This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

Guarding Deep Learning Systems with Boosted Evasion Attack Detection and Model Update

Xiangru Chen, Dipal Halder, *Graduate Student member, IEEE*, Kazi Mejbaul Islam, *Student member, IEEE*, and Sandip Ray, *Senior Member, IEEE*

***Abstract*—Deep learning systems are susceptible to evasion attacks, which represent a significant category of security vulnerabilities. These attacks entail the alteration of input data in such a way that the victim Deep Neural Network (DNN) misclassifies it. Researchers have devised detection and defense methods to counter evasion attacks; however, these techniques impose a significant computational burden and are not suitable for real-time detection on devices with limited resources. Our paper presents an infrastructure, GERALT designed to improve the efficiency of evasion attack detection for real-time execution on edge devices. It involves a partition analysis that optimizes detection methods and allows for the use of a smaller detection network. Additionally, we propose a hardware architecture that accelerates inter-network inference using intermediate data reuse techniques and enables a different pattern of model updates between cloud servers and edge devices in real-world applications. Furthermore, it is also extended to a principle of inter-network accelerator design, which is evaluated at different PE ratios. Our evaluations demonstrate that GERALT achieves more than 3x improvement in performance compared to standard accelerators like Eyeriss, without affecting detection and classification accuracy. The boosted model update system avoids the bandwidth limit between edge devices and the cloud server, saving 14 hours when updating the model for a new evasion attack.**

***Index Terms*—Evasion Attack Detection, Inference Accelerator, Deep Neural Network, Adversarial Training, Adversarial Example, Model Update from Cloud, Image Classification**

I. INTRODUCTION

We are seeing an explosive proliferation of Deep Neural Networks (DNN) across a variety of applications. Image classification is a key area of application of DNNs, which involves the determination of one or more target labels to a given image, and is used in a variety of safety-critical systems, including autonomous vehicles, healthcare, surveillance, etc. Unfortunately, recent research has shown that DNN applications are vulnerable to a number of adversarial attacks. *evasion attack* [26], [37] stands out as a critical threat, which entails introducing perturbations, such as noise, to the input image with the aim of inducing misclassification. For instance, in automotive Computer Vision systems, evasion attacks could lead to misclassifying stop signs as speed limit signs [9]. Given the critical nature of the targeted applications, an unmitigated evasion attack on a DNN can result in catastrophic consequences.

There has been significant recent research to address evasion attacks [21], [23], [41]. A promising approach is *self-evolvable defense*, which deploys defense strategies that can be reconfigured in runtime to prevent the adversary from exploiting knowledge of the implemented protection. These methods rely on adversarial training [10] to provide a robust defense against evasion attacks. The idea is to use adversarial examples to generate a trained model that is robust against evasion attack [4]. However, since they depend on extensive training with adversarial images, it is infeasible to deploy them on edge devices with limited computational resources to enable real-time self-evolution. On the other hand, moving the training to cloud servers would require high communication bandwidth to enable the transmission of large quantities of raw data from edge devices. To address the first issue, the gap between the new attack and the updated model should be minimized. For the second problem, resource occupancy should be reduced for edge devices. These challenges can be addressed by a real-time detection system with high performance and energy efficiency.

In this paper, we propose a framework, GERALT, to enable real-time detection of evasion attacks and only uploading attack images to the cloud for adversarial training to improve the robustness, which boosts the current model update system. GERALT is an integrated combination of a partition analysis of neural network (GERALT-part) and hardware architecture (GERALT-arch) that work together to accelerate inter-networking inference and provide architecture support for optimizing detection neural networks. More precisely, GERALT uses a smaller neural network for detection and skips the computation of the classification network's previous layers by sharing intermediate data. The approach is inspired by Transfer Learning [42]. We demonstrate how to use the technique effectively to reduce inter-network computation with 3x improvement in performance. Note that the reduction in computation is critical for viability of the approach in edge devices where stringent resource constraints preclude high computation-intensive solutions.

The paper makes the following important contributions.

- We propose a software-hardware system design to enable real-time detection using neural networks for evasion attacks, reducing the gap between attack and model update.

A preliminary version of the work, titled "GERALT: Real-time Detection of Evasion Attacks in Deep Learning Systems", appears in the 2023 International Symposium on Circuits and Systems. This paper significantly expands the conference version with new analysis, experiments, as well as more detailed presentation.

Xiangru Chen, Dipal Halder, Kazi Mejbaul Islam, and Sandip Ray are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611. USA. Email: cxr1994816@ufl.edu, dipal.halder@ufl.edu, kmejbaulislam@ufl.edu, sandip@ece.ufl.edu

"Copyright (c) 20xx IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org."

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

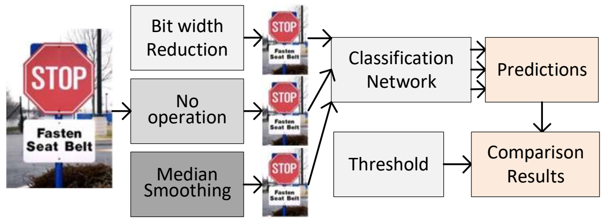

Fig. 1. Feature Squeezing

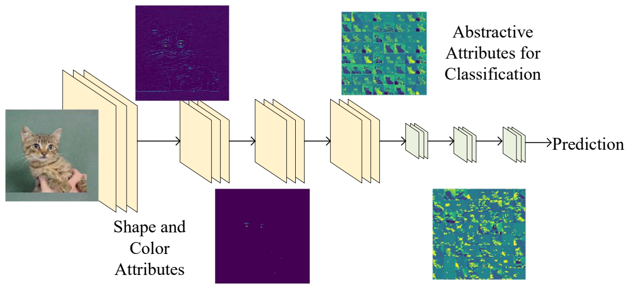
Abstractive Attributes for Classification
Prediction
Shape and Color Attributes

Fig. 2. Different Information of Neural Network Layers

- GERALT is developed to optimize the computation of the detection neural network to fit in the edge devices with similar detection and classification accuracy.
- To our knowledge, GERALT is presented as the first inter-network inference accelerator with intermediate data reused to reduce computation with special storage patterns and dataflow.
- The accelerator is further extended to the principle of inter-network computation to guide the accelerator design under similar conditions.
- GERALT is applied on an example detection method, Feature Squeezing [41] with both software and hardware evaluation to prove the advantages.
- The boosted model update system is evaluated with real-world data to prove the advantage, which saves 14 hours when updating the model for new evasion attacks.

The remainder of the paper is organized as follows. Section II provides the relevant background. Section IV-A introduces the GERALT-part approach and shows the design idea. GERALT-arch is developed as the supporting hardware in Section IV-B . Section V presents our evaluation of GERALT-arch. We discuss related work in Section VI and conclude in Section VII.

II. BACKGROUND

A. Adversarial Training

Adversarial training has been demonstrated as a useful technique for improving the robustness of machine learning systems to evasion attacks. In this approach, attack images are included in the dataset to train the neural network model. These attack images are appropriately labeled with the correct classes so that the model can learn the information about added noise and make correct predictions. Although this method may slightly reduce normal performance [31], [36], it holds tremendous potential for defense when given sufficient training time and an adequate supply of attack images, which is the key challenge to be addressed. A generative adversarial network(GAN) can be used for generating existing attack images, but it does not help with new attacks. This is why we focus on improving the efficiency of detection.

B. Evasion Attack Detection

The evasion attack is named because of the purpose of evading the correct classification of the neural network, which is recognized by adding noise to input images to get adversarial examples. The small amount of noise will be amplified by the nonlinear factor of the neural network and cause mislabeling in the final prediction [22], [26], [29], [37], [39].

The current detection methods for evasion attacks can be categorized into two classes. The first class involves the use of neural networks for attack detection. In this approach, the neural network learns the distinctive characteristics of attacked images or temporary results derived from images in order to recognize them as adversarial examples. The second class of methods relies on alternative data sources, such as natural scene statistics or network gradients, to build the classifier for detecting evasion attacks.

Feature Squeezing [41] is one of the promising approaches for detecting evasion attacks. The input images are squeezed to get two new images which are forward propagated through the neural network to get three predictions. By comparing the dissimilarities between the DNN's predictions from the compressed inputs and the original prediction, attacks can be identified. The process is illustrated in Fig. 1. Each input image is squeezed in two ways before being processed. With features squeezed in color or bit depth, the noises on the image will generate different output vectors. The dissimilarities over the threshold between the DNN's predictions from compressed inputs and the original one are used to identify attacks. To train the threshold and get the best performance, training images are loaded into the model for forward propagation, and the distances of images are sorted to set up the threshold. However, the substantial overhead associated with these detection techniques makes them impractical for real-world applications.

C. Inference Accelerators

Advanced architectures with specific dataflow and storage patterns have been proposed to improve the performance of DNN inference. General purpose accelerators like TPU, GPU [17], [27] can reduce parts of the under-utilization caused by the variability of neural network structure and dataset. However, the requirement of applications other than neural networks limits its potential for performance improvement. Specific designed inference accelerators further explore the opportunity of neural network computation, which increases the data locality and reduces movement. Nevertheless, none of these designs notices the relationship between networks which can boost the usability of deep learning systems under certain conditions like evasion attack detection.

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

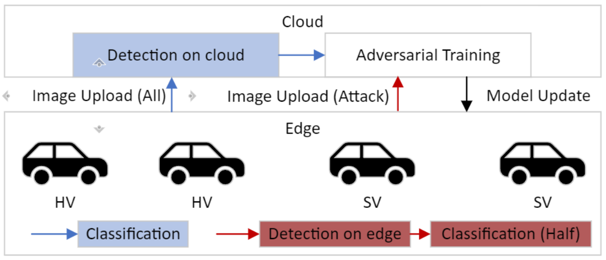

Fig. 3. GERALT-System Overview

D. *Transfer Learning*

Transfer learning [42] entails leveraging a pre-trained model from one task as a starting point for a model on a different task. When applying Convolutional Neural Networks (CNN) for image classification, the initial layers of the network learn basic shape features such as edges, colors, and lines [24], which stays the same in different classes of images shown in Fig. 2. This information can be reused between networks for different domains, reducing training time without compromising accuracy. Since the previous layers are fixed without weight update during transfer learning, the intermediate data also stays the same enabling inter-network reuse.

III. REAL-TIME DETECTION SYSTEM ENABLED BY GERALT

An essential feature of the GERALT system is to conserve bandwidth and deliver attack images in a timely manner. To illustrate the importance of this feature, let's consider the model update process for the traffic sign recognition system for self-driving vehicles. In this system, the collected images are leveraged to enhance the robustness of the neural network model through adversarial training. Fig 3 shows the high-level overview.

Original (in blue): Vehicle collects images and sends all data back to the cloud for detection and adversarial training, requiring huge bandwidth, which is not practical. According to data from Tesla [38], the Full Self-Driving chip (FSD) can process 2300 frames from eight cameras on the vehicle. Assume that the traffic sign images per frame are 5. Then the images to be uploaded to the cloud are 11500 images per second. With an image size of 5Kb, the requirement of bandwidth for image uploading is 56.15 Mbps. In the meantime, the speed of widely used LTE is 10Mbps. Although the network bandwidth continues to increase, *e.g.*, 10Gbps with 5G technology, this is generally offset by the corresponding increase in image resolution. Using the speed of image collection and network bandwidth, we observe that uploading all images to the cloud can not be realized under real-time requirements. [1]

Boosted (in red): Here, detection is moved to the edge with GERALT architecture, and only the attack images are uploaded for Adversarial Training, which reduces most parts of the image upload and shortens the gap between the attack and model update. The total computation will be reduced without sacrificing any accuracy. To illustrate the impact of this approach, consider a scenario where one attack image is present for every ten images. In such a case, the bandwidth requirement is reduced by a factor of ten compared to the previous setup where all images were uploaded. The reduction of computation resources is shown in Section V and indicates that human-driving vehicles (HV) can also be equipped with detection hardware to provide enough attack images for training together with self-driving vehicles (SV).

[1]In practice, the uploading of self-driving data is typically performed using the user's home internet outside of the operational time.

IV. GERALT OVERVIEW

Real-time detection using neural networks poses a significant challenge due to the added computational overhead. In a conventional implementation, the additional detection neural network requires approximately twice the computation of normal inference. GERALT offers a solution to this issue through a combination of adjusting model structure and accelerating inter-network inference. Fig 4 outlines the flow of our design. The design flow can be divided into two main components: partition analysis (GERALT-part) and architecture details (GERALT-arch). In our design, the actual computation is determined by the connection point between networks. Therefore, before going deep into the architecture details, we will explain the partition analysis of the inter-network system. Details will be discussed later in subsections.

A. GERALT-*part: Optimizing the Detection Network*

To design the architecture with inter-network optimization, it is crucial to analyze and select the most efficient connecting method. GERALT-part aims to optimize the detection of adversarial examples by separating the computation required for detecting them from the classification process. It enables the detection of adversarial examples while the intermediate data from the detection neural network is transmitted to the classification network for further computation. This approach reduces overhead and enables efficient processing of data. To reuse the intermediate data, several factors need to be determined through partition analysis. This includes identifying which results should be used for subsequent computations and determining which layer should accept the intermediate data. Correspondingly, GERALT is guided by extensive experiments targeted to identify optimal settings of different parameters.

Using this strategy includes a number of challenges. The first issue arises from the fact that when employing a separate neural network for detection, the size of the intermediate feature map may not be appropriate for loading into the classification network for inference directly. Additionally, the number of channels can vary between layers even when both networks use the same-sized feature map. Finally, it can be challenging to identify the layer that has the most pertinent data for classification. The processes in GERALT-part involve overcoming these difficulties and designing an inter-network structure that performs similarly to the original model.

Gathering Baseline Accuracy:

A common neural network is chosen as the benchmark for classification accuracy in order to assess the efficacy of the

© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

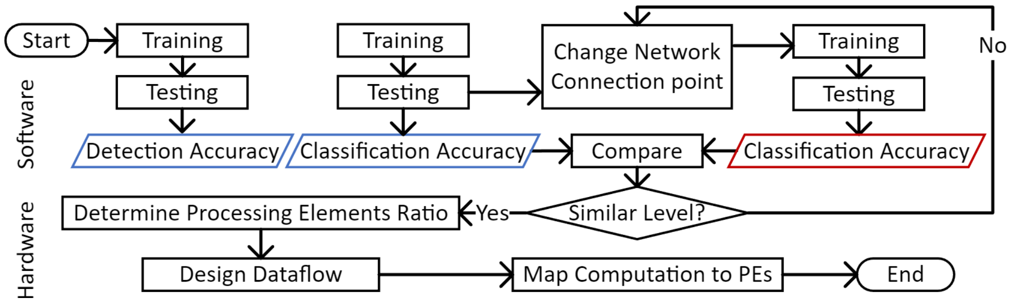

Fig. 4. GERALT Design Flow

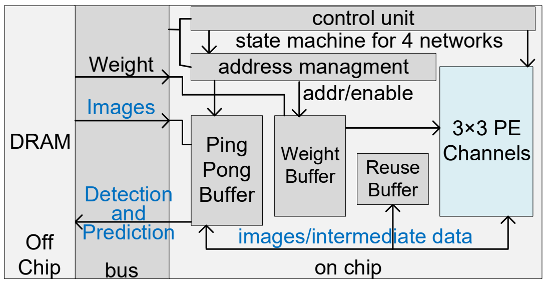

Fig. 5. GERALT-arch Overview

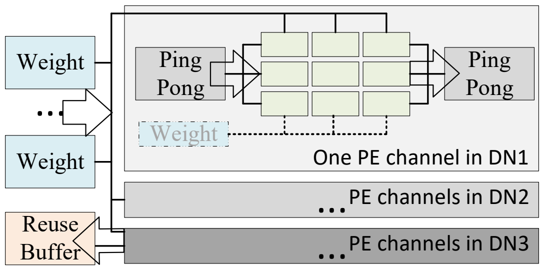

Fig. 6. GERALT Storage: There are only 3 groups of Ping-pong buffers of DN in this figure. CN's Ping-Pong buffer is not shown.

proposed GERALT technique. Additionally, a different smaller neural network will act as the baseline detection network. To produce adversarial examples, noises are then applied to the dataset. The efficiency of the detection network is assessed using both the original and the adversarial images to ascertain the fundamental detection accuracy.

Investigating the Optimal Inter-Network Inference Structure: To connect the two neural networks and optimize the proposed approach, two options are employed. The results of these options are compared to determine the best configuration of GERALT-part, as outlined in Section V.

1) To ensure that the classification model can efficiently process intermediate feature maps from different layers, the detection model's structure is modified in terms of dimensions. This modification ensures that the classification model receives feature maps of an appropriate size from various layers.
2) We only scale the size of feature maps that are sent to the classification model. This step is crucial for reducing overhead and optimizing the inter-network structure but may cause an accuracy drop. By transmitting only the resized feature maps, the classification model can efficiently process the data without the need for additional computation or data transfer.

B. GERALT-*arch: Accelerating Inter-Network Inference*

While GERALT-part optimizes the evasion attack detection calculation using a smaller detection network, current accelerators do not support the corresponding computation. This problem is solved by GERALT-arch, which offers an inter-network design for reusing intermediate data. This design, shown in Fig. 5, makes use of three important observations. First of all, unlike neural network training with back-propagation and weight updates, the inference phase does not require the storing of intermediate data, which reduces the need for off-chip storage and bandwidth. As a result, DRAM can be smaller and use less energy. Second, the threshold of on-chip delay rises when image loading gets more frequent and computation time for one image is relatively low. Finally, additional "write" operations from the detection network, rather than just "read" operations from the classification network, are included in the reuse of intermediate data. Therefore, pipelining the computation of detection and classification networks would result in the sequential execution of various operations within the same memory space. The GERALT-arch framework offers a practical design that permits the reuse of intermediate feature maps between the detection and classification networks in order to address these issues and optimize inter-network data transfer. This design minimizes the requirement for additional data transfer and storage, improving energy efficiency and lowering overhead.

1) Storage Pattern: In light of the aforementioned design decisions, we put together the storage architecture for GERALT-arch in Fig. 6. The computing results are stored in four groups of ping-pong buffers, three for detection and one for classification. All kernel weights are stored in two groups of weight buffers, one of which is shared by three smaller detection networks and the other by the classification network. In order to facilitate the pipeline between networks, a collection of specialized reuse buffers is installed to store the intermediate data of a particular layer.

2) Inter-Network Interface Dataflow: The changes made to the data flow in GERALT-arch are shown in Fig. 7. Weight stationary is used as the standard flow type. Adder Trees

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

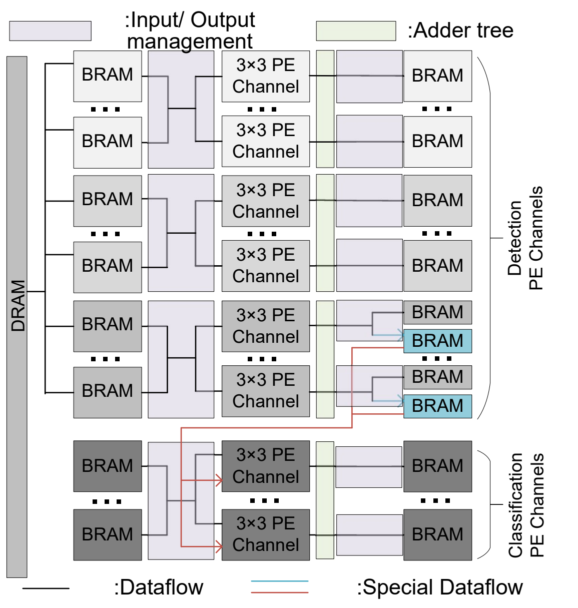

Fig. 7. GERALT Dataflow

are utilized to collect the results from the 3×3 PE channel. The input/output management module handles the pattern of loading and storing the intermediate data. Three detection networks receive the image from off-chip memory, put it into their ping-pong buffers, and then do inference on the front layers until they reach the connection point. To avoid idleness, the connection point's outputs are saved in the reuse buffer at the same speed as the ping-pong buffer. As seen in Fig. 8, the detection networks then move on to the next connection point for a new input and provide data to construct the pipeline. The calculation time for classification should be the same as that for detection for the best latency. As a result, we establish the PE channel ratio to be 63:1.

3) Principle of Reusing Intermediate Data in Hardware Level: Based on the discussions above, we expand the design idea to a principle of reusing intermediate data between any two Neural Networks. There are two factors to account for when designing an inter-network inference accelerator: (1) the ratio of processing elements for two networks to keep the pipeline working; and (2) the reasonable gap period to avoid data conflict on the connection point.

The following formula is used to calculate the ratio of computation resources to balance the pipeline:

$$R_g = \frac{(C + \sum_{i=1}^{m} I_i \times O_i \times (K_i)^2) \div U_1}{\sum_{i=c_2}^{n} I_i \times O_i \times (K_i)^2 \div U_2}$$

Here U_1 and U_2 is the utilization of different architecture used to process the computation of two networks; I_i,O_i,K_i are the feature map dimensions and used to calculate the computation of layer i; m,n is the total layers of two network; c_1,c_2 are the connection points, c_1 is not shown because it only affect the data to be reused; C means the additional computation for special operations, like the preprocessing/squeezing of input images in our example.

To avoid data conflict, the start point of classification in the control flow should be later than the storage of the final results of the Connection point – SQZ, which varies in different architectures. The following formulas indicate the components of the gap before the start of classification computation:

$$G_g = \frac{C_{pl} + C_{ir}}{U_1 \times R_1}$$

Here C_{pl} is the computation required for finishing the computation of previous layers, and C_{ir} means the computation required for finishing part of the connection point's results required by classification and storing them to the reuse buffer. R_1 represents the allocated resources for the detection neural networks. Normally the C_{ir} can be ignored because it requires a dual-port BRAM with higher energy consumption. Note that although the defined gap can be reduced by increasing utilization or PE numbers, it cannot be used to determine the exact computation resources.

V. GERALT EVALUATION

A. *Evaluation of* GERALT*-part*

In the experiments, we use AlexNet [20] and ResNet-50 [12] as the classification network. They are trained with a traffic sign dataset including 51,839 images in 43 classes [33]. Besides, for evasion attack detection, we choose SqueezeNet [14] and ResNet-18 with smaller sizes. The models are trained under the learning rate of 0.01 with the batch size of 16. The loss function is CrossEntropyLoss and the optimizer is SGD. To get adversarial examples, we use Deepfool [26] attack with the knowledge of the target neural network's structure.

The experimental results from different configurations are presented in Table I and Table II. We have selected the configuration settings aiming at achieving maximum classification accuracy. In the configuration setting, the names "Connection point – SQZ" and "Connection point – AN" configurations imply that the corresponding layer's outputs in SqueezeNet will be reused as intermediate data, to be the input of another specific layer in AlexNet. More precisely, we build a classification inter-network system that performs the computation before "Connection point – SQZ" and after "Connection point – AN". Similarly, the "Connection point – 18" and "Connection point – 50" configurations are employed in experiments involving ResNet. In addition, the methods of "Resize" and "Structure" are employed to adjust the network sizes in accordance with the techniques outlined in Section IV-A. Furthermore, experiments are constrained to layers with equivalent channel numbers.

Remark 1: Note that the configuration of the connection points only affects the accuracy of the classification and does not charge any computation of detection. Our work primarily focuses on *accelerating* the detection process and the ability to counter evasion attacks. The effectiveness of the adversarial training itself is not the focus of this work, since it has been comprehensively covered in previous works [2], [45].

Table I presents the accuracy of Alexnet connected with SqueezeNet. For the "Resize" method, the best accuracy drops 13% but we get some significant observations here. Our first observation is that the information contained in "Connection

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

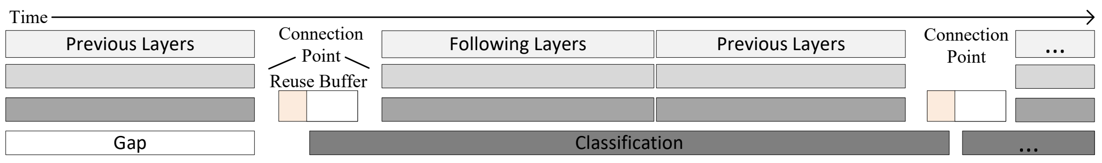

Fig. 8. GERALT Pipeline

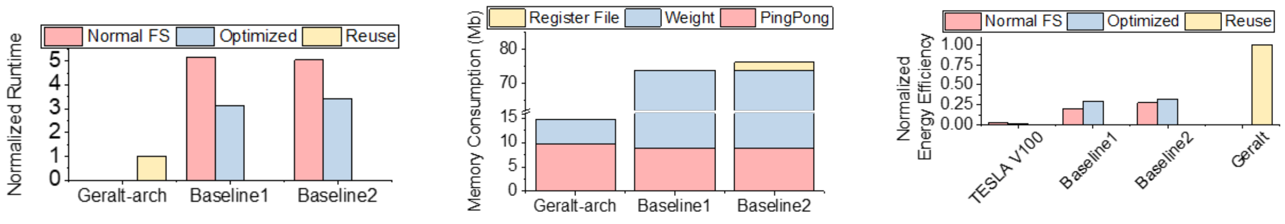

Fig. 9. GERALT-arch Evaluation. (a) Normalized Run-time; (b) Memory Consumption; (c) Energy Efficiency

point-SQZ", determines the accuracy. Even if we change "Connection point – AN", the accuracy barely varies. Next, the accuracy is affected by the information suitability of "Connection point - AN". The latter layer performs better as the "Connection point – AN", shown in settings 4, 7, and 10. The optimal accuracy is attained by setting 17 which indicates significant pertinent information is imprinted in the "fire5" layer of SqueezeNet while the concluding layers of AlexNet are better equipped to accommodate this data.

The experiments indicate that modifying the network structure yields better outcomes than resizing the feature map directly. This disparity can be attributed to information loss during resizing operations and inconsistencies between connection points. These findings allow for efficient computation optimization between any two CNNs. Accordingly, GERALT employs setting 17 as the functional configuration to guide the hardware design. Additionally, Table II and Table III corroborates the connection point's generalizability.

We have also evaluated the detection accuracy of two detection methods after optimization with GERALT-part. According to our findings, the implementation of optimized Feature Squeezing demonstrates a slightly superior detection accuracy of 0.5% when compared to the traditional method, while maintaining a 94.8% true positive rate. It is worth noting that this marginal improvement is within the expected range of normal fluctuation, as our optimization process does not alter the detection algorithm and computation.

TABLE I
ACCURACY OF ALEXNET CONNECTED WITH SQUEEZENET. THE NAMES "CONNECTION POINT – SQZ" AND "CONNECTION POINT – AN" IMPLY THAT THIS LAYER'S OUTPUTS IN SQUEEZENET WILL BE REUSED AS INTERMEDIATE DATA, TO BE THE INPUT OF A LAYER IN ALEXNET.

Setting	Connection Point-SQZ	Connection Point-AN	Resize	Structure	Accuracy
Origin	N/A	N/A	N/A	N/A	93.63
0	fire6	relu3	Yes	No	62.29
1	fire7	relu3	Yes	No	62.29
2	fire6	conv4	Yes	No	62.29
3	fire7	conv4	Yes	No	62.29
4	fire4	relu4	Yes	No	60
5	pool2	relu4	Yes	No	32.57
6	fire5	relu4	Yes	No	56.57
7	fire4	conv5	Yes	No	60
8	pool2	conv5	Yes	No	32.57
9	fire5	conv5	Yes	No	56.57
10	fire4	relu5	Yes	No	76.57
11	pool2	relu5	Yes	No	32.57
12	fire5	relu5	Yes	No	68.57
13	fire4	relu5	No	Yes	87.43
14	fire5	relu5	No	Yes	89.14
15	fire4	conv5	No	Yes	91.20
16	pool2	conv5	No	Yes	94.23
17	fire5	conv5	No	Yes	94.70

B. Evaluation of GERALT-arch

We use the Xilinx VCU118 evaluation board to implement GERALT-arch which has 75 Mb of BRAM on chip and 20 Gb of DDR4 off-chip memory. Details of the chip and board are shown in Table IV. In total, 64 3×3 PE channels are deployed. Each detection PE channel is equipped with one pair of ping-pong buffers with a size of 0.15Mb. All detection PE channels share 21 weight buffers for 0.24 Mb weights. An optimized weight stationary architecture is used as the first baseline which is also employed in Dandelion [5], designed

TABLE II
ACCURACY OF RESNET-50 CONNECTED WITH RESNET-18

Setting	Connection Point-18	Connection Point-50	Resize	Structure	Accuracy
Origin	N/A	N/A	N/A	N/A	96.00
0	fire5	layer3_1	No	Yes	97.47
1	conv1x	conv3x	No	Yes	96.29
2	conv1x	conv4x	No	Yes	96.37

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

TABLE III
ACCURACY OF VGG-16 CONNECTED WITH SQUEEZENET

Setting	Connection Point-18	Connection Point-50	Resize	Structure	Accuracy
Origin	N/A	N/A	N/A	N/A	96.00
0	fire5	layer3-1	No	Yes	97.47
1	fire5	layer3-2	No	Yes	97.83
2	maxpooling3	layer4-1	No	Yes	96.71
2	maxpooling3	layer4-2	No	Yes	96.71
2	fire9	layer4-1	No	Yes	95.27
2	fire9	layer4-2	No	Yes	95.65

TABLE IV
HARDWARE SPECIFICATION FOR EXPERIMENTAL EVALUATION

Technology	Virtex UltraScale+ XCVU9P-L2FLGA2104
Chip Size (mm)	47.5 × 47.5
Clock Frequency (MHz)	200 (On-chip); 50 (Off-chip)
Voltage (V)	0.825-0.876 (Chip); 12 (Board)
System Logic Cells (K)	2,586
CLB Flip-Flops (K)	2,364
CLB LUTs (K)	1,182
Total Block RAM (Mb)	75.9

to address the significant variation in kernel size while utilizing similar hardware configuration like GERALT-arch. The second baseline is Eyeriss [6], a popular inference accelerator using row-stationary dataflow to reduce under-utilization. The baselines are utilized in two different modes: (1) a straightforward approach that involves running three classification networks sequentially to achieve joint detection and prediction, and (2) a smart implementation of the Feature Squeezing method with optimized computation, but does not reuse intermediate data. The computation resources are constrained to the same level to demonstrate the comparative performance.

The runtime improvement of GERALT-arch is depicted in Fig. 9(a). The efficient implementation of Feature Squeezing benefits from the utilization of a smaller network for detection, which leads to reduced computation. Additionally, the reuse of intermediate data results in further improvement in performance, yielding a speedup of up to 3×. This approach skips the computations of previous layers in the classification process, which consume a significant portion of the baseline's runtime. Memory usage is indicated in Fig. 9(b). The GERALT-arch occupies a slightly larger BRAM space due to its pipeline execution for intermediate data, but a large amount of storage space required for kernel weights of the larger classification network can be avoided. Furthermore, the weight buffer size gap between the GERALT-arch and the 50x larger AlexNet model is significant. Our design achieves more than 4× reduction on memory consumption. Finally, Fig. 9(c) depicts the energy efficiency of each architecture in order to consider their viability on edge devices. Due to the difficulty of reducing GPU computation, we solely evaluate its energy efficiency, which indicates elevated power consumption and under-utilization when executing less dense

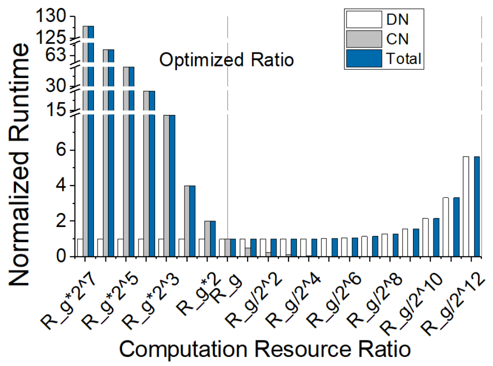

Fig. 10. Evaluation of Computation Resource Ratio

computations. GERALT experiences reduced off-chip DRAM accesses, resulting from the advantageous employment of less and fully-buffered weights. Furthermore, greater efficiency is achieved due to the reduced on-chip BRAM size and minimized register file requirements.

To evaluate the principle of inter-network design, we change the ratio of computation resources around the optimized value from the formula. Fig. 10 indicates how the runtime will be affected by the ratio with fixed computation resources. It is clear that the distance from the optimized ratio decreases the runtime in different ways. The reason is because of the big value of the R_g, meaning that most of the resources are allocated to the detection network to balance the pipeline. This also explains why the runtime of the detection network does not increase significantly at the right side of the reference line.

Note that our proposed method only works on systems using two or more connected neural networks. Furthermore, given the concept of transfer learning mentioned in Section II, the reuse of intermediate data is also limited to convolutional neural networks, *e.g.*, for image recognition. However, we believe that these limitations will not significantly affect the applicability of our method since CNNs are central components for many current applications like autonomous driving; furthermore, these applications typically involve more than one CNN to address the computational demands of the task.

C. Evaluation of System Boosted by GERALT

As discussed in previous sections, GERALT enables real-time detection and can change the current model update pattern of self-driving systems. To evaluate the impact of GERALT on the update of the traffic sign recognition model, we collect necessary information from multiple sources and calculate the time used for retraining the neural network model with reasonable assumptions. Note that the model is not constrained to a specific structure and size. Obviously, details of dataset characteristics and training methodology are proprietary to the industry and not disclosed. Consequently, we develop estimates for different metrics involved indirectly through other available parameters as follows. The computation of model retraining is represented by the required GPU hours on Tesla's cloud server and other necessary information. This evaluation focuses on the speed-up of image uploading and the

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

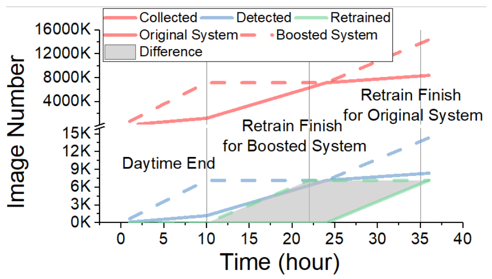

Fig. 11. Finished Image Number in Original or Boosted System

improvement of the model update's delay. The model update is divided into three steps: image collection, image detection, and model retraining. The original system and boosted system are evaluated in these three stages.

Step 1: Recall from Section III that the currently used Hardware 3 of the Autopilot system can collect 2300 frames from real-time traffic videos per second and requires a bandwidth of 56Mbps for real-time uploading. Only in the original system, the uploading of images to the cloud will be limited by the bandwidth of the current connection on the vehicle or at home. The average driving time in the US is 52 minutes every day [34]. An assumption is made that the driving behavior happens mostly in the daytime which is about 10 hours a day and most of the images are collected during the daytime evenly.

Step 2: In the original system, the uploaded images will be processed on the cloud in the original system to detect attack images. Since the computation resources on the cloud are abundant and the pipeline is utilized to reduce latency, the delay is ignored for detection. For the boosted system, the detection is performed together with classification and can be considered with no extra computation delay. We assume that there is 1 attack image among 1000 images and the bandwidth requirement becomes 57Kbps meaning that the uploading delay can be ignored as well.

Step 3: Attack images are used for adversarial training on the cloud in both systems. The training of 48 networks takes 70000 GPU hours to finish [35]. With 5760 GPUs in Tesla's AI Training center, it takes 12.15 hours to retrain only one network with 120 GPUs.

Fig. 11 compares the number of retrained images between the Original and Boosted systems. Since the image collection in the original system is affected by the bandwidth limit, the time used to collect the same number of attack images and retrain the model is delayed for 14 hours with the upload speed of 0.57Mb per second using Tesla's Wall Connector during charging. This will significantly slow down the reaction to new attacks.

VI. Related Work

There has been significant research on detecting and defending against evasion attacks. One notable approach is SafetyNet [23], which identifies abnormal Relu results to reject inputs that have been attacked. Carrara *et al.* [4] employ LSTM training to learn how to transfer input representations that map to different areas for attacked images. To improve the adaptability of detecting accuracy, a detector neural network [25] is trained together with a larger network. Debicha *et al.*, developed a transfer learning-based adversarial detector comprised of multiple detectors which showed enhanced detectability of adversarial traffic compared to a single detector [7]. In [11], Guo *et al.*, came up with the very first hardware accelerator based on memristor crossbar arrays for adversarial attacks which significantly improves the throughput of a visual adversarial perturbation system along with the robustness and security of future deep learning systems. Cao *et al.*, proposed region-based classification to ensemble information in a hypercube centered to predict the correct label which significantly mitigates evasion attacks without sacrificing classification accuracy [3]. Herath *et al.*, [13] proposed LAM (Log Anomaly Mask) to perturb streaming logs with minimal modifications to design it as a reinforcement learning agent that operates in a partially observable environment to predict the best perturbation action and it significantly reduces the accuracy of the model. A fault sneaking attack on DNNs is developed by Hu *et al.*, [44] where the adversary aims to misclassify certain input images into any target labels by modifying the DNN parameters and it can inject multiple sneaking faults without losing the overall test accuracy performance.

For model updates in edge intelligence, there has been work on minimizing the data transferred and reducing the size of the model. Song *et al.* [32] shows how to use the unsupervised learning method to distinguish useful data from big raw data to save the bandwidth requirement. Correspondingly, compression and quantization of machine learning models have been successfully used for minimizing model size [40] [30]. However, these methods only focus on the condition of one neural network and are orthogonal to our proposed design.

Numerous accelerator designs have been proposed to optimize the performance of deep learning applications. The Equinox architecture, as described in [8], enables interleaved training during idle inferences, which enhances the overall utilization of the system. PUMA [1] is an energy-efficient accelerator designed to support scalable computation of various neural network models using advanced memristor technology. GCONV [43] utilizes computation mapping techniques to enable efficient inference or training accelerator design by mapping different computations to the same pattern. Unlike GERALT, which leverages inter-network computation, the approaches mentioned, these approaches(together with GPU [28], TPU [18] and other architectures [15], [16], [19]) do not utilize the opportunity in inter-network computation.

VII. Conclusion

GERALT is proposed in this paper to optimize the computation of evasion attack detection and accelerate inter-network inference. First, the detection method is analyzed to partition the computation and connect two neural networks for the following architecture design in GERALT-part. Then GERALT-arch provides necessary architecture support

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

to enable the reuse of intermediate data for inter-network acceleration. What's more, the design of inter-network accelerators is extended to a general principle. It is proved by our experiments that GERALT improves the performance of inter-network inference while obtaining high energy efficiency using less memory. The period of model update can also be shortened without bandwidth limitation.

In future work, we will extend GERALT for more functions like enabling efficient evasion attack defense, *e.g.*, denoising and generative adversarial network (GAN).

REFERENCES

[1] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W.-m. W. Hwu, J. P. Strachan, K. Roy, and D. S. Milojicic. Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 715–731, New York, NY, USA, 2019. Association for Computing Machinery.

[2] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

[3] X. Cao and N. Z. Gong. Mitigating evasion attacks to deep neural networks via region-based classification, 2019.

[4] F. Carrara, R. Becarelli, R. Caldelli, F. Falchi, and G. Amato. Adversarial examples detection in features distance spaces. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[5] X. Chen, J. Zhang, and S. Ray. Dandelion: Boosting dnn usability under dataset scarcity. *IEEE Transactions on Computers*, pages 1–1, 2021.

[6] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017.

[7] I. Debicha, R. Bauwens, T. Debatty, J.-M. Dricot, T. Kenaza, and W. Mees. Tad: Transfer learning-based multi-adversarial detection of evasion attacks against network intrusion detection systems. *Future Generation Computer Systems*, 138:185–197, 2023.

[8] M. Drumond, L. Coulon, A. Pourhabibi, A. C. Yüzügüler, B. Falsafi, and M. Jaggi. Equinox: Training (for free) on a custom inference accelerator. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 421–433, New York, NY, USA, 2021. Association for Computing Machinery.

[9] V. S. Gireesh Chamarthi, X. Chen, B. B. Yedla Ravi, and S. Ray. Exploration of machine learning attacks in automotive systems using physical and mixed reality platforms. In *2023 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4, 2023.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] H. Guo, L. Peng, J. Zhang, F. Qi, and L. Duan. Hardware accelerator for adversarial attacks on deep learning neural networks. In *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*. IEEE, oct 2019.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] J. D. Herath, P. Yang, and G. Yan. Real-time evasion attacks against deep learning-based anomaly detection from distributed system logs. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, CODASPY '21, page 29–40, New York, NY, USA, 2021. Association for Computing Machinery.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[15] H. Jia, M. Ozatay, Y. Tang, H. Valavi, R. Pathak, J. Lee, and N. Verma. 15.1 a programmable neural-network inference accelerator based on scalable in-memory computing. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, volume 64, pages 236–238, 2021.

[16] J. Jo, S. Cha, D. Rho, and I.-C. Park. Dsip: A scalable inference accelerator for convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 53(2):605–618, 2018.

[17] N. P. Jouppi, C. Young, and Patil. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, page 1–12, New York, NY, USA, 2017. Association for Computing Machinery.

[18] N. P. Jouppi, C. Young, and Patil. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2):1–12, jun 2017.

[19] J. H. Kim, B. Grady, R. Lian, J. Brothers, and J. H. Anderson. Fpga-based cnn inference accelerator synthesized from multi-threaded c software. In *2017 30th IEEE International System-on-Chip Conference (SOCC)*, pages 268–273, 2017.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[21] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[22] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[23] J. Lu, T. Issaranon, and D. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[24] Mangodoc. Visualizing the output of each cnn layers. https://www.imangodoc.com/82091153.html. Accessed June 2022.

[25] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[26] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[27] J. Nickolls and W. J. Dally. The gpu computing era. *IEEE Micro*, 30(2):56–69, 2010.

[28] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[29] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[30] B. Rokh, A. Azarpeyvand, and A. Khanteymoori. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Transactions on Intelligent Systems and Technology*, sep 2023.

[31] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[32] M. Song, K. Zhong, J. Zhang, Y. Hu, D. Liu, W. Zhang, J. Wang, and T. Li. In-situ ai: Towards autonomous and incremental deep learning for iot systems. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 92–103, 2018.

[33] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[34] A. D. Survey. Research Brief. https://aaafoundation.org/wp-content/uploads/2018/01/19-0226_AAAFTS-2018-ADAS-Research-Brief-Update_v1.pdf.

[35] Tesla. Ai & robotics. https://www.tesla.com/AI. Accessed March 2023.

[36] F. Tramer and D. Boneh. Adversarial training and robustness for multiple perturbations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[37] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[38] Wikipedia. Tesla autopilot. https://en.wikipedia.org/wiki/Tesla_Autopilot. Accessed March 2023.

[39] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[40] C. Xu and J. McAuley. A survey on model compression and acceleration for pretrained language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):10566–10575, Jun. 2023.

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3324568

[41] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[42] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[43] J. Zhang, X. Chen, and S. Ray. Gconv chain: Optimizing the whole-life cost in end-to-end cnn acceleration. *IEEE Transactions on Computers*, pages 1–1, 2021.

[44] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin. Fault sneaking attack: A stealthy framework for misleading deep neural networks. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, New York, NY, USA, 2019. Association for Computing Machinery.

[45] W. Zhao, S. Alwidian, and Q. H. Mahmoud. Adversarial training methods for deep learning: A systematic review. *Algorithms*, 15(8), 2022.

Xiangru Chen received a B.S. degree in Electronic Information Engineering from Shandong University. After graduating with a Master's degree in Electrical and Computer Engineering from the University of Florida, he went on to complete a Ph.D. degree in the same field at the University of Florida in 2023. His research focuses on exploring ways to increase the usability of edge intelligence systems. Through extensive research and experimentation, he concentrated on designing machine accelerators that could enhance the efficiency and security of such systems.

Dipal Halder is a second year PhD student at the RISING Lab in University of Florida. Dipal's research topic includes System-on-Chip (SoC) design and hardware security. He has received master's degree in Computer Aided Engineering from Eastern Michigan University and actively engaged in an autonomous vehicle project. Furthermore, he has worked as a system engineer intern in R&D section of Groupe SEB in France. He also received an advance master's degrees in Embedded Systems from Grenoble INP, ESISAR, France as a part of full scholarship provided by IDEX, University of Grenoble Alpes. Dipal's bachelor was in Electrical and Electronic Engineering that he completed from Khulna University of Engineering and Technology, Bangladesh.

Kazi Mejbaul Islam has completed his B.Sc in Electrical and Electronic Engineering from Ahsanullah University of Science and Technology and M.Sc in Computer Science from Florida State University. Currently he is pursuing a Ph.D. in Electrical and Computer Engineering at University of Florida. His research area is security of multi-tenant devices.

Sandip Ray is an Endowed IoT Term Professor at the Department of Electrical and Computer Engineering, University of Florida. His research involves developing correct, dependable, secure, and trustworthy com- puting through cooperation of specification, synthesis, architecture and validation technologies. He focuses on next generation computing applications, including IoT, autonomous automotive systems, etc. Before joining University of Florida, he was a Senior Principal Engineer at NXP Semiconductors, where he led the R&D on security architecture and validation of hardware platforms for automotive and IoT applications. Prior to that, he was a Research Scientist at Intel Strategic CAD Labs, where he led research on validation technologies for security and functional correctness of SoC designs. Dr. Ray is the author of three books and over 90 publications in international journals and conferences. He has a Ph.D. from University of Texas at Austin and is a Senior Member of IEEE.

Authorized licensed use limited to: University of Florida. Downloaded on February 08,2024 at 15:54:49 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.