MENDNet: Just-in-time Fault Detection and Mitigation in AI Systems with Uncertainty Quantification and Multi-Exit Networks

Shamik Kundu*, Mirazul Haque*, Sanjay Das, Wei Yang and Kanad Basu
The University of Texas at Dallas, *Richardson, TX, USA*

Abstract

Hardware faults in AI accelerators, particularly in accelerator memory, can alter pre-trained deep neural network parameters, leading to errors that compromise performance. To address this, just-in-time (JIT) fault detection and mitigation are crucial. However, existing fault detection/mitigation approaches, either interrupt continuous execution or introduce significant latency, making them less ideal for JIT implementation. To circumvent this issue, this paper explores uncertainty quantification in deep neural networks as a means of facilitating an efficient and novel fault detection approach in AI accelerators. Furthermore, in order to mitigate the impact of such faults, we propose MENDNet, which leverages the properties of multi-exit neural networks, coupled with the proposed uncertainty quantification framework. By tuning the confidence threshold for inference in each exit and leveraging the energy-based uncertainty quantification metric, MENDNet can make accurate predictions even in the presence of faults in the accelerator. When evaluated on state-of-the-art network-dataset configurations and with multiple fault rate-fault position combinations, our proposed approach furnishes up to 80.42% improvement in accuracy over a traditional DNN implementation, thereby instilling the reliability of the AI accelerator in mission mode.

Keywords: Deep Neural Network Accelerator, Fault Tolerance, Uncertainty Quantification, Multi-exit Networks.

1 Introduction

The popularity of Deep Neural Networks (DNN) has grown, particularly for IoT edge devices. Specialized energy-efficient hardware, such as AI accelerators, have been developed for DNN applications in mission-critical scenarios, *e.g.,* smart healthcare, autonomous vehicles, and military operations. Malfunctions in the hardware caused by manufacturing variation, voltage noise, and temperature fluctuations introduce *faults* in the AI accelerators during inference at the edge [3]. Hence, it is important to develop just-in-time (JIT) techniques to detect and mitigate these faults. A proper JIT strategy needs to address two concerns to ensure the safety and reliability of the AI hardware: (1) The execution of the system ideally should not be halted/hindered.(2) The latency overhead for the detection and mitigation schemes should be negligible.

For detection, while existing research to detect faults in AI accelerators primarily focuses on functional testing strategies; these techniques can not incorporate JIT constraints. In these approaches, the functional test patterns are executed periodically, in tandem with the target application to detect run-time fault manifestation in the accelerator [9]. However, these approaches require additional ground truth of the test inputs, and they are unable to identify faults that manifest after the execution of the patterns have been completed, thus, failing to comply being JIT. Other methods to detect faults in non-AI systems like Built-in-self-test (BIST) and error correction codes (ECCs) furnish even higher overhead issues, when compared to the aforementioned functional testing strategy [1]. Furthermore, methods like ATPG, typically used for manufacturing testing, cannot seamlessly detect faults during mission mode, since they require shift from functional to scan clocks for operation. Hence, it is imperative to develop a JIT fault detection technique, catering to mission-critical AI systems.

Existing research to address fault mitigation in AI accelerators based on the re-mapping and retraining the application DNN prevents continuous execution and furnishes significant overhead in terms of latency [14]. Moreover, it is important to note that the process necessitates access to training data, which may not always be practically feasible or readily available. Another common approach is redundancy-based fault tolerance, where multiple copies of critical units are used to perform redundant computations [6]. However, redundancy comes at the cost of increased area, power consumption, and complexity; hence, the technique might not be feasible for low overhead edge applications.

In this paper, we introduce an innovative and efficient data-free framework for JIT fault detection in AI accelerators at the edge. Our method, designed for mission-critical environments, eliminates the need for test patterns during routine AI hardware tasks. Central to our approach is an energy-based uncertainty quantification metric, measuring AI system deviations without relying on groundtruth. We demonstrate the metric's adaptability in detecting manifested faults, where in-distribution samples in faulty hardware are misclassified as out-of-distribution. Importantly, the energy-based metric incurs no additional training cost, making it well-suited for practical deployment in accelerators at the edge.

Furthermore, in order to mitigate the impact of such faults, we propose MENDNet, that enables on-line fault mitigation, entirely eliminating the need for expensive traditional fault detection, diagnosis, mapping, retraining and redundancy-based schemes. The proposed MENDNet framework facilitates a proactive approach, that ensures uninterrupted operation and reduces downtime in AI hardware in mission mode. MENDNet leverages the properties of multi-exit networks to address faults in AI accelerators. In systems with limited ECC capabilities that incorporate AI accelerators, MENDNet acts as a secondary defense against in-field faults at the edge. The key contributions of this paper are as follows:

- We, propose a JIT fault detection technique in AI accelerators at the edge in mission critical environments, by adapting energy-based uncertainty quantification in DNNs.

*Both authors contributed equally to this research.
Corresponding Author: Shamik Kundu (Email: *shamik.kundu@utdallas.edu*)
This research is supported by Semiconductor Research Corporation (SRC GRC Task: 3243.001) and National Science Foundation (NSF CNS 2235137 & CCF 2146443).

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3656506

- We propose MENDNet, a JIT framework based on multi-exit neural networks, to mitigate the impact of faults on inference classification accuracy. Utilizing multiple exits in MENDNet, we decide the final prediction based on confidence thresholds and uncertainty quantification, thereby circumventing the need for costly remapping, retraining or redundancy approaches.
- We evaluate MENDNet on three metrics: effectiveness, efficiency and sensitivity. When evaluated on three popular DNN models and two popular datasets, our proposed MENDNet framework furnishes up to 80.42% improvement in classification accuracy, over a conventional DNN implementation, with and without multi-exit, in the AI accelerator.

2 Background

Uncertainty Quantification. Uncertainty quantification in deep neural networks (DNNs) is a burgeoning research area focused on estimating and characterizing the uncertainty associated with model predictions. This plays a critical role in Out-of-Distribution (OOD) detection, where the aim is to identify samples differing from the trained data distribution (*e.g.*, unexpected inputs, corrupted input data, or adversarial examples). Bayesian Neural Networks [4], Monte Carlo Dropout [8], and an energy-based method [11] have been proposed for OOD detection.

The energy score, introduced by the energy-based method, gauges how well a given DNN handles data and serves as a widely adopted metric for discriminating between observed and unobserved inputs [11]. Generally, the energy score tends to be higher for unobserved data and lower for observed data. We propose employing the energy score for uncertainty quantification in DNNs due to its minimal computational overhead compared to techniques like Monte Carlo Dropout or ensemble methods [12], crucial for JIT fault detection and mitigation. Importantly, calculating energy score requires no additional retraining, as it is based on an Energy-based Model (EBM) mapping each input point x to a non-probabilistic scalar energy score. Let's assume f is a DNN, and E represents the energy score given f and specific input x. Then, E can be defined as follows,

$$E(x; f) = -T \cdot log \sum_{i=1}^{K} e^{f_i(x)/T}$$

Here, K represents the number of labels, f_i represents the logit score of the i^{th} label (logit score is the final intermediate value of the DNN calculated before the softmax layer), and T is called the temperature parameter. For experiments, we put the value of T as 1, following existing research [11].

Multi-exit Neural Networks. Multi-exit neural networks [7] are a class of deep learning architectures that have gained attention for their ability to efficiently process inputs with varying levels of complexity. Multi-exit networks incorporate multiple paths or exits within the network, allowing for early termination of the computation based on the input's complexity.

Next, we discuss the working mechanism of traditional multi-exit networks. Let's assume, $\mathcal{F}$ is a multi-exit network with N exits. Each exit has a classifier that can provide a prediction given an input x. For each ulti-exit network, a confidence score threshold γ is predetermined. For any exit i, if $max(\mathcal{F}^i(x)) > \gamma$, then the inference is stopped and final prediction is provided. In this paper, we leverage multi-exit networks to improve fault resilience in AI accelerators. The redundant nature of such networks, with multiple exits producing predictions, offers built-in fault tolerance. In the presence of faults, multi-exit networks can rely on alternative exits that have not been affected, ensuring reliable predictions.

3 Fault Detection by Uncertainty Quantification

3.1 Intuition for Fault Detection

Our approach to fault detection relies on identifying changes in weight values induced by permanent hardware faults in deep neural networks (DNNs). We consider faults as static noise (δ) modifying weight values (w) for the inference of all inputs. Since DNN output is determined by arithmetic operations involving weights and inputs, faults impact the hidden states of the DNN. Our goal is to identify a technique capable of detecting these alterations in hidden states caused by such faults.

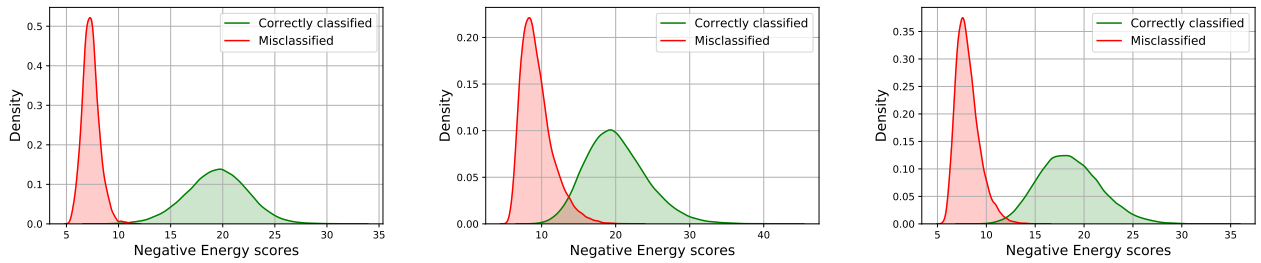
Changes in DNN hidden states can also occur with the introduction of static noise to the input, where natural corruptions are regarded as static noise (δ') added to input x [5]. Epistemic uncertainty quantification techniques, commonly used for detecting naturally corrupted inputs, characterize uncertainty as low for observed inputs (belonging to the training data distribution) and high for unobserved inputs. Natural corruption introduces variability to inputs, disrupting their distribution. In turn, the process of quantifying epistemic uncertainty can pinpoint these altered inputs by identifying shifts in hidden state outputs. Since faults in weights induce similar alterations in DNN hidden states, our intuition is that epistemic uncertainty quantification techniques can effectively detect changes in hidden state outputs caused by faults.

3.2 Study Setup

Fault Model: In this paper, we focus on permanent faults in accelerator memory (which stores the model weights), particularly relevant to DNNs, arising from factors like manufacturing variation, voltage noise, and temperature fluctuations causing unpredictable circuit delays [9]. Our fault injection framework introduces permanent bit-flips in the Most Significant Bit (MSB) of randomly selected weights distributed across the memory to illustrate a worst-case scenario, aligning with existing research [9, 10]. We consider three fault rates (FR) — 10%, 30%, and 50% — to showcase varying scenarios. FR represents the percentage of erroneous weight parameters among all parameters in a given DNN layer. It's important to note that higher fault rates significantly degrade accuracy, posing reliability challenges for AI accelerators. Our proposed approach aims to mitigate this accuracy drop, regardless of the specific fault rate, addressing reliability issues in AI accelerators.
Networks: For the experiments, we use three popular DNN models VGG16, Wideresnet32, and MobileNet.
Datasets: For the datasets we choose two image datasets CIFAR-10 and CIFAR-100, which consist of 32×32 RGB images of 10 and 100 classes respectively.
Methodology: We propose identifying patterns in uncertainty scores when a model exhibits faults. Our investigation centers on determining a threshold in uncertainty, specifically the negative energy score, to detect misclassified inputs resulting from faults. To achieve this, we compare the negative energy scores of correctly classified inputs on non-faulty models with those of incorrectly classified inputs on the faulty model. If these sets of values exhibit distinct ranges, we can establish a negative energy score threshold for identifying misclassified inputs due to faults.

3.3 Summary of Findings

Figure 1 depicts our research findings on the negative energy score threshold for misclassified inputs due to faults, specifically at a



(a) Model: VGG16, 30% FR **(b)** Model: WideResNet, 30% FR **(c)** Model: MobileNet, 30% FR

Figure 1. Density plots of negative energy scores of correctly classified inputs in the non-faulty model (green) and mis-classified inputs (red) from faulty model on training dataset.

fault rate of 30%, with an end classifier fault position and using CIFAR-100 data. The green curve represents scores from correctly classified inputs using the non-faulty model, while the red curve corresponds to scores from incorrectly classified inputs using the faulty model. Notably, the green curve consistently shows a higher negative energy score range than the red curve, suggesting the potential for identifying misclassified inputs due to faults. Similar outcomes have been observed for fault rates 10% and 50%, as well as for other network-dataset configurations in different fault positions. However, **due to brevity of space, we could not include them in the paper. These results can be found at our website**[1].

4 Proposed MENDNet Framework

In this work, we present MENDNet, a novel approach that employs an energy score-driven uncertainty estimation metric in multi-exit networks to mitigate the errors caused by hardware faults. As described in Section 3, the negative energy score effectively detects misclassified inputs in faulty models. Leveraging this insight, MENDNet incorporates a novel exit strategy for multi-exit networks. Subsequently, any multi-exit network employing the MENDNet strategy is referred to as a **MENDNet model** throughout the paper.

4.1 Problem Formulation

The primary aim of our study is to develop a robust framework utilizing the structure of multi-exit networks to effectively mitigate the faults. Multi-exit networks exhibit a redundant nature characterized by the presence of multiple exits that generate predictions. We leverage this inherent redundancy to build a fault tolerance mechanism. Based on our study, illustrated in Section 3, we propose the incorporation of uncertainty quantification mechanisms within each exit of the multi-exit network.

Let's assume $\mathcal{F}$ is a multi-exit network that has N exits and K classes. Generally, for a given exit i and input x, if $max(\mathcal{F}^i(x)) > \gamma$, the network takes the exit (γ is the predefined confidence score threshold). We propose to involve an additional uncertainty score threshold for each exit of the classification task. This exit strategy would help to ascertain if the network is faulty or not. As a result, we can discern whether the inference predictions can be deemed reliable from the AI hardware.

In accordance with the aforementioned concept, we engage in a comprehensive analysis of four distinct fault positions within multi-exit networks, as depicted in Figure 2. We then proceed to explore the potential of our newly devised exit strategy in mitigating these

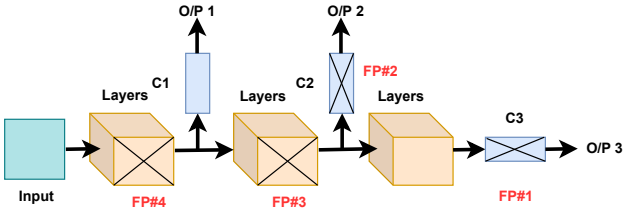


Figure 2. Different fault positions in a multi-exit network. C represents different classifiers, where the cross sign signifies different fault positions.

faults. Subsequently, we delve into the details of our algorithm, which encapsulates the findings and insights from this discussion.
Fault Positions: The first fault position, denoted as fault position #1 (FP#1), is positioned at the final classifier exit. In a traditional DNN, such a fault would significantly impair the model's accuracy. Fault position #2 (FP#2) is located at an internal classifier exit, and notably, it does not affect the preceding or subsequent classifiers. Fault position #3 (FP#3) is situated at an intermediate convolutional layer, and in a standard DNN, its impact could propagate to the final output, potentially compromising accuracy. Lastly, fault position #4 (FP#4) is at the first convolutional layer, resulting in an impact on every exit within the network.

4.2 Proposed Algorithm

Based on the previously mentioned four case scenarios, we develop our mitigation approach. *The goal of the mitigation approach is to alleviate the faults in FP#1, FP#2 and FP#3, while localizing the fault while the fault position is FP#4.* The mitigation algorithm takes following inputs: a multi-exit neural network $\mathcal{F}$, a pre-defined confidence threshold γ, a pre-defined negative energy threshold τ, an input x; *while producing two outputs: output prediction label and a boolean value to inform if the fault position is before the first exit.*

Algorithm 1 delineates our proposed MENDNet framework for exiting multi-exit network. For each input x, the network $\mathcal{F}$ would calculate output on various exits. If, for a specific exit, confidence score is higher than threshold γ and negative energy score is higher than threshold τ, the network takes the exit *(Lines 4-10)*. If the negative energy score is higher than threshold τ, it ensures that the decision is not impacted by the fault. If the confidence score for an exit does not cross the threshold γ, but negative energy score is higher than threshold τ, the output label is added to a predefined list(or array) *(Lines 11-12)*. This list would be helpful specifically to mitigate faults at FP#1 and FP#3. If the negative energy score is less

[1] https://sites.google.com/view/mendnet/home

Algorithm 1 Proposed MENDNet Framework

Require:
A multi-exit neural network $\mathcal{F}$;
A pre-defined confidence threshold γ;
A pre-defined negative energy threshold τ;
An input x;

```
1:  list=emptyList()  // Used for storing different exit predictions
2:  lowest_unc_label=-1   // To store prediction with lowest unc
3:  min_unc=-inf    // To store highest −ve energy or lowest unc
4:  for i=1.....N do
5:    if −Eng(F^i(x)) > min_unc then
6:      min_unc = −Eng(F^i(x))
7:      lowest_unc_label = argmax(F^i(x))
8:    end if
9:    if max(F^i(x)) > γ and − Eng(F^i(x)) > τ then
10:     return argmax(F^i(x)), False                    // Exiting
11:   else if −Eng(F^i(x)) > τ then
12:     list.add(argmax(F^i(x)))    // Prediction added to the list
        (For FP#1 and FP#3)
13:   else if −Eng(F^i(x)) <= τ then
14:     if list.isempty() or i<=2 then
15:       continue
16:     else
17:       return list.pop(), False
18:     end if
19:   end if
20:   if i==N then
21:     if list.isempty() then
22:       return lowest_unc_label, True    // Fault location FP#4
23:     else
24:       return list.pop(), False        // latest pred with low unc
25:     end if
26:   end if
27: end for
```

than threshold τ and if the exit is after the second exit, the output of the final classifier is returned *(Lines 13-18)*.

If the input reaches the final exit *(Line 20)*, and both confidence and negative energy scores haven't surpassed their thresholds, the list length is examined. If the list is empty *(Line 21)*, indicating no exit has a negative energy score beyond the threshold, the algorithm identifies the fault to be located before the first exit. In this case, the output prediction with the lowest uncertainty is returned, accompanied by a $True$ fault boolean value, indicating the detection of the fault. Otherwise, the last list elements is returned *(Line 24)*, representing the prediction from the latest exit with a high negative energy score (low uncertainty).

5 Evaluation

Through the evaluation, we focus on three research questions about MENDNet approach: **Effectiveness**, **Efficiency**, and **Sensitivity**.

Table 1. Different τ and γ selected for all models for CIFAR-10 and CIFAR-100 datasets

	CIFAR-10			CIFAR-100		
Threshold	VGG-16	WideRN	MobileNet	VGG-16	WideRN	MobileNet
γ	0.5	0.7	0.6	0.5	0.6	0.7
τ	8	10	15	15	10	8

RQ1 (Effectiveness) : What is the effectiveness of the MENDNet w.r.t mitigating faults in hardware? In this RQ, we discuss the fault mitigation effectiveness through measuring the accuracy of the model against different fault rates and fault positions.
RQ2 (Efficiency) : What is the efficiency of MENDNet compared to the baseline? In this RQ, we discuss the latency of MENDNet model against baseline approaches during inference.
RQ3 (Sensitivity) : What is the effect of MENDNet conditional threshold values on the model-accuracy? Here, we measure the MENDNet model accuracy on different values of τ and γ.

5.1 Experimental Setup

Networks and Datasets: We use Shallow-deep Network (SDN) architecture, a popular multi-exit network. As the backbone of SDN [7], we use the same DNN-dataset configurations, mentioned in Section 3. Due to space constraints, we could not include all the results in the paper. However, they are available on our website[1].
Baseline: As a baseline we use the following two: (1) the traditional DNN mechanism, where the model has a single exit after all layers, (2) the SDN mechanism [7] (using only confidence score) as the second baseline mechanism. *Both SDN and MENDNet models use the same model architecture, but the exit strategies of the models are different.* The existing strategies that require remapping, retraining schemes are not considered as baseline, since they do not facilitate JIT mitigation, and require halting the application to address the manifested fault. Instead, we compared our proposed technique with two state-of-the-art approaches, that do not require remapping or retraining, yet facilitates online fault management in AI accelerators to boost the classification accuracy [2, 13]. However, they do not offer JIT mitigation, as demonstrated in our MENDNet framework. The method proposed in [13] employs median feature selection to mitigate bit errors before each layer's execution, whereas ERDNN introduces an Error Correction Layer (ECL) in each convolution layer and a Piece-wise Rectified Linear Unit (PwReLU) to selectively suppress errors [2]. We compare MENDNet with these approaches to highlight its efficacy in enhancing AI hardware robustness.
Metric: We compare the classification accuracy of models on held-out test sets to evaluate the effectiveness of MENDNet against faults. For efficiency evaluation, we measure the inference latency.
Threshold Values. We select the threshold values τ and γ by evaluating MENDNet models' accuracy after faults and we use the thresholds for a given model-dataset pair based on the best average accuracy generated by using specific thresholds, as demonstrated in Table 1. For SDNs, we only use the best accuracy threshold γ.

5.2 RQ1 (Effectiveness)

Table 2 presents the evaluation results, demonstrating that, on average, MENDNet outperforms DNN and SDN across all FP and FR for each model-dataset pair. Except for FP#4, the accuracy of MENDNet models remains consistently high across different FPs and FRs. Notably, the average accuracy of the SDN model improves for the CIFAR-100 dataset. However, for the CIFAR-10 dataset, the average accuracy of the SDN model is notably lower compared to that of the MENDNet model. Although the accuracy of the MENDNet model is affected by faults for FP#4, the fault position can be localized based on the uncertainty scores across different exits.

Additionally, we observe that utilizing MENDNet with faults can result in higher accuracy compared to non-faulty DNNs. For the VGG16-CIFAR-100, WideRN-CIFAR-10, MobileNet-CIFAR-10, and MobileNet-CIFAR-100 pairs, MENDNet achieves higher accuracies

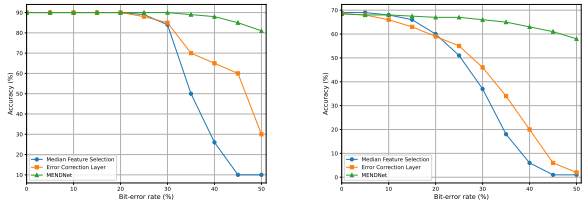
Table 2. Accuracy(%) of DNN, SDN and MENDNet for three model-backbones and two datasets through various FPs and FRs

Models	Dataset	Techniques	FP#1			FP#2			FP#3			FP#4			Avg
			FR:10%	FR:30%	FR:50%	FR:10%	FR:30%	FR:50%	FR:10%	FR:30%	FR:50%	FR:10%	FR:30%	FR:50%	
VGG16	cifar-10	DNN	92.6	92.5	22.5	92.6	92.6	92.6	92.6	90.3	11.5	85	40.4	11.5	68.05
		SDN	85.43	85.42	85.42	85.41	85.42	85.43	85.43	85.4	85.3	72.68	31.79	17.41	74.2
		MENDNet	92.3	92.26	92.26	92.3	92.25	92.28	92.26	92.04	91.99	89.25	36.17	17.27	**81.13**
	cifar-100	DNN	69.92	65.83	3.36	70.47	70.47	70.47	66.86	43.89	11.51	56.34	10.04	7.11	45.52
		SDN	70.08	70.09	70.08	69.8	69.74	69.92	69.5	67.6	66.9	56.89	12.02	6.68	58.27
		MENDNet	72.1	72.13	72.13	71.92	71.8	71.83	71.46	70.28	70.14	51.03	9.79	6.24	**59.23**
WideRN	cifar-10	DNN	91.35	75.62	38.25	93.95	93.95	93.95	93.91	93.36	92.72	80.57	71.01	37.72	79.69
		SDN	84.1	84.07	84.07	83.97	83.73	83.06	83.95	83.62	83.13	74.77	30.2	15.68	72.86
		MENDNet	93.97	93.97	93.96	93.8	94.1	94.14	94.04	95.6	90.43	70.17	52.74	13.43	**81.67**
	cifar-100	DNN	65.97	38.83	14.91	75.68	75.68	75.68	75.37	74.33	72.77	56.74	3.24	6.9	53.01
		SDN	69.9	69.75	69.68	68.87	68.25	68.93	69.75	68.75	68.25	54.69	16.88	6.56	58.35
		MENDNet	74.34	74.28	74.28	74.21	74.97	74.99	74.22	72.52	69.48	52.34	6.35	6.56	**60.71**
MobileNet	cifar-10	DNN	90.57	89.67	15.63	90.75	90.75	90.75	89.44	75.08	30.08	86.41	60.55	27.06	69.72
		SDN	86.27	86.26	86.26	86.19	86.12	86.12	86.22	85.75	85.04	70.67	23.71	11.08	73.3
		MENDNet	91.11	91.11	91.11	91.16	91.15	91.15	90.37	87.94	85.13	84.87	40.81	31.31	**80.60**
	cifar-100	DNN	61.53	41.92	4.59	65.1	65.1	65.1	63.35	43.51	8.47	44.44	18.04	13.5	41.22
		SDN	68.4	68.32	68.46	68.25	67.6	68.05	67.54	62.36	57.41	34.32	28.38	11.62	55.89
		MENDNet	68.33	68.43	68.43	67.77	67.48	67.81	66.82	62.86	59.65	51.93	15.67	10.86	**56.33**

than non-faulty DNNs. There are two possible reasons for this phenomenon: 1) By increasing negative energy score threshold during the exit decision-making process, MENDNet ensures that the model produces decisions with high certainty, thus reducing mis-classifications. 2) Early exits in MENDNet model can correctly predict labels for a number of inputs, while the final exit fails to do so. These findings provide strong support for the usage of the MENDNet model over traditional DNNs.

> **Finding 1**: *With respect to average accuracy, MENDNet outperforms baselines for all model-dataset pairs across all fault positions and fault rates.*
> **Finding 2**: *For all model-dataset pairs, MENDNet with fault can improve the accuracy of non-faulty DNNs.*

Comparison with existing approaches. In Figure 3, we showcase MENDNet's superior robustness against fault manifestations using the VGG16 model trained on the CIFAR-10 and CIFAR-100 dataset. The results reveal a significant accuracy decline with increasing fault rates for median feature selection and ERDNN, dropping to 10% and 30%, respectively, at 50% fault rate on CIFAR-10. In contrast, MENDNet maintains accuracy, only dropping to 81% under the same conditions, demonstrating its superiority.



(a) (b)

Figure 3. Comparing the efficiency of our proposed framework with existing approaches: VGG16 model trained and evaluated on (a) CIFAR10 and (b) CIFAR100.

5.3 RQ2 (Efficiency)

We evaluate the efficiency of the MENDNet model against baselines by feeding held-out test data to the faulty models with the combination of different fault rates (FR) and fault positions (FP) and measure the latency of the models. To measure efficiency, we feed each input thrice to the models and calculate the median latency to remove unwanted noise in the measurement.

Figure 4 shows efficiency evaluation results for VGG16 model trained with CIFAR-10 data. Results for other model-dataset combinations can be found on our website[1]. From the results, we observe that MENDNet is more efficient than DNN across different FP-FR combinations. SDN shows higher efficiency than MENDNet as SDN exits if the confidence threshold is exceeded; however, MENDNet also considers the uncertainty threshold. Therefore, to ensure high accuracy during faults, MENDNet incurs a minimal increase in latency compared to SDNs.

> **Finding 3**: *MENDNet's efficiency is higher than DNN, but SDN is more efficient than both MENDNet and DNN.*

5.4 RQ3 (Sensitivity)

We aim to assess the impact of varying thresholds on the accuracy of MENDNet models by conducting sensitivity evaluations. During the sensitivity evaluation, while adjusting one threshold (such as

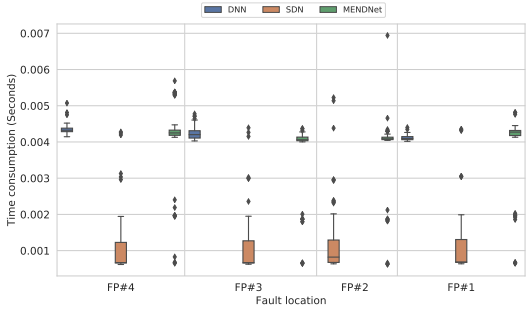
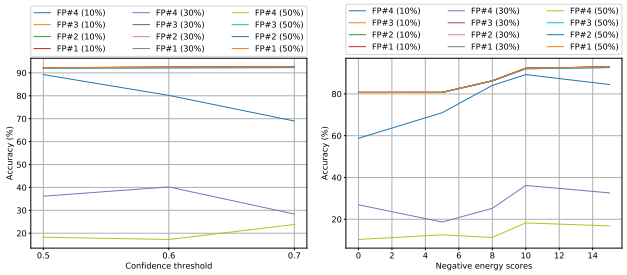


Figure 4. Latency for VGG16 model on CIFAR-10 dataset.



(a) VGG16 with fixed τ **(b)** VGG16 with fixed γ

Figure 5. Accuracy with varying confidence threshold γ and negative energy score threshold τ for different models for CIFAR-10.

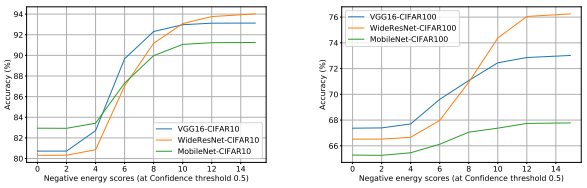
confidence threshold γ), we maintain the other one (negative energy score threshold τ) at a constant level.

The sensitivity evaluation results are depicted in Figure 5. Figure 5a illustrates the evaluation conducted by varying τ, while Figure 5b displays the evaluation conducted by varying γ. It is evident that, for FP#1–FP#3 across all fault rates, the model's accuracy scores exhibit minimal changes when either threshold is varied. The selection of the optimal threshold pairs is primarily based on the model's accuracy after faults are induced in FP#4. When inducing faults at FP#4, we did not identify any specific pattern in the model's accuracy while varying γ. However, by varying τ, we discovered that τ values ranging from 8 to 10 yield improved model accuracy for faults at FP#4. Evaluation results for CIFAR-100 data can be found on the website[1].

> **Finding 4**: *Best τ and γ values are selected majorly based on the model-accuracy after faults are induced at FP#4.*

6 Discussion

Fault-localization for first layer faults. In this section, we discuss if the MENDNet models can localize faults induced at earlier layers. Previously, we observed that MENDNet is able to mitigate faults induced at FP#1-FP#3. However, for faults induced at FP#4, the accuracy of MENDNet model is similar to the accuracy of traditional DNN. Although traditional DNN can not localize the faults at FP#4, where we provide evidence that MENDNet model is able to do so. If faults are induced at FP#4, the negative energy score at all the exits will be in the lower range, showing high uncertainty. Thus, we can ensure that the faults are induced before the first exit. This fault localization feature in evaluated by utilizing a metric, *Uncertainty Violation Rate (UVR). UVR is calculated by the percentage of test inputs for which negative energy score is lower than a predefined threshold τ*. Upon experimentation with varying model-dataset configurations and fault rates, the proposed framework showcases its effectiveness in localizing faults in the MENDNet architecture. The experimental results for fault localization using UVR is presented in our website[1]. It is noticed that for all three models and for all exits, UVR is higher than 50% for the faulty model, while for non-faulty model, the UVR is lower for at least one exit. The results provide evidence that MENDNet can localize faults at early layers.



(a) For CIFAR-10 **(b)** For CIFAR-100

Figure 6. Model accuracy with varying τ.

Improving non-faulty SDN's accuracy with MENDNet. We demonstrate that employing the MENDNet exit strategy can significantly enhance the accuracy of multi-exit networks compared to using a confidence threshold as the exit strategy. While we have previously shown that for some model-dataset pairs, faulty MENDNet models can outperform non-faulty DNNs, the improvement was not substantial. Figure 6 illustrates that for all model-dataset pairs, the accuracy of SDN is improved by up to 10% when utilizing the MENDNet model. Here, we set the confidence threshold γ to 0.5 and vary the negative score threshold τ. A value of $\tau = 0$ represents the SDN exit strategy. By increasing the value of τ, we observe an improvement in accuracy. This is because a higher τ rejects uncertain predictions from exits, leading to enhanced accuracy.

7 Conclusion

In this work, we propose a JIT framework to detect and mitigate faults in the AI hardware accelerators. First, we use uncertainty quantification techniques to detect faults. Next, we propose MENDNet framework to use both uncertainty quantification and multi-exit networks to mitigate the faults. Based on the evaluation on popular model-dataset pairs, we find that MENDNet can improve the accuracy of DNN against faults at different fault positions and fault rates. We have open-sourced the code and it is available at https://github.com/isnadnr/MENDNet.

References

[1] [n. d.]. Martian Computing Is Light on RAM, Heavy on Radiation Shielding | WIRED. https://www.wired.com/2012/08/martian-computing-is-light-on-ram-heavy-on-radiation-shielding/. (Accessed on 01/03/2022).

[2] Muhammad Salman Ali et al. 2020. ERDNN: Error-Resilient Deep Neural Networks With a New Error Correction Layer and Piece-Wise Rectified Linear Unit. *IEEE Access* 8 (2020), 158702–158711.

[3] Arjun Chaudhuri et al. 2023. Functional Test Generation for AI Accelerators using Bayesian Optimization. In *2023 IEEE 41st VTS)*. IEEE, 1–6.

[4] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. PMLR, 1050–1059.

[5] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261* (2019).

[6] Younis Ibrahim et al. 2020. Soft error resilience of deep residual networks for object recognition. *IEEE Access* 8 (2020), 19490–19503.

[7] Yigitcan Kaya et al. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*. PMLR, 3301–3310.

[8] Alex Kendall et al. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems* 30 (2017).

[9] Shamik Kundu et al. 2021. Toward Functional Safety of Systolic Array-Based Deep Learning Hardware Accelerators. *IEEE TVLSI* 29, 3 (2021), 485–498.

[10] Shamik Kundu et al. 2023. Trouble-shooting at GAN Point: Improving Functional Safety in Deep Learning Accelerators. *IEEE Trans. Comput.* (2023).

[11] Weitang Liu et al. 2020. Energy-based out-of-distribution detection. *Advances in neural information processing systems* 33 (2020), 21464–21475.

[12] Siyu Luan et al. 2023. Timing performance benchmarking of out-of-distribution detection algorithms. *Journal of Signal Processing Systems* (2023), 1–16.

[13] Elbruz Ozen and Alex Orailoglu. 2020. Boosting Bit-Error Resilience of DNN Accelerators Through Median Feature Selection. *IEEE TCAD* 39, 11 (2020), 3250–3262. https://doi.org/10.1109/TCAD.2020.3012209

[14] Geng Yuan et al. 2021. Improving dnn fault tolerance using weight pruning and differential crossbar mapping for reram-based edge ai. In *2021 22nd ISQED*. IEEE, 135–141.