed activation circuits described in Chapter 3.2.2, the new gene comprises all four activation circuits and their respective physical quantities $\boldsymbol{q}$. Additionally, a learnable pointer directs to one of the activation circuits, which will then be utilized in the forward pass of the pNC.

In the mutation process, the physical quantities $\boldsymbol{q}$ of all activation circuits, irrespective of whether their corresponding activation circuits are pointed (activated), undergo random alterations (see Figure 4.12). Concurrently, the pointer will also mutate with a certain probability to randomly point to another activation circuit.



(a) numerical mutation (b) discrete decision mutation

Figure 4.12: Mutation of the node gene that is related to selectable and learnable activation circuits. Sourced from [34].

4.3.4 Experiment

We implement the training framework with Python[2] and conduct training on the pNC utilizing the EA methodology and test it on 13 benchmark datasets (described in Table 4.1) against gradient-based optimization techniques as a baseline as the experiment.

Experiment setup. 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. To achieve a sufficient good result, the population for these experiments is set at $N = 1{,}000$. Each node is initialized to have a random activation circuit among the given design.

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 terms of the mutation mechanisms, 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 at 0.3. In this way, the topology of the network can grow to a larger scale. In addition, there exists a 0.1 chance that the edges will toggle its state from enabled to disabled, or vice versa. Furthermore, the mutation rate of changing the pointer of selected activation circuit is 0.1.

In training (evolution) process, we utilize a full-batch training, with termi-

[2] `https://github.com/Neuromophic/eNAS_leanrable_selectable_LNC`.

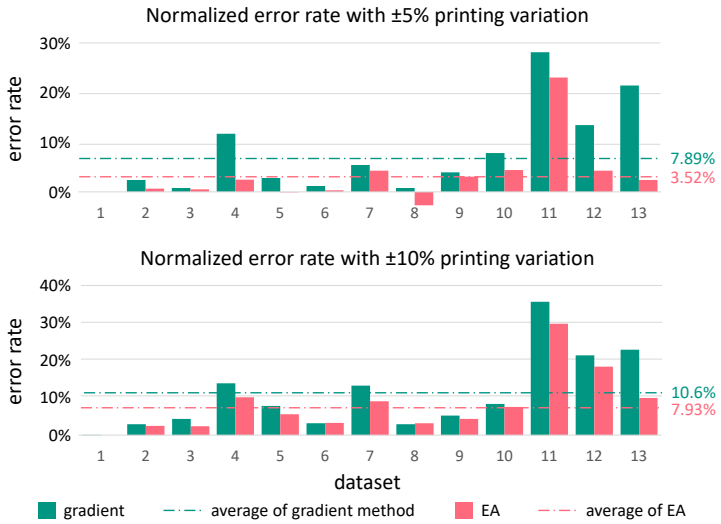


Figure 4.13: Normalized error rate with 5% and 10% printing variations. Sourced from [34].

nation upon a patience threshold, i.e., $\mathcal{K}_3$, 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, employing different random seeds varying from 1 to 10.

Note that, this configuration will be treated as the *default setup* for all the trainings with EAs in this thesis.

As training objective, we do not include all the factors described in Chapter 4.2. Rather, we take only printing variation into consideration to exhibit the effectiveness of the EA. We take $\varepsilon \sim \mathcal{U}[1-\mathrm{e}, 1+\mathrm{e}]$ with $\mathrm{e} = 5\%$ and $\mathrm{e} = 10\%$ to exam the robustness for both high and low printing precision, respectively.

Baseline. As the baseline, we employ the gradient-based training method, following the default setup as described in Chapter 4.1.3. Moreover, To provide an upper bound classification accuracy of each dataset, we also trained the

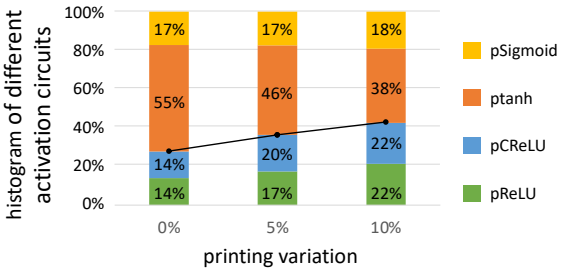


Figure 4.14: Histogram of different selected activation circuits with 0%, 5%, and 10% printing variations. Sourced from [34].

pNCs without any variation, i.e., $e = 0$. 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*.

Result. After training the pNCs on training datasets and early-stopping the training on validation datasets, we test the trained pNCs on test sets. Table 4.5 reports the classification accuracy and the running time of the training.

Given that the classification accuracy under variation is close to the theoretical upper bound (i.e., the reference accuracy), the absolute magnitude of the increment in accuracy becomes less significant due to boundary effects [40]. Consequently, we normalize the accuracy by their perspective reference accuracy, subsequently, we focus on the error rate instead of the improvement of the classification accuracy. The result is plotted in Figure 4.13. We can conclude that, compared to the baseline, pNCs trained from EA provide a significant reduction of error rate by 55.38% in $\pm 5\%$ and 25.11% in $\pm 10\%$ printing variations.

In addition to the performance regarding classification results, we also analyze the robustness of different activation circuits to provide further guidance for future design. To this end, we summarize the percentages of different activation circuits used in pNCs after the training, as illustrated in Figure 4.14. A clear trend can be observed where the number of ptanh circuits decreases significantly with increasing printing variation, while the number of pCReLU and pReLU circuits increases. We speculate that, there are two factors influ-

Table 4.5: Classification accuracy and runtime of gradient-based approach without variation and comparison with EA with baseline in high precision printing (5% variation) and low-precision printing (10% variation) on 13 benchmark datasets. Sourced from [34].

Dataset	Reference accuracy	High-precision printing (±5%)		Low-precision printing (±10%)		Runtime	
	(without variation)	Baseline	EA	Baseline	EA	Baseline (min)	EA (min/pop)
1	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.999 ± 0.012	1.000 ± 0.000	183.9	9.5
2	0.902 ± 0.017	0.880 ± 0.004	0.896 ± 0.008	0.877 ± 0.008	0.881 ± 0.012	205.9	21
3	0.971 ± 0.001	0.963 ± 0.008	0.966 ± 0.006	0.931 ± 0.039	0.949 ± 0.012	180.9	11
4	0.879 ± 0.007	0.774 ± 0.004	0.857 ± 0.005	0.763 ± 0.002	0.794 ± 0.007	178.0	19
5	0.915 ± 0.019	0.889 ± 0.032	0.916 ± 0.026	0.847 ± 0.012	0.866 ± 0.011	194.6	15
6	0.894 ± 0.016	0.883 ± 0.023	0.891 ± 0.038	0.867 ± 0.026	0.866 ± 0.021	189.4	10
7	0.965 ± 0.005	0.912 ± 0.034	0.923 ± 0.050	0.843 ± 0.045	0.882 ± 0.039	178.8	7.3
8	0.788 ± 0.003	0.782 ± 0.017	0.810 ± 0.018	0.766 ± 0.053	0.764 ± 0.055	190.9	5.0
9	0.577 ± 0.054	0.554 ± 0.038	0.559 ± 0.039	0.548 ± 0.047	0.553 ± 0.050	198.3	14
10	0.891 ± 0.031	0.820 ± 0.034	0.851 ± 0.023	0.820 ± 0.041	0.827 ± 0.007	176.0	6.4
11	1.000 ± 0.001	0.713 ± 0.012	0.765 ± 0.018	0.660 ± 0.017	0.716 ± 0.019	177.1	6.4
12	0.830 ± 0.007	0.716 ± 0.007	0.794 ± 0.004	0.661 ± 0.000	0.685 ± 0.004	180.6	4.6
13	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	$\mathbf{0.879 \pm 0.013}$	$\mathbf{0.809 \pm 0.023}$	$\mathbf{0.848 \pm 0.019}$	$\mathbf{0.786 \pm 0.029}$	$\mathbf{0.809 \pm 0.023}$	**181.9**	**10.8**

encing the robustness of the pNCs against printing variation. The first factor is the intrinsic robustness of the circuit to variance, i.e., whether a small change in the physical quantity $\boldsymbol{q}$ leads to a substantial change in $\boldsymbol{\eta}$. Secondly, the circuit should also exhibit a smaller slope in its transfer characteristic, because the resistor crossbar weighted-sum suffers from variation as well, causing fluctuations in the input voltages to the activation circuits. Consequently, the activation function should have a low slope to prevent drastic changes in the output due to minor input variations. By checking Figure 4.10, our speculation is justified, as the experimental results align with our expectation. Therefore, we conclude that, at high printing variation scenarios, pCReLU circuits will become dominant. Conversely, at low printing variation, ptanh is capable to finely distinguish small input differences, and thus can grant the pNCs a greater expressiveness, making ptanh circuits dominant in low variation cases.

4.3.5 Discussion

The introduction of NAS elevates the optimization of pNCs to a higher level, as demonstrated by the experiments. Although other factors like aging and sensing errors were not considered in the objective function, their implementation is technically straightforward.

Furthermore, we obtain a useful insight for selecting or designing pNCs to withstand hardware variations. Under conditions of high variation, it is advisable to design or utilize activation circuits that are intrinsically robust to resist variation inside activation circuits, and exhibit small slops to resist input perturbation.

4.4 Fault Analysis

Previous sections have examined the reliability of pNCs in terms of factors that influence its classification accuracy. These variations are typically anticipated and expected during the design process. However, unexpected problems can arise during the fabrication process of PE, such as catastrophic faults like open or short of circuit components. An example of the transistor fault is shown in Figure 4.15. As these faults are rare and should often be mitigated by improved

printing techniques or circuit testing, this section primarily examines the impact of catastrophic faults on the pNCs [33], and less about its algorithmic level solution. This section justifies the importance of circuit testing or post-printing for pNCs.

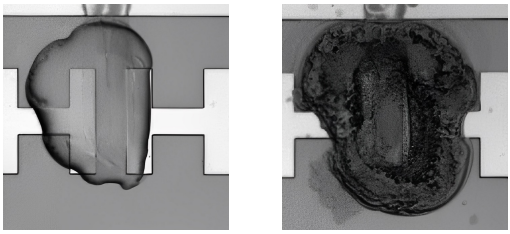


Figure 4.15: Micrographs of the electrolyte-gated transistor (EGT): functioning EGT (left) compared to EGT with exploded electrolyte (right). Sourced from [13].

4.4.1 Modeling of Printing Faults

In order to incorporate the device faults into the forward pass of pNCs, we first study and model the faults of printed devices. Afterwards, we propose an algorithm to emulate the random faults into pNCs.

Empirically [13], the faults of PE can be categorized into the defects in resistors and in transistors. For printed resistors, as illustrated in Figure 4.16(a), the faults can be modeled as either open or short circuits. In this case, the resistor can be simply modeled by setting its resistance $R = \infty$ (open circuit) or $R = 0$ (short circuit), correspondingly, the conductance is modeled as $g = 0$ and $g = \infty$.

In contrast, the fault models of transistors are more complex. As depicted in Figure 4.16(b), transistor faults can be summarized into four types: Gate-Drain (G-D) short, Gate-Source (G-S) short, Drain-Source (D-S) short, and Gate open. In this work, we typically consider transistors in the nonlinear circuits, such as activation circuits or negation circuits, as a whole unit. Consequently, we do not model transistor faults independently into the pNCs. In-

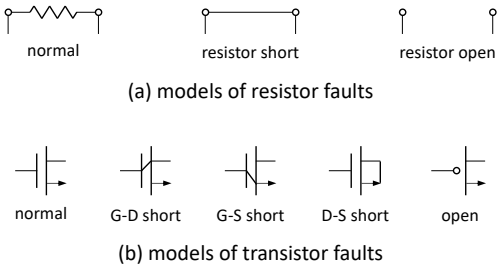


Figure 4.16: Circuit schematics of the resistor and transistor faults.

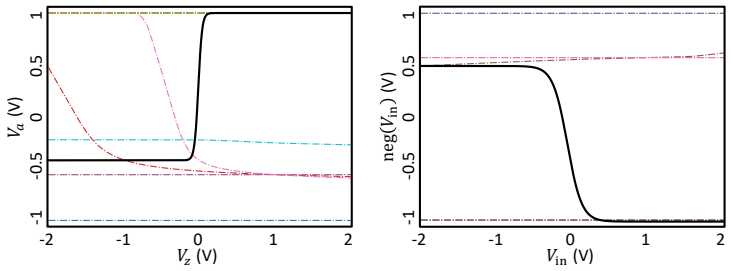
stead, we analyze the activation circuit or negation circuit as complete entities to understand and model their faults.

Due to the low probability of faults occurring, the scenario of more than two faults in the same nonlinear circuit can be disregarded. Therefore, it is sufficient to consider only one fault when modeling the nonlinear circuit with fault. For ptanh circuit (shown in Figure 2.6), we account for the short and an open R_1^{A} or R_2^{A}, and the G-D short, G-S short, D-S short, and Gate open in T_1^{A} or T_2^{A}. This results in 12 different fault situations, and we plot the characteristic curves of the ptanh circuit for those faults in Figure 4.17(a). Analogous, the faults of the negation circuit are shown in Figure 4.17(b). Evidently, the faults in nonlinear circuits can lead to catastrophic consequence. Given the streamlined and compact design of these circuits, each component plays a crucial role. For instance, if a pull-up resistor is short, the transistor can no longer be activated, resulting in a nearly constant characteristic curve. For the negation circuit, as it consists of only one transistor, once the transistor is faulty, the circuit will lose the nonlinearity, as shown in Figure 4.17(b). Regarding the ptanh circuit, if a transistor, which functions as an inverter, fails, the tendency of its characteristic curve will be totally in opposition, see Figure 4.17(a).

To model the faulty negation circuit, denoted by $\mathrm{neg}^{\mathrm{f}}(\cdot)$, we simply employ a linear function

$$\mathrm{neg}_i^{\mathrm{f}}(V_{\mathrm{in}}) = k_i^{\mathrm{fN}} \cdot V_{\mathrm{in}}$$

to fit the curves in Figure 4.17(b) through the parameter k_i^{fN}. As a result, we



(a) Printed tanh-like circuit with faults (b) Negative weight circuit with faults

Figure 4.17: Impact of printing fault on nonlinear circuits of pNCs: (a) characteristic curves of ptanh circuit with different faults in its components, and (b) characteristic curves of printed negation circuit with different faults in its components. The black bold curves refer to the fault-free cases, while the colored dash-dot lines indicate different faults. Sourced from [33].

obtained 14 faulty negation functions, encompassing two types of faults for each of the five resistors and four types of faults for the transistor. They are denoted by $\text{neg}_i^{\text{f}}(\cdot), i = 1, \cdots, 14$.

As the characteristics of faulty ptanh, denoted by $\text{ptanh}^{\text{f}}(\cdot)$, exhibit negative tanh-like behavior, we thus use the

$$\text{ptanh}^{\text{f}}(V_z) = -\left(\eta_1^{\text{fA}} + \eta_2^{\text{fA}} \cdot \tanh\left((V_z - \eta_3^{\text{fA}}) \cdot \eta_4^{\text{fA}}\right)\right)$$

to fit the curves of the faulty ptanh circuits. Thus, we obtained 12 faulty activation functions, including two types of faults for each of the resistors and four types of faults for the two transistors. They are denoted by $\text{ptanh}_i^{\text{f}}(\cdot), i = 1, \cdots, 12$.

4.4.2 Injection of Faults

To simulate the occurrence of faults in algorithmic level, we introduce the mask-based method to inject faults. Specifically, for the surrogate conductances within a resistor crossbar, $\boldsymbol{\theta}$ is multiplied by a mask vector before in-

volved into the forward pass of the pNC, i.e.,

$$\boldsymbol{\theta}^{\mathrm{f}} = \boldsymbol{\theta} \odot M^{\mathrm{fC}}, \tag{4.6}$$

where M^{fC} has the same dimension as $\boldsymbol{\theta}$ and its elements are all set to 1 by default. Therefore, the default faulty crossbar conductance $\boldsymbol{\theta}^{\mathrm{f}}$ equals to the designed conductances $\boldsymbol{\theta}$. When injecting a fault, we randomly set a value in M^{fC} to 0 (for open circuit) or ∞ (for short circuit).

For the negation circuits, we employ

$$\mathrm{neg}^{\mathrm{f}}(\cdot) = \mathrm{NEG}^{\mathrm{f}} \cdot M^{\mathrm{fN}}, \tag{4.7}$$

where the first term includes all the negation functions, i.e.,

$$\mathrm{NEG}^{\mathrm{f}} = [\mathrm{neg}(\cdot), \mathrm{neg}_1^{\mathrm{f}}(\cdot), \mathrm{neg}_2^{\mathrm{f}}(\cdot), \cdots, \mathrm{neg}_{14}^{\mathrm{f}}(\cdot)]$$

and the mask matrix is

$$M^{\mathrm{fN}} = [m_0^{\mathrm{fN}}, m_1^{\mathrm{fN}}, \cdots, m_{14}^{\mathrm{fN}}]^{\top},\ m_i^{\mathrm{fN}} \in \{0,1\},\ \sum_i m_i^{\mathrm{fN}} = 1.$$

In this way, by setting $m_0^{\mathrm{fN}} = 1$, the final negation circuit $\mathrm{neg}^{\mathrm{f}}(\cdot)$ is identical to the original and functional circuit $\mathrm{neg}(\cdot)$ as introduced in Equation (2.6). Otherwise, by setting $m_i^{\mathrm{fN}} = 1$, $i = 1, \cdots, 14$, the i-th faulty negation circuit will be employed in the forward pass of the pNC.

Similarly, we utilize

$$\mathrm{ptanh}^{\mathrm{f}}(\cdot) = \mathrm{PTANH}^{\mathrm{f}} \cdot M^{\mathrm{fA}}, \tag{4.8}$$

to handle the faults in ptanh circuit. Here, the first term includes all the ptanh functions, i.e.,

$$\mathrm{PTANH}^{\mathrm{f}} = [\mathrm{ptanh}(\cdot), \mathrm{ptanh}_1^{\mathrm{f}}(\cdot), \mathrm{ptanh}_2^{\mathrm{f}}(\cdot), \cdots, \mathrm{ptanh}_{12}^{\mathrm{f}}(\cdot)]$$

and the mask matrix is

$$M^{\mathrm{fA}} = [m_0^{\mathrm{fA}}, m_1^{\mathrm{fA}}, \cdots, m_{12}^{\mathrm{fA}}]^{\top},\ m_i^{\mathrm{fA}} \in \{0,1\},\ \sum_i m_i^{\mathrm{fA}} = 1.$$

In this way, by setting $m_0^{\mathrm{fA}} = 1$, the final activation circuit $\mathrm{ptanh}^{\mathrm{f}}(\cdot)$ is identical to the original and functional circuit $\mathrm{ptanh}(\cdot)$ as introduced in Equation (2.5). Otherwise, by setting $m_i^{\mathrm{fN}} = 1$, $i = 1, \cdots, 12$, the i-th faulty ptanh circuit will be employed in the forward pass of the pNC.

After injecting faults into the circuit primitives, Equation (4.6) - Equation (4.8) are then employed to replace the corresponding terms in Equation (3.1).

4.4.3 Experiment

To investigate the impact of faults on pNCs and evaluate the tolerance of pNCs trained with different strategies, we employed three training methods: nominal training, variation-aware training, and training with dropout. We then introduced 1, 2, and 4 faults respectively into the trained pNCs and observed the effect on classification accuracy. The implementation can be found in the GitHub repository[3].

Experiment setup. The major training setups are kept identical to the default one described in Chapter 4.1.3. Additionally, in variation-aware training, we take $\mathrm{e} = 10\%$, while in dropout training, we set the dropout rate to 10%. Additionally, we also combine both variation-aware training and dropout to train pNCs. Note that, the variation or dropout is only introduced during training. In test stage, we only inject device faults without any additional variation or dropout.

Baselines. Nominal training refers to train pNCs with cross-entropy loss [29] as the objective function to straightforwardly increase the classification accuracy. As it forms the most basic training method, we treat the nominal training as the baseline. Meanwhile, variation-aware training, as introduced earlier in this chapter, considers parametric variations that may impact the classification accuracy of the pNCs. Therefore, we want to study whether it can also provide robustness against catastrophic faults. Lastly, dropout [45] is a common training trick in ML, referring to deactivate some neurons randomly during training to prevent overfitting. Since the deactivation in dropout training resembles the

[3] `https://github.com/Neuromophic/FaultAnalysis`.

open of the device in the pNCs, we speculate dropout training may enhance the circuit tolerance to faults. For all these training strategies, we consider the fault-free cases as the baselines in this experiment.

Result. Figure 4.18 represents the performance of pNCs trained with different strategies and tested with 0, 1, 2, and 4 faults in a whole pNC. It is clear that there is a significant decrease in the average classification accuracy of pNCs on all the datasets with the number of faults increases. Additionally, the variance of the classification accuracy increases as well, meaning that the reliability of pNCs becomes hard to estimate.

Another conclusion is that, unfortunately, neither variance-aware training, dropout training, nor a combination of these methods significantly improves the robustness of the circuit to faults. This justifies the development of novel algorithms that can improve the circuit robustness against faults or detect the faults efficiently.

4.4.4 Discussion

This section suggests typical faults in pNCs and model their faulty behaviors. Subsequently, we propose a mask-based method to resemble fault injection. Finally, the robustness of multiple training methods are studied in the experiment, however, none of them can offer significant improvement against faults. Therefore, it necessitates to improve existing hardware technologies or develop other techniques to overcome circuit faults, such as upgrading ink qualities or printing techniques. In terms of circuit design, more robust circuit schematics can be proposed with enhanced resilience to combat component faults. Additionally, developing circuit testing is also crucial. Because, due to the additive manufacturing characteristics of PE, it should be possible to compensate for circuit faults by reconfiguring and reprinting some of the components once the faults have been detected and located. The effectiveness of this technique, named *in-situ tuning*, was initially demonstrated in [20].

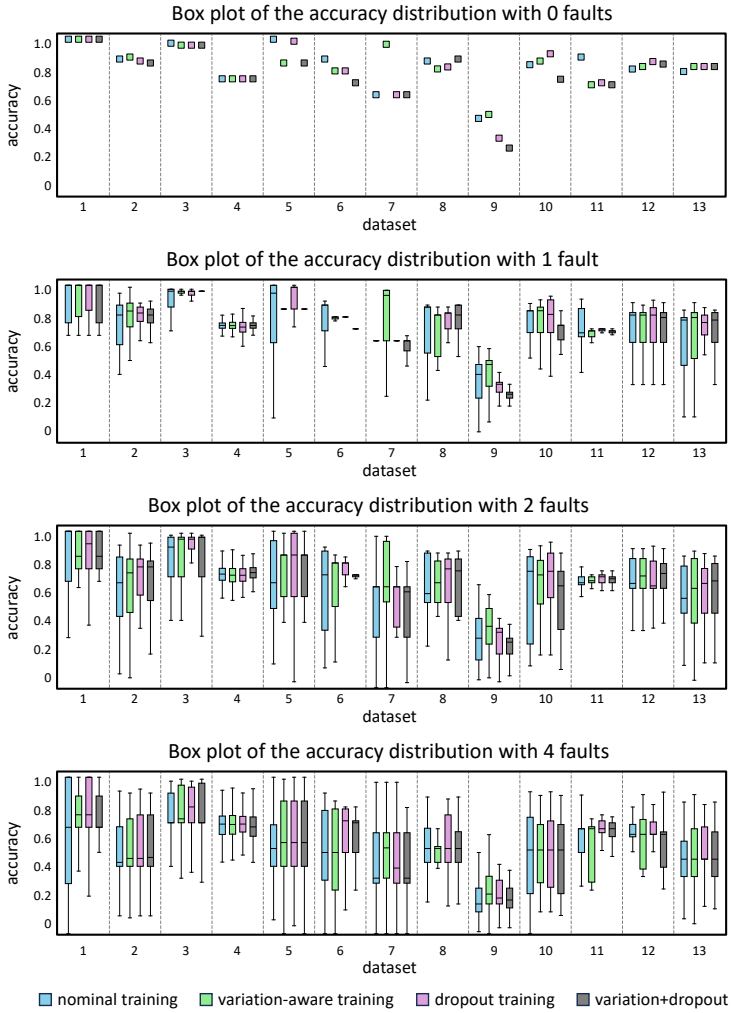


Figure 4.18: Box plot of the classification accuracy of pNCs from nominal and variation-aware training, dropout training, and dropout with variation-aware training on test set. In the test phase, pNCs are tested under 4 scenarios: fault-free, single-fault, double-fault, and quadruple-fault conditions. Sourced from [33].

4.5 Summary

In this chapter, we investigate and improve the reliability and robustness of pNCs. Firstly, we identify several key factors that affect the reliability of pNCs, including aging, sensing errors, printing variation, and component faults. Then, we analyze the behavior of these factors, develop their mathematical models, and incorporate them into the forward pass of the pNCs training frameworks.

In terms of algorithmic perspective, this chapter first proposes objective functions that can improve the expected classification accuracy under aging, sensing errors, and printing variation. Afterwards, by introducing MC estimation, the proposed objective functions can be efficiently solved through gradient-based optimization methods. EAs can further bring the training of pNCs to a higher level. By facilitating the optimization of discrete variables, the search space of pNCs is expanded to include the circuit architecture and decision for selecting different activation circuits. The effectiveness of these methods has been demonstrated through experiments.

By analyzing the experimental results, further suggestions for pNCs design can be derived. Regarding the fabrication process, the experiments in Chapter 4.2 indicates that, the impact of printing variation is significantly covered by aging. Therefore, developing techniques for anti-aging or packaging may be prioritized over reducing printing errors. Conversely, sensing error is independent of aging and printing variation, thus, improving sensor accuracy and incorporating sensing uncertainty-aware training are always critical. Moreover, due to the lack of optimization algorithms for fault tolerance, reducing the printing failure rate remains an important objective as well. In terms of circuit design, the experiment in Chapter 4.3 reveals that, improving circuit robustness against variations depends not only on the enhancing the inherent robustness of the circuit itself, but also on designing characteristic curves with small slopes to resist fluctuations in the input signals caused by variation of preceding components.

Ensuring the reliability of the circuit is the first step in transitioning pNCs from labor concept to real-world deployment. In the next chapter, this thesis will address various issues in the application of pNCs, as enhancing the practicality of pNCs is the next critical step towards its deployment.

Bibliography

[1] Fevzi Alimoglu, Dr Doc, Ethem Alpaydin, and Yagmur Denizhan. “Combining Multiple Classifiers for Pen-based Handwritten Digit Recognition”. In: (1996).

[2] Edgar Anderson. “The Irises of The Gaspe Peninsula”. In: *Bull. Am. Iris Soc.* 59 (1935), pp. 2–5.

[3] Diogo Ayres-de-Campos, Joao Bernardes, Antonio Garrido, Joaquim Marques-de-Sa, and Luis Pereira-Leite. “Sisporto 2.0: A Program for Automated Analysis of Cardiotocograms”. In: *Journal of Maternal-Fetal Medicine* 9.5 (2000), pp. 311–318.

[4] Eric Berthonnaud, Joannès Dimnet, Pierre Roussouly, and Hubert Labelle. “Analysis of The Sagittal Balance of The Spine and Pelvis Using Shape and Orientation Parameters”. In: *Clinical Spine Surgery* 18.1 (2005), pp. 40–47.

[5] U Bielecka, P Lutsyk, K Janus, J Sworakowski, and W Bartkowiak. “Effect of Solution Aging on Morphology and Electrical Characteristics of Regioregular P3HT FETs Fabricated by Spin Coating and Spray Coating”. In: *Organic Electronics* 12.11 (2011), pp. 1768–1776.

[6] Léon Bottou. “Online algorithms and stochastic approximations”. In: *Online learning in neural networks* (1998).

[7] Małgorzata Charytanowicz, Jerzy Niewczas, Piotr Kulczycki, Piotr A Kowalski, Szymon Łukasik, and Sławomir Żak. “Complete Gradient Clustering Algorithm for Features Analysis of X-ray Images”. In: *Information technologies in biomedicine*. Springer, 2010, pp. 15–24.

[8] Zheng Cui. *Printed electronics: materials, technologies and applications*. John Wiley & Sons, 2016.

[9] Jacek Czerniak and Hubert Zarzycki. “Application of Rough Sets in The Presumptive Diagnosis of Urinary System Diseases”. In: *Artificial intelligence and security in computing systems*. Springer, 2003, pp. 41–51.

[10] Frederik Michel Dekking. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005.

[11] Ronald H Dieck. *Measurement uncertainty: methods and applications*. ISA, 2007.

[12] Matthias Elter, Rüdiger Schulz-Wendtland, and Thomas Wittenberg. “The Prediction of Breast Cancer Biopsy Outcomes Using Two CAD Approaches that both Emphasize an Intelligible Decision Process”. In: *Medical physics* 34.11 (2007), pp. 4164–4172.

[13] Ahmet Turan Erozan, Simon Bosse, and Mehdi B Tahoori. “Defect detection in transparent printed electronics using learning-based optical inspection”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29.8 (2021), pp. 1505–1517.

[14] Ian Farrance and Robert Frenkel. “Uncertainty of measurement: a review of the rules for calculating uncertainty components through functional relationships”. In: *The Clinical Biochemist Reviews* 33.2 (2012), p. 49.

[15] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. “Do we need hundreds of classifiers to solve real world classification problems?” In: *The journal of machine learning research* 15.1 (2014), pp. 3133–3181.

[16] Harley Flanders. “Differentiation Under The Integral Sign”. In: *The American Mathematical Monthly* 80.6 (1973), pp. 615–627.

[17] B Fraboni, Piero Cosseddu, YQ Wang, RK Schulze, ZF Di, A Cavallini, M Nastasi, and A Bonfiglio. “Aging Control of Organic Thin Film Transistors via Ion-Implantation”. In: *Organic Electronics* 12.9 (2011), pp. 1552–1559.

[18] Mohammad M Hamasha, Tara Dhakal, Khalid Alzoubi, Shehab Albahri, Awni Qasaimeh, Susan Lu, and Charles R Westgate. “Stability of ITO Thin Film on Flexible Substrate under Thermal Aging and Thermal Cycling Conditions”. In: *Journal of Display Technology* 8.7 (2012), pp. 385–390.

[19] Michael Hefenbrock. “Modelling and Training Printed Neuromorphic Circuits”. PhD thesis. Dissertation, Karlsruhe, Karlsruhe Institute of Technology (KIT), 2022.

[20] Michael Hefenbrock, Dennis D Weller, Jasmin Aghassi-Hagmann, Michael Beigl, and Mehdi B Tahoori. “In-situ tuning of printed neural networks for variation tolerance”. In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2022, pp. 72–75.

[21] Edwin T Jaynes. “Information theory and statistical mechanics”. In: *Physical review* 106.4 (1957), p. 620.

[22] Saleem Khan, Leandro Lorenzelli, and Ravinder S Dahiya. “Technologies for printing sensors and electronics over large flexible substrates: A review”. In: *IEEE Sensors Journal* 15.6 (2014), pp. 3164–3185.

[23] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[24] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[25] Changwoo Lee, Hyunkyoo Kang, Hojoon Kim, Ho Anh, Duc Nguyen, and Keehyun Shin. “Quality control with matching technology in roll to roll printed electronics”. In: *Journal of Mechanical Science and Technology* 24.1 (2010), p. 315.

[26] Qian Liu and Steve 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.

[27] Kuankuan Lu, Rihui Yao, Yiping Wang, Honglong Ning, Dong Guo, Xianzhe Liu, Ruiqiang Tao, Miao Xu, Lei Wang, and Junbiao Peng. “Effects of Praseodymium Doping on the Electrical Properties and Aging Effect of InZnO Thin-film Transistor”. In: *Journal of Materials Science* 54.24 (2019), pp. 14778–14786.

[28] Olvi L Mangasarian and William H Wolberg. *Cancer Diagnosis Via Linear Programming*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 1990.

[29] Anqi Mao, Mehryar Mohri, and Yutao Zhong. “Cross-entropy loss functions: Theoretical analysis and applications”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 23803–23828.

[30] Christopher J Matheus and Larry A Rendell. “Constructive Induction on Decision Trees.” In: *IJCAI*. Vol. 89. Citeseer. 1989, pp. 645–650.

[31] Douglas C Montgomery and George C Runger. *Applied statistics and probability for engineers*. John wiley & sons, 2010.

[32] M Serdar Onses, Erick Sutanto, Placid M Ferreira, Andrew G Alleyne, and John A Rogers. “Mechanisms, capabilities, and applications of high-resolution electrohydrodynamic jet printing”. In: *Small* 11.34 (2015), pp. 4237–4266.

[33] Priyanjana Pal, Florentia Afentaki, Haibin Zhao, Gurol Saglam, Michael Hefenbrock, Georgios Zervakis, Michael Beigl, and Mehdi B Tahoori. “Fault Sensitivity Analysis of Printed Bespoke Multilayer Perceptron Classifiers”. In: *2024 IEEE European Test Symposium (ETS)*. IEEE. 2024, pp. 1–6.

[34] Priyanjana Pal, Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Neural Architecture Search for Highly Robust Printed Neuromorphic Circuits”. In: *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 2024, pp. 1–9.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[36] Lutz Prechelt. “Automatic early stopping using cross validation: quantifying the criteria”. In: *Neural networks* 11.4 (1998), pp. 761–767.

[37] Farhan Rasheed, Michael Hefenbrock, Michael Beigl, Mehdi B Tahoori, and Jasmin Aghassi-Hagmann. “Variability modeling for printed inorganic electrolyte-gated transistors and circuits”. In: *IEEE transactions on electron devices* 66.1 (2018), pp. 146–152.

[38] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. “A survey of neuromorphic computing and neural networks in hardware”. In: *arXiv preprint arXiv:1705.06963* (2017).

[39] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. “Activation functions in neural networks”. In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[40] Ronald W Shephard and Rolf Färe. “The law of diminishing returns”. In: *Production Theory: Proceedings of an International Seminar Held at the University at Karlsruhe May–July 1973*. Springer. 1974, pp. 287–318.

[41] Robert S Siegler. “Three Aspects of Cognitive Development”. In: *Cognitive psychology* 8.4 (1976), pp. 481–520.

[42] Surya A Singaraju, Dennis D Weller, Thurid S Gspann, Jasmin Aghassi-Hagmann, and Mehdi B Tahoori. “Artificial Neurons on Flexible Substrates: A Fully Printed Approach for Neuromorphic Sensing”. In: *Sensors* 22.11 (2022), p. 4000.

[43] Nihal Sinnadurai and K Wilson. “The Aging Behavior of Commercial Thick-film Resistors”. In: *IEEE Transactions on Components, Hybrids, and Manufacturing Technology* 5.3 (1982), pp. 308–317.

[44] Enrico Sowade, Maxim Polomoshnov, and Reinhard R. Baumann. “The design challenge in printing devices and circuits: Influence of the orientation of print patterns in inkjet-printed electronics”. In: *Organic Electronics* 37 (2016), pp. 428–438. ISSN: 1566-1199.

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[46] Mohammadamin Tavakoli, Forest Agostinelli, and Pierre Baldi. “Splash: Learnable activation functions for improving accuracy and adversarial robustness”. In: *Neural Networks* 140 (2021), pp. 1–12.

[47] Athanasios Tsanas and Angeliki Xifara. “Accurate Quantitative Estimation of Energy Performance of Residential Buildings Using Statistical Machine Learning Tools”. In: *Energy and Buildings* 49 (2012), pp. 560–567.

[48] David A Van Dyk and Xiao-Li Meng. “The art of data augmentation”. In: *Journal of Computational and Graphical Statistics* 10.1 (2001), pp. 1–50.

[49] Dennis D Weller, Michael Hefenbrock, Michael Beigl, Jasmin Aghassi-Hagmann, and Mehdi B Tahoori. “Realization and training of an inverter-based printed neuromorphic computing system”. In: *Scientific reports* 11.1 (2021), p. 9554.

[50] Jin Xu, Zishan Li, Bowen Du, Miaomiao Zhang, and Jing Liu. “Reluplex made more practical: Leaky ReLU”. In: *2020 IEEE Symposium on Computers and communications (ISCC)*. IEEE. 2020, pp. 1–7.

[51] Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Aging-Aware Training for Printed Neuromorphic Circuits”. In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–9.

[52] Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Highly-dependable printed neuromorphic circuits based on additive manufacturing”. In: *Flexible and Printed Electronics* 8.2 (2023), p. 025018.

[53] Haibin Zhao, Brojogopal Sapui, Michael Hefenbrock, Zhidong Yang, Michael Beigl, and Mehdi B Tahoori. “Highly-Bespoke Robust Printed Neuromorphic Circuits”. In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2023, pp. 1–6.

5 Practicality Design

Printed electronics (PE) exhibits unique properties like stretchability [35], porosity [20], bio-compatibility [24], and cost-effectiveness, making pNCs the ideal candidate for integrating computing ability and smartness into edge products such as the packaging of fast-moving consumer goods [1], product labels [6], and bandages [41]. However, for effective implementation of pNCs in these areas, only the classification accuracy and device reliability (as described in previous chapters) are insufficient. The product must also address key concerns of the target customers (either consumer or business). For instance, whether the additional cost from pNCs is sufficiently low; given the pNCs are generally integrated in disposable electronics, whether they provide meaningful battery lifetime; and considering the large feature size of PE, whether pNCs can be easily embedded in small scale scenarios.

This chapter studies, models these critical issues, and proposes corresponding algorithmic solutions to address these problems to facilitate the real-world deployments of pNCs. Notably, due to the *no free lunch theorem in optimization* [49], parametric optimization of the pNCs can only provide a trade-off between the utility factors and classification accuracies. Therefore, this chapter employs Pareto-analysis [44] to facilitate Pareto-optimization under varying requirements on hardware or classification accuracy.

5.1 Split Manufacturing for Ultra-low Cost

Although the additive manufacturing strategy of PE can already significantly reduce the manufacturing cost of pNCs, different manufacturing technologies still possess unique and distinct technical characteristics. Therefore, by combining the advantages of various printing technologies, the printing cost of pNCs may be further reduced. Fortunately, the additive process of PE allows

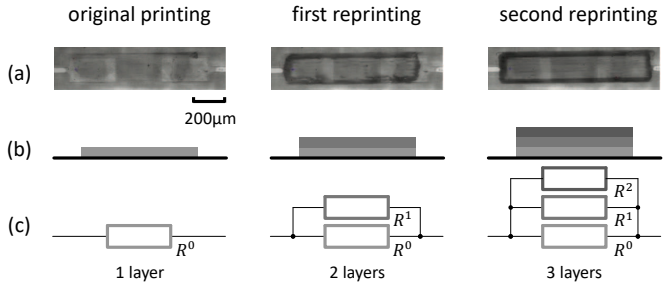


Figure 5.1: Resistor reprinting by adding layers: (a) microscope photos, (b) physical schematics, and (c) circuit diagrams. Sourced from [52].

for the effective integration of multiple printing technologies by simply layering materials printed by different methods on top of each other.

As introduced in Chapter 2.1.1, replication printing technologies, such as gravure and screen printing (Figure 2.2, bottom part), are designed for high-volume production. Gravure printing involves engraving a pattern onto a cylinder, which can then produce a large number of circuits by rotating the cylinder. Screen printing uses a stencil with hollowed-out patterns, and a squeegee to apply the printing material onto the substrate. These methods are efficient and cost-effective for mass production, as the masks (cylinder or stencil) are reusable but difficult to modify once created. In opposite, jet-printing technologies (Figure 2.2, top part) like inkjet printing are better suited for individual manufacturing. It allows for highly customizable and bespoke printing routes and materials, resulting in diverse functionalities but at the cost of higher production times and variable costs per circuit.

This section aims to combine both high- and low-volume printing technologies to further reduce the printing cost for pNCs. Specifically, high-volume methods are employed to print a large common part of all pNCs to ensure the efficiency and low cost, while low-volume but highly flexible printing process will be utilized to do the point-of-use correction of each individual pNC. In this way, both printing cost and classification accuracy of the pNCs can be guaranteed.

5.1.1 Resistor Reprinting

Thanks to the additive manufacturing characteristic of PE, printed circuits can be easily adjusted post-fabrication through adding material or modifying the geometric shape of a component. Figure 5.1 shows a printed resistor with additional layers of conductive ink added post-fabrication to adjust its conductivity. For brevity, this procedure will be referred to as *reprinting* in the following.

As shown in Figure 5.1, resistor reprinting can be seen as printing an additional, parallel conductive path, with the total conductance being the sum of the conductances of the each print, i.e.,

$$\frac{1}{R} = \sum_i \frac{1}{R^i}.$$

Thus, the reprinted corresponding surrogate conductance can be denoted by

$$\theta = \sum_i \theta^i,$$

As the resulting conductance can be split into multiple sub-conductances, this work proposes to produce the conductance via an initial fabrication step (high-volume) followed by an individual customization via, e.g., inkjet-printing. Thus, the surrogate conductance can be expressed as $\theta = \theta^C + \theta^I$. Here, θ^C (common surrogate conductance) denotes the conductance of the initial printing. As it is fabricated with a high-volume process, it is shared by all circuits fabricated with the same mask. Subsequently, θ^I (individual surrogate conductance) denotes the conductance value of the additional material printed to customize the device in a reprinting step. In the following, we refer to the vector $\boldsymbol{\theta}$ as the summary of all conductance values in a pNC. Analogously, $\boldsymbol{\theta}^C$ and $\boldsymbol{\theta}^I$ summarize their corresponding common and individual conductance values.

5.1.2 Training Framework for Multiple pNCs

Reprinting allows to combine high- and low-volume technologies. To find a large and common part in a set of pNCs and the perspective individual parts that retain acceptable classification accuracies, we introduce a training framework that allows to train multiple pNCs for different tasks simultaneously. The

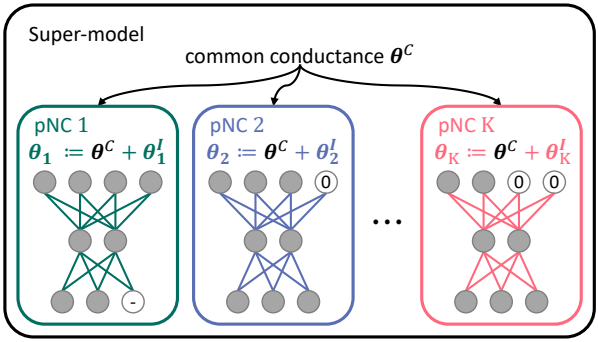


Figure 5.2: Structure of learnable parameters in an exemplary super-model for joint training of multiple pNCs. Each sub-model has its own individual conductance $\boldsymbol{\theta}_k^I$, while the common conductance $\boldsymbol{\theta}^C$ is shared across all sub-models. The resulting conductance of the pNC for the k-th task $\boldsymbol{\theta}_k$ equals $\boldsymbol{\theta}^C + \boldsymbol{\theta}_k^I$ and determines the resulting weights in the pNC. The input/output layers of all pNCs in a super-model are padded to the same dimensionality. The inputs denoted by 0 will be connected to ground. Sourced from [52].

framework considers a decomposition of the conductances of multiple pNCs into a common part, which is shared across all pNCs, and individual parts, which vary among the pNCs of different tasks. By fabricating the circuits with combined technologies, costs and production time can be saved, while attaining comparable performances for the circuits.

Super training model. When different pNCs should be fabricated for a set of different tasks $k = 1, \cdots, \mathrm{K}$, we need to train K different pNCs to address them. If these pNCs are trained independently, little commonality can be expected between them. Thus, there is little potential for joint production and split manufacturing. To increase this potential, the training of these pNCs should be done jointly. For this purpose, we introduce the super-model for training pNCs of different tasks jointly to achieve a high commonality and thus high potential for joint production via split additive manufacturing. For a set of K tasks, a super-model has a set of parameters $\boldsymbol{\theta}^C$, which is shared among all pNCs, and

several sets of parameters $\boldsymbol{\theta}_k^I$ with $k = 1, \cdots, \mathrm{K}$ that are unique to each individual task k. The (surrogate) conductance vector of a pNC trained for task k is then implicitly determined by $\boldsymbol{\theta}_k := \boldsymbol{\theta}^C + \boldsymbol{\theta}_k^I$. A conceptual illustration can be seen in Figure 5.2. Naturally, for this to work, the architecture of all pNCs need to be compatible in the sense that they have the same number of input, hidden, and output neurons. To address this, we take the maximal number of input/output among all pNCs as the number of input/output for all pNCs (see Figure 5.2). For tasks with fewer inputs, zero-padding is employed in training, which relates to connecting those inputs to 0 V (GND) in the circuits. As for output, irrelevant outputs can be simply ignored in the classification tasks.

Constraints. To achieve valid pNCs, several constraints of the printing technology need to be respected. Firstly, as $\boldsymbol{\theta}^C$ and $\boldsymbol{\theta}_k^I$ are independent printing through different methods, each element of them should follow the range of printable conductance values i.e., $[-\mathrm{g}_{\max}, -\mathrm{g}_{\min}] \cup \{0\} \cup [\mathrm{g}_{\min}, \mathrm{g}_{\max}]$, where $\mathrm{g}_{\max}$ and $\mathrm{g}_{\min}$ depend on the specific technology and $\theta = 0$ refers to no printing. Beyond this, we also have to consider that reprinting can only increase the conductances from their original values, i.e., $|\boldsymbol{\theta}_k| \geq |\boldsymbol{\theta}^C|$ for any task k. In other words, we cannot change the choice of connecting either V_i or $\mathrm{neg}(V_i)$, to adjust what would relate to the sign of the weights via reprinting. To respect this constraint, $\boldsymbol{\theta}^C$ determines the signs of the entries of $\boldsymbol{\theta}_t$ via

$$\boldsymbol{\theta}_k := \mathrm{sign}(\boldsymbol{\theta}^C) \cdot |\boldsymbol{\theta}_k^I| + \boldsymbol{\theta}^C,$$

while $\boldsymbol{\theta}_k^I$ is only able to adjust the absolute value of the resulting conductances.

Training objective. Through the proposed training framework, a connection between different tasks is established via the common $\boldsymbol{\theta}^C$. However, this formulation does not necessitate high commonality between the individual pNCs. For example, the solution after training may likely have $\boldsymbol{\theta}^C = \mathbf{0}$ and express everything via the uncoupled $\boldsymbol{\theta}_k^I$. Therefore, to encourage high commonality, we add a penalty term to the training objective to keep the individual conductances $\boldsymbol{\theta}_k^I$, and thus the reprinting effort, low. In this work, the penalty term is

formulated as the ℓ_1 norm of all the individual conductances $\boldsymbol{\theta}_k^I$, i.e.,

$$C(\boldsymbol{\theta}^I) = \sum_k \left\|\boldsymbol{\theta}_k^I\right\|_1,$$

because both printing times and the amount of the printing materials of the individual printing are approximately proportional to the size of the entries of $\boldsymbol{\theta}_k^I$.

Consequently, the training objective of the super pNC considering both accuracy and reprinting costs is then given by

$$\mathcal{L}(\boldsymbol{\theta}^C, \boldsymbol{\theta}^I) = (1-\mu)\sum_k L(\mathcal{D}_k, \boldsymbol{\theta}^C, \boldsymbol{\theta}_k^I) + \mu \cdot C(\boldsymbol{\theta}^I),$$

where $L(\cdot)$ cross-entropy loss and $\mathcal{D}_t$ denotes the training data of the k-th task. Furthermore, the coefficient $\mu \in [0,1]$ denotes a hyperparameter adjusting the influence of the costs $C(\boldsymbol{\theta}^I)$. Notably, we do not include the nonlinear circuits $\boldsymbol{q}$ as learnable parameters, because with some initial experiments, we have concluded that the learning of $\boldsymbol{q}$ does not significantly improve the effectiveness of the proposed method.

For $\mu = 0$, the training objective is unaffected by the cost $C(\boldsymbol{\theta}^I)$ of the reprinting required for each task. In this case, the training of the super pNC can be conceptually equated to the completely independent training of K pNCs for K tasks. Consequently, there may be little commonality between the different pNCs and thus little potential for a sensible production of $\boldsymbol{\theta}^C$ via a high-volume production process. However, the individual pNCs may also achieve the best accuracy as they are not bound together. On the other hand, for high values of μ (i.e., $\mu \to 1$), the cost $C(\boldsymbol{\theta}^I)$ will completely dominate the loss term in training. This should lead to a solution where $\forall k : \boldsymbol{\theta}_k^I = \mathbf{0}$. In this case, the individual pNCs are solely determined by $\boldsymbol{\theta}^C$, and are thus all the same. While this is the most economical in terms of production costs, the resulting pNCs likely provide no useful accuracy for their respective tasks. Thus, μ may be used to express a trade-off between cost and accuracy. To find an appropriate value of μ, we suggest training the super pNC for different values of μ and draw a Pareto-front.

5.1.3 Experiment

We implement[1] the proposed method, and conduct an experiment with 30 benchmark datasets, whose complexities and use cases match the PE and pNC profile. The results are additionally analyzed regarding the accuracy-cost trade-off by generating a Pareto-front of possible solutions.

Datasets. Unlike the default datasets listed in Table 4.1, we have to collect more datasets to test the effectiveness of the proposed method on a large number of datasets. Specifically, a subset of the 121 classification benchmark datasets summarized in [11] was taken. We select datasets which are suitable for PE and pNCs, most notably, tasks with a limited number of inputs and outputs (≤ 10). Moreover, since numerous pNCs are trained simultaneously, we limit the experiments to datasets with the number of data points between 100 and 1 000, which leaves 30 datasets. Finally, we scale all the inputs to $[0,1]$ to simulate the electrical signals from sensors and put all inputs on the same scale. We split each dataset into training (60%), validation (20%), and test (20%) sets. The detailed information about the datasets can be found in Table 5.1.

Experiment setup. As described in previous section, the architecture of all pNCs is determined by the maximal number of input and output of all the datasets. In this experiment, the architecture is chosen to be 9-3-8, where the number of inputs and outputs are determined by the datasets. For training, still follow the default gradient-based training configuration introduced in Chapter 4.1.3. Additionally, to investigate the trade-off between accuracy and cost, we select 50 different $\mu \in [0,1]$ with equidistant.

The training is repeated 30 times (with seeds varying from 1 to 30) for different initialization for each value of μ to make sure to achieve a sufficiently good solution for each value of μ.

Baseline. As the baseline, we report the performance of the pNCs with $\mu = 0$, which equivalent to train the pNC considering only individual $\boldsymbol{\theta}^I$. This result

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

Table 5.1: Benchmark datasets and baseline accuracy for split additive manufacturing tasks. Sourced from [52].

Dataset	**# Input # Output**	**# Data**	**Baseline accuracy**	**Source**
Acute Inflammation	6-2	120	0.904	[8]
Acute Nephritis	6-2	120	0.925	[8]
Balance Scale	4-3	625	0.671	[42]
Blood	4-2	748	0.747	[50]
Breast Cancer	9-2	286	0.724	[58]
Breast Cancer Wisconsin	9-2	699	0.955	[30]
Breast Tissue	9-6	106	0.409	[40]
Ecoli	7-8	336	0.592	[18]
Energy (y_1)	8-3	768	0.816	[47]
Energy (y_2)	8-3	768	0.761	[47]
Fertility	9-2	100	0.857	[13]
Glass Identification	9-6	214	0.439	[12]
Haberman's Survival	3-2	306	0.788	[15]
Hayes-Roth	3-3	132	0.342	[17]
Indian Liver Patient Dataset	9-2	583	0.684	[45]
Iris	4-3	150	0.701	[2]
Mammographic Mass	5-2	961	0.728	[10]
MONK's Problem 1	6-2	124	0.598	[46]
MONK's Problem 2	6-2	169	0.617	[46]
MONK's Problem 3	6-2	122	0.576	[46]
Pima Indians Diabetes	8-2	768	0.644	[43]
Pittsburgh Bridges MATERIAL	7-3	106	0.902	[39]
Pittsburgh Bridges SPAN	7-3	92	0.509	[39]
Pittsburgh Bridges T-OR-D	7-2	102	0.800	[39]
Pittsburgh Bridges TYPE	7-6	105	0.665	[39]
Seeds	7-3	210	0.454	[5]
Teaching Assistant Evaluation	5-3	151	0.397	[28]
Tic-Tac-Toe Endgame	9-2	958	0.632	[32]
Vertebral Column (2 cl.)	6-2	310	0.635	[3]
Vertebral Column (3 cl.)	6-3	310	0.586	[3]

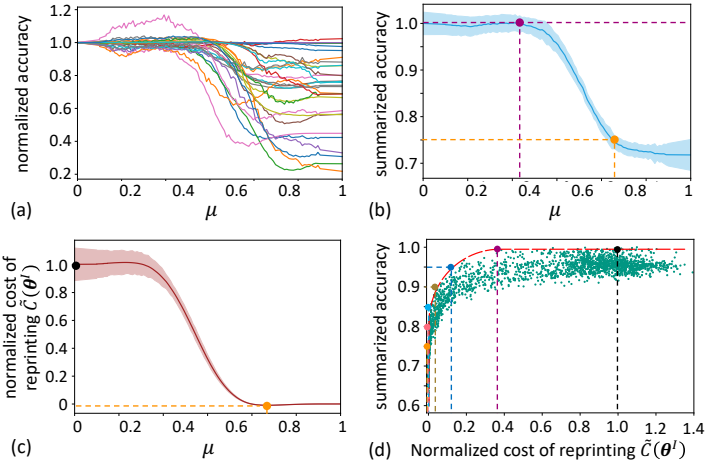


Figure 5.3: Results of experiment with 100 different μ values: (a) normalized accuracies of 30 tasks. Each task is indicated by a different color, (b) summarized accuracies (average normalized accuracies), the blue curve and area denote the mean and standard deviation respectively, (c) normalized cost of the individual (point-of-use) reprinting, the red curve and area denote the mean and standard deviation respectively, (d) scatter plot of summarized accuracy versus cost of reprinting for all the runs. The red curve displays the Pareto-front and the bold points denote different possible trade-offs on the Pareto front. Sourced from [52].

can be regarded as the upper bound of the circuit performance. We also refer to this as *individual* pNCs.

Result. After training, we evaluate the pNCs on the test sets. Table 5.1 reports the accuracies of baseline. To analyze the impact of μ more clearly and to eliminate the disparate difficulties among different tasks, we normalize the accuracy by the baseline through

$$\text{normalized accuracy} = \frac{\text{accuracy}}{\text{baseline accuracy}},$$

where the baseline accuracy should theoretically indicate the best accuracy that can be achieved by the perspective pNCs. Note that this is not always achieved

in practice due to the complex nature of the nonlinear optimization problem that neural network training resembles. The resulting curves are displayed in Figure 5.3(a).

To summarize the overall implications of μ for all the pNCs, we report the *summarized accuracy*, which refers to the average value of the normalized accuracies over all the tasks trained from a super-model. The result is shown in Figure 5.3(b). We also show the relationship between μ and the cost of reprinting in Figure 5.3(c). To obtain the Pareto-front, we plot all the $C(\boldsymbol{\theta}^I)$ versus their summarized accuracy in Figure 5.3(d). Based on the scatters, we draw the Pareto-front as the red dashed line.

As can be seen in Figure 5.3(a)-(c), at $\mu = 0$, which refers to fully individual printing, the summarized accuracy is normalized to 1, and the expected cost of individual reprinting is the highest. Analogous to the treatment of the accuracy, we normalize all the printing costs by the cost at $\mu = 0$, as shown by the black point in Figure 5.3(c), where $\tilde{C}(\boldsymbol{\theta}^I)$ denotes the normalized reprinting cost. With increasing μ, the normalized accuracy, summarized accuracy, and the cost of reprinting $C(\boldsymbol{\theta}^I)$ will decrease. For μ greater than a certain threshold, i.e., $\mu \geq 0.7$ in this experiment, the cost penalty completely dominates the training objective. Thus, as expected, $\boldsymbol{\theta}^I_k = \mathbf{0}$ for all pNCs, and the training results are equivalent to performing all tasks with the same pNCs. We refer to this pNC as *common* pNC. The orange points in Figure 5.3(b), (c) and (d) show the common pNC with the best accuracy.

Contrary to our expectation, the summarized accuracy does not decrease immediately as μ is increased from 0. This could be due to two possible reasons: Firstly, the $C(\boldsymbol{\theta}^I)$ term may act as a regularization and mitigates overfitting to the training set of the specific task. Therefore, the accuracy of some pNCs could even slightly increase, e.g., the pink curve in Figure 5.3(a). Secondly, μ only explicitly affects $\boldsymbol{\theta}^I$ rather than $\boldsymbol{\theta}^C$. Hence, $\boldsymbol{\theta}^C$ would be adjusted accordingly during training to compensate for the loss of accuracy caused by the penalty term. This phenomenon allows reducing the cost of individual printing, while retaining an acceptable accuracy. Such a solution can be found at the purple point in Figure 5.3(b), where $\mu \approx 0.37$.

The Pareto-curve in Figure 5.3(d) reveals the relationship between the cost

for individual configuration and the summarized accuracies of super pNCs. Compared to fully individual printing (black point), the point-of-use printing cost can be reduced to 38.6% without any noticeable loss in accuracy (purple point). Moreover, if the summarized accuracy is allowed to be reduced by e.g., 5%, the reprinting cost can be further reduced to 15.7% (blue point). Finally, using a single *common* pNC (orange point) leads to a 75% summarized accuracy, but consequently also no reprinting costs. Other exemplary trade-off options are summarized in Table 5.2.

Table 5.2: Different trade-offs between reprinting cost and classification accuracy, drawn from the Pareto-front. Sourced from [52].

Printing technology	**Summarized accuracy**	**Reprinting cost** $\tilde{C}(\boldsymbol{\theta}^I)$
individual	100%	100.0%
ours	100%	38.6%
	95%	15.7%
	90%	5.7%
	85%	2.9%
	80%	1.4%
common	75%	0.0%

5.1.4 Discussion

In this section, we propose a design strategy for multiple and different pNCs to leverage the additive manufacturing nature of the PE. Through resistor reprinting technology, the gap between high- and low-volume printing are bridged, and their advantages are combined. The experiment shows that, given pNCs that are trained together (co-designed) with shared conductances, a substantial amount of point-of-use printing cost can be saved, while still obtaining pNCs with sufficient accuracies. This can result in substantial time savings when fabricating pNCs even in case each circuit is different from others and each individual circuit is only required in a small batch. Additionally, depending on the requirements, manufacturers may be able to adjust their production strategy based on the cost-accuracy trade-off visualized by the Pareto-front.

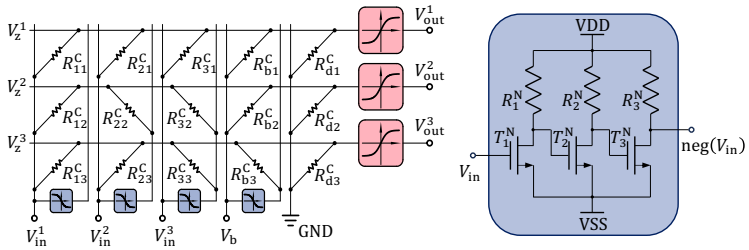
5.2 Power-Efficient Circuit Design

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 [7] or printed energy harvesters [25]. 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 section, we modify the existing circuit structure in a more power-efficient way, and then propose power-aware training for pNC by explicitly integrating power models into the objective function. Specifically, we derive the accurate power models for the circuit primitives in the pNC. Afterwards, by integrating these models into the pNC 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. Further, this section employs augmented Lagrangian method to enable the constraint training with prescribed power budget. This can significantly reduce the training effort compared to drawing the Pareto-front and thus accelerate the circuit design cycle.

5.2.1 Power-Efficient Circuit Structure

In previous design (Figure 2.7), negation 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. To eliminate this redundancy and thus reduce the power, we modified the circuit design, as shown in Figure 5.4(a). With this modified structure, only one single negation 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.

Moreover, as we notice, the previous negation circuit design consumes significantly higher power (in mW) than other primitives (in µW), we propose a new design of the negation circuit, as illustrated in Figure 5.4(b). The new



(a) modified resistor crossbar (b) modified negation circuit

Figure 5.4: Modified pNC design for low power consumption: (a) modified crossbar with each input voltage negated maximally once, sourced from [53], and (b) modified negation circuit consuming lower power, sourced from [54].

negation circuit requires also µW-level power consumption. The feasible design space of the modified negation circuit is reported in Table 5.3.

To validate the new circuit design and assess other characteristics such as latency of the new circuit designs, we performed SPICE simulation with the pPDK [37]. The new circuit structure yields the similar input-output behavior and similar latency as the previous design, however, the power consumption decreases due to the reduced power and reduced number of negation circuits.

Table 5.3: Feasible design space of the modified negation circuit.

	R_1 (kΩ)	R_2 (kΩ)	R_3 (kΩ)	W_1 (µm)	L_1 (µm)	W_2 (µm)	L_2 (µm)	W_3 (µm)	L_3 (µm)
min	250	6	300	80	40	40	30	80	50
max	2000	32	500	100	80	60	40	200	150

5.2.2 Power Consumption Model

Due to the structural simplicity of resistor crossbar arrays, we derive analytical solutions for the power consumption through physics-informed modeling method (Chapter 3.1.1). In contrast, due to the complexity of the nonlinear circuits, we obtain the power models through approximation-based modeling approach (Chapter 3.1.2) with data collected from SPICE simulation.

Power consumption of resistor crossbar. Due to the pure resistivity of the crossbar array (excluding the negation 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 negation circuits can be modeled as

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

where $(\cdot)^2$ denotes an element-wise square operation, moreover $(\cdot)^{\mathrm{Ex}}$ refers to the stack of repeated vectors, i.e.,

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

and

$$\boldsymbol{V}_{\mathrm{z}}^{\mathrm{Ex}} = \begin{bmatrix} \boldsymbol{V}_{\mathrm{z}} \\ \vdots \\ \boldsymbol{V}_{\mathrm{z}} \end{bmatrix} \in \mathbb{R}^{(M+2)\times N}.$$

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

$$\mathcal{P}^{\mathrm{C}} = \mathbf{1}_{M+2}^{\top} \cdot \boldsymbol{P}^{\mathrm{C}} \cdot \mathbf{1}_N, \tag{5.1}$$

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.

Additionally, we can see that, the weights are scale-invariant with respect to the resistances. Thus, the resistances can be scaled up to save power, while the weights remain unchanged. Consequently, for estimating the lowest power for the crossbar with given weights, the resistances are first up-scaled to the highest feasible values, which depends on the printing technology and the latency of

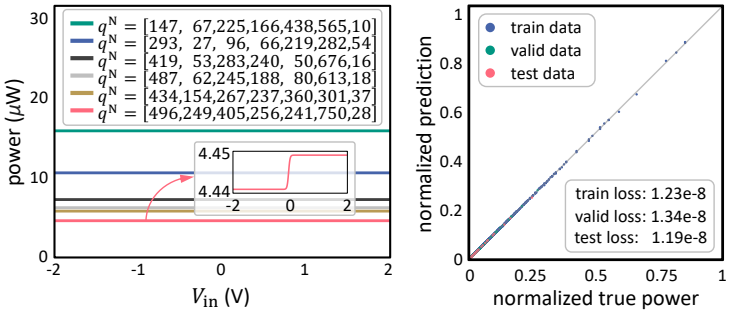


Figure 5.5: Left: Power of some negative weight circuits with input voltages V_{in} ranging from $-2\,\text{V}$ to $2\,\text{V}$, the legend shows the configuration of the circuit components $\boldsymbol{q}^{\text{N}}$, the right bottom box shows the shape of the pink curve. Right: visualization of the results from the surrogate power consumption model. The x-axis and the y-axis refer to the normalized true power and predicted value. Blue, green, and red colors denotes the data from training, validation, and test sets. Sourced from [53].

the circuit. In this work, the maximal feasible resistance has been identified to be $1\,\text{M}\Omega$ through SPICE simulation.

Power consumption of nonlinear circuits. For the nonlinear circuits, i.e., negation circuits and ptanh circuits, estimating the power consumption based on the physical quantities $\boldsymbol{q}^{\text{N}}$ and $\boldsymbol{q}^{\text{A}}$ is challenging. We therefore train ANNs to approximate the power consumption of these circuits based on SPICE simulations. Here, we establish the power consumption models for both ptanh circuit and negation circuit, in addition, as the negation circuit is a newly proposed circuit, we establish its surrogate model for calculating its transfer characteristic curve as well. The modeling process follows the pipeline introduced in Chapter 3.1.2.

The left side in Figure 5.5 exemplifies the power of the negation circuits with different physical quantity values. The input voltage V_{in} ranges from $-2\,\text{V}$ to $2\,\text{V}$. The legend denotes the corresponding circuit configuration $\boldsymbol{q}^{\text{N}}$. It is notable that, although the power varies with changing input voltage, as shown by

the pink curve in the right bottom box, the variation is so small that the power consumption can be regarded as a constant with respect to 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 $-2\,\text{V}$ and $2\,\text{V}$ according to the principle of maximum entropy [19]. Consequently, the expected power consumption P^{N} is represented by the mean value with respect to input voltages.

With these data, we finally obtain a 15-layer ANN as the surrogate power model. The performance of the surrogate model is demonstrated on the right side of Figure 5.5, 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:

$$\mathcal{P} = \mathcal{P}^{\text{C}} + N^{\text{N}} \cdot \mathcal{P}^{\text{N}} + N^{\text{A}} \cdot \mathcal{P}^{\text{A}}, \tag{5.2}$$

where N^{N} and N^{A} denote the number of negation circuits and ptanh circuits. Moreover, $\mathcal{P}^{\text{N}}$ and $\mathcal{P}^{\text{A}}$ are the estimated power consumption from the surrogate power models.

It is notable from Equation (5.2) that, the overall power consumption of the pNC can not only be reduced through lowering the power consumption of the nonlinear circuits themselves, but also through reducing the number of the nonlinear circuits. Fortunately, PE natively allows for such highly flexible printing patterns, meaning that, whenever arbitrary subcircuits are modified or excluded, PE can be easily adjusted to the updated circuits through simply modifying its printing trajectories and without costly extras.

However, since we primarily 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} and N^{A} in Equa-

tion (5.2), representing the number of negation circuits and ptanh circuits, depends on $\boldsymbol{\theta}$ but represents a piece-wise constant function. Specifically, the count of negation circuits N^{N} in a M-input N-output crossbar array is expressed by

$$N^{\mathrm{N}} = \mathrm{column\,max}\left\{\mathbb{1}_{\{\boldsymbol{\theta}<\mathbf{0}\}}\right\} \cdot \mathbf{1}_{M+2}, \tag{5.3}$$

where $\mathrm{column\,max}(\cdot)$ returns the column-wise maximum values, because each column in $\boldsymbol{\theta}$ in Figure 5.4(a) is relating to one input voltage. If the whole column of surrogate conductances is positive, the corresponding input voltage does not require to be passed through any negation circuit. Similarly, N^{A} is calculated through

$$N^{\mathrm{A}} = \mathbf{1}_N^{\top} \cdot \mathrm{row\,max}\left\{\mathbb{1}_{\{|\boldsymbol{\theta}|>\mathbf{0}\}}\right\}, \tag{5.4}$$

meaning that, if any weight corresponding to a ptanh activation circuit (i.e., a row of resistors) is non-zero, the ptanh circuit should be printed to activate the input. In contrast, if all the weights corresponding to the ptanh circuit are zero-valued, the corresponding ptanh circuit can be removed from printing.

Evidently, the count of the devices, i.e., the $\mathbb{1}_{\{\cdot\}}$ function, is a piece-wise constant function. To address this issue and enable the optimization of N^{N} and N^{A} through $\boldsymbol{\theta}$, we soft-count of the nonlinear circuits as shown in Figure 3.5. Specifically, we relax the function by a $\mathrm{sigmoid}(\cdot)$, denoted by $N^{\mathrm{N}}_{\mathrm{soft}}$, $N^{\mathrm{A}}_{\mathrm{soft}}$. In the forward pass of the soft-count, $N^{\mathrm{N}}_{\mathrm{soft}}$ and $N^{\mathrm{A}}_{\mathrm{soft}}$ are still calculated by Equation (5.3) and Equation (5.4), however, in the backpropagation, relaxed functions,

$$\mathrm{column\,max}\left\{1-\mathrm{sigmoid}(\boldsymbol{\theta})\right\} \cdot \mathbf{1}_{M+2}^{\top},$$

and

$$\mathbf{1}_N^{\top} \cdot \mathrm{row\,max}\left\{\mathrm{sigmoid}(|\boldsymbol{\theta}|)\right\},$$

are employed to generate the gradient for updating $\boldsymbol{\theta}$ for the counts of negation circuits and ptanh circuits respectively.

By replacing N^{N} and N^{A} in Equation (5.2) with soft-counts, the resulting power estimation of the printed neuron can be formulated as

$$\mathcal{P} = \mathcal{P}^{\mathrm{C}} + N^{\mathrm{N}}_{\mathrm{soft}} \cdot \mathcal{P}^{\mathrm{N}} + N^{\mathrm{A}}_{\mathrm{soft}} \cdot \mathcal{P}^{\mathrm{A}}. \tag{5.5}$$

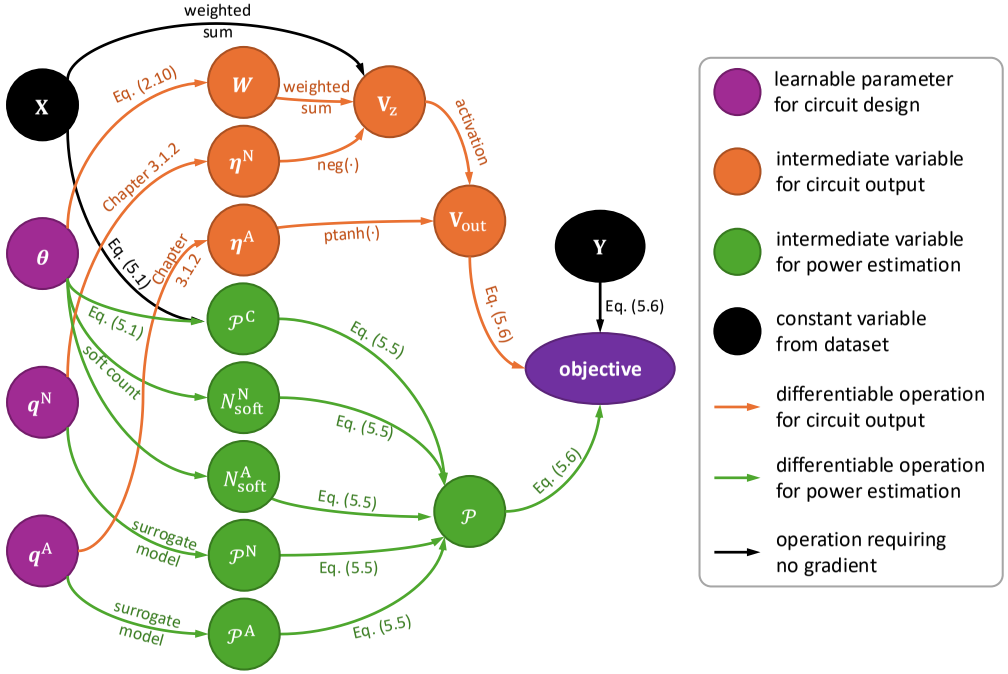


Figure 5.6: Computational graph of the power-aware training of the pNC within one neuron. The orange part refers to the classification related variables, while the green part denotes the power consumption related variables introduced in this section. Sourced from [53].

The computational graph for the complete power estimation is shown by the green part in Figure 5.6. Note that this figure only represents the computational graph for one neuron. In case multiple neurons are adopted, the $\boldsymbol{V}_{\text{out}}$ 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 of the pNCs. Moreover, the power consumption of all neurons will be summed up, serving as the final estimate for the power consumption. It can also be seen that, the changes in $\boldsymbol{q}^{\text{N}}$ and $\boldsymbol{q}^{\text{A}}$ not only impact the circuit power, but also influence their transfer characteristics, and thus, the accuracy of the classification. Therefore, $\boldsymbol{q}^{\text{N}}$ and $\boldsymbol{q}^{\text{A}}$ can not be simply chosen to provide lower power consumption, but also have to be jointly considered with the classification accuracy. This justifies to propose an appropriate objective that can involve both metrics during training.

5.2.3 Power-Aware Training

For nominal training of pNCs, we typically employ cross-entropy [31] as the loss function to ensure the classification accuracy. However, to jointly optimize both classification accuracy and power consumption, power-aware training objective should be considered.

Power-aware training with penalty method. A naive approach is to add two terms with a balance factor μ:

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) = (1-\mu) \cdot L(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}) + \mu \cdot \mathcal{P}(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{q}), \tag{5.6}$$

where $L(\cdot)$ denotes the cross-entropy loss and $\mu \in \mathbb{R}^+$ denotes a scaling factor to express the trade-off between loss and power consumption. If $\mu = 0$, the training objective entirely corresponds to the accuracy of the classification tasks. In this case, the trained pNC 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 $\mu = 1$, power $\mathcal{P}$ dominates the training objective whereas the accuracy is disregarded. Therefore, the trained pNCs 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 μ, and, considering that a specific trade-off will be chosen based on different application scenarios, we decide to train pNCs with different $\mu \in [0, 1]$ and construct a Pareto-front [44] to facilitate the selection of various trade-offs with Pareto-optimality.

Power-aware training with augmented Lagrangian. There is an obvious drawback of the penalty method: in practice, circuit design typically requires ensuring a prescribed power consumption to guarantee factors such as the circuit operation time. However, attempting to determine the optimal circuit classification accuracy for a given power consumption by plotting the Pareto-front necessitates hundreds or even thousands of trainings. To address the problem, we introduce the augmented Lagrangian method (as described in Algorithm 3 and Algorithm 4), which can constrain the power consumption of the pNC to a predefined value (or less) with one single training.

Fine-tuning. It is worthy to highlight that pNCs trained with either penalty method (for Pareto-front) or the augmented Lagrangian (for constrained training) may not reach optimal trade-off between power and accuracy. Because the power consumption term, functioning as a penalty, suppresses the conductances through *soft-count* for decreasing the device counts, and thus circuit power. However, if, e.g., any input conductance of a neuron cannot be all suppressed to zero for a given μ, the corresponding ptanh circuit can not be removed. In this case, such suppression on the input conductances through $N^{\mathrm{A}_{\mathrm{soft}}}$ not only fails to reduce the count of ptanh circuits, 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^{C} for each surrogate conductance θ and multiply them to indicate the parameters after removing the useless components (i.e., the resistors with zero conductances, the ptanh circuits with all input conductances being zero, and the negation circuits

for the voltages that do not need to be negated), i.e.,

$$\theta \leftarrow m^{\mathrm{C}} \cdot \theta,$$

where m^{C} is either 1 or 0, indicating whether the parameter is removed.

Regarding the removal of ptanh circuits, if all input parameters of a neuron are removed, the output voltage of this neuron will be multiplied with a mask value equaling 0, i.e.,

$$a \leftarrow m^{\mathrm{A}} \cdot a,$$

where a refers to the activated value by ptanh circuit, and m^{A} is mask being either 1 or 0. If a ptanh circuit is removed, its corresponding mask will be 0 to emulate a 0 V open circuit.

As for the negation circuits, we introduce

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

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

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

5.2.4 Experiment

To evaluate the effectiveness of the power-aware training of pNCs, we implemented the proposed approaches[2][3] with PyTorch [36] and conduct experiments on the benchmark datasets described in Table 4.1.

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

Experiment setup. For both Pareto and augmented Lagrangian approaches, we employ gradient-based training and follow the default pipeline employed in Chapter 4.1.3 for basic training. Subsequently, in fine-tuning, as it can be seen as a warm-start near the optimum, we select a smaller initial learning rate as 0.01. Other setups in fine-tuning are kept the same as the default setup. Regarding augmented Lagrangian, we utilize the inequality version, because the power consumption of pNCs is generally a discrete value due to the discrete device counts. Thus, equality constraints are hardly to be guaranteed.

To investigate the trade-off between accuracy and power, we uniformly select 50 values in $\mu \in [0,1]$ to draw the Pareto-front as the benchmark of the power-accuracy problem. In augmented Lagrangian, we take 20%, 40%, 60%, 80% of the maximal power consumption of the pNCs as the constraints.

Result. After training, we evaluate the trained pNCs on the test sets. In order to obtain the Pareto-front, we plot the entirety of powers versus their respective accuracies for all runs (random seeds) and all values of μ by the cyan points in Figure 5.7. Subsequently, we can delineate the Pareto-front by the valid pink curve.

The Pareto-front illustrates the relationship between power and accuracy. In comparison to power-unaware training (i.e., conceptually the top right corner maximal accuracy and maximal power consumption), the power can be slightly reduced without any accuracy loss. This is because

1. The original pNC architectures for the target datasets might be redundant. Therefore, after removing few of the components, the accuracy does not reduce.
2. The power term functions as a regularization term, which can avoid overfitting on training data, and thus improve the classification accuracy. This may compensate the accuracy loss due to the penalty on power.

Furthermore, this approach enables other Pareto-optimal trade-offs for any given power and accuracy. These trade-offs can be chosen in consideration of the specific design requirements and application contexts.

However, in real circuit design, there is generally a prescribed power budgets. This can be ensured by the augmented Lagrangian methods during train-

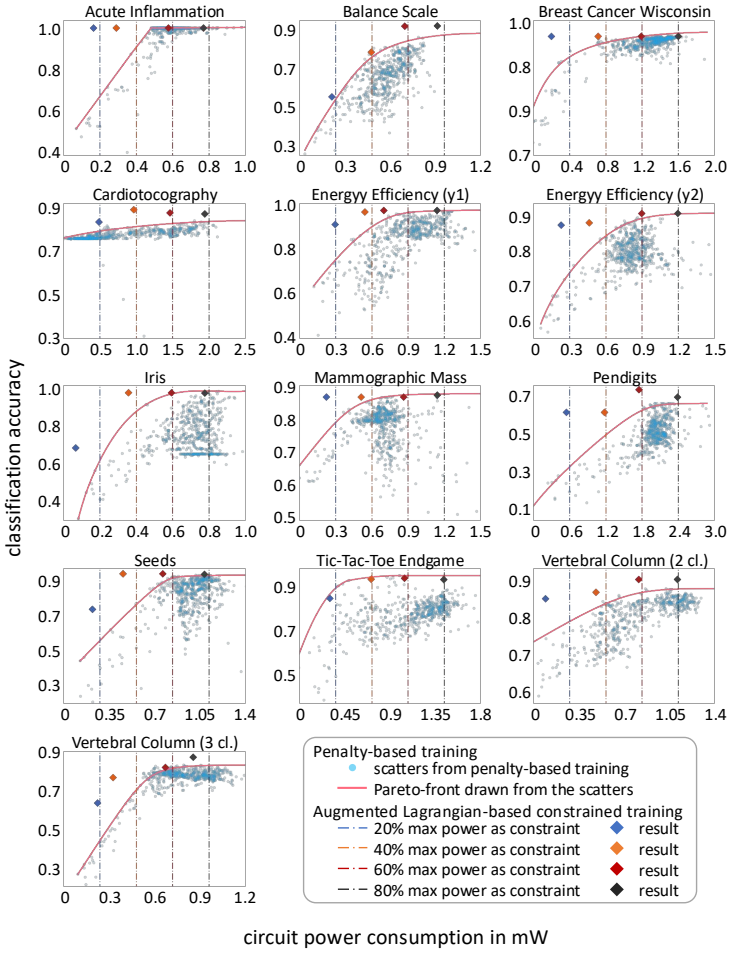


Figure 5.7: Classification accuracy and power consumption of pNCs from penalty-based and augmented Lagrangian-based training approaches. The blue scatters are results from penalty-based training, while the pink curves are Pareto-fronts drawn from the scatters. The vertical lines indicate the power constraints in the augmented Lagrangian approach, whereas the rhombus with the same color refer to the results from the augmented Lagrangian method.

ing. The vertical lines in Figure 5.7 show the predefined power constraints during training. They are namely 20% (blue), 40% (orange), 60% (red), 80% (black) of the maximal power consumption of the perspective datasets. The corresponding training results are scattered by the diamond symbols with the identical colors. It can be seen, the augmented Lagrangian approach can effectively ensure the inequality constraints (i.e., the points locate on the left side of the vertical lines), while achieving comparable or even better results than the optimalities trained from penalty-based objective. We speculate two main reasons for this

1. In the penalty method, when the balance factor of power consumption significantly outweighs that of the cross-entropy loss to encourage low power consumption, the optimization problem may become ill-conditioned. This is why the augmented Lagrangian outperforms penalty-based method especially in case of low power trade-offs.
2. The penalty method can be explained as Lagrangian method through

$$
\begin{aligned}
&\underset{\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}}\ (1-\mu)\cdot L(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})+\mu\cdot\mathcal{P}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})\\
=\ &\underset{\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}}\ (1-\mu)\cdot L(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})+\mu\cdot\mathcal{P}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})-\mathcal{C}\\
=\ &\underset{\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}}\ L(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})+\underbrace{\frac{\mu}{(1-\mu)}}_{=:\lambda^*}\Big(\mathcal{P}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})\Big)-\frac{1}{(1-\mu)}\mathcal{C}\\
=\ &\underset{\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}}\ L(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})+\lambda^*\Big(\mathcal{P}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})-\underbrace{\frac{1}{\mu}\mathcal{C}}_{\mathcal{C}'}\Big)\\
=\ &\underset{\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}}\ L(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})+\lambda^*\Big(\mathcal{P}(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})-\mathcal{C}'\Big).
\end{aligned}
$$

   In other word, for any given μ, the training is likely to converge at the point with $\lambda^* = \frac{\mu}{1-\mu}$ being the Lagrangian multiplier, and a certain $\mathcal{C}'$ being the constraint on power consumption. However, the given $\lambda^* = \frac{\mu}{1-\mu}$ may not correspond to any specific $\mathcal{C}'$, meaning that λ^* is not a Lagrangian multiplier for any constraint $\mathcal{C}'$. Consequently, the training result does not lie on the Pareto-front.
3. Even if λ^* is a valid Lagrangian multiplier associated with a specific

and existing constraint $\mathcal{C}'$, whose corresponding solution will locate on the Pareto-front, the gradient-based method is not guaranteed to converge to that solution. Because according to the KKT condition, the optimal solution of the Lagrange equation is only necessarily to have zero-gradient, while its Hessian (second-order derivative) can be either positive-definite or not. In contrast, the gradient method converges only at points where the gradient is zero and the Hessian is positive-definite. Therefore, in case the Hessian of the optimum of the Lagrangian equation is not positive-definite, the gradient method fails to achieve.

5.2.5 Discussion

In this section, we target the design of power-efficient pNCs. By establishing physics-informed and approximation-based power consumption models, the circuit power can be explicitly incorporated into the design objective of the pNCs. Thanks to the highly flexible processing of PE, device parameters and even circuit structures can be easily adjusted to realize low power pNCs. By introducing a variable trade-off factor in the training process of pNCs, a Pareto-front can be drawn, from which any optimal trade-offs between accuracy and power can be chosen according to specific requirements or application scenarios. We further propose an augmented Lagrangian-based training method, that can avoid the costly training of the penalty-based method, and can achieve even better results due to its better formulation of the optimization problem.

5.3 Highly Compact Circuit Design

Despite the unique advantages of PE and pNCs, pNCs still face the challenges inherent to PE, i.e., large feature sizes and low device counts. Such drawback imposes considerable limitation when applying pNCs to scenarios with constrained areas, such as smart band-aids [41] or compact smart packaging [1]. Therefore, the architectural topology of the pNC, especially the number of neurons and connections, need to be considered in such area-scarce applications.

To enable effective training for compact pNCs, it is imperative to explicitly incorporate the circuit area into the training objective. In this work, the circuit

area A is estimated the area of the individual devices and their counts. Since device count is an integer variable related to circuit topology, considering the area in the training objective necessitates training algorithms that are capable of topology optimization. To this end, we employ an EA-based method for the simultaneous training of both crossbar conductances (weights) and circuit topologies (neural architecture). The algorithmic details were introduced in Chapter 3.2.2.

5.3.1 Modeling of Circuit Footprint Area

To achieving the compact circuit design, it is required to establish a precise model for estimating the circuit area. Therefore, we first build the circuit area model that can estimate circuit footprint of the pNCs from their architecture. Since the trained circuits need to go through a placement and a routing process before area estimation, the relationship between circuit netlist and the final area exhibits high complexity. Usually, 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. Consequently, we employ approximation-based method (Chapter 3.1.2) to precisely model the circuit area estimator.

Area of Circuit Primitive. The schematics of a printed resistor is depicted in Figure 5.8(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 remaining unchanged. Moreover, Figure 5.8(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 [48] as $A^{\mathrm{R}} = 0.15\,\mathrm{mm}^2$, $A^{\mathrm{N}} = 22\,\mathrm{mm}^2$ and $A^{\mathrm{A}} = 30\,\mathrm{mm}^2$, correspondingly.

Software for Circuit Area Calculation. We utilize mature and well-developed commercial electronic design automation (EDA) tools to facilitate automatic

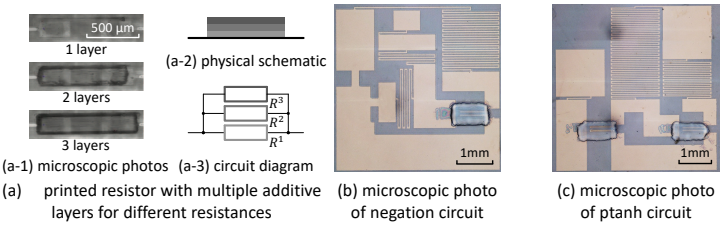


Figure 5.8: Schematics and photos of the primitives in pNCs. (b) and (c) sourced from [48] with permission for reprinting.

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[4] tool for the automatic placement and routing of pNCs.

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 5.9(c), we can print insulating materials, in our setup, 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 PCBs 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.

Circuit Area Estimator. As it is hard to integrate the EasyEDA into the training approach of the pNCs, 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

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

adopt an ANN-based model as the area estimator, because it is proven to be a universal approximator. The approximation-based 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 5.8, which includes dimensions and pin configurations. Subsequently, we randomly generate 500 different pNCs 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 5.9. After this, we use the minimum bounding rectangles for each circuit to denote the area footprint of the pNCs. Since the algorithm provided by EasyEDA is not deterministic, i.e., each conduction may produce a different result, we repeated the algorithm ten times per pNC, and record their bounding box 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

$$A = \text{AreaEstimator}(N^{\text{A}}, A^{\text{A}}, N^{\text{N}}, A^{\text{N}}, N^{\text{R}}, A^{\text{R}}).$$

  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 [21] and widely used as generative AI models [34]. 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

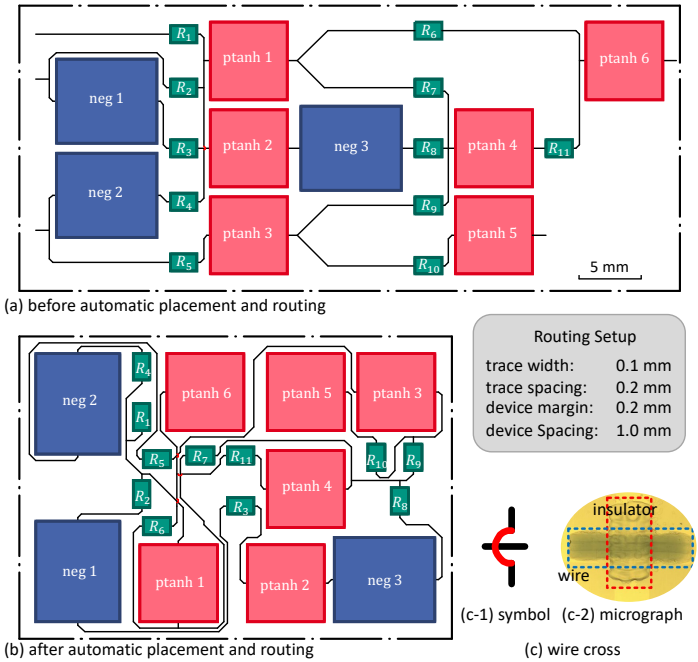


Figure 5.9: Placement and routing of the pNCs. (a) A naive placement that mimics the form of neural networks described in Figure 5.11(b), and (b) the solution of the automatic placement and routing from EasyEDA software. The gray box shows the major setups of the algorithm (with technology specification 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 DMSO (dimethyl sulfoxide) as the insulator, sourced from [38].

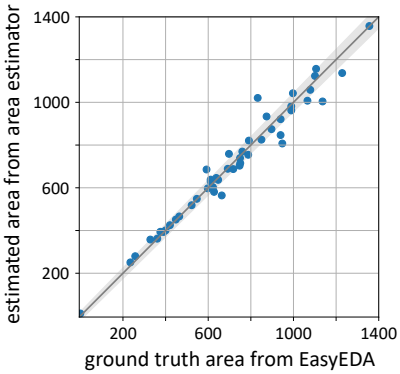


Figure 5.10: Performance of the variational autoencoder (VAE)-based area estimator on the test data. The x-axis is the ground truth area from the EasyEDA, while the y-axis denotes the estimated areas. Each blue point is a test data. The gray diagonal line refers to the ideal case where the estimation equals the ground truth. The gray area around the line represents the variation of the ground truth areas.

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 pNCs. Figure 5.10 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.

5.3.2 Area-Aware Training

Although this chapter primarily considers EA as the primary method for the compact design of pNCs, there are also gradient-based methods as viable solution for the architectural design. Therefore, this chapter also reviews, analyzes, and conducts experiments on gradient-based methods, assessing their potential and effectiveness in optimizing circuit architecture, and serve as the contrast to EA-based methods.

EA-based training. The fundamental EA algorithm has already been introduced in Algorithm 2. As it can be directly utilized for the compact pNCs design, we only do minor modification regarding the objective (fitness) function and the initialization of the pNC architecture.

As for training objective for the classification accuracy, we employ the combination of the cross-entropy [31] and the classification accuracy as introduced in Equation (3.3), namely:

$$\mathcal{O}(\mathcal{D}, \text{  }) = \text{CE}\left(\mathcal{D}, \text{}\right) - \text{ACC}\left(\mathcal{D}, \text{}\right).$$

With the consideration of the circuit area, the overall objective is defined as

$$\underset{\text{  }}{\text{minimize}}\ (1-\mu)\mathcal{O}(\mathcal{D}, \text{}) + \mu \frac{A(\text{})}{A'}, \tag{5.7}$$

where $\mu \in \mathbb{R}^+$ expresses the balance between accuracy and area, and A' is a constant multiplier to calibrate the area term of the similar magnitude with the accuracy term $\mathcal{O}(\mathcal{D}, \text{  })$. 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.

Regarding the initialization of the circuit architecture, the EA starts with only output nodes to facilitate the generation of more compact pNCs. From there, it progressively increases the number of neurons and their connectivity. After the evolution, the associated topological structures and parameters can be mapped to the respective hardware primitives and flexibly printed.

Gradient-based training. SOTA gradient-based strategies for optimizing network architectures are *NAS* [9] and *network pruning* [23]. Unfortunately, NAS approaches are mainly designed for deep neural networks (DNNs) with block structures [4, 14, 55, 57] 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)* [26], 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) [16] and *structured pruning* (targeting groups of parameters) [22]. The former typically incorporates the ℓ_p norms of parameters into the regularizer independently, e.g., through

$$\|g_1\|_1 + \|g_2\|_1 + \cdots + \|g_i\|_1 + \cdots \tag{5.8}$$

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

$$\|[g_1, g_2, \cdots, g_i, \cdots]\|_2 \tag{5.9}$$

to foster the elimination of complete neurons. Therefore, the latter is also named grouped pruning [27, 51].

In ML, structured pruning is more favored as it streamlines 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 pNCs, 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 (5.8), and structured pruning, Equation (5.9), methods as a SOTA baseline for the proposed evolutionary architecture algorithm.

In addition to existing network pruning methods, this work also enhances

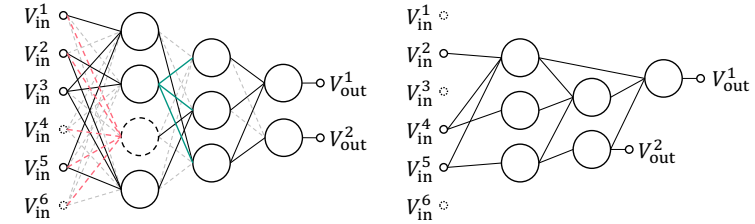


Figure 5.11: Comparison of the circuit architecture from (a) gradient and (b) evolutionary approaches. In (a), the dashed edges and nodes are pruned. The red color refers to pruning a neuron when all its input weights are pruned. The green color represents the case of pruning a negation circuit when all the weights associated to a voltage are positive.

the existing pruning method to specifically encourage the compact design of pNCs, namely the *area-aware training*.

Despite the large amount of research on the functional forms [29, 33, 56] 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 pNC design, we aim to incorporate circuit area of pNCs directly as the regularization in the training objective. This approach promotes the explicit optimization of compact pNCs. Additionally, as the circuit area is significantly influenced by the circuit architecture, which is non-differentiable and fails to offer valid gradient information, we employ gradient-relaxation methods to heuristically guide the gradient-based training. The basic idea of this gradient relaxation was introduced in Chapter 3.2.1.

As illustrated by the dashed neuron in Figure 5.11(a), the presence of a neuron can be expressed by the existence of its input weights embodied by the conductance g_i, i.e.,

$$\max_i \left\{ [\mathbb{1}_{\{g_1>0\}}, \mathbb{1}_{\{g_2>0\}}, \cdots, \mathbb{1}_{\{g_i>0\}}, \cdots] \right\}. \tag{5.10}$$

This method belongs to structured pruning, as it aims to eliminate the entire

neuron. Different from traditional regularization like Equation (5.9), Equation (5.10) 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. As the indicator function $\mathbb{1}_{\{\cdot\}}$ is a piece-wise constant function, we employ the soft-count to enable the gradient-based training. Specifically, we still use the result of Equation (5.10) in the forward pass, while calculating gradients using a smooth relaxation called soft-counts, 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 (5.10) is given by

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

as shown in by the orange function in Figure 3.5.

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 5.11. If none of the corresponding weights is negative, the output voltage from the preceding neuron does not need to be negated. Similarly, the gradient of this function is relaxed by

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

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

$$\text{sigmoid}(g_i)$$

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

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

Finally, we use the cross-entropy loss to guide the training for higher accuracy.

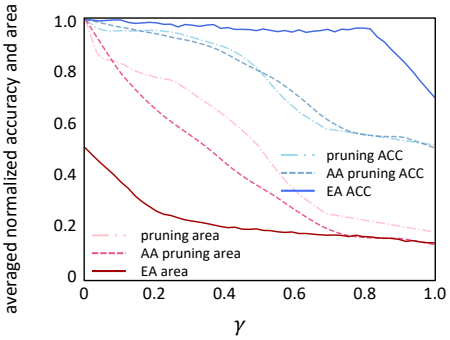


Figure 5.12: Results of the experiment: averaged normalized accuracy and area from three methods, namely the evolutionary algorithm (EA)-based training, the area-aware (AA) pruning, and the existing pruning method.

Comparable to Equation (5.7), the training objective for the pruning is

$$\underset{\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}}\ (1-\mu)\,\mathrm{CE}\,(\mathcal{D},\boldsymbol{\theta},\boldsymbol{q})+\mu\frac{A^{\text{soft}}(\boldsymbol{\theta})}{A'}. \tag{5.11}$$

Similar to Chapter 5.2, to mitigate the noneffective penalty introduced by the soft-count, we employ the mask-based pruning method after the training process, and conduct a fine-tuning stage to further improve the classification accuracy without any increase in the circuit areas.

5.3.3 Experiment

To evaluate the efficacy of the proposed methods, we implemented both evolutionary and pruning algorithms[5] with PyTorch and conducted experiments on 13 benchmark datasets as introduced in Table 4.1.

Experiment Setup. To conduct the EA-based training, we follow the default evolution setup suggested in Chapter 4.3.4, whereas for gradient-based pruning methods, we employ the pipeline introduced in Chapter 4.1.3.

[5] `https://github.com/Neuromophic/Area-Aware-Training`.

In addition, in EA, to preserve a minimum size of the circuits, the architecture for all datasets are initialized as unconnected networks, i.e., featuring only #output nodes. But in the network pruning, as it 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 sections.

To investigate the trade-off between accuracy and area, we run 50 experiments, for both EA and pruning, with 50 uniformly selected values in $\mu \in [0,1]$. The whole process is repeated ten times (with random seed varying from 1 to 10) for each μ to ensure achieving a sufficiently good solution.

Result. After training, results are calculated on the corresponding test sets. It is evident that, with increasing μ, the training objective gradually transitions from prioritizing classification accuracy to minimizing circuit area. Conceptually, $\mu = 0$ 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 5.4.

Meanwhile, as μ 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 μ is described in Figure 5.12.

Existing pruning vs. area-aware training. By comparing the SOTA 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.

Table 5.4: Simulation result and runtime of three approaches on 13 benchmark datasets with $\mu = 0$. Sourced from area paper.

Dataset	Existing (area-unaware) Pruning			Area-Aware Pruning			Proposed EA Approach		
	Accuracy	**Area** (mm^2)	**Time** (min)	**Accuracy**	**Area** (mm^2)	**Time** (min)	**Accuracy**	**Area** (mm^2)	**Time** (min)
1	1.00 ± 0.00	688 ± 47	8.3 ± 1.1	1.00 ± 0.00	743 ± 74	7.9 ± 1.5	1.00 ± 0.00	165 ± 17	32.8 ± 10.4
2	0.97 ± 0.06	890 ± 40	11.5 ± 2.4	0.93 ± 0.03	894 ± 28	11.5 ± 2.5	0.93 ± 0.02	269 ± 5	101.1 ± 31.1
3	0.97 ± 0.00	827 ± 62	10.4 ± 2.2	0.97 ± 0.00	769 ± 136	11.3 ± 1.1	0.96 ± 0.01	176 ± 18	83.2 ± 13.7
4	0.88 ± 0.01	1378 ± 97	12.4 ± 1.9	0.87 ± 0.01	1299 ± 94	11.4 ± 2.6	0.86 ± 0.03	238 ± 16	118.9 ± 26.0
5	0.97 ± 0.01	959 ± 29	15.6 ± 3.8	0.97 ± 0.02	970 ± 32	13.6 ± 4.8	1.00 ± 0.00	228 ± 32	88.9 ± 36.4
6	0.92 ± 0.01	965 ± 62	11.2 ± 0.8	0.92 ± 0.02	988 ± 82	10.4 ± 1.1	0.90 ± 0.01	188 ± 3	82.9 ± 26.3
7	0.98 ± 0.01	832 ± 71	8.6 ± 0.7	0.96 ± 0.01	859 ± 59	7.5 ± 0.8	0.94 ± 0.00	227 ± 6	32.6 ± 12.2
8	0.86 ± 0.01	876 ± 40	9.8 ± 0.8	0.86 ± 0.01	827 ± 33	9.1 ± 1.2	0.86 ± 0.02	213 ± 2	71.3 ± 21.6
9	0.32 ± 0.01	1121 ± 30	21.6 ± 3.8	0.34 ± 0.07	1076 ± 177	22.2 ± 2.5	0.49 ± 0.10	551 ± 13	196.3 ± 66.5
10	0.92 ± 0.02	957 ± 34	8.7 ± 0.8	0.94 ± 0.05	915 ± 62	8.9 ± 0.8	0.88 ± 0.02	262 ± 19	47.0 ± 19.5
11	1.00 ± 0.00	883 ± 90	9.7 ± 0.5	0.97 ± 0.02	982 ± 36	9.4 ± 0.5	0.95 ± 0.01	199 ± 5	75.3 ± 26.3
12	0.83 ± 0.01	811 ± 12	7.3 ± 0.5	0.83 ± 0.01	816 ± 2	7.9 ± 0.5	0.81 ± 0.01	238 ± 3	46.5 ± 20.3
13	0.82 ± 0.02	978 ± 12	8.8 ± 1.2	0.81 ± 0.03	939 ± 60	8.6 ± 1.3	0.84 ± 0.01	238 ± 7	55.2 ± 16.8
Average	$\mathbf{0.88 \pm 0.01}$	$\mathbf{936 \pm 48}$	$\mathbf{11.2 \pm 1.6}$	$\mathbf{0.87 \pm 0.02}$	$\mathbf{929 \pm 67}$	$\mathbf{10.7 \pm 1.6}$	$\mathbf{0.88 \pm 0.02}$	$\mathbf{246 \pm 11}$	$\mathbf{79.4 \pm 25.2}$

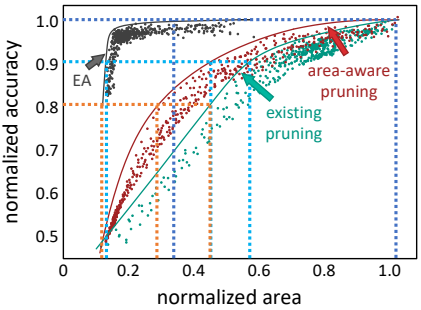


Figure 5.13: Scatters and Pareto fronts of three methods, green for existing pruning that is **unaware** of circuit area, red for proposed area-aware pruning, and black for evolutionary algorithm (EA)-based area-aware training.

EA approach vs. area-aware training. It can be seen that even with $\mu = 0$, 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 μ increases, although both methods decrease the accuracy and area of pNCs, the EA method produces a more modest degradation in accuracy.

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 μ in Figure 5.13 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 5.5 displays several trade-off points from the Pareto-fronts. It is evident that the EA approach offers 3.1$\times$ area-savings without any accuracy degradation compared to pruning. In

Table 5.5: Accuracy-area trade-offs with Pareto-optimality.

Normalized	**Area-Aware Pruning**		**EA (superiority over pruning)**	
Accuracy (%)	**Area (%)**	**Power** (mW)	**Area (%)**	**Power** (mW)
100	100	78	32 ($\downarrow$ 3.1$\times$)	26 ($\downarrow$ 3.0$\times$)
90	44	37	15 ($\downarrow$ 2.9$\times$)	13 ($\downarrow$ 2.9$\times$)
80	31	22	12 ($\downarrow$ 2.5$\times$)	4 ($\downarrow$ 5.5$\times$)

case a 10% reduction in normalized accuracy is permissible, only 15% of the reference area is needed, which is 2.9$\times$ area reduction compared to area-aware pruning. Moreover, 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$\times$ and 2.9$\times$ power reduction respectively while providing 100% and 90% normalized accuracies.

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 5.4.

In our experiments, EA demands significantly more time ($\approx 7\times$) than its gradient counterpart. However, we have the following comments on this issue: The most time-consuming steps in EAs are evaluating genomes and producing offspring, which are nearly proportional to the population size N. We selected a sufficient large N to explore the full capabilities of EAs, which can be reduced in real design and optimization scenarios. Genome evaluation and offspring production are well-suited to parallelization, although parallelization is not included in this thesis, it can be easily addressed in the future. Despite longer training times in our setup (30 minutes to 3 hours), the duration is still acceptable within the broader product development cycle, as circuit optimization is only part of NRE and target applications of PE often require small-scale circuits.

5.3.4 Discussion

This section focuses on the inherent challenges of PE, i.e., the large feature sizes and limited device counts, restricting the broader application of pNCs in compact scenarios. We leveraged the capability of PE for flexibly printing any bespoke circuit architecture for specific target objectives. To explore this potential, we proposed an area-aware training objective and adapted it to two different approaches, namely EA-based method and gradient-based pruning approaches. Simulation results reveal that the proposed method is capable to facilitate compact design of pNCs, and EA presents a superiority over the gradient-based pruning benchmarks. This significantly expands the range of application scenarios and enhances the practicality of pNCs.

5.4 Summary

This chapter focuses on the utility of pNCs. Leveraging the additive manufacturing of PE, this chapter explores a hybrid fabrication strategy that merges the cost-effectiveness and scalability of high-volume printing with the highly bespoke fabrication of low-volume printing. This approach optimizes both cost and classification accuracy in the production of pNCs across varying specifications and batches. Power consumption is a pivotal factor in circuit design. This chapter manages the power-aware training that precisely models the power-consumption of pNCs and facilitates optimal power-accuracy trade-offs in different situations. Additionally, this chapter delves into the compact design of pNCs, which is critical for area-constrained applications. By establishing an accurate model to predict circuit area after placement and routing, this chapter proposes an objective function that supports area-aware training, balancing circuit footprint with classification accuracy.

Methodologically, this chapter primarily utilizes the Pareto-analysis with penalty methods to identify the best compromises among multiple objectives, offering valuable insights for circuit design. It also introduces the augmented Lagrangian method (for both equality and inequality constraints) to achieve or even surpass Pareto-optimal within single training. This method significantly enhances the efficiency of pNC design and optimization with given budgets.

Bibliography

[1] Arif U Alam, Pranali Rathi, Heba Beshai, Gursimran K Sarabha, and M Jamal Deen. “Fruit quality monitoring with smart packaging”. In: *Sensors* 21.4 (2021), p. 1509.

[2] Edgar Anderson. “The Irises of The Gaspe Peninsula”. In: *Bull. Am. Iris Soc.* 59 (1935), pp. 2–5.

[3] Eric Berthonnaud, Joannès Dimnet, Pierre Roussouly, and Hubert Labelle. “Analysis of The Sagittal Balance of The Spine and Pelvis Using Shape and Orientation Parameters”. In: *Clinical Spine Surgery* 18.1 (2005), pp. 40–47.

[4] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. “Efficient architecture search by network transformation”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[5] Małgorzata Charytanowicz, Jerzy Niewczas, Piotr Kulczycki, Piotr A Kowalski, Szymon Łukasik, and Sławomir Żak. “Complete Gradient Clustering Algorithm for Features Analysis of X-ray Images”. In: *Information technologies in biomedicine*. Springer, 2010, pp. 15–24.

[6] Leonardo Weiss Ferreira Chaves and Christian Decker. “A survey on organic smart labels for the Internet-of-Things”. In: *2010 Seventh International Conference on Networked Sensing Systems (INSS)*. IEEE. 2010, pp. 161–164.

[7] CM Costa, R Gonçalves, and S Lanceros-Méndez. “Recent advances and future challenges in printed batteries”. In: *Energy Storage Materials* 28 (2020), pp. 216–234.

[8] Jacek Czerniak and Hubert Zarzycki. “Application of Rough Sets in The Presumptive Diagnosis of Urinary System Diseases”. In: *Artificial intelligence and security in computing systems*. Springer, 2003, pp. 41–51.

[9] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. “Neural architecture search: A survey”. In: *Journal of Machine Learning Research* 20.55 (2019), pp. 1–21.

[10] Matthias Elter, Rüdiger Schulz-Wendtland, and Thomas Wittenberg. “The Prediction of Breast Cancer Biopsy Outcomes Using Two CAD Approaches that both Emphasize an Intelligible Decision Process”. In: *Medical physics* 34.11 (2007), pp. 4164–4172.

[11] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. “Do we need hundreds of classifiers to solve real world classification problems?” In: *The journal of machine learning research* 15.1 (2014), pp. 3133–3181.

[12] B. German. *Datatset: Glass Identification*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5WW2P. 1987.

[13] David Gil, Jose Luis Girela, Joaquin De Juan, M. Jose Gomez-Torres, and Magnus Johnsson. "Predicting seminal quality with artificial intelligence methods". In: *Expert Syst. Appl.* 39 (2012), pp. 12564–12573.

[14] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. "LSTM: A search space odyssey". In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.

[15] S. Haberman. *Datatset: Haberman's Survival*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XK51. 1999.

[16] Song Han, Jeff Pool, John Tran, and William Dally. "Learning both weights and connections for efficient neural network". In: *Advances in neural information processing systems* 28 (2015).

[17] Barbara Hayes-Roth and Frederick Hayes-Roth. *Datatset: Hayes-Roth*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5501T. 1989.

[18] Paul Horton and Kenta Nakai. "A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins". In: *Proceedings. International Conference on Intelligent Systems for Molecular Biology* 4 (1996), pp. 109–15. URL: `https://api.semanticscholar.org/CorpusID:964254`.

[19] Edwin T Jaynes. "Information theory and statistical mechanics". In: *Physical review* 106.4 (1957), p. 620.

[20] Dong Jin Kang, Lola Gonzaléz-García, and Tobias Kraus. "Soft electronics by inkjet printing metal inks on porous substrates". In: *Flexible and Printed Electronics* 7.3 (2022), p. 033001.

[21] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. "Pruning Filters for Efficient ConvNets". In: *arXiv preprint arXiv:1608.08710* (2016).

[23] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. "Pruning and quantization for deep neural network acceleration: A survey". In: *Neurocomputing* 461 (2021), pp. 370–403.

[24] Xiaoping Liang, Haifang Li, Jinxin Dou, Qi Wang, Wenya He, Chunya Wang, Donghang Li, Jin-Ming Lin, and Yingying Zhang. "Stable and biocompatible carbon nanotube ink mediated by silk protein for printed electronics". In: *Advanced Materials* 32.31 (2020), p. 2000165.

[25] Tong-Hong Lin, Jo Bito, and Manos M Tentzeris. “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.

[26] Hanxiao Liu, Karen Simonyan, and Yiming Yang. “Darts: Differentiable architecture search”. In: *arXiv preprint arXiv:1806.09055* (2018).

[27] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. “Learning efficient convolutional networks through network slimming”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2736–2744.

[28] Wei-Yin Loh. *Datatset: Teaching Assistant Evaluation*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55P6M. 1997.

[29] Rongrong Ma, Jianyu Miao, Lingfeng Niu, and Peng Zhang. “Transformed ℓ_1 Regularization for Learning Sparse Deep Neural Networks”. In: *Neural Networks* 119 (2019), pp. 286–298.

[30] Olvi L Mangasarian and William H Wolberg. *Cancer Diagnosis Via Linear Programming*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 1990.

[31] Anqi Mao, Mehryar Mohri, and Yutao Zhong. “Cross-entropy loss functions: Theoretical analysis and applications”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 23803–23828.

[32] Christopher J Matheus and Larry A Rendell. “Constructive Induction on Decision Trees.” In: *IJCAI*. Vol. 89. Citeseer. 1989, pp. 645–650.

[33] Rahul Mazumder, Jerome H Friedman, and Trevor Hastie. “Sparsenet: Coordinate descent with nonconvex penalties”. In: *Journal of the American Statistical Association* 106.495 (2011), pp. 1125–1138.

[34] Ashish Mishra, Shiva Krishna Reddy, Anurag Mittal, and Hema A Murthy. “A generative model for zero shot learning using conditional variational autoencoders”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 2188–2196.

[35] Mohammed G Mohammed and Rebecca Kramer. “All-printed flexible and stretchable electronics”. In: *Advanced Materials* 29.19 (2017), p. 1604965.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[37] Farhan Rasheed, Michael Hefenbrock, Michael Beigl, Mehdi B Tahoori, and Jasmin Aghassi-Hagmann. “Variability modeling for printed inorganic electrolyte-gated transistors and circuits”. In: *IEEE transactions on electron devices* 66.1 (2018), pp. 146–152.

[38] Farhan Rasheed, Michael Hefenbrock, Rajendra Bishnoi, Michael Beigl, Jasmin Aghassi-Hagmann, and Mehdi B Tahoori. “Crossover-aware placement and routing for inkjet printed circuits”. In: *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 16.2 (2020), pp. 1–22.

[39] Yoram Reich and Steven Fenves. *Datatset: Pittsburgh Bridges*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5RP5H. 1990.

[40] JP S and J Jossinet. *Datatset: Breast Tissue*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5P31H. 2010.

[41] Ehsan Shirzaei Sani, Changhao Xu, Canran Wang, Yu Song, Jihong Min, Jiaobing Tu, Samuel A Solomon, Jiahong Li, Jaminelli L Banks, David G Armstrong, et al. “A stretchable wireless wearable bioelectronic system for multiplexed monitoring and combination treatment of infected chronic wounds”. In: *Science Advances* 9.12 (2023), eadf7388.

[42] Robert S Siegler. “Three Aspects of Cognitive Development”. In: *Cognitive psychology* 8.4 (1976), pp. 481–520.

[43] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert Scott Johannes. “Using the ADAP learning algorithm to forecast the onset of diabetes mellitus”. In: *Proceedings of the annual symposium on computer application in medical care*. American Medical Informatics Association. 1988, p. 261.

[44] Joseph E Stiglitz. “Pareto optimality and competition”. In: *The Journal of Finance* 36.2 (1981), pp. 235–251.

[45] Isabel Straw and Honghan Wu. “Investigating for bias in healthcare algorithms: a sex-stratified analysis of supervised machine learning models in liver disease prediction”. In: *BMJ Health & Care Informatics* 29 (2022). URL: `https://api.semanticscholar.org/CorpusID:248390297`.

[46] Sebastian Thrun. “The MONK’s Problems: A Performance Comparison of Different Learning Algorithems”. In: *Technical Report of Carnegie Mellon University* (1991).

[47] Athanasios Tsanas and Angeliki Xifara. “Accurate Quantitative Estimation of Energy Performance of Residential Buildings Using Statistical Machine Learning Tools”. In: *Energy and Buildings* 49 (2012), pp. 560–567.

[48] Dennis D Weller, Michael Hefenbrock, Michael Beigl, Jasmin Aghassi-Hagmann, and Mehdi B Tahoori. “Realization and training of an inverter-based printed neuromorphic computing system”. In: *Scientific reports* 11.1 (2021), p. 9554.

[49] David H Wolpert and William G Macready. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.

[50] I-Cheng Yeh. *Datatset: Blood Transfusion Service Center*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5GS39. 2008.

[51] Jaehong Yoon and Sung Ju Hwang. “Combined group and exclusive sparsity for deep neural networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3958–3966.

[52] Haibin Zhao, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Split Additive Manufacturing for Printed Neuromorphic Circuits”. In: *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2023, pp. 1–6.

[53] Haibin Zhao, Priyanjana Pal, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Power-Aware Training for Energy-Efficient Printed Neuromorphic Circuits”. In: *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE. 2023, pp. 1–9.

[54] Haibin Zhao, Priyanjana Pal, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Towards Temporal Information Processing – Printed Neuromorphic Circuits with Learnable Filters”. In: *Proceedings of the 18th ACM International Symposium on Nanoscale Architectures*. 2023, pp. 1–6.

[55] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. “Practical block-wise neural network architecture generation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2423–2432.

[56] Yexu Zhou, Haibin Zhao, Michael Hefenbrock, Siyan Li, Yiran Huang, and Michael Beigl. “Deep Neural Network Pruning with Progressive Regularizer”. In: *2024 IEEE International Joint Conference on Neural Network (IJCNN 2024)*. IEEE. 2024.

[57] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. “Learning transferable architectures for scalable image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8697–8710.

[58] Matjaz Zwitter and Milan Soklic. *Datatset: Breast Cancer*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C51P4M. 1988.

6 Extension of Printed Neuromorphic Circuits

Previous chapters primarily adhered to pNCs with computational paradigm of MLPs. However, this paradigm inherently lacks the capability to handle temporal information due to the absence of time-dependent operations. As illustrated in Figure 6.1, research in neuromorphic computing hardware extends beyond feedforward ANNs or MLPs to include recurrent neural networks (RNNs) and spiking neural networks (SNNs).

This chapter first introduces a PE implementation of printed recurrent neuromorphic circuits (pRNCs). By including learnable filters into existing pNCs and modifying the circuit architecture, the new circuit can process temporal signals similarly in the way of RNNs. Moreover, SNN is one of the most biological plausible computing paradigms, are recognized for its low power consumption and resilience to noise. This chapter also proposes a printed spike-generator and its inclusion into existing pNCs to emulate printed spiking neuromorphic circuits (pSNCs).

For both circuit designs, we propose corresponding circuit modeling methods and training schemes to leverage the agile manufacturing of PE, enabling highly customized circuit training for specific tasks.

6.1 Printed Recurrent Neuromorphic Circuits

In many scenarios like stress detection [22], the concrete signal values might be less informative due to physiological variation among individuals. In contrast, the temporal changes in the signal often exhibits more meaningful information. However, this kind of time-series data processing is beyond the reach of the existing pNCs with MLP-paradigms. This is because of their lack of component to memorize historical signals.

To address this limitation, we propose to introduce learnable printed capaci-

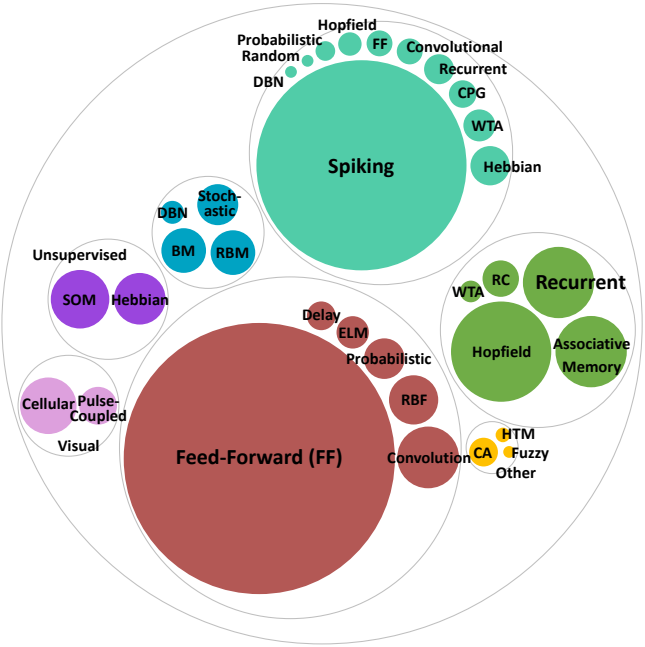


Figure 6.1: A breakdown of network models in neuromorphic implementations, grouped by overall type and sized to reflect the number of associates papers. Sourced from [18] (2017).

tors into the existing pNCs, which allow the circuit to memorize, and thus, process temporal sensory data. By combining the capacitors with existing primitives in pNCs, we construct the printed temporal processing block (pTPB). Additionally, by stacking multiple pTPBs, pRNCs can be constructed to provide more sophisticated computing functionalities.

6.1.1 Circuit Design of Recurrent pNCs

Before propose the circuit design of pRNCs, we shortly review the preliminary of RNNs and analyze their essential advancement over the MLPs.

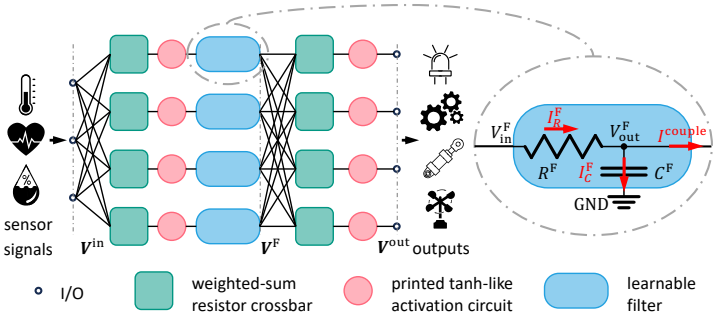


Figure 6.2: Schematic of a 3-input 4-output printed temporal processing block (pTPB) that receives sensor signals and yields outputs to subsequent devices. Sourced from [21].

Recurrent neural networks. RNNs were initially proposed to handle inputs with variable lengths, as the input dimensionality for an MLP is always predetermined. By incorporating an internal hidden state $\boldsymbol{h}$, an RNN is then capable of processing variable length inputs through repeated state updates. RNNs have demonstrated remarkable success in areas such as handwriting recognition [7]. Notably, RNNs are theoretically Turing-complete, meaning that they can execute arbitrary programs to process any given input sequences [10]. A general formulation of RNN state equations is given by

$$\begin{aligned} \boldsymbol{h}_t &= f_1(f_2(\boldsymbol{h}_{t-1}) + f_3(\mathbf{X}_t)), \\ \hat{\boldsymbol{Y}}_t &= f_4(\boldsymbol{h}_t), \end{aligned} \tag{6.1}$$

where the subscript $t \in \{0, 1, \cdots, T\}$ refer to the iteration (time step), $\boldsymbol{h}_t$ is the internal hidden state at the t-th step, $\mathbf{X}_t$ denotes the input at the t-th step, and $\hat{\boldsymbol{Y}}_t$ represents the output at the t-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 [17], $f_2(\cdot)$ 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.

Printed Recurrent Neuromorphic Circuits. Notably, the essential capability of RNNs to process temporal information is the update of the hidden state containing previous information. Inspired by the similar behavior of the capacitors, we introduce printed capacitors into the existing pNCs to construct the printed circuits that may emulate RNN-paradigms. Further, to include other operations in ANNs as shown in Equation (6.1), we combine the filters with crossbars and ptanhs circuits. Consequently, the pTPB is proposed and represented in Figure 6.2. 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 to provide time-dependencies. To enable bespoke design of the proposed circuits, we also establish the corresponding model to facilitate the training of the circuit components for target tasks. For this, we will first model the filters while considering their coupling with the rest of the circuit. Afterwards, we will develop the model to cover the entire pTPB.

6.1.2 Modeling of Recurrent pNCs

Different application scenarios generally necessitate different temporal processing behaviors. Conveniently, the highly flexible additive printing techniques of PE can enable such task-specific fabrication. To design the bespoke signal processing behaviors for target tasks, we develop the mathematical model of the proposed pTPB and the corresponding pRNCs, along with an optimization objective. With this approach, the components in pTPBs (e.g., the capacitance) can be optimized alongside the resistances in the crossbars (representing weights and biases).

Modeling of single filter. Initially, we concentrate on modeling the filter without considering its coupling to the successive circuit. Taking the filter unit in Figure 6.2 as an example, we obtain:

$$\begin{aligned} I_R^{\mathrm{F}} &= \left(V_{\mathrm{in}}^{\mathrm{F}} - V_{\mathrm{out}}^{\mathrm{F}}\right)/R^{\mathrm{F}}, \\ I_C^{\mathrm{F}} &= C^{\mathrm{F}}\mathrm{d}V_{\mathrm{out}}^{\mathrm{F}}/\mathrm{d}t, \\ I_R^{\mathrm{F}} &= I_C^{\mathrm{F}}, \end{aligned} \tag{6.2}$$

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

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

By using backward Euler integration [2], we obtain the update of the V^{F} from time step to time step:

$$\begin{aligned} V_{\mathrm{out}t}^{\mathrm{F}} &= \underbrace{\frac{R^{\mathrm{F}}C^{\mathrm{F}}}{R^{\mathrm{F}}C^{\mathrm{F}}+\Delta t}}_{=:\beta} V_{\mathrm{out}t-1}^{\mathrm{F}} + \underbrace{\frac{\Delta t}{R^{\mathrm{F}}C^{\mathrm{F}}+\Delta t}}_{=:1-\beta} V_{\mathrm{in}t}^{\mathrm{F}} \\ &= \beta V_{\mathrm{out}t-1}^{\mathrm{F}} + (1-\beta)V_{\mathrm{in}t}^{\mathrm{F}}, \end{aligned} \tag{6.3}$$

where Δt refers to the step size of the temporal discretization, $V_{\mathrm{in}t}^{\mathrm{F}}$ and $V_{\mathrm{out}t}^{\mathrm{F}}$ are the input and output of the filter at time step t. 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. As these values will be learned jointly with the crossbar resistors to fit the specific tasks, they are referred to as *learnable filters*.

Modeling of coupled filter. To connect the learnable filters with the resistor crossbars, it is imperative to take the impact of their interface into account. This impact 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 6.2. To reflect this interface in our model, we modify Equation (6.2) to

$$I_R^{\mathrm{F}} = I_C^{\mathrm{F}} + I^{\mathrm{couple}} =: \mu I_C^{\mathrm{F}},$$

with I^{couple} refers to the coupling current flows towards crossbar, and

$$\mu := 1 + \frac{I^{\mathrm{couple}}}{I_C^{\mathrm{F}}}$$

is a decoupling factor. Consequently, Equation (6.3) is reformulated to

$$\begin{aligned} V_{\mathrm{out}t}^{\mathrm{F}} &= \underbrace{\frac{\mu R^{\mathrm{F}}C^{\mathrm{F}}}{\mu R^{\mathrm{F}}C^{\mathrm{F}}+\Delta t}}_{=:\beta'} V_{\mathrm{out}t-1}^{\mathrm{F}} + \underbrace{\frac{\Delta t}{\mu R^{\mathrm{F}}C^{\mathrm{F}}+\Delta t}}_{=:1-\beta'} V_{\mathrm{in}t}^{\mathrm{F}} \\ &= \beta' V_{\mathrm{out}t-1}^{\mathrm{F}} + (1-\beta')V_{\mathrm{in}t}^{\mathrm{F}}, \end{aligned} \tag{6.4}$$

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 minimize the coupling effect, the resistances in the filters are designed with lower values ($\leq 1\,\mathrm{k}\Omega$) than that of the resistors in crossbars ($100\,\mathrm{k}\Omega$-$10\,\mathrm{M}\Omega$), while the capacitances are designed as high as the printing technology allows ($100\,\mathrm{nF}$-$100\,\mu\mathrm{F}$). By analyzing of signal frequencies in the experimental datasets (see Table 6.1) and conducting SPICE simulations with pPDK [16], we empirically determined $\mu \in [1, 1.3]$ for the given applications.

Modeling of temporal processing block. Although Equation (6.4) emulates Equation (6.1), 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 is designed.

As sketched in Figure 6.2, to match analogous computational capabilities of classic Elman RNNs [17], 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 $I^{\mathrm{C}}_{\mathrm{out}}$. 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 entire circuit exemplified in Figure 6.2. Consequently, the mathematical model of a pTPB can be described as

$$\begin{aligned} \boldsymbol{V}^{\mathrm{F}}_t &= \boldsymbol{\beta}' \odot \boldsymbol{V}^{\mathrm{F}}_{t-1} + (1-\boldsymbol{\beta}') \odot \mathrm{ptanh}(\boldsymbol{W}_1 \boldsymbol{V}^{\mathrm{in}}_t + \boldsymbol{b}_1), \\ \boldsymbol{V}^{\mathrm{out}}_t &= \mathrm{ptanh}(\boldsymbol{W}_2 \boldsymbol{V}^{\mathrm{F}}_t + \boldsymbol{b}_2), \end{aligned} \tag{6.5}$$

where $\boldsymbol{V}^{\mathrm{in}}_t \in \mathbb{R}^{N_{\mathrm{in}}}$ and $\boldsymbol{V}^{\mathrm{out}}_t \in \mathbb{R}^{N_{\mathrm{out}}}$ vectorize the input and output voltages of the pTPB at time point t. $\boldsymbol{V}^{\mathrm{F}}_t \in \mathbb{R}^{N_{\mathrm{F}}}$ summarizes the output voltages of the filter layer and $\boldsymbol{\beta}' \in \mathbb{R}^{N_{\mathrm{F}}}$ collects the β' values of each filter. Moreover, $\boldsymbol{W}_1 \in \mathbb{R}^{N_{\mathrm{F}} \times N_{\mathrm{in}}}$, $\boldsymbol{b}_1 \in \mathbb{R}^{N_{\mathrm{F}}}$, $\boldsymbol{W}_2 \in \mathbb{R}^{N_{\mathrm{out}} \times N_{\mathrm{F}}}$, and $\boldsymbol{b}_2 \in \mathbb{R}^{N_{\mathrm{out}}}$ denotes the weighted-sum

operations emulated by the corresponding crossbars. Additionally, $\odot$ indicates element-wise multiplication.

By comparing Equation (6.5) with Equation (6.1), 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 pRNC may consist of multiple pTPBs connected successively for accomplishing more intricate computational tasks. In case of multiple pTPBs, we denote the initial input voltages (typically sensor signals) by $\mathbf{X}_k$, and represent the final output of the last layer by $\hat{\boldsymbol{Y}}_t(\boldsymbol{\beta}', \boldsymbol{\theta}, \boldsymbol{q}, \mathbf{X}_t, \mathbf{X}_{t-1}, \cdots, \mathbf{X}_0)$, which is a function of $\boldsymbol{\beta}'$ in the learnable filters, the crossbar conductances $\boldsymbol{\theta}$, the components in nonlinear circuit $\boldsymbol{q}$, and the input voltages at all time steps $\mathbf{X}_t, \cdots, \mathbf{X}_0$.

6.1.3 Training of pRNCs

In case of training basic pNCs (without pTPB), the cross-entropy loss [12] can be minimized with respect to learnable surrogate conductances $\boldsymbol{\theta}$ to decrease the mismatch between the label $\mathbf{Y}$ and the circuit prediction $\hat{\boldsymbol{Y}}(\boldsymbol{\theta}, \boldsymbol{q}, \mathbf{X})$ for an input $\mathbf{X}$, and thus improve the classification accuracy. In contrast, the pRNCs is time-dependent and allows obtaining predictions for each time step $\hat{\boldsymbol{Y}}_t$ based on the current input $\mathbf{X}_t$ and previous inputs $\mathbf{X}_{t-1}, \cdots, \mathbf{X}_0$. 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

$$\underset{\boldsymbol{\beta}', \boldsymbol{\theta}, \boldsymbol{q}}{\text{minimize}} \underbrace{\frac{1}{T} \sum_{t=0}^{T} L\left(\hat{\boldsymbol{Y}}_t(\boldsymbol{\beta}', \boldsymbol{\theta}, \boldsymbol{q}, \mathbf{X}_t, \mathbf{X}_{t-1}, \cdots, \mathbf{X}_0), \mathbf{Y}\right)}_{\mathcal{L}(\mathcal{D}, \boldsymbol{\beta}', \boldsymbol{\theta}, \boldsymbol{q})}. \tag{6.6}$$

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 the modeling of pTPBs, while the latter is generally caused by the preceding input signal. To reduce the dependencies of the circuit coupling (μ) and the initial voltage $\boldsymbol{V}_0^{\mathrm{F}}$ 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 $\boldsymbol{V}_0^{\mathrm{F}}$.

Through this, we should achieve a configuration of that learnable parameters $\boldsymbol{g}$ and $\boldsymbol{\beta}'$ that is robust to the choice of either value, which leads to the training objective of

$$\underset{\boldsymbol{\beta}',\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}} \int \mathcal{L}(\mathcal{D},\boldsymbol{\beta}',\boldsymbol{\theta},\boldsymbol{q},\mu,\boldsymbol{V}_0^{\mathrm{F}})p(\mu)\,\mathrm{d}\mu\,p(\boldsymbol{V}_0^{\mathrm{F}})\,\mathrm{d}\boldsymbol{V}_0^{\mathrm{F}}.$$

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

$$\underset{\boldsymbol{\beta}',\boldsymbol{\theta},\boldsymbol{q}}{\text{minimize}}\ \mathbb{E}_{p(\mu),p(\boldsymbol{V}_0^{\mathrm{F}})}\left\{\mathcal{L}(\mathcal{D},\boldsymbol{\beta}',\boldsymbol{\theta},\boldsymbol{q},\mu,\boldsymbol{V}_0^{\mathrm{F}})\right\}, \tag{6.7}$$

which allows to obtain MC estimates of the function value (and its gradients) whenever needed. Consequently, based on the ranges for $\boldsymbol{V}_0^{\mathrm{F}}$ and μ, we choose $p(\boldsymbol{V}_0^{\mathrm{F}}) = \mathcal{U}[0,1]$ 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 (6.7). By encouraging more correct classifications at each time step, the circuit should be trained to achieve correct output as fast as possible.

6.1.4 Experiment

To evaluate the pRNCs, we implemented the proposed approach[1] with PyTorch [14] and conduct experiments on 15 benchmark time-series datasets.

Datasets. To find benchmark datasets for testing the temporal information processing, we first sourced all datasets from the UCR time-series classification archive [3]. Afterwards, we filtered out datasets based on their complexity. Only datasets with N_{in} and N_{out} below ten 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 sig-

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

nal values to the range of $[-1, 1]$, reshuffle and split the datasets into training (60%), validation (20%), and test (20%) sets. The information of the datasets is listed in Table 6.1. Then, we leveraged a 2-layer RNNs as a baseline to remove datasets whose difficulty surpassed the capabilities of general RNNs, because these are not the target applications of pNC. Ultimately, the top 15 datasets with optimal RNN performance are retained for the further experiment.

Experiment setup. For each dataset, we adopt a 2-layer pRNC (consisting of two consecutive pTPBs) with the number of learnable filters N_{F} equaling to N_{out}. For training-related setups, we follow the standard pipeline as introduced in Chapter 4.1.3.

Baselines. For comparison, we consider 2-layer basic pNCs without pTPB as a baseline. The topology is kept the same as the pRNCs, i.e., N_{in}-N_{F}-N_{out} with $N_{\mathrm{F}} = N_{\mathrm{out}}$. This comparison intends to assess the temporal processing ability of both pNCs and pRNC. Since pNCs 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 pRNCs 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 pRNCs, which is the default training setup.

Result. After training, we select the best models (trained with three different random seeds) for each dataset according to the accuracy on the validation set. Note that, in accordance with the proposed objective, we have computed the classification accuracy at every time step, and subsequently average these accuracies over time to yield the overall classification accuracy on each 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 6.1. To

Table 6.1: Result on 15 benchmark time-series datasets: mean and standard deviation of accuracy from random guess, previous printed analog neuromorphic circuit (pNC), hardware-agnostic Elman recurrent neural network (RNN), and printed recurrent neuromorphic circuit (pRNC). Sourced from [21].

	Dataset	**Random Guess**	**pNC**	**Elman RNN**	**pRNC**
1	CBF	0.335	0.456 ± 0.038	0.683 ± 0.036	0.907 ± 0.015
2	DistalPhalanxTW	0.441	0.507 ± 0.006	0.764 ± 0.012	0.654 ± 0.007
3	FreezerRegularTrain	0.520	0.597 ± 0.120	0.795 ± 0.030	0.761 ± 0.076
4	FreezerSmallTrain	0.492	0.509 ± 0.066	0.798 ± 0.068	0.765 ± 0.015
5	GunPointAgeSpan	0.390	0.452 ± 0.003	0.768 ± 0.023	0.682 ± 0.106
6	GunPointMaleVersusFemale	0.567	0.637 ± 0.054	0.829 ± 0.108	0.891 ± 0.163
7	GunPointOldVersusYoung	0.557	0.540 ± 0.007	1.000 ± 0.000	1.000 ± 0.000
8	MiddlePhalanxOutlineAgeGroup	0.483	0.560 ± 0.042	0.708 ± 0.035	0.712 ± 0.006
9	MixedShapesRegularTrain	0.283	0.261 ± 0.008	0.625 ± 0.068	0.503 ± 0.208
10	PowerCons	0.445	0.651 ± 0.010	0.982 ± 0.008	0.801 ± 0.040
11	ProximalPhalanxOutlineCorrect	0.655	0.711 ± 0.001	0.724 ± 0.006	0.743 ± 0.005
12	SelfRegulationSCP2	0.464	0.489 ± 0.011	0.742 ± 0.010	0.782 ± 0.010
13	Slope	0.501	0.559 ± 0.002	0.963 ± 0.036	0.898 ± 0.159
14	SmoothSubspace	0.268	0.447 ± 0.011	0.648 ± 0.010	0.694 ± 0.063
15	Symbols	0.152	0.141 ± 0.002	0.660 ± 0.049	0.670 ± 0.052
	Average	**0.437**	$\mathbf{0.501 \pm 0.025}$	$\mathbf{0.779 \pm 0.033}$	$\mathbf{0.764 \pm 0.062}$

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 6.1.

As can be seen in Table 6.1, basic pNCs 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 pRNCs and basic pNCs, it reveals that pRNCs are indeed capable of processing time-series data. By comparing the performance of pRNCs with RNNs, we conclude that the pRNCs can attain a comparable (98%) classification accuracy to their completely hardware-agnostic Elman RNN counterparts. Interestingly, a closer observation of Table 6.1 reveals that pRNCs and RNNs do not consistently yield comparable performance on every dataset. Their accuracy differs significantly on datasets such as *CBF*, *DistalPhalanxTW*, *PowerCons*, and *SmoothSubspace*. This may be due to the physical limitations of the circuits (and consequently their distinct computational models).

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 pNCs and the proposed pRNCs in different application scenarios . Analogously, we averaged the hardware costs across all datasets to report a comprehensive comparison regarding the hardware costs between the pRNC and its pNC counterpart. The results can be seen in Table 6.2. Evidently, the capability of pRNCs for temporal signal processing requires only approximately $1.5\times$ more devices and $1.3\times$ more power consumption.

6.1.5 Discussion

This section addresses the inherent limitations of pNCs based on the MLP paradigm in processing temporal signals, i.e., time-series data. By incorporating capacitors, the circuit gains time-dependency, enabling to memorize and process historical input signals. Leveraging this feature of the capacitors, we designed a learnable filter sub-circuit as a primitive. Subsequently, we considered the architecture for integrating the learnable filter primitives into the

Table 6.2: Hardware costs of basic printed analog neuromorphic circuits (pNCs) and printed recurrent neuromorphic circuits (pRNCs). Sourced from [21].

Dataset	#Transistor		#Resistor		#Capacitor		**#Total Device**		**Power** (mW)	
	pNC	pRNC	pNC	pRNC	pNC	pRNC	pNC	pRNC	pNC	pRNC
1	18	24	57	84	–	6	75	114	0.471	0.653
2	36	48	150	222	–	12	186	282	1.069	1.501
3	12	16	34	50	–	4	46	70	0.272	0.372
4	12	16	34	50	–	4	46	70	0.276	0.342
5	12	16	34	50	–	4	46	70	0.221	0.374
6	12	16	34	50	–	4	46	70	0.302	0.389
7	12	16	34	50	–	4	46	70	0.289	0.324
8	18	24	57	84	–	6	75	114	0.454	0.625
9	30	40	115	170	–	10	145	220	0.862	1.188
10	12	16	34	50	–	4	46	70	0.312	0.363
11	12	16	34	50	–	4	46	70	0.226	0.381
12	12	16	44	60	–	4	56	80	0.294	0.472
13	12	16	34	50	–	4	46	70	0.320	0.388
14	18	24	57	84	–	6	75	114	0.436	0.610
15	36	48	150	222	–	12	186	282	1.143	1.526
Average	**18**	**23**	**60**	**88**	**–**	**6**	**78**	**118**	**0.463**	**0.634**

main body of pNCs and addressed their interaction (coupling), resulting in the pTPB. The pTPB then serves as a fundamental block of the pRNCs to process more complex computing tasks.

In addition, we developed a parameterized model of pRNCs and designed a training objective function tailored for specific tasks. This enables the pRNCs to learn distinct filtering behaviors for the target tasks. Thanks to the agile manufacturing capabilities of PE, these bespoke designs can be easily implemented without additional costs.

Experiments have demonstrated that the proposed pRNCs require only minor additional hardware to provide efficient temporal signal processing capabilities nearly equivalent to the Elman RNNs. In sum, this new circuit design significantly broadens the application scenarios of the general pNCs family.

6.2 Printed Spiking Neuromorphic Circuits

As introduced in Chapter 2.2 and Figure 6.1 from [18], spiking neural network (SNN) [6] is one of the most initial intention of neuromorphic computing and remains one of the most frequently investigated types of neuromorphic hardware. This is due to its plausibility of the biological signal processing and its competitiveness in terms of circuit area, energy efficiency, and real-time processing capabilities. Some researchers even claim, in the extreme, that only SNNs can be truly considered as "neuromorphic computing". Several studies have successfully implemented analog printed spiking neurons [9, 19]. However, these implementations primarily rely on organic transistors, necessitating high operating voltages, which do not align with the targeted edge application scenarios considered in this thesis.

Moreover, those works often focus solely on the functional implementation at circuit level, such as mimicking the behavior of spiking neurons with the integrate-and-fire (I&F) or leaky-integrate-and-fire (LIF) model. However, they fail to propose corresponding parametric modeling to facilitate the training of circuits containing these spiking neurons. A critical reason for this problem originates from the inconsistency between the designed circuits and the claimed operating principles. For instance, both [9] and [19] assert to design analog printed spiking neurons based on the I&F principle. However, this is

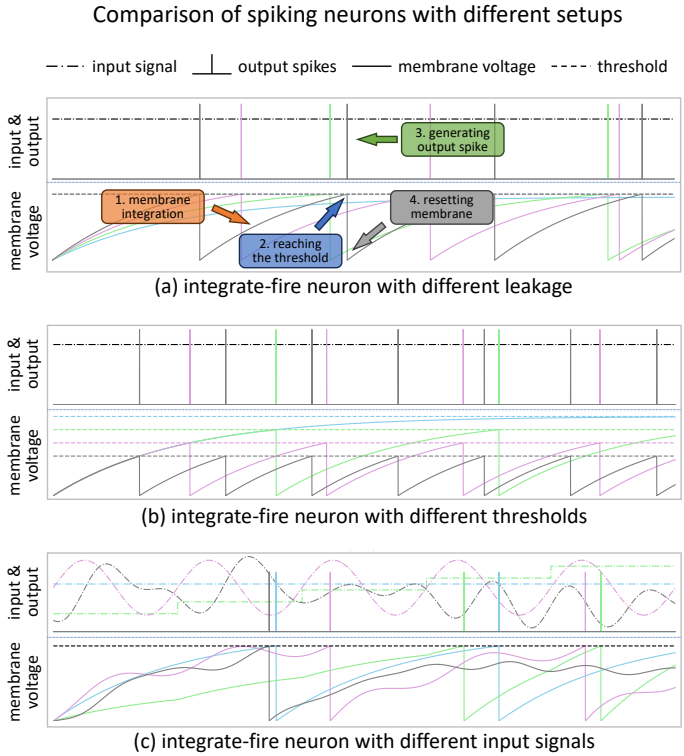


Figure 6.3: Illustration of the behaviors of integrate-and-fire (I&F) neurons with (a) different membrane voltage leakages, (b) different firing thresholds for the membrane voltage, and (c) different input signals. In each sub-figure, lines of the same color represent the corresponding inputs, membrane voltages, thresholds, and outputs. The blocks and arrows in (a) indicate the working principle of the spiking neuron: leaky integration of input voltage, comparison between membrane voltage and threshold, firing, followed by the reset of membrane voltage.

technically impossible for analog circuits, because, as shown in Figure 6.3, the execution of an I&F neuron requires an ideal comparator (necessitating an ideal transistor as switch with no linear region between cut-off and saturation region; and can be charged or discharged instantaneously) and momentary reset/switch of the membrane and output voltages (requiring instantaneous changes of capacitor voltages). Consequently, lacking accurate modeling of these hardware neurons, one can only train the circuits with the ideal I&F model and then naïvely employ the proposed circuit (with another behavior) to replace the ideal I&F model. Evidently, this will significantly reduce the classification accuracy of the circuits or even render them completely unusable.

To address these challenges, this section introduces a hardware-software co-design that emulates spike generation using a set of circuits with fast charging and discharging functions (referred to as a spike-generator). This design is feasible, compatible, and friendly with analog circuits. Although this circuit does not replicate any existing SNN model, we employ the approximation-based modeling approach to accurately model the relationship between the input and output signals of the spike-generator, rather than claiming it with another behavior like LIF. This accurate surrogate model allows us to integrate the proposed spike-generator into existing pNCs, resulting in printed spiking neuromorphic circuit (pSNC), and train them effectively. This end-to-end training approach ensures consistency between the algorithm optimization and the real hardware implementation, enabling the proposed circuit to be fully leveraged and trained for the target tasks.

6.2.1 Circuit Design of Spiking pNCs

Before diving into the concrete circuit design of the pSNCs, we briefly go through the models of the spiking neurons.

Spiking neural networks. Both SNNs and ANNs consists of weighted-sum operations and nonlinear activations. The main difference between SNNs and ANNs is their nonlinear activation behaviors. In ANNs, weighted-summed data are activated by nonlinear activations in magnitude domain, i.e., nonlinear conversion of one magnitude value to another magnitude value. In contrast,

SNNs transform weighted-summed data into temporal spike trains for the delivery and processing of information. As shown in Figure 6.3, the primary working principle of a I&F neuron is to

1. integrate the input signal as membrane voltage, then
2. membrane voltage increases and may reach the threshold value,
3. once the threshold voltage is reached,
4. a spike is generated at the output side, and
5. the membrane voltage is reset.

However, step 3 and step 5 are challenging for the implementation of analog circuits because they require idea comparators and instantaneous reset of capacitor voltages. To avoid these difficulties, we analyze the purpose of this design, and employ an end-to-end design method to design the printed spiking neuron.

We notice that, the design of I&F neuron aims to link the magnitude of input voltages (step 1) with the time interval for the membrane voltage (step 2 and step 5) to reach a certain threshold (step 3), and thus the output spike (step 4). In other word, the magnitude of the input signals is encoded through the occurrence of the output spikes. Follow this idea, we propose the guideline of our pSNCs, namely

1. the number of spikes in a certain temporal interval should monotonically relate to the input magnitude, and
2. the implementation of spikes can be relaxed through a fast charging stage followed by a discharging stage.

According to these principles, we proposed the printed spike-generator that comprises a charging and a discharging circuits that can encode the input signals to the corresponding spike-like pulses.

Printed spike-generator. The design of a printed spiking neuron is shown in Figure 6.4. The resistor crossbar is identical to that in the basic pNCs.

After the weighted-sum, the resulting voltage performs as the gate voltage of T^{ch}, which controls the drain-source (DS) current to charge the succeeding

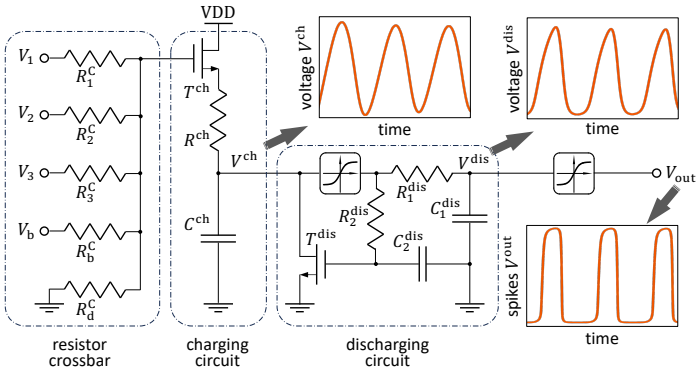


Figure 6.4: Circuit level implementation of printed spiking neuromorphic circuit (pSNC) which includes two primitives: resistor crossbar for weighted-sum operation and the printed spike-generator. The spike-generator can be further decomposed into a charge network and a discharge network. Sourced from [13].

charging circuit. The charging circuit primarily consists of a R^{ch} and C^{ch}, forming a circuit to provide V^{ch} to the amplifier with a delay directly proportional to both $R^{ch} \cdot C^{ch}$ and the frequency of the spikes.

V^{ch} is then strengthened through an amplifier circuit that has identical architecture as the ptanh circuit as introduced in Figure 2.6. The amplifier output is connected to two RC pairs: the first one consists of the pull-up resistor R_1^{dis} with C_1^{dis}, while the second pair is R_2^{dis} and C_2^{dis}. These two RC pairs are essential for the oscillation functionality by adding a small phase shift due to capacitance charging. The voltage across the capacitor C_2^{dis} controls transistor T^{dis}, which manages the discharging of C^{ch}. As the gate voltage of T^{dis} exceeds the cut-off region, the DS current flows through T^{dis}, initiating C^{ch} to discharge. However, as the input voltage from the crossbar continues to supply the charge network, C^{ch} gets recharged again, maintaining the spike oscillations of the circuit.

The amplification of the signal at C^{ch} is shown in Figure 6.4. As the output signal of C_1^{dis} still remains considerably below supply voltage VDD, a second amplifier is connected to boost the amplitude of the output spikes (i.e., V_{out})

to facilitate the activation of succeeding neurons. The specific values of the components in the spike-generator is reported in Table 6.3.

Table 6.3: Component values in the spike-generator. Sourced from [13].

Component	Values
R^{ch}, R_1^{dis}, R_2^{dis}	10 kΩ
W^{ch}, W^{dis}	600 µm
L^{ch},L^{dis}	40 µm
C^{ch},C^{dis_1},C^{dis_2}	10 µF

6.2.2 Modeling of Spike-Generator

To fully leverage the computing functionalities of the proposed circuit, we also propose the corresponding modeling method to precisely describe the hardware behaviors. After that, we integrate the established model into the existing training framework of pNCs to enable the bespoke design of pSNCs.

To enable gradient-based training via backpropagation [8], a fully differentiable model to describe the transfer characteristic of the printed spike-generator circuit is needed. However, given the high circuit complexity and the unusual circuit mechanism, the common hardware-agnostic SNN training frameworks, e.g., the snnTorch [5], are incompatible with proposed circuits. Therefore, we employ approximation-based modeling approach (see Chapter 3.1.2) to build the precise surrogate model of the spike-generator. Different from the modeling of nonlinear activation circuit such as ptanh circuit, the behavior of the spike-generator should be described by a sequence-to-sequence model to incorporate the temporal dimension of the circuit. To this end, we utilize a Transformer-based as the surrogate spike-generator model to learn the circuit behavior for mapping the input voltage sequences into the output voltage sequences. A Transformer [20] 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 SOTA models like

BERT [4] and ChatGPT [15]. Notably, this Transformer-based model is not the one being printed, but rather an algorithmic surrogate model that serves the training of the circuit components.

To prepare the data required for training the surrogate spike-generator model, we conducted 5000 SPICE simulations for a single spike-generator circuit based on the pPDK [16]. The duration of the simulation is 3 s and the temporal step size is 1 ms. 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, namely,

1. constant voltages, ranging from 0 V to 2 V, serving to represent the case of stable inputs;
2. the output voltages obtained from step 1, i.e., V_{out}, representing the case of a cascade of multiple neurons; and
3. diverse harmonic signals with varying frequencies $(0-5\,\text{Hz})$, amplitudes $(0-1\,\text{A})$, phases $(0-2\pi)$, and their combinations, expressing the circuit behavior in other complex situations.

Considering the causality nature of the hardware, i.e., the output of the circuit must not depend on the future input signals, we introduce causal attention layers in the Transformer. After hyperparameter tuning, we select a 3-layer Transformer architecture, with each layer having three attention heads. Following the pipeline of from Chapter 3.1.2, we split the dataset, train and finally obtain the surrogate spike-generator model. The mean squared error (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. Figure 6.5 exemplifies some results to show the precision of the surrogate model compared to SPICE simulation.

6.2.3 Training of pSNCs

In the training of existing pNCs the cross-entropy loss $L(\cdot)$ is minimized with respect to the crossbar conductances $\boldsymbol{\theta}$ for maximizing the classification accuracy. However, given that the output of pSNC is a temporal data series, temporal dynamics of the circuit output need to be considered. Therefore, to

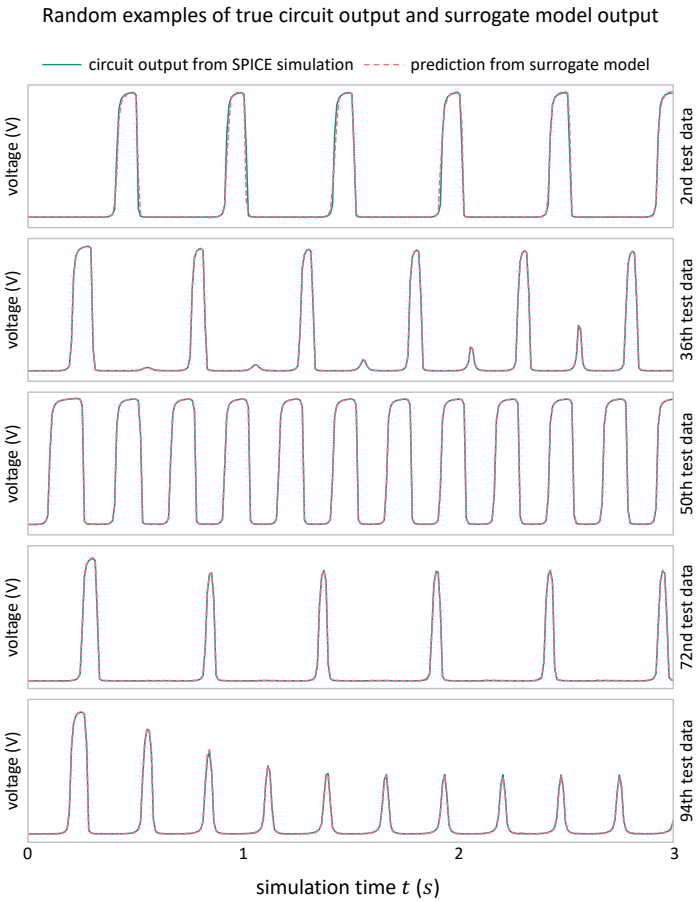


Figure 6.5: Visualization of random examples from test data. The green curves indicate the ground truth circuit output obtained from SPICE simulation, while the red dot lines illustrate the predicted circuit output through the trained Transformer-based surrogate spike generator model.

encourage the overall classification accuracy at every time step, a modified training objective can be formulated as

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \underbrace{\frac{1}{T}\sum_{t=0}^{T} L(\mathbf{X}_t, \mathbf{Y}, \boldsymbol{\theta})}_{\mathcal{L}(\mathcal{D},\boldsymbol{\theta})}, \tag{6.8}$$

where $\mathbf{X}_t$ is input data series at each time step t and $\mathbf{Y}$ denotes the corresponding classes. They are summarized by $\mathcal{D}$, denoting the target dataset for pSNCs, and $\boldsymbol{\theta}$ summarizes all the learnable conductances in the pSNC. Subsequently, as all the operations in Equation (6.8) are fully differentiable, and can be updated through gradient based optimizers, such as Adam [11] and SGD [1].

Notably, unlike the objective for pNCs, which includes physical quantities $\boldsymbol{q}$ for the nonlinear circuits as optimization variables, the learnable parameters in Equation (6.8) only include the crossbar resistors, representing the weights and biases. Because, in this thesis, we do not include the circuit components in the spike-generator in a parameterized way to train the surrogate model. Rather, these circuit components are considered predefined and fixed. However, this limitation can be easily addressed by collecting a more comprehensive simulation dataset that is parameterized by the circuit components as well.

6.2.4 Experiment

To evaluate the proposed pSNC, we implement the proposed training framework[2] with PyTorch [14] and conducted a comparative study of pSNC against the prior pNCs and the benchmark hardware-agnostic SNNs [5].

Datasets. Although SNNs or pSNCs 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, such as the datasets described in Table 4.1. 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 6.6. Therefore,

[2]Code available at `https://github.com/Neuromophic/Printed_Spiking_NN`.

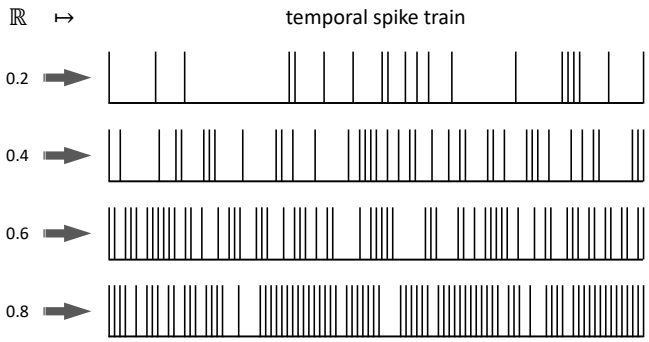


Figure 6.6: Dataset temporization that encodes real numbers into the density of the temporal spike trains.

this section directly takes the temporized datasets listed in Table 4.1. This not only keeps the consistency with other SNNs works, but also facilitates the fair comparison with previous pNCs.

Experiment setup. The major training setup follows the suggestion proposed in Chapter 4.1.3. As Equation (6.8) only includes crossbar conductances instead of the parameters in the spike-generator. For fair comparison, we also exclude the physical quantities $\boldsymbol{q}$ in the training of the baseline pNCs and only train $\boldsymbol{\theta}$. For SNNs, as we want to explore the potential classification abilities of the fully functional SNNs as a reference, we not only train the weights and biases, but also two critical learnable parameters inside the spike neuron, namely the leakage of the membrane voltage and the firing threshold. The impact of these factors can be observed in Figure 6.3(a) and (b).

Baselines. The basic pNCs are employed as a baseline to provide the hardware reference. Additionally, considering the target computing paradigm, pSNCs are also compared with its hardware-agnostic counterpart, SNNs, with the LIF mechanism [5].

Result. After training the networks, we select the models with the lowest loss on the validation set, as they are the ones that would be printed. Note that,

Table 6.4: The experiment result of spiking neural networks (SNNs), basic printed analog neuromorphic circuits (pNCs), and proposed printed spiking neuromorphic circuits (pSNCs) on 13 benchmark datasets. In addition to classification accuracy, the power and energy consumption is also reported. Sourced from [13].

Dataset	**SNN**	**pNC**			**pSNC**		
	Accuracy	**Accuracy**	**Power** (mW)	**Energy** (mJ)	**Accuracy**	**Power** (mW)	**Energy** (mJ)
1	1.00 ± 0.00	1.00 ± 0.00	5.51 ± 0.01	6.61 ± 0.01	1.00 ± 0.00	1.83 ± 0.11	2.19 ± 0.11
2	0.87 ± 0.03	0.89 ± 0.02	4.41 ± 0.04	5.29 ± 0.04	0.73 ± 0.03	1.42 ± 0.03	1.70 ± 0.03
3	0.97 ± 0.02	0.96 ± 0.01	11.3 ± 0.03	13.5 ± 0.03	0.97 ± 0.00	2.60 ± 0.01	3.12 ± 0.01
4	0.85 ± 0.03	0.75 ± 0.02	19.5 ± 0.01	23.3 ± 0.01	0.75 ± 0.06	5.30 ± 0.06	6.36 ± 0.06
5	0.99 ± 0.02	0.99 ± 0.01	7.88 ± 0.01	9.45 ± 0.01	0.92 ± 0.01	2.32 ± 0.03	2.78 ± 0.03
6	0.90 ± 0.09	0.89 ± 0.02	8.33 ± 0.04	9.99 ± 0.04	0.85 ± 0.06	2.24 ± 0.02	2.68 ± 0.02
7	0.97 ± 0.10	0.65 ± 0.06	5.48 ± 0.08	6.57 ± 0.08	0.87 ± 0.01	1.89 ± 0.02	2.26 ± 0.02
8	0.86 ± 0.12	0.83 ± 0.01	6.45 ± 0.02	7.74 ± 0.02	0.81 ± 0.05	1.81 ± 0.02	2.17 ± 0.02
9	0.32 ± 0.10	0.48 ± 0.06	16.1 ± 0.48	19.3 ± 0.48	0.46 ± 0.10	4.49 ± 0.21	5.38 ± 0.21
10	0.95 ± 0.13	0.85 ± 0.09	7.72 ± 0.03	9.26 ± 0.03	0.83 ± 0.02	1.71 ± 0.05	1.98 ± 0.05
11	0.97 ± 0.05	0.84 ± 0.06	11.5 ± 0.03	13.8 ± 0.03	0.85 ± 0.05	1.66 ± 0.04	1.99 ± 0.04
12	0.84 ± 0.10	0.84 ± 0.01	5.84 ± 0.02	7.00 ± 0.02	0.84 ± 0.04	1.52 ± 0.00	1.82 ± 0.00
13	0.83 ± 0.11	0.84 ± 0.02	7.23 ± 0.02	8.67 ± 0.02	0.82 ± 0.13	1.98 ± 0.02	2.30 ± 0.02
Average	$\mathbf{0.87 \pm 0.07}$	$\mathbf{0.83 \pm 0.03}$	$\mathbf{9.02 \pm 0.06}$	$\mathbf{10.8 \pm 0.06}$	$\mathbf{0.82 \pm 0.04}$	$\mathbf{2.37 \pm 0.05}$	$\mathbf{2.82 \pm 0.05}$

in accordance with the objective proposed in Equation (6.8), we computed the classification accuracy at every time step and subsequently averaged the accuracies over time to yield the overall classification accuracy of a dataset. These selected models are then evaluated on the test set. Finally, for each dataset, we summarized the mean accuracy and the power consumption with respect to the random seeds. The result is presented in Table 6.4. To get insights into the effectiveness of each model in various scenarios, we also averaged the accuracy and standard deviation with respect to target tasks. The comparative analysis among SNN, pNCs, and pSNCs (illustrated in Table 6.4) reveals that pSNCs exhibits a similar level of accuracy as SNN and pNCs. Across the 13 benchmark datasets, pSNCs achieved an average accuracy only 1% lower than the established reference pNCs.

In terms of power consumption, a significant improvement of pSNC over pNCs can be observed, which is around 3.86$\times$ higher in power and energy efficiency. This advancement is a consequence of the inherent sparsity of voltage activations within the network, aligning with the requirements of PE, particularly in low-power applications where energy efficiency is a critical concern.

Hardware cost. To investigate the additional hardware resources required by the pSNCs, we collected the device counts and total power saving of both the previous pNCs and the proposed pSNCs in different application scenarios. Analogously, we averaged the hardware costs across all datasets to provide a comparison regarding the hardware costs between pSNCs and its pNCs counterpart. The results can be seen in Table 6.5.

It can be seen, the total number of devices, on average, increased by 50% when employing pSNCs compared to pNCs due to a higher transistor count and the addition of capacitors. Although the area footprint expanded, pSNCs achieved a significant reduction in power consumption due to their spiking nature. Also, the datasets of various sizes provide a wide range of spectra to showcase the circuit capability, tailored for PE applications.

Table 6.5: Hardware costs of basic printed analog neuromorphic circuits (pNCs) and printed spiking neuromorphic circuits (pSNCs). Sourced from [21].

Dataset	#Transistors		#Resistors		#Capacitors		#Total Device	
	pNC	pSNC	pNC	pSNC	pNC	pSNC	pNC	pSNC
1	18	54	85	96	-	27	103	177
2	14	42	79	77	-	21	93	140
3	24	72	118	129	-	36	142	237
4	48	144	268	264	-	72	316	480
5	22	66	127	121	-	33	149	220
6	22	66	131	121	-	33	153	220
7	14	42	83	77	-	21	97	140
8	16	48	82	85	-	24	98	157
9	38	114	254	230	-	57	292	401
10	20	60	107	110	-	30	127	200
11	24	72	116	129	-	36	140	237
12	18	54	81	96	-	27	99	177
13	18	54	100	99	-	27	118	180
Average	**23**	**69**	**126**	**126**	**-**	**35**	**149**	**228**

6.2.5 Discussion

This section highlights the potential of merging printed manufacturing techniques and innovative algorithms in the field of PE and SNNs. Specifically, we by analyzing the working principle of general SNNs, we propose the circuit design of pSNCs. Subsequently, we established a precise surrogate spike-generator model to incorporate into the spiking behaviors into the training framework of pNCs and thus formed the training method for pSNCs. In this way, the trained circuits can exactly predict and reflect the real hardware pNCs.

The presentation of pSNCs completes the last major piece of the puzzle of implementing neuromorphic computing paradigms through PE (as shown in Figure 6.1). This progress significantly enhances the functionality of the general pNCs series and their adaptability to various edge application scenarios.

6.3 Summary

This chapter highlights the intrinsic limitations of the computational paradigm employed by the previous pNCs, i.e., the MLPs. Specifically, the MLP-like paradigms are unable to process temporal signals, restricting its application horizon. Additionally, as it is not an event-driven algorithm, the circuits must operate continuously, leading to inefficient energy consumption. These intrinsic challenges can not be addressed by the methodologies introduced in previous chapters. To tackle these issues and broaden the existing pNCs, this chapter introduces two novel variations of pNCs: the pRNC and the pSNC.

The new circuit designs leverage printed capacitors to incorporate temporal dependencies. We utilized capacitors and resistors to construct learnable low-pass filters, which serve as the foundation for the pTPB. As the iterative behavior of the pTPBs aligns with that of the RNNs, the proposed pRNCs are able to handle application scenarios that necessitate RNNs. For the pSNC, we analyzed the working principles of SNNs and developed a hardware-friendly spike-generator that contains a charging and a discharging circuit.

Beyond proposing novel circuit designs, we also developed accurate optimization models for these circuits to describe their hardware behaviors. Considering the complexity of the circuits, we employed physics-informed modeling for the pRNCs and approximation-based modeling for the pSNCs. Both modeling approaches exhibit a high level of hardware consistency, accurately reflecting the circuit behaviors in algorithmic level. Based on these models, we proposed the circuit objective functions for highly bespoke training tailored to specific tasks.

Although this chapter does not involve the reliability and practicality considerations discussed in the previous chapters as metrics and objectives, those methods can be seamlessly applied to new circuits. This chapter mainly focuses on exploring the advantages of the circuit innovations, because the newly proposed circuits allow PE-based neuromorphic computing hardware to cover the mainstream computing paradigm, enabling the pNC family to provide efficient, low-energy computing in most scenarios, and laying a solid foundation for the popularization of pNC in edge scenarios.

Bibliography

[1] Léon Bottou. “Online algorithms and stochastic approximations”. In: *Online learning in neural networks* (1998).

[2] Germund Dahlquist and Åke Björck. *Numerical methods*. Courier Corporation, 2003.

[3] Hoang Anh Dau et al. *The UCR Time Series Classification Archive*. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`. Oct. 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[5] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. “Training spiking neural networks using lessons from deep learning”. In: *Proceedings of the IEEE* (2023).

[6] Samanwoy Ghosh-Dastidar and Hojjat Adeli. “Spiking neural networks”. In: *International journal of neural systems* 19.04 (2009), pp. 295–308.

[7] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. “A novel connectionist system for unconstrained handwriting recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 31.5 (2008), pp. 855–868.

[8] Robert Hecht-Nielsen. “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[9] Mohammad Javad Mirshojaeian Hosseini, Elisa Donati, Tomoyuki Yokota, Sunghoon Lee, Giacomo Indiveri, Takao Someya, and Robert A Nawrocki. “Organic electronics Axon-Hillock neuromorphic circuit: towards biologically compatible, and physically flexible, integrate-and-fire spiking neural networks”. In: *Journal of Physics D: Applied Physics* 54.10 (2020), p. 104004.

[10] Heikki Hyötyniemi. “Turing machines are recurrent neural networks”. In: *Proceedings of step* 96 (1996), p. 15.

[11] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[12] Anqi Mao, Mehryar Mohri, and Yutao Zhong. “Cross-entropy loss functions: Theoretical analysis and applications”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 23803–23828.

[13] Priyanjana Pal, Haibin Zhao, Maha Shatta, Michael Hefenbrock, Sina B Mamaghani, Sani Nassif, Michael Beigl, and Mehdi B Tahoori. “Analog Printed Spiking Neuromorphic Circuit”. In: *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2024, pp. 1–6.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[16] Farhan Rasheed, Michael Hefenbrock, Michael Beigl, Mehdi B Tahoori, and Jasmin Aghassi-Hagmann. “Variability modeling for printed inorganic electrolyte-gated transistors and circuits”. In: *IEEE transactions on electron devices* 66.1 (2018), pp. 146–152.

[17] Paul Rodriguez, Janet Wiles, and Jeffrey L Elman. “A recurrent neural network that learns to count”. In: *Connection Science* 11.1 (1999), pp. 5–40.

[18] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. “A survey of neuromorphic computing and neural networks in hardware”. In: *arXiv preprint arXiv:1705.06963* (2017).

[19] Vanessa Tischler, Piotr Dudek, Jayawan Wijekoon, Leszek A Majewski, Yasunori Takeda, Shizuo Tokito, and Michael L Turner. “An integrate-and-fire neuron circuit made from printed organic field-effect transistors”. In: *Organic Electronics* 113 (2023), p. 106685.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[21] Haibin Zhao, Priyanjana Pal, Michael Hefenbrock, Michael Beigl, and Mehdi B Tahoori. “Towards Temporal Information Processing – Printed Neuromorphic Circuits with Learnable Filters”. In: *Proceedings of the 18th ACM International Symposium on Nanoscale Architectures*. 2023, pp. 1–6.

[22] Haibin Zhao, Alexander Scholz, Michael Beigl, Si Ni, Surya Abhishek Singaraju, and Jasmin Aghassi-Hagmann. “Printed Electrodermal Activity Sensor with Optimized Filter for Stress Detection”. In: *Proceedings of the 2022 ACM International Symposium on Wearable Computers*. 2022, pp. 112–114.

7 Conclusion and Outlook

Printed electronics (PE), due to its flexible additive manufacturing, diverse material choices, and extremely low fabrication cost, has emerged as a pivotal enabler of edge devices within the context of IoT. Printed analog neuromorphic circuits (pNCs) leverage these unique advantages and also integrate the excellent computational capabilities of neuromorphic computing. Consequently, pNCs hold the potential to informatize, electronify, and intellecturalize the most edge scenarios in daily life, such as smart packaging, smart band-aids, and smart clothing.

However, pNCs also face challenges inherent to additive manufacturing, such as large feature size, low integration density, and high variations. While existing research has proposed some circuit primitives of pNCs and validated the basic concept of these circuits, significant improvement is needed to enable pNCs being practically deployed. This thesis addresses these challenges through a series of studies around pNCs.

The first major contribution of this thesis is developing the methodology for establishing optimization models for pNCs, while incorporating physical and technical constraints. After the modeling, this thesis enhances an existing ML-based training method, addressing the weakness of gradient-based training. By introducing heuristic gradient, the training process is enabled even with non-differentiable variables and operations. In addition, an EA-based training method is proposed to efficiently search for optimal circuit architectures, bringing the training to a higher level. Importantly, these methodologies are not only applicable to modeling and training for pNCs, but can also be transplanted to other circuit problems, which lays a solid foundation for subsequent contributions.

The second primary contribution focuses on improving the reliability of pNCs. This dissertation experimentally measures and models the stochastic

aging behavior of the organic printed resistors (PEDOT:PSS) and introduces aging-aware training to enhance classification accuracy over the circuit lifetime. Besides, this work identifies three main factors affecting the circuit reliability, namely sensing error, printing variation, and device aging. To this end, this work proposes a training objective that considers all these factors to significantly boost circuit reliability. To even further enhance the circuit reliability, new nonlinear activation circuits are designed and optimally selected through an EA. Experimental results demonstrate that, to achieve high robustness, the activation circuits need not only to be stable against variation, but also to exhibit a low slope of its characteristic to resist input variations caused by the previous layer. Lastly, this thesis examines the impact of catastrophic faults on pNCs and highlights the limitations of algorithmic optimization in such cases, justifying and opening new research areas for the testing and post-printing measures.

The next contribution enhances the practical utility of pNCs. Reliability is only an initial prerequisite of the deployment of pNCs. There are further challenges like circuit cost, battery lifetime, and circuit compactness. Given the target application of PE in low-cost edge applications, such as smart packaging, pNCs must be inexpensive. By leveraging the additive manufacturing of PE, the thesis combines the unique benefits of both high- and low-volume manufacturing, and significantly reducing fabrication costs through split additive manufacturing. Regarding the power consumption, this work precisely models the power of pNCs and proposes power-aware training based on the augmented Lagrangian method. With this approach, the circuit can be trained to provide the best classification accuracy within the given power budget. This approach outperforms the existing penalty-based methods. Lastly, this dissertation also focuses on circuit compactness by developing an accurate area model to predict circuit area footprint from the netlist. Notably, the area model can provide precise prediction of the circuit area after placement and routing. As the circuit compactness is strongly related to the circuit architecture, an EA in employed to conduct area-aware training. This approach enhances compactness, making the circuits suitable for area-constrained applications like smart band-aids.

The final contribution of this work extends the computational paradigm of

pNCs. Existing designs of pNCs primarily based on MLP-scheme, which lacks capabilities for temporal signal processing. This is essentially due to the absence of circuit components with time-dependencies. To address this, we utilize printed capacitors to create low-pass filters, and further structure printed recurrent neuromorphic circuits (pRNCs). Subsequently, a hardware-software co-design framework is proposed to enable the bespoke training of both conductances (weights) and capacitances (filtering behaviors) for specific tasks, achieving a performance nearly equivalent to hardware-agnostic Elman RNNs. This work also designs printed spiking neuromorphic circuits (pSNCs). By proposing the corresponding parametric model as well as the training framework, the pSNCs can be trained on target datasets to yield results that are comparable to hardware-agnostic LIF SNNs. The extension of computing paradigms enriches the family of pNCs, and thus expands their application range and scenarios.

Despite the progress made in this thesis, which lays a robust foundation for the practical deployment of pNCs, there remain multiple opportunities for further exploration and enhancement of pNCs.

Regarding the reliability, while parameter variations such as aging and printing errors have been incorporated into the training process, catastrophic faults pose a significant challenge at the algorithmic level, where intuitive solutions fall short. This underscores the need for future research on the testing of pNCs, such as designing input patterns to efficiently locate faults. Similar to [1], once a fault is localized, additive manufacturing techniques can be utilized for post-printing recovery.

pNCs currently lack specialized placement and routing algorithms. Although preliminary work [3] has explored this issue, considering the number of wire crossovers, it employs EA, which is inefficient, and is targeted to digital pNCs. Although we have borrowed the algorithm from mature and efficient commercial PCB design tools in this thesis, yet this approach is not optimal neither. Future algorithms should aim to support the routing and placement for pNCs at the transistor level, while respecting various technical characteristics and specifications of the PE.

New materials often drive disruptive innovations. With the recent realiza-

tion of printed memristors [2], memristor-based pNCs could emerge in the future. The unique properties of memristors allow for reconfigurable circuits by adjusting the resistance. This feature can be utilized to compensate for conductance degradation due to aging, or to reconfigure the identical circuits for different tasks, which can significantly reduce the fabrication costs for bespoke printing. Additionally, memristors may also enable the development of entirely new computational paradigms in the future.

Furthermore, algorithms for modeling and training pNCs can also be improved. In modeling, particularly approximation-based modeling, obtaining a representative dataset through extensive SPICE simulations is necessary before building the ANN-based surrogate models. However, the data preparation can be time-consuming, and large-scale surrogate models are often required to accurately describe circuit behavior, increasing the cost of training surrogate models and reducing the efficiency of the training framework of pNCs. Recent work have directly linked PyTorch, the training framework for pNCs, to the SPICE [4]. Future research should follow this progress for accelerated pNCs training. Moreover, to conduct comprehensive training for pNCs taking account for both aging, printing variation, fabrication cost, power consumption, and circuit area footprint, numerous MC samples will be introduced. Therefore, employing analytic or hybrid analytic-numerical methods can be considered in future work to handle of stochastic variables in the modeling. Also, the training will be a multi-objective, requiring several trade-off parameters. This may significantly reduce training efficiency. To this end, more advanced methods to achieve Pareto-optimality should be adopted.

It is hoped that the methods, discussions, and ideas presented in this thesis will provide support and inspiration for future work on the algorithm-driven design and optimization of pNCs.

Bibliography

[1] Michael Hefenbrock, Dennis D Weller, Jasmin Aghassi-Hagmann, Michael Beigl, and Mehdi B Tahoori. “In-situ tuning of printed neural networks for variation tolerance”. In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2022, pp. 72–75.

[2] Hongrong Hu, Alexander Scholz, Christian Dolle, Alexander Zintler, Aina Quintilla, Yan Liu, Yushu Tang, Ben Breitung, Gabriel Cadilha Marques, Yolita M Eggeler, et al. “Inkjet-Printed Tungsten Oxide Memristor Displaying Non-Volatile Memory and Neuromorphic Properties”. In: *Advanced Functional Materials* 34.20 (2024), p. 2302290.

[3] Farhan Rasheed, Michael Hefenbrock, Rajendra Bishnoi, Michael Beigl, Jasmin Aghassi-Hagmann, and Mehdi B Tahoori. “Crossover-aware placement and routing for inkjet printed circuits”. In: *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 16.2 (2020), pp. 1–22.

[4] Mikail Yayla, Simon Thomann, Md Mazharul Islam, Ming-Liang Wei, Shu-Yin Ho, Ahmedullah Aziz, Chia-Lin Yang, Jian-Jia Chen, and Hussam Amrouch. “Reliable Brain-inspired AI Accelerators using Classical and Emerging Memories”. In: *2023 IEEE 41st VLSI Test Symposium (VTS)*. IEEE. 2023, pp. 1–10.

Curriculum Vitae

General

Haibin Zhao
Born in 1995, China
haibin.zhao@kit.edu
https://haibinzhao-950102.github.io/MyPage/

Education

Karlsruhe Institute of Technology	Oct. 2018 - Dec. 2020
M.Sc. Mechatronics	Karlsruhe, Germany
Chongqing University	Sep. 2013 - Jul. 2017
B.Sc. Mechanical Engineering	Chongqing, China
Qingdao No.2 Middle School	Sep. 2010 - Jul. 2013
High School	Shandong, China
Qingdao No.21 Middle School	Sep. 2007 - Jul. 2010
Middle School	Shandong, China
Qingdao Yiyang Road Primary School	Sep. 2001 - Jul. 2007
Elementary School	Shandong, China

Working Experience

Karlsruhe Institute of Technology	Jan. 2021 - now
Scientific Researcher	TECO, KIT, Germany
Karlsruhe Institute of Technology	Apr. 2020 - Nov. 2022
Student Research Assistant	ISAS, KIT, Germany
Karlsruhe Institute of Technology	Oct. 2019 - Jun. 2020
Student Research Assistant	IPR, KIT, Germany
Qingdao Yonghe Packing Machinery Co., Ltd.	Jul. 2017 - Nov. 2017
Product Design Assistant	Shandong, China
Qingdao Yonghe Packing Machinery Co., Ltd.	Jul. 2016 - Aug. 2016
Machinery Assembling Assistant	Shandong, China

Other Publications

This section provides an overview of other publications made by the author during the doctoral research period. While these publications exhibit irrelevance to the primary topic of this work, they have not been included in the main body of this dissertation. They serve as a supplement to the papers listed in Chapter 1.3.

Haibin Zhao, Christopher Funk, Benjamin Noack, Uwe Hanebeck, and Michael Beigl. "Kalman Filtered Compressive Sensing Using Pseudo-Measurements". In: *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE. 2021, pp. 1–8.

Haibin Zhao, Tobias Röddiger, and Michael 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.

Tobias Röddiger, Christopher Clarke, Paula Breitling, Tim Schneegans, **Haibin Zhao**, Hans Gellersen, and Michael Beigl. "Sensing with earables: A systematic literature review and taxonomy of phenomena". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6.3 (2022), pp. 1–57.

Yexu Zhou, **Haibin Zhao**, Yiran Huang, Till Riedel, Michael Hefenbrock, and Michael Beigl. "TinyHAR: A lightweight deep learning model designed for human activity recognition". In: *Proceedings of the 2022 ACM International Symposium on Wearable Computers*. 2022, pp. 89–93.

Haibin Zhao, Yexu Zhou, Till Riedel, Michael Hefenbrock, and Michael Beigl. "Improving Human Activity Recognition Models by Learnable Sparse Wavelet Layer". In: *Proceedings of the 2022 ACM International Symposium on Wearable Computers*. 2022, pp. 84–88.

Yiran Huang, **Haibin Zhao**, Yexu Zhou, Till Riedel, and Michael Beigl. "Standardizing Your Training Process for Human Activity Recognition Models – A Comprehensive Review in the Tunable Factors". In: *Proceedings of the 20th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2023)*. Springer, 2023.

Yiran Huang, Yexu Zhou, **Haibin Zhao**, Till Riedel, and Michael Beigl. "A Survey on Wearable Human Activity Recognition: Innovative Pipeline Development for Enhanced Research and Practice". In: *2024 IEEE International Joint Conference on Neural Network (IJCNN 2024)*. IEEE. 2024.

Yiran Huang, Yexu Zhou, **Haibin Zhao**, Till Riedel, and Michael Beigl. "Optimizing AutoML for Tiny Edge Systems: A Baldwin-effect Inspired Genetic Algorithm". In: *22nd IEEE International Conference on Pervasive Computing and Communicaitons (PerCom 2024)*. IEEE. 2024.

Yexu Zhou, Tobias King, Yiran Huang, **Haibin Zhao**, Till Riedel, Tobias Röddiger, and Michael Beigl. "Enhancing Efficiency in HAR Models: NAS Meets Pruning". In: *22nd IEEE International Conference on Pervasive Computing and Communicaitons (PerCom 2024)*. IEEE. 2024.

Yexu Zhou, **Haibin Zhao**, Michael Hefenbrock, Siyan Li, Yiran Huang, and Michael Beigl. "Deep Neural Network Pruning with Progressive Regularizer". In: *2024 IEEE International Joint Conference on Neural Network (IJCNN 2024)*. IEEE. 2024.

Yexu Zhou, **Haibin Zhao**, Yiran Huang, Tobias Röddiger, Murat Kurnaz, Till Riedel, and Michael Beigl. "AutoAugHAR: Automated Data Augmentation for Sensor-based Human Activity Recognition". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8.2 (2024), pp. 1–27.

Yiran Huang, Yexu Zhou, **Haibin Zhao**, Likun Fang, Till Riedel, and Michael Beigl. "ExTea: An Evolutionary Algorithm-Based Approach for Enhancing Explainability in Time-Series Models". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2024.

Haibin Zhao, Tobias Röddiger, Yufei Feng, and Michael Beigl. "Fit2Ear: Generating Personalized Earplugs from Smartphone Depth Camera Images". In: *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '24)*. ACM, 2024.

Yexu Zhou, Tobias King, **Haibin Zhao**, Yiran Huang, Till Riedel, and Michael Beigl. "MLP-HAR: Boosting Performance and Efficiency of HAR Models on Edge Devices with Purely Fully Connected Layers". In: *ACM International Symposium on Wearable Computers (ISWC '24)*. ACM, 2024.