Towards Functional Safety of Neural Network Hardware Accelerators: Concurrent Out-of-Distribution Detection in Hardware Using Power Side-Channel Analysis

Vincent Meyers[1], Michael Hefenbrock[2], Mahboobe Sadeghipourrudsari[1], Dennis Gnad[1], Mehdi Tahoori[1]
[1]Karlsruhe Institute of Technology, [2]RevoAI GmbH
Karlsruhe, Germany
firstname.lastname@kit.edu,michael.hefenbrock@revoai.de

Abstract

For AI hardware, functional safety is crucial, especially for neural network (NN) accelerators used in safety-critical systems. A key requirement for maintaining this safety is the precise detection of out-of-distribution (OOD) instances, which are inputs significantly distinct from the training data. Neglecting to integrate robust OOD detection may result in possible safety hazards, diminished performance, and inaccurate decision-making within NN applications. Existing methods for OOD detection have been explored for full-precision models. However, the evaluation of methods on quantized neural network (QNN), which are often deployed on hardware accelerators such as FPGAs, and on-device hardware realization of concurrent OOD detection (COD) is missing in literature. In this paper, we provide a novel approach to OOD detection for NN FPGA accelerators using power measurements. Utilizing the power side-channel through digital voltage sensors allows on-device OOD detection in a non-intrusive and concurrent manner, without relying on explicit labels or modifications to the underlying NN. Furthermore, our method allows OOD detection before the inference finishes. Additionally to the evaluation, we provide an efficient hardware implementation of COD on an actual FPGA.

CCS Concepts

• **Hardware → Safety critical systems**.

Keywords

out of distribution detection, neural network accelerators, power side channel

ACM Reference Format:
Vincent Meyers[1], Michael Hefenbrock[2], Mahboobe Sadeghipourrudsari[1], Dennis Gnad[1], Mehdi Tahoori[1]. 2025. Towards Functional Safety of Neural Network Hardware Accelerators: Concurrent Out-of-Distribution Detection in Hardware Using Power Side-Channel Analysis. In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3658617.3697745

ASPDAC '25, January 20–23, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3697745

1 Introduction

The rapid advancement in artificial intelligence (AI) and deep learning algorithms has led to the widespread deployment of NNs in various applications, including safety-critical domains such as autonomous driving, medical diagnosis, and authentication [6]. As the complexity and size of NN models have grown, the need for specialized hardware accelerators, such as graphical processing units (GPUs), tensor processing units (TPUs), and field-programmable gate arrays (FPGA), has become increasingly important [15, 27]. These hardware accelerators enable efficient and high-speed execution of NN models, facilitating the real-time analysis required in many AI applications.

However, despite their widespread use and remarkable performance, NN accelerators are not immune to certain limitations and vulnerabilities. One critical concern is the inability of NNs to identify inputs that are outside of their training (intended) data distribution, commonly known as OOD inputs [6]. When NNs encounter OOD inputs, they often provide unreliable or incorrect predictions, which can have severe consequences in safety-critical domains. Detecting and effectively handling OOD inputs is thus of paramount importance when deploying NN accelerators in real-world scenarios. This promotes the *fail-safe* property in NNs in which the system either makes trusted outputs (predictions) or flags the existence of faulty (untrusted) predictions. Recently, researchers have explored various techniques to address this challenge, including confidence thresholding [2, 6], entropy maximization [1], and bayesian NN [3].

Most existing techniques require computationally expensive procedures, such as input reconstruction or density estimation, which can hinder low-latency performance in resource-constrained edge environments. Additionally, OOD detection methods require either the NNs input, output probabilities, logits or even activation patterns. This requirement delays the detection until the end of the inference, thus, adding latency, which might not be acceptable in safety-critical systems. Here, a form of COD is required, which can be performed during the inference and on the hardware. Furthermore, existing OOD detection methods have yet to be evaluated on QNN for edge deployment, which have reduced precision for weights and activation values. Addressing these limitations is crucial to ensure robust and reliable OOD detection in NN accelerators and enable their safe deployment in a wide range of applications.

In this work, we propose a novel approach to OOD detection by leveraging power side-channel measurements to distinguish between the power profiles of in-distribution (ID) samples and OOD samples. We harness the availability of on-chip FPGA voltage

sensors to remotely capture the power consumption of the NN accelerator during the execution of ID samples. By collecting a large dataset of power measurements from these ID samples, we can extract the underlying patterns and features of normal power consumption behavior. We analyse the collected power traces for ID and OOD samples and determine thresholds for OOD detection based on a scoring-function, such as minimum or summation.

Aside from the ability to perform OOD detection in a non-intrusive manner, this approach also offers additional advantages. Most notably, OOD detection can be performed concurrently to the inference on the same hardware, and the detection mechanism can be implemented directly on the hardware NN accelerator in a low-overhead and zero-latency manner.

The contributions of this paper can be summarized as:

- Non-intrusive and concurrent power side-channel based detection of OOD samples for QNN FPGA accelerators
- Zero-latency and low-overhead hardware implementation of the proposed approach

In the remaining paper in Section 2 we introduce OOD detection methods and power side-channels of NN accelerators. Our method for OOD detection is presented in Section 3. We explain our experimental setup for the power trace acquisition in Section 4. The results are presented and discussed in Section 5. The paper is concluded in Section 6.

2 Background and Related Work

2.1 Out-of-Distribution Detection

OOD detection is a crucial task in machine learning models, aiming to identify samples that differ significantly from the training distribution [6]. Several methods have been proposed to tackle this challenge, which we briefly review in the following. As our proposed approach is non-intrusive to the model, we focus on related work, which does not rely on re-training the model or interfering with the model's training process.

2.1.1 Maximum Softmax Probability (MSP) [6]. OOD samples can be identified by setting a confidence threshold for softmax output probabilities x, with low class probabilities that fall below the threshold corresponding to OOD samples [2].

$$f(x) = \max_i \left\{ \frac{e^{x_i}}{\sum_j e^{x_j}} \right\} \quad \text{for} \quad i = 1, \ldots, n \tag{1}$$

However, it has been shown that NNs can have overconfident predictions even for samples far from the training data [4].

2.1.2 MaxLogits [5]. In a neural network classifier, the logits are the raw, unnormalized predictions produced by the final layer before applying the softmax function. The maximum logit refers to the highest value among these pre-softmax outputs x.

$$f(x) = \max_i \{x_i\} \quad \text{for} \quad i = 1, \ldots, n \tag{2}$$

The intuition behind this approach is that for ID samples, the model is likely to produce a high maximum logit value, indicating strong confidence in its prediction. Conversely, for OOD samples, the maximum logit is expected to be lower, as the model should be less confident when encountering unfamiliar data.

2.1.3 Entropy Maximization [1]. The method involves calculating the softmax entropy of the network's output x.

$$f(x) = -\sum_{i=1}^{n} x_i \log(x_i) \tag{3}$$

For ID samples, the network is expected to produce confident (low entropy) predictions, while for OOD samples, the predictions should be less confident (high entropy).

2.1.4 Energy-based [14]. Energy-based methods [14], on the other hand, model the distribution of the training samples using a score function. OOD samples can then be identified based on their higher energy score, which has a lower likelihood of occurrence.

$$f(x) = -t \log \left(\sum_{i=1}^{n} e^{\frac{x_i}{t}} \right) \tag{4}$$

While this method is more computationally efficient and parameter-free, we notice that it does not perform well in detecting input image corruptions or OOD samples with pixel values close to the range of the ID dataset.

2.1.5 ODIN [13]. ODIN combines the following key components:

(1) Temperature scaling: The authors apply temperature scaling to the softmax function, which helps to separate the softmax score distributions between ID and OOD samples.
(2) Input perturbation: Small controlled perturbations are added to the input images, which further enhances the separation between ID and OOD samples.

The method works by observing that when these two techniques are applied, the softmax scores for ID images tend to increase, while those for OOD images typically decrease.

2.1.6 Summary. Whilst each of these methods has its merits, they also suffer from certain drawbacks. Most notably, the absence of efficient hardware implementation capabilities. Furthermore, the existing methods can not be performed concurrently to the inference, adding additional latency after the final output is calculated. These challenges motivate the need for further research in developing robust and accurate techniques for OOD detection for machine learning models on dedicated hardware accelerators.

2.2 Power Side-Channel of Neural Network Accelerators

The power side-channel can be measured as data-dependent voltage fluctuations that leak information during the execution of a device [9]. These side-channels expose information about the computations being performed, which can jeopardize system security and privacy. Power consumption is influenced by factors such as input data patterns, model architecture, weights, and specific operations during NN execution.

However, power side channels of NN accelerators also play a pivotal role in the security and privacy landscape. While side channels typically pose vulnerabilities, they can also be harnessed for constructive purposes within the context of hardware accelerators, such as hardware Trojan detection [20] and anomaly detection [24]. These works emphasize the utilization of power side channels for strengthening security measures and enhancing privacy protection.

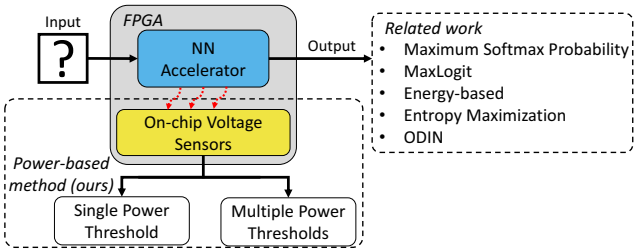


Figure 1: Setup for our OOD detection using on-chip voltage measurements and related work using outputs or internal values of the neural network.

Measurements can be done locally, with probes and an oscilloscope, or as in our case, remotely, with dedicated voltage sensing logic. In this work, we utilize multiple routing delay sensors (RDS) to measure voltage fluctuations in FPGAs [22]. In contrary to other on-chip sensors, which use a tapped delay line or ring oscillators, RDS utilize the FPGA routing resources for sensing voltage fluctuations. By utilizing RDS for voltage sensing, we remove the requirement for additional peripheral hardware. We have opted for RDS because they have demonstrated superior performance compared to other sensors and are more resilient to fluctuations in temperature[22].

3 Concurrent OOD Detection

In this section, we present our method for utilizing the power side-channel of QNN FPGA accelerators for OOD detection. Our approach for the detection mechanism, relies on calculating a power profile of expected power consumption for ID data. The power profile can be represented as a single or multiple thresholds τ_i.

Figure 1 shows the general setup for the power side-channel based OOD detection. On-chip sensors measure voltage during the inference, and a score function f_{score} maps the trace to a single value. If the score crosses a threshold (or multiple), we set the OOD flag. The steps of our method can be summarized as:

(1) Collect power measurements from the training or test set
(2) Offline analysis of power traces and determine threshold(s)
(3) Write threshold(s) to memory and evaluate on hardware

We assume that the traces are represented as fixed-point numbers in hardware, which leads to the same results in the software evaluation with float-32. This is due to the simple calculations, which do not require high precision and the sensor values being represented as integer values. The fractional part is only required for taking the average of sensors.

We evaluate established criteria [6], such as the FPR-95, receiver operating characteristic (ROC) curve and area under ROC curve (AUC). The ROC curve illustrates the balance between true positive rate (TPR) and false positive rate (FPR) at various decision thresholds and the AUC is a summary statistic. The FPR-95 is the FPR of OOD examples at 95% TPR of ID examples. Lower FPR-95 indicates better OOD detection performance.

3.1 OOD Detection Hardware Implementation

To ensure the efficiency of the proposed technique, the design is implemented together with the FINN accelerator IP on the FPGA.

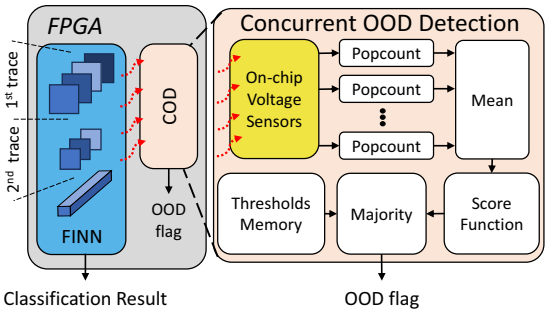


Figure 2: Overview of hardware implementation of power side-channel based COD on FPGA with FINN.

The score of a sensor value v_i for the one clock cycle is denoted as

$$score = f_{\text{score}}\left(\frac{\sum_i^{\#\text{sensors}} \text{Popcount}(v_i)}{\#\text{sensors}}\right) \tag{5}$$

and the detection mechanism for a single threshold is defined as follows, where τ is the threshold

$$f(score) = \begin{cases} \text{accept}, & \text{if score} \leq \tau \\ \text{reject}, & \text{otherwise} \end{cases} \tag{6}$$

We implement Equation 5 and Equation 6 as displayed in Figure 2. This figure shows the FINN accelerator and the hardware implementation of the power side-channel OOD detection mechanism. The RDS are started with the start of the inference, and stop after a configurable number of clock cycles. In each clock cycle the sensors measure voltage fluctuations as an array of '1's and '0's. We utilize `popcount` modules, which count all '1's with a combinational module and pass them to `mean`. The module `mean` calculates the average of all popcount outputs. These two modules work simultaneously within a single clock cycle. The average sensor result is passed to the `score function`, which maps the power trace to a single value. Here, we explored simple computations, such as summation or minimum. For the case of multiple thresholds, each section of the trace must be compared to different thresholds. As these thresholds can be written to memory during runtime, there is no need to reconfigure the device. The `majority` module performs the comparison and tracks the results for each section and issues the OOD detection flag after the last trace is processed.

4 Experimental Setup

We employ the publicly accessible FINN framework [23] to generate all neural network accelerators for our experiments. Subsequently, we deploy these accelerators onto a Zynq Ultrascale ZCU104. RDS [22] are used to measure voltage fluctuations. These sensors are co-located on the same FPGA. The floorplan for the setup is displayed in Figure 3, with the accelerator in yellow, sensors in red and the detection mechanism in orange. In our experiments, the sensors run at 200MHz and the accelerator at 50MHz, however, the sensors can also run at higher speeds with the correct calibration. The frequency of the sensors is thus not a limitation.

We evaluate our method on a multi-layer-perceptron with 4 layers and 64 neurons in each layer, which we will call MLP-64. VGG-like [21] CNNs with 6 convolutional layers, followed by 3 fully-connected layers, are also considered for the GTSRB [7] and

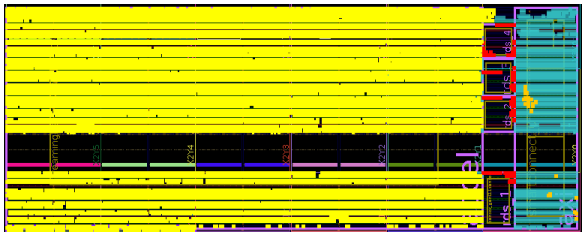


Figure 3: Floorplan of the FINN accelerator (yellow) and multiple RDS (red), OOD detection (orange), remaining control and debug logic (cyan).

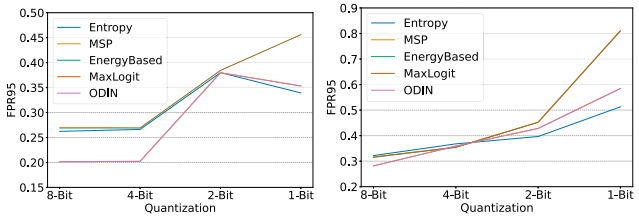
Figure 4: Exemplary samples from MNIST with various corruptions as applied in MNIST-C [17].

CIFAR-10 dataset. The classification accuracy of the MNIST, CIFAR-10 and GTSRB model are 92.95%, 84.61% and 98.06% respectively. The models are quantized to 1-bit weights and activations with the Brevitas [19] library. We use the following four OOD datasets for models trained on MNIST [12]: (1) FashionMNIST [25], (2) CIFAR10 [11] (grayscale, resized to 28×28), (3) MNIST-C [17] and (4) independent Gaussian noise drawn from $\mathcal{N}(0, 1)$.

Each of these datasets consists of 10k samples. The MNIST-C dataset consists of various corruptions applied to all sample of the original MNIST dataset. Some of the corruptions, such as change in brightness, do not have an effect on binary NNs, as the pixel values need to cross a threshold to make a change. Thus, we use only the following corruptions from the MNIST-C dataset: *rotate*, *line*, *dotted_line*, *zigzag*, *inverse*, *stripe*, *canny_edges*, *impulse_noise*. For the CNNs, we use the following OOD datasets: (1) TinyImageNet [16], (2) SVHN [18], (3) GTSRB / CIFAR-10 (respectively) and (4) independent Gaussian noise drawn from $\mathcal{N}(0, 1)$. For the model trained on GTSRB, we reduce the number of OOD samples to fit the number of ID samples, so that the ROC is not skewed and the FPR95 and AUC remain informative. Here, the number of samples is 3870. We repeat the experiments 20× and find no significant variation in the results.

The intermediate states of the FINN accelerator do not reset after an inference. As the data-dependent power consumption of the accelerator is driven by the switching activity, this makes the power measurements related to an input sample dependent on the previous one. Thus, we can determine our threshold by collecting power traces for transitions from ID to ID samples. Then, if an OOD samples is queried after an ID sample, the score function of the power trace will be remarkably different. A visualization for this behavior can be found in Figure 7, which displays the average power trace for MNIST and corresponding OOD datasets. Here, the traces can be easily distinguished by the first voltage drop or the minimum of the trace.

For the comparison with existing methods we use the python library *pytorch-ood* by Kirchheim et al. [8]. We consider *Entropy Maximization*, *MaxLogit*, *MaxSoftmax*, *Energy-based* OOD-detection and *ODIN*. The performance on each OOD dataset for these methods is presented in Figure 6.



(a) CIFAR10 **(b) FashionMNIST**

Figure 5: Evaluation of FPR-95 for related work with models trained on MNIST on three quantization levels and CIFAR10 and FashionMNIST as OOD datasets.

5 Results

We begin by evaluating related work on QNNs and then report the results for our proposed method. For the comparison with related work, we consider the results from a fixed point implementation, as it would be in hardware.

5.1 Evaluation of Related Work

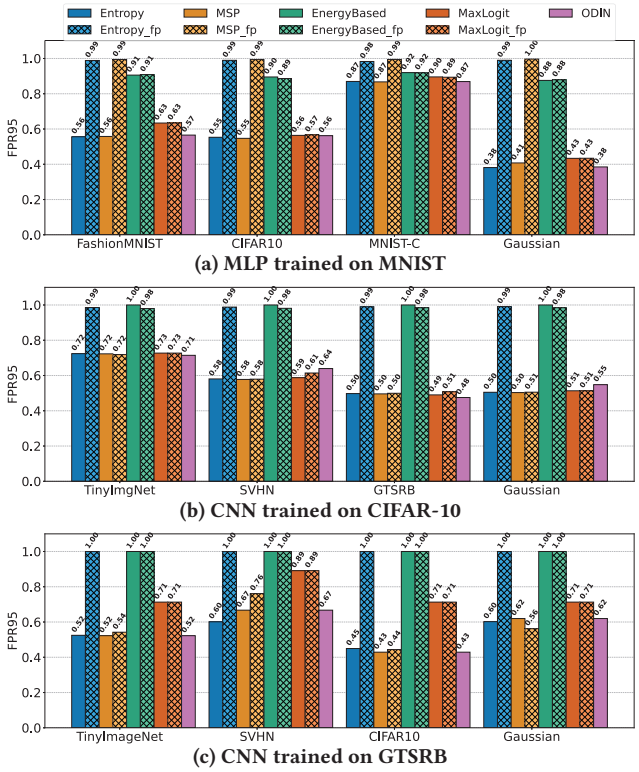
As a preliminary experiment, we compare the FPR-95 of the considered related work for different levels of quantization. The results for this experiment are presented in Figure 5, which shows the FPR-95 for the MLP64 with 1,2 4, and 8-Bit quantization. We notice that with decreasing precision of weights and activations, the FPR-95 increases, displaying an deterioration of the detection mechanism.

On top of model quantization, the calculations in hardware, like entropy and softmax, are mostly performed with fixed-point representation. Thus, we implement fixed-point variants of the evaluated methods from related work. Here, we consider binary neural networks. For softmax we also consider an hardware-efficient approximation as in [10]. We use 10-bit fixed-point numbers with 2 bit for the integer and 8 bit for the fractional part, which is sufficient for representing the logits and softmax values without loss of accuracy in any of the models. ODIN [13] is excluded from this comparison, as it requires back-propagation, which cannot be performed efficiently on the FPGA and would also result in 2× latency for the second inference of the perturbed input.

Figure 6a to Figure 6c show the results of this comparison. In all cases, entropy maximization [1] does not function properly anymore with an FPR95 above 90%. We find that the effect of fixed-point numbers on existing methods is minimal in most other cases, however also breaks MSP [6] for MNIST. Also for complex CNN tasks, detection capabilities of existing methods slightly deteriorate.

5.2 Results of our Side-Channel Based COD

Initially, we analyze power traces of the MLP-64 trained on MNIST to evaluate our method on a simple dataset and model. Figure 7 displays, how the power traces are significantly different, supporting our assumption that the transition from ID samples to OOD samples can be detected by the power consumption. Here, it is sufficient to determine the threshold by the minimum of the trace, as the first layer creates the most notable voltage drop. For the GTSRB model, we select the sum as the score function with a single threshold. Finally, for the CIFAR-10 model, we use 9 thresholds for the detection and can still get the OOD detection result at around 3/4 of the inference.



(a) MLP trained on MNIST

(b) CNN trained on CIFAR-10

(c) CNN trained on GTSRB

Figure 6: Comparison of OOD Detection methods with floating point and fixed point on a binary MLP trained on MNIST.

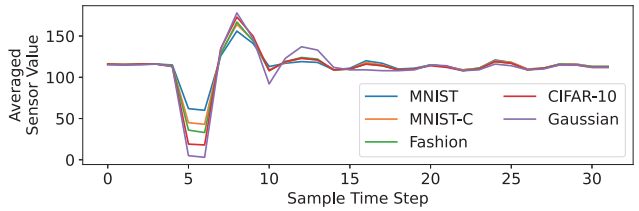


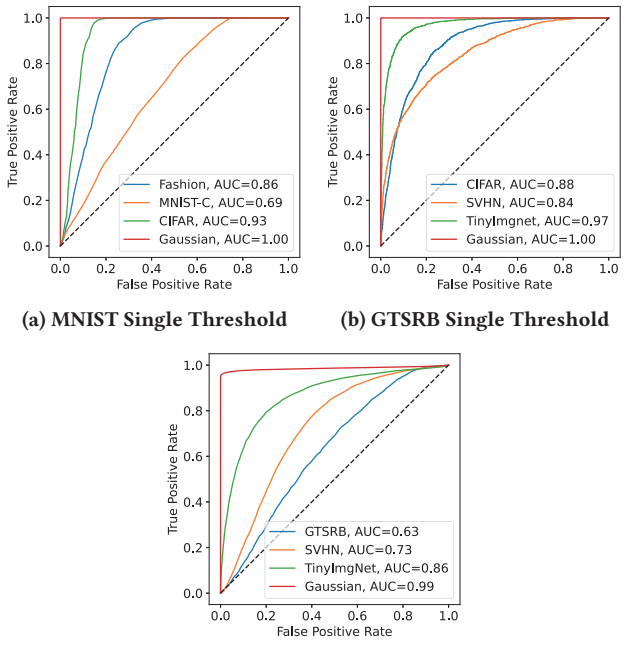
Figure 7: Averaged power traces for each dataset for an MLP trained on MNIST.

Figure 8 shows the ROC for our power side-channel-based OOD detection for the MNIST (a), GTSRB (b) and CIFAR-10 (c) model. We notice that gaussian noise seems to have distinguishable features from the ID sets, independent of model or training dataset. Noise can be perfectly detected in almost any case with our method. As shown earlier in Figure 6, this is not possible using methods from related work when considering QNN.

Table 1, Table 2 and Table 3 summarize all of our experiments and show a comparison to related work. Our method beats related work in FPR-95 in almost every case. This method outperforms related work for any of the chosen OOD datasets, noteworthy also MNIST-C, and is also computationally efficient. It is easily implemented on an FPGA as it requires no complex calculations.

5.3 Result Summary and Discussion

For all the evaluated datasets, we find that the hardest task is detection of samples from the MNIST-C set. These samples are closest to



(a) MNIST Single Threshold

(b) GTSRB Single Threshold

(c) CIFAR-10 Multiple Threshold

Figure 8: ROC of the OOD detection for different datasets and models

actual MNIST samples, making the detection a more challenging task. Still, we can achieve an FPR-95 of 0.67, an improvement of around 25% from the best result of related work. Yang et al. [26] describe these as near OOD sets, which only have semantic shift in comparison with the ID dataset. We notice that the detection mechanism performs worse on the VGG trained on CIFAR-10, which is likely related to the lower accuracy of the model, which in turn results in a less pronounced power profile for ID samples.

We find that the evaluated existing methods [1, 5, 6, 13, 14] in almost all cases show worse performance compared to our method, even though the AUC is in some cases higher. Furthermore, our proposed detection method can be efficiently implemented in hardware and can be performed concurrently to the detection, as it is sufficient to analyze a part of the trace. Thus, we conclude that our method beats the evaluated existing methods in performance. The suboptimal performance of current methods in OOD detection could potentially be attributed to their lack of adaptation to quantized neural network (QNN), which have a reduced parameter space. The difference in parameters of QNNs might reduce the distinctiveness of the logits and softmax values, which in turn lowers the effectiveness of existing methods. We show that our method still works effectively for the worst case of single bit activations and weights, which have a less pronounced power profile than less quantized networks, due to lower switching activity.

Furthermore, our approach works out of the box and does not rely on any other data than the power measurement and the explicit output label of the accelerator. By solely relying on power measurements, we reduce the amount of data-transfers between ARM-core and FPGA required to detect OOD samples.

Table 4 evaluates the efficiency of the proposed COD technique in terms of hardware costs compared to the overall system. As

Table 1: False positive rate at 95% TPR (FPR-95) and area under curve (AUC) of the OOD-Detection with fixed-point / float32 for MLP-64 (MNIST). ↓ means lower values are better and ↑ means high values are better.

Method (ID=MNIST)	FashionMNIST		MNIST-C		CIFAR-10 (Grayscale)		Gaussian Noise	
	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑
Power Side-Channel (ours)	**0.31**	**0.86**	**0.67**	**0.69**	**0.13**	**0.93**	**0.0**	**1.00**
MaxLogit [5]	0.63 / 0.63	0.85 / 0.85	0.89 / 0.89	0.72 / 0.72	0.56 / 0.56	0.85 / 0.85	0.43 / 0.43	0.88 / 0.88
MSP [6]	0.99 / 0.54	0.18 / 0.85	0.99 / 0.86	0.31 / 0.72	0.99 / 0.54	0.18 / 0.85	0.99 / 0.40	0.15 / 0.88
Entropy Maximization [1]	0.98 / 0.55	0.23 / 0.85	0.98 / 0.86	0.43 / 0.72	0.98 / 0.55	0.23 / 0.85	0.98 / 0.38	0.20 / 0.88
Energy-based [14]	0.90 / 0.90	0.58 / 0.59	0.91 / 0.91	0.55 / 0.55	0.88 / 0.89	0.60 / 0.60	0.87 / 0.87	0.61 / 0.62
ODIN [13]	0.56	0.85	0.86	0.72	0.56	0.85	0.38	0.88

Table 2: False positive rate at 95% TPR (FPR-95) and area under curve (AUC) of the OOD-Detection with fixed-point / float32 for a CNN (CIFAR-10). ↓ means lower values are better and ↑ means high values are better.

Method (ID=CIFAR-10)	TinyImgNet		SVHN		GTSRB		Gaussian Noise	
	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑
Power Side-Channel (ours)	**0.50**	**0.86**	**0.68**	**0.73**	**0.78**	**0.62**	**0.00**	**0.99**
MaxLogit [5]	0.73 / 0.73	0.76 / 0.76	0.61 / 0.59	0.79 / 0.79	0.51 / 0.49	0.82 / 0.82	0.51 / 0.51	0.81 / 0.81
MSP [6]	0.72 / 0.72	0.75 / 0.75	0.58 / 0.58	0.79 / 0.78	0.50 / 0.50	0.82 / 0.81	0.51 / 0.50	0.81 / 0.81
Entropy Maximization [1]	0.99 / 0.72	0.25 / 0.75	0.99 / 0.58	0.19 / 0.78	0.99 / 0.50	0.22 / 0.81	0.99 / 0.50	0.19 / 0.80
Energy-based [14]	0.98 / 1.00	0.28 / 0.50	0.98 / 1.00	0.25 / 0.50	0.98 / 1.00	0.22 / 0.50	0.98 / 1.00	0.22 / 0.50
ODIN [13]	0.71	0.76	0.64	0.77	0.48	0.82	0.55	0.80

Table 3: False positive rate at 95% TPR (FPR-95) and area under curve (AUC) of the OOD-Detection with fixed-point / float32 for a CNN (GTSRB). ↓ means lower values are better and ↑ means high values are better.

Method (ID=GTSRB)	TinyImgNet		SVHN		CIFAR-10		Gaussian Noise	
	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑	FPR-95 ↓	AUC ↑
Power Side-Channel (ours)	**0.13**	**0.97**	**0.59**	**0.84**	**0.38**	**0.88**	**0.0**	**1.00**
MaxLogit [5]	0.71 / 0.71	0.91 / 0.91	0.89 / 0.89	0.87 / 0.87	0.71 / 0.71	0.92 / 0.92	0.71 / 0.71	0.91 / 0.91
MSP [6]	0.54 / 0.52	0.91 / 0.92	0.76 / 0.67	0.88 / 0.89	0.44 / 0.44	0.92 / 0.93	0.56 / 0.62	0.91 / 0.91
Entropy Maximization [1]	1.00 / 0.92	0.08 / 0.52	1.00 / 0.89	0.11 / 0.60	1.00 / 0.93	0.07 / 0.45	1.00 / 0.91	0.09 / 0.60
Energy-based [14]	1.00 / 1.00	0.16 / 0.81	1.00 / 1.00	0.19 / 0.77	1.00 / 1.00	0.15 / 0.82	1.00 / 1.00	0.19 / 0.81
ODIN [13]	0.52	0.92	0.67	0.89	0.43	0.93	0.62	0.91

Table 4: Hardware utilization and power of FINN, RDS, and the proposed OOD detection mechanism

Module	LUT	Registers	Power(W)
FINN	97508 (42.32%)	123001 (26.69%)	0.929W
RDS	2136 (0.93%)	1404 (0.30%)	0.275W
OOD Detection	60 (0.03%)	37 (< 0.01%)	0.014W

shown in Table 4, the needed FPGA resource requirement for OOD detection is less than 1% and 0.3% of the LUTs and registers, respectively. This amounts to only 0.2% of the hardware consumption compared to the FINN accelerator. Furthermore, the power consumption for OOD detection is less than 24% of the total power consumption of the entire system. It should be noted that 20% of this power consumption is due to the higher frequency of the sensors compared to the rest of the system. As mentioned before, our approach is zero-latency as it only requires the configured amount of clock cycles and can finish anytime before the inferences finishes.

6 Conclusion

We propose an innovative approach to efficiently and effectively detect OOD samples in NN FPGA accelerators, using power side-channel measurements and one or multiple thresholds learned from the ID data. Unlike the other existing methods, our method is not only suitable for quantized NNs as commonly used for edge hardware deployment, but can also be efficiently implemented within the device and detect OOD samples before inference finishes. The proposed approach can be extended to different domains and datasets and enhances the robustness and reliability of edge accelerators in safety-critical domains, ensuring their safe and secure operation even in the presence of OOD inputs. In the future, we are looking into expanding our experiments and method for other types of NNs, such as transformer models, and tasks beyond image classification.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG), grant 501300923 (SecureNN).

References

[1] Robin Chan, Matthias Rottmann, and Hanno Gottschalk. 2021. Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation. In *International Conference on Computer Vision*. IEEE.

[2] Terrance DeVries and Graham W Taylor. 2018. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865* (2018).

[3] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*. PMLR.

[4] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. 2019. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Conference on Computer Vision and Pattern Recognition*. IEEE.

[5] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joe Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. 2019. Scaling out-of-distribution detection for real-world settings. *arXiv preprint arXiv:1911.11132* (2019).

[6] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[7] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. 2013. Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*.

[8] Konstantin Kirchheim, Marco Filax, and Frank Ortmeier. 2022. PyTorch-OOD: A Library for Out-of-Distribution Detection Based on PyTorch. In *International Conference on Computer Vision and Pattern Recognition*. IEEE.

[9] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *CRYPTO*. Springer.

[10] Ioannis Kouretas and Vassilis Paliouras. 2020. Hardware implementation of a softmax-like function for deep learning. *Technologies* 8, 3 (2020), 46.

[11] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[13] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690* (2017).

[14] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems* (2020).

[15] Microsoft. 2022. Deploy ML models to field-programmable gate arrays (FPGAs) with Azure Machine Learning. https://learn.microsoft.com/en-us/azure/machine-learning/v1/how-to-deploy-fpga-web-service

[16] Mohammed Ali mnmoustafa. 2017. Tiny ImageNet. https://kaggle.com/competitions/tiny-imagenet

[17] Norman Mu and Justin Gilmer. 2019. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337* (2019).

[18] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, Vol. 2011. Granada, 4.

[19] Alessandro Pappalardo. 2023. *Xilinx/brevitas*. https://doi.org/10.5281/zenodo.3333552

[20] Roshni Shende and Dayanand D Ambawade. 2016. A side channel based power analysis technique for hardware trojan detection using statistical learning approach. In *International Conference on Wireless and Optical Communications Networks (WOCN)*. IEEE.

[21] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[22] David Spielmann, Ognjen Glamočanin, and Mirjana Stojilović. 2023. RDS: FPGA Routing Delay Sensors for Effective Remote Power Analysis Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2023).

[23] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. 2016. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. (12 2016). https://doi.org/10.1145/3020078.3021744

[24] Xiao Wang, Quan Zhou, Jacob Harer, Gavin Brown, Shangran Qiu, Zhi Dou, John Wang, Alan Hinton, Carlos Aguayo Gonzalez, and Peter Chin. 2018. Deep learning-based classification and anomaly detection of side-channel signals. In *Cyber Sensing*. SPIE.

[25] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv:cs.LG/1708.07747 [cs.LG]

[26] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, et al. 2022. Openood: Benchmarking generalized out-of-distribution detection. *Advances in Neural Information Processing Systems* (2022).

[27] Maohua Zhu, Liu Liu, Chao Wang, and Yuan Xie. 2016. Cnnlab: a novel parallel framework for neural networks using gpu and fpga-a practical study with trade-off analysis. *arXiv preprint arXiv:1606.06234* (2016).