Processing-in-Memory Architecture with Precision-Scaling for Malware Detection

Sreenitha Kasarapu*, Sathwika Bavikadi*, Sai Manoj Pudukotai Dinakarrao
Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA, USA
{skasarap, sbavikad, spudukot}@gmu.edu

***Abstract*—The wide adaptations of embedded systems in multiple fields have led to smart connectivity across devices and enhanced computation capabilities. Despite the vast applications in different areas, embedded systems face huge security threats. One of the critical security vulnerabilities is caused by malicious software *a.k.a* malware. Successful malware detection by employing Machine Learning (ML) is widely adopted in many systems. One of the prominent challenges in implementing neural network (NN) architectures is the requirement to have a large number of computational resources. Furthermore, the frequent movement of data between logic and memory units adds large overheads. Conversely, the IoT and edge devices are often limited in terms of the number of available resources. As a panacea, we introduce a PIM-based architecture to address such concerns and improve memory access latency. Such a paradigm further enriches the malware detection latency by mitigating the data transfer latency. To further improve the throughput and energy consumption, we employ precision scaling for the PIM-based malware detection in this work. We observe a malware detection accuracy of 98% with the proposed technique. Our proposed PIM architecture has 1.09× higher throughput than other traditional PIM architectures. Furthermore, precision scaling and PIM improve the energy efficiency by 1.5× compared to the full-precision operation without any penalty in performance.**

I. Introduction

With the technical developments in hardware architecture and embedded systems, IoT devices and applications have procured enormous interest in the past few decades [1]. These IoT and edge computing devices connect to the internet over a network for communicating between devices and with the base station(s). As these devices handle vast amounts of user data and are considered a soft target by cyber-attackers [2]. Among multiple cyber threats, malicious applications *a.k.a* malware is considered a prominent threat [2] due to its feasibility to propagate and ease of development. Malware is malicious software developed to infect a system to exploit and steal information such as passwords and financial data, manipulating the stored data without the user's consent. In 2021 alone, over 5.4 billion recorded malware attacks [3]. In the first half of 2022, nearly 2.8 billion malware attacks were recorded on IoT and edge devices.

Anti-virus software has been introduced to detect malware. However, anti-virus software often involves large overheads regarding resource usage and memory footprint. Static and dynamic analysis [4] is employed in anti-virus software for malware detection. Static analysis [4] is performed in a non-runtime environment by examining malware binaries' internal structure and not by executing the binary executable files. In dynamic analysis, the binary applications are inspected for malware traces by executing them in a harmless, isolated environment [4]. Unlike static analysis, dynamic analysis is a functionality test. The static analysis serves as quick testing but is not efficient. However, efficient dynamic analysis is a bulky and time-consuming process [5].

Malware detection using Machine Learning (ML) is seen as an efficient technique [6]. Nataraj et al. [7] introduced a technique for malware detection using image processing where binary applications are converted into grayscale images and classified using ML algorithms such as SVM. Among the ML-based malware detection techniques, the Convolutional Neural Network (CNN)-based image classification technique [8] is more robust and efficient due to its prime ability to learn image features. However, one of the main challenges with adopting such a technique is the massive computations involved. This challenge is exacerbated by the overheads involved in moving the data between memory and logic blocks. To alleviate these challenges, processing-in-memory (PIM) [9] has been proposed as an alternative paradigm in recent years. PIM architectures perform computations inside the memory, alleviating data movement overheads.

In this study, we propose a novel approach for malware detection to utilize the in-memory computing technique for the Processing-in-memory (PIM) platform. PIM is a novel computing paradigm in which the memory chip is enhanced with computing capabilities. This essentially restricts the circulation of the data within the memory chip and thereby drastically minimizes the power consumption and latency caused by the data movements. In addition, a PIM architecture takes advantage of its proximity to the data to perform massively parallel computing. Therefore, such a PIM paradigm is particularly suitable for data-intensive applications like deep learning (DL) and optimization problems.

Several recent studies have proven that PIM architectures outperform GPU and CPU designs for training deep neural networks (DNN) and combinatorial optimization problems in terms of throughput and energy efficiency. While traditional PIM architecture, such as bitline-wise architecture [10] and analog crossbar array architecture [11], have been regarded as better alternatives to conventional computing hardware for executing the heavy computational load of DNN [9]. These architectures suffer from the complexity and overhead of digital-to-analog (DAC) and analog-to-digital (ADC) conversions. Unlike the bitwise processing PIMs, the recently developed Look-up-table (LUT) based PIMs are more flexible, with

§*Both authors contributed equally to this research

superior energy efficiency for a similar level of performance, such as LAcc [12], pPIM [13]. This feature makes the LUT-based PIM architectures superior for DNN computations [13].

By considering the benefits of superior malware detection performance by DNNs and hardware efficiency by LUT-based PIMs, we propose a LUT-based hardware accelerator for malware detection in this work. This is primarily targeted toward the edge and resource-constrained devices. We further employ precision scaling to decrease the power consumption of malware detection.

The novel contributions of this work are outlined as follows:

- We introduce a malware detection approach that utilizes a LUT-based Processing-in-Memory computational paradigm.
- Precision scaling is introduced to achieve lower power consumption for malware detection without trading off the performance.
- We evaluate the malware detection on various CNN architectures, including AlexNet, ResNet-18, -34, -50, VGG-16, and MobileNet V2.

II. Related Work

A. Malware Detection Techniques

Static analysis [4] on malware data is performed by comparing the opcode sequences of binary executable files, control flow graphs, and code patterns. The main drawback of static analysis is it is unable to detect malware when adversaries add junk of unrelated functionalities, which decreases the malware similarity score [14]. Malware detection using dynamic analysis is performed based on detecting system calls, or HPC [4]. However, they are computationally expensive and are inefficient in detecting hidden malware code blocks.

Later [7] introduced a technique for malware detection using image processing where binary applications are converted into grayscale images. The generated images have identical patterns because of the executable file structural distributions. The paper used the K-Nearest Neighbour ML algorithm to classify malware images. Other approaches [6] include image visualization and classification using ML algorithms such as SVM. However, the involved latency is significant and the achieved performance is limited. Neural networks such as deep neural networks (DNNs) are used extensively to solve the problem [15], as neurons can capture the features of the images more accurately than other ML algorithms. But, the fully connected layers of artificial neural networks tend to exhaust computational resources. In [8], authors use convolutional neural networks (CNNs) due to their ability to efficiently handle image data through feature extraction by Convolutional 2D layers and using Maxpooling 2D layers to downsample the input parameters. However, the involved computations and depth of CNNs make it challenging to embed them on edge and IoT devices.

B. Processing-in-Memory (PIM)

In recent years, PIM designs have received a lot of attention from DNN/CNN applications. The PIM architecture can reduce data movement's latency and energy costs. Moreover, with the integration of memory and processing capability, the PIM architecture can efficiently execute matrix-vector multiplication (MVM) operations, which are fundamental computing operations in various disciplines of research such as signal processing, machine learning [12], [13], deep learning [16], and stochastic computing, image processing, and recognition [17], data mining [13], and cryptographic [18], [19].

Numerous works have been proposed on in-memory computing hardware accelerators on different memory platforms including the traditional memory platforms of Static and Dynamic Random Access Memory (SRAM and DRAM) [20], [12], [17], [21], non-volatile Resistive RAM (ReRAM) [22], Phase-changing Memory (PCM), and Magnetic RAMs such as Spin Transfer Torque MRAM (STT-MRAM), and Spin-Orbit Torque MRAM (SOT-MRAM) technologies. It has been found that a satisfactory level of accuracy can be retained even despite performing various levels of quantization/down-scaling of data parameters in CNN algorithms [16]. This opens up an exploration space for high-performance and low-power CNN implementations for real-time application domains such as IoT, mobile, and edge applications. The PIM architecture is gaining popularity in real-time application domains. To the best of our knowledge, the PIM architecture has not been utilized for malware detection.

III. Problem Formulation

With technology advancements, attackers are introducing complex malware families, making it impossible for low-scale embedded systems to detect malware under constrained resources. Even advanced anti-malware software fails to detect these advanced malware families in real-time. One can define the problem of reliable malware detection in low-scale embedded devices as follows:

$$\begin{aligned} & C(\mathbb{D}) : X \rightarrow Y \\ \text{s.t.} \quad & S = \sum_{i=1}^{mem} \sum_{j=1}^{t} C[M^{-i} \cdot T^{-j}] \end{aligned} \tag{1}$$

In Equation 1, C is a classifier model trained with dataset $\mathbb{D}$ to perform malware detection. After training, the classifier C will be able to classify any input sample X and map it to either malware class M or benign class B. The output class is represented as Y. As Equation 1 represents, malware detection in real-time requires addressing the memory and timing constraints, represented as M^{-i} and T^{-j} respectively.

This work utilizes the PIM architecture designed to support the data-intensive malware detection framework. Utilizing PIM-based processing and also employing precision scaling on the input operands improves the resource consumption and throughput of the ML model used for malware detection. In bitwise logic-based architectures, which by nature employ bit-wise parallel processing to accomplish operations with variable bit-width, it becomes challenging to perform large-data (8-bit/16-bit) operations. These architectures allow sacrifices of accuracy but are useful for low-power applications. LUT-based PIMs can prevent this by employing decomposition algorithms instead of extensive adder chains and shifters to produce products of operands. Therefore we use a novel LUT-based

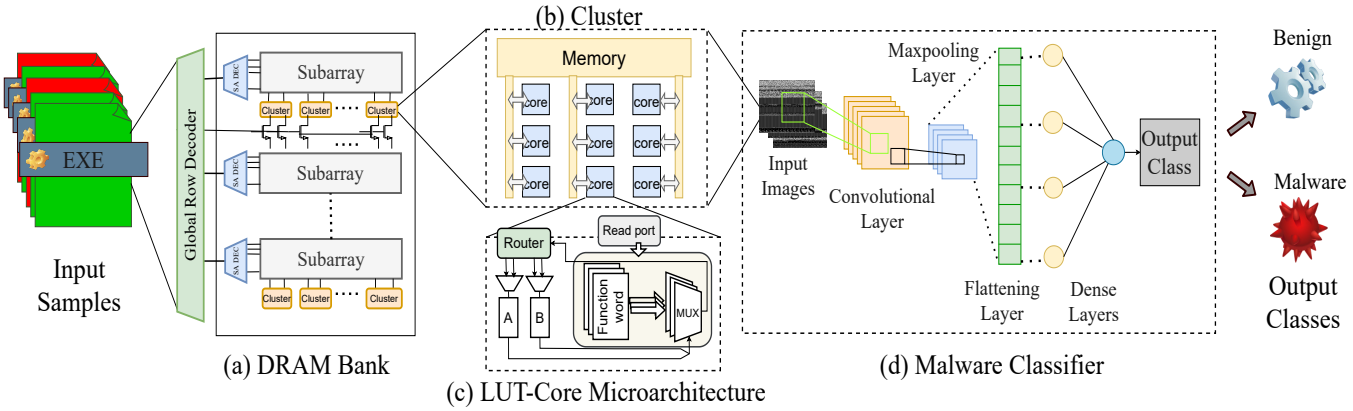

Fig. 1. Hierarchical view of the architecture implementation of malware detection on the processing in-memory architecture

PIM [13] architecture in order to tackle the computational loads of the CNN layer operations of large data (8-bit/ 16-bit). These are inherently capable of offering CNN/ DNN inference with higher data precision with better accuracy while not sacrificing performance and efficiency.

IV. Proposed Technique

A. Overview of the Proposed Technique

The overview of the proposed technique follows the flow as shown in Figure 1. The input data samples are stored in the DRAM memory bank as represented in Figure 1(a). In-memory processing is employed using a DRAM cluster to improve memory access time. The architecture of each DRAM cluster is represented in 1(b). Each cluster's LUT-core is represented as 1(c). The binary input samples are processed in the memory and converted into images. As this is done using an in-memory processing technique, there was no need for excessive data movement. Once the data is accessed from memory, it is given as input to the malware classifier given in 1(d). The test data used for inference is precision scaled using uniform quantization. The different elements in the proposed technique are:

- *PIM Unit*: To overcome the memory and resource constraints in low-scale embedded systems for malware detection, a novel PIM architecture is employed. In-memory processing improves throughput and limits resource consumption.
- *Malware Classifier*: Binary malware and benign samples are fed to the Convolutional Neural Network to train a malware classifier.
- *Precision-scaling*: While retaining the malware detection accuracy, a low-precision data version is employed to decrease the power consumption to make the task applicable to real-time detection.

B. Malware Detection Model

Computer vision-based machine learning (ML) techniques need images for localized feature extraction. In this work, the input data is stored in the memory. As the CNNs/DNNs in the computer vision operate on the pixel information, we employ the stored digital data of the applications for processing, i.e., directly classifying the applications stored in the memory. This alleviates the conversion complexity and overheads as well. Each stored binary file value is represented using an 8-bit vector format. Thus, each 8-bit vector represents a value of the programming file or application in the machine-readable format stored in the memory. These values are stored and accessed in a row-column format, representing a tabular input. This tabular input will be fed to the PIM-based CNN for malware detection. However, one of the challenges in comparison with the traditional computer vision applications is that the size of the gray-scale image varies with the size of the binary file.

To address this challenge, we perform table resizing and scaling to make its size uniform. As the pattern or sub-pattern of malware cannot alter despite embedding the malware to launch a malicious payload, through this technique, the malware can be detected with a higher performance (around 98% accuracy). To minimize the computational latency and resources, we employ uniform quantization. While retrieving the binary data stored in DRAM memory banks, uniform quantization is applied to quantize them from floating point 32-bit to integer types 16-bit, 8-bit, and 4-bit and feed to the PIM-based CNNs. This drastically reduces the associated computational overheads and the memory consumption.

In the proposed work, the training data consists of several sequences for a variety of classes of malware (backdoor, rootkit, trojan, virus, and worm) and benign applications stored in the DRAM. The CNN model is trained (offline) using these data samples for classification, as shown in Figure 1(d). The inference is processed in real-time in low-scale embedded systems. The multiply-and-accumulate (MAC) operations in each of the CNN layers are accommodated by PIM architecture by programming the LUT cores inside the cluster, as shown in Figure 1(c). The CNN classifier trained on input data is used to classify the output class. The inference is performed during runtime.

The classification accuracy $a(\cdot)$ can be defined as the difference in probability of the predicted class $P(Y_{pred})$ to the real class $P(Y_{true})$.

$$a(\cdot) \leftarrow P(Y_{true}) - P(Y_{pred}) \tag{2}$$

The novel PIM-based architecture utilized for malware detection is discussed in detail in IV-C.

C. The PIM Architecture

This work utilizes a PIM architecture designed to support compute-intensive applications, including convolution neural

networks (CNNs) and DNNs. This PIM architecture is depicted hierarchically in Figure 1, including (a) the arrangement of clusters within a DRAM bank, (b) the architecture of a cluster, and (c) the architecture of the LUT core.

1) Core Architecture: At the center of the PIM architecture is the proposed core, which facilitates programmable operations on two 4-bit inputs. A LUT-based design is adopted for the PIM core instead of a pre-defined logic circuit to provide functional programmability. The LUT-based PIM can perform in-memory arithmetic operations such as addition, multiplication, substitution, and comparison operations with a significantly lower delay than bitwise computations. Therefore, a collection of these operations can be used to implement various ML algorithms.

Figure 1(c) shows a detailed view of the architecture of a single LUT core. The LUT cores inside the cluster are formed of an 8-bit 256:1 multiplexer, accompanied by eight 256-bit latch arrays. The pre-calculated outputs of any particular 8-bit operation are stored in the latches as eight 256-bit function words. These latches can read new function words from the bit lines of the DRAM subarrays. Each LUT can produce a 4-bit data output for two input data operands with a size of 4-bit width, as shown by A and B registers in Figure 1(c). These registers together drive the select pins of the multiplexers and make them 'look up' specific 8-bit data from the eight latches that represent the operation's output.

2) Cluster Architecture: To perform operations necessary for CNN acceleration, such as convolution operations, the PIM cluster integrates all the operations done by the LUT core. Nine PIM cores are arranged as shown in Figure 1(b) and placed inside the memory unit to form a PIM cluster and perform in-memory computations. Inside the PIM cluster, the PIM core performs various logic and arithmetic operations associated with the CNN acceleration for malware detection. To perform a specific operation, such as multiplication and accumulation operations all the cores inside the cluster are connected by a router. The router also makes it possible to access data from any core at any moment throughout the implementation to perform the operations.

3) Router Architecture: All nine LUT cores in a cluster are connected via a router mechanism as shown in Figure 1(c), which enables direct and parallel communication between them. A router that connects the read/write ports on all components of the cluster in order to facilitate parallel communication among all the cores in the cluster. This enables the router to access any data at any implementation point to perform operations required for CNN acceleration for malware detection.

D. Implementation on the PIM Architecture

Malware detection performs similar mathematical operations as the CNN/DNN classifier, which consists of convolution layers, activation layers, max-pooling layers and fully-connected layers. The PIM architecture can easily accommodate these operations by programming the LUT cores inside the PIM cluster to perform these operations. For our baseline design, we choose a single 8-bit operand or a pair of 4-bit operands since it represents the majority of image & video pixel data. At the center of the architecture, each core can facilitate programmable operations on a pair of 4-bit inputs. With the help of these cores and the routing mechanism, the PIM architecture can perform any 8-bit mathematical operations required by the CNN/DNN layers. Since the clusters in the PIM architecture can perform the required mathematical operations for CNN, an array of these clusters can be utilized to implement different layers of the CNN model. The LUT cores in the PIM need to be re-programmed whenever there is a need to perform a different operation than the prior implemented operation. The key advantage of using LUTs in a PIM architecture is that the LUT core in the PIM architecture can be re-programmed and is faster compared to traditional in-memory computations. As a result, the functional flexibility required for implementing malware detection using CNNs is provided.

In this work, we employ offline learning for ML detection, thus the need for programming is minimal. The inference, i.e., classification of the malware (as described in Section IV-B), is performed when a new application or programming file is loaded into the memory. This facilitates real-time malware detection.

V. Experimental Results

A. Experimental Setup

The proposed methodology is implemented on multiple Jetson Nanos containing 128-core NVIDIA Maxwell architecture-based GPUs. The PIM performance evaluations are obtained through ICC-based simulations. We have obtained malware applications from VirusTotal [23] with 12500 malware samples encompassing five malware classes: backdoor, rootkit, Trojan, virus, and worm, and 13700 benign application files. The files are stored in an 8-bit vector format and they are accessed in a row-column format, representing a tabular input. This tabular input is represented as a gray-scale image and fed to the PIM-based CNN for malware detection. The grayscale images are re-sized to 32 x 32 and split into 70% training set and 30% test set. After training, we analyze the models' malware detection accuracy, energy efficiency, and throughput. Further, the inference accuracy of these models on precision scaling schemes such as 16-bit, 8-bit, and 4-bit are evaluated. We also evaluated the performance using different metrics(*such as* operational latency, power consumption and active area) from HDL synthesis on Synopsys Design Compiler using 28 nm standard cell library from TSMC. The characteristics (delay, power consumption, and active area) of the PIM cores and the PIM cluster that can perform a single 8-bit MAC operation are shown as 0.8 ns, 2.7 mW, 4196.64 μm^2, 6.4 ns, 8.2-11 mW, 37769.81 μm^2 respectively.

B. Evaluation of Malware Detection Accuracy on Various Precision Schemes

The integer precision schemes such as 16-bit, 8-bit, and 4-bit are applied to data given as input to various ML algorithms. The effect of precision scaling on malware detection capability is observed in various algorithms. The base accuracy of models trained using 32-bit floating point data is compared to that

of 16-bit, 8-bit, and 4-bit data. As shown in Figure 2, the performance of various CNN models on different data precisions is compared. We can observe a considerable performance decay with decreased precision for models such as AlexNet and ResNet18. But other models such as ResNet34, ResNet50, VGG-16, and MobileNetV2 retain the accuracy despite the low precisions. We can observe a malware detection accuracy of about 98% in models VGG-16 and MobileNetV2 for an 8-bit precision scheme. And for 4-bit precision models ResNet34, VGG-16, and MobileNetV2 have an accuracy of about 95%. So even with precision scaling schemes, we can still retain effective malware detection capability.

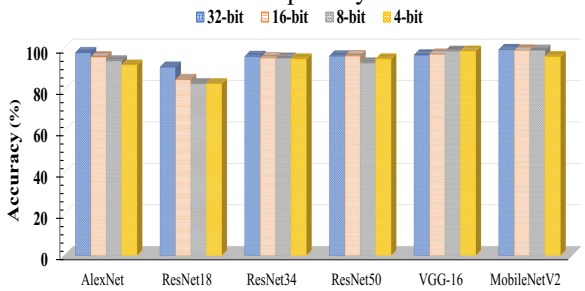

Fig. 2. Performance Evaluation of AlexNet, ResNet18, ResNet34, ResNet50, VGG16, and MobileNetV2 on the PIM accelerator with precision scaling (a) 32-bit floating point, (b) 16-bit integer type, (c) 8-bit integer type and (d) 4-bit integer type

Table I presents the comparison of the proposed technique with the existing malware detection techniques. We compare the performance of the proposed technique in terms of accuracy, F1 score, and recall. All the models in table I focus on malware detection based on malware and benign features. Compared to the existing techniques the proposed PIM-based malware detection achieves high throughput without performance decay. It maintains a malware detection accuracy of 98%. It is also evident that the proposed technique achieves efficient malware detection accuracy even in low-precision settings.

TABLE I
COMPARISON WITH EXISTING HPC-BASED DETECTION TECHNIQUES

Model	Accuracy (%)	F1-score	Recall
OneR [24]	0.81	0.81	0.82
JRIP [24]	0.83	0.83	0.84
PART [24]	0.81	0.815	0.831
J48 [24]	0.82	0.82	0.82
Adaptive-HMD [25]	0.853	0.853	0.858
SVM [26]	0.739	0.736	0.772
RF [26]	0.835	0.834	0.822
NN [26]	0.811	0.811	0.816
SMO [27]	0.932	0.933	0.931
Proposed	**0.987**	**0.987**	**0.982**

C. Performance Evaluation

This section presents the comparative analysis of the algorithm implemented on PIM in terms of throughput (in Frames per second) and Energy Efficiency (Frames/Joules). For evaluation purposes, we have implemented AlexNet, ResNet 18, 34, 50, VGG16, and MobileNetV2 networks on the PIM accelerator. Figure 3 presents comparisons of the throughput (in Frames per second) and Figure 4 energy efficiency (in Frames per Joule) of inference on all these CNNs deployed on the PIM accelerator.

From Figure 3 it can be observed that the proposed PIM model is capable of performing malware detection tasks with

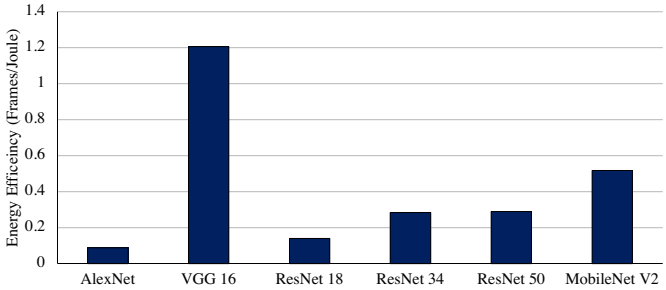

Fig. 3. Comparison of Energy efficiency (Frames/Joules) for AlexNet, ResNet18, ResNet34, ResNet50, VGG16, and MobileNetV2 on the PIM accelerator

an impressive performance of low latency. For example, ResNet-50, the largest network consisting of 50 layers and thirty-eight billion computations, is processed within 10 mS.

A similar trend is observed for energy consumption, from Figure 4 it can be observed that the proposed PIM model is capable of performing malware detection tasks with high energy efficiency. This is because the PIM module supports 8-bit precision mode in order to perform the operations required for CNN acceleration. These tasks can be performed by distributing the data across the cluster which contains nine cores connected by a router which inherently offers a higher degree of parallelization and performs all the operations in comparatively fewer steps.

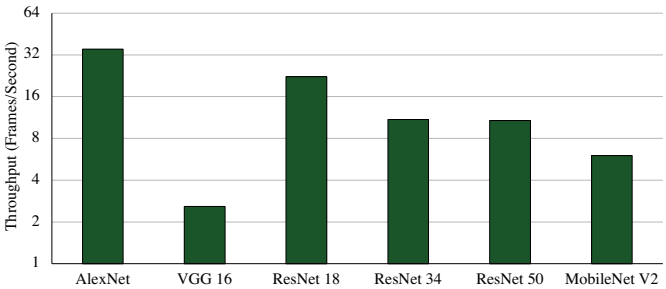

Fig. 4. Comparison of Throughput (Frames/second) for AlexNet, ResNet18, ResNet34, ResNet50, VGG16, and MobileNetV2 on the PIM accelerator

D. Performance Comparison with State-of-the-Art Hardware Accelerators for CNN Implementation

Performance is evaluated by comparing the proposed architecture with state-of-the-art hardware accelerator architectures in terms of power consumption (Watt) and throughput (Frames/second), as shown in Figure 5.

As a proof of concept, we evaluate and implement AlexNet [28] on the proposed architecture with the 8-bit width precision. We envision a 256 PIM cluster arrangement in a DRAM chip as this configuration provides a fine balance between performance, power consumption, and on-chip area overhead. An input dimension of 224x224x3 has been considered for performance benchmarking with the other CNN accelerators and different operational modes of PIM. The PIM architectures under comparison in this section include DRAM-based bulk bit-wise processing devices DRISA [20], and LUT-based PIM implemented on the DRAM platforms such as LAcc [12]. On the other hand, the von Neumann devices under comparison are Intel Knights Landing (KNL), a state-of-the-art CPU [29], and Pascal Titan X, a state-of-the-art GPU.

It can be observed from Figure 5, that the PIM architectures, in general, outperform both the CPU and the GPU by a huge

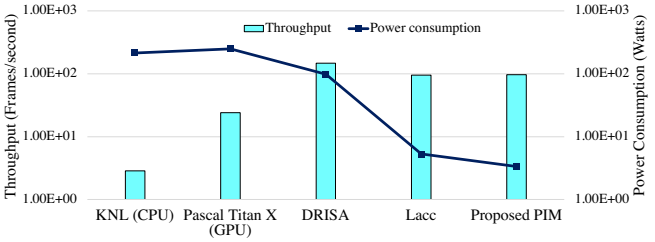

Fig. 5. Comparative performance analysis of PIM with respect to state-of-the-art hardware accelerator architectures in terms of throughput (Frames/second) and power consumption (Watt)

margin since all these PIMs can avoid the significant overhead and latency associated with off-chip communications, unlike the CPU and the GPU. In fact, the most computation-intensive 8-bit fixed-point operation mode PIM ideally provides 4.02×, 45× higher throughput compared to Pascal Titan X GPU and Knights Landing Processor while being 74.62×, 64.13× more energy-efficient.

On the other hand, a relatively higher throughput is observed for DRISA [20] due to its ability to parallelize operations across multiple banks, albeit at significantly low power efficiency. The benefits of adopting LUTs in order to utilize pre-calculated results instead of performing in-memory logic operations are convincingly demonstrated by LAcc [12] which achieves impressive inference performance at quite a low power consumption. From Figure 5, it is also observed that the proposed PIM outperforms DRISA and LAcc in both the throughput by 0.065×, 1.09× as well as power efficiency by 29.25×, 1.5× respectively for AlexNet inference.

VI. Conclusion

In this paper, we proposed a PIM-based ML modeling technique for malware detection. The proposed approach not only achieves low latency for implementing malware detection tasks but also provides high energy efficiency. Such a methodology makes the real-time malware detection task in embedded devices adaptable. The performance of the proposed PIM is evaluated by comparing it with state-of-the-art CPU, GPU, and other PIM architectures. The experimental results indicate that the proposed PIM is 74.62×, 64.13× more energy-efficient and has 4.02×, 45× higher throughput compared to the GPU and CPU respectively. It is also observed that the PIM is 1.5× energy efficient and has 1.09× higher throughput than other LUT-based PIM architecture. Multiple CNN models were trained using data that was precision scaled into 16-bit, 8-bit, and 4-bit samples, to further aid the energy efficiency and throughput. The performance of these models is compared with state-of-the-art malware detectors. From experimental results, it is evident that the proposed technique is efficient for malware detection as it does not experience performance decay.

References

[1] T. Adiono, "Challenges and opportunities in designing internet of things," *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*, 2014.

[2] O. Abbas and et al., "Big data issues and challenges," 2016.

[3] J. Johnson, "Number of malware attacks per year 2020," Aug 2021. [Online]. Available: https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/

[4] A. Damodaran *et al.*, "A comparison of static, dynamic, and hybrid analysis for malware detection," *Journal of Computer Virology and Hacking Techniques*, 2015.

[5] C. Rossow and et.al, "Prudent practices for designing malware experiments: Status quo and outlook," *Symposium on Security and Privacy*, 2012.

[6] K. Kancherla and et.al, "Image visualization based malware detection," in *Computational Intelligence in Cyber Security (CICS)*, 2013.

[7] L. Nataraj and et al., "Malware images: Visualization and automatic classification," in *Int. Symposium on Visualization for Cyber Security*, 2011.

[8] D. Gibert and et.al, "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, 2019.

[9] S. Bavikadi *et al.*, "A review of in-memory computing architectures for machine learning applications," in *GLSVLSI*, 2020.

[10] V. S. et al., "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," *IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[11] A. D. P. et al., "An mram-based deep in-memory architecture for deep neural networks," in *IEEE International Symposium on Circuits and Systems*, 2019.

[12] Q. Deng and et al., "Lacc: Exploiting lookup table-based fast and accurate vector multiplication in DRAM-based CNN accelerator," in *ACM/IEEE Design Automation Conf. (DAC)*, 2019.

[13] P. R. Sutradhar *et al.*, "pPIM: A programmable processor-in-memory architecture with precision-scaling for deep learning," *IEEE Computer Architecture Letters*.

[14] A. Moser and et.al, "Limits of static analysis for malware detection," in *Annual Computer Security Applications Conference (ACSAC 2007)*, 2007.

[15] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," in *Int. Conf. on Data Management, Analytics and Innovation (ICDMAI)*, 2017.

[16] P. R. Sutradhar *et al.*, "Look-up-table based processing-in-memoryarchitecture with programmable precision-scalingfor deep learning applications," *IEEE TPDS*, 2021.

[17] S. Bavikadi *et al.*, "upim: Performance-aware online learning capable processing-in-memory," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2021.

[18] P. R. Sutradhar *et al.*, "An ultra-efficient look-up table based programmable processing in memory architecture for data encryption," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*.

[19] S. Liu *et al.*, "Accelerating adversarial attack using process-in-memory architecture," in *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*, 2022, pp. 325–330.

[20] S. L. et al., "Drisa: A dram-based reconfigurable in-situ accelerator," in *IEEE/ACM International Symposium on Microarchitecture*, 2017.

[21] S. Bavikadi *et al.*, "Heterogeneous multi-functional look-up-table-based processing-in-memory architecture for deep learning acceleration," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–8.

[22] P. C. et al., "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016.

[23] "Virustotal package," 2021. [Online]. Available: https://www.rdocumentation.org/packages/virustotal/versions/0.2.1

[24] N. Patel *et al.*, "Analyzing hardware based malware detectors," in *ACM/EDAC/IEEE Design Automation Conference (DAC)*.

[25] Y. Gao *et al.*, "Adaptive-hmd: Accurate and cost-efficient machine learning-driven malware detection using microarchitectural events," in *IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2021.

[26] A. P. Kuruvila, S. Kundu, and K. Basu, "Analyzing the efficiency of machine learning classifiers in hardware-based malware detectors," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2020.

[27] H. Sayadi and et.al, "Customized machine learning-based hardware-assisted malware detection in embedded devices," in *IEEE International Conference On Trust, Security And Privacy In Computing And Communications*, 2018.

[28] M. Z. Alom *et al.*, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv*, 2018.

[29] A. Sodani, "Knights landing (knl): 2nd generation intel® xeon phi processor," in *2015 IEEE Hot Chips 27 Symposium (HCS)*.