SEALing Neural Network Models in Encrypted Deep Learning Accelerators

Pengfei Zuo[*†], Yu Hua[*✉], Ling Liang[†], Xinfeng Xie[†], Xing Hu[‡], Yuan Xie[†]
[*]Huazhong University of Science and Technology
[†]University of California, Santa Barbara
[‡]SKL of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences

Abstract—**Deep learning (DL) accelerators suffer from a new security problem, i.e., being vulnerable to physical access based attacks. An adversary can easily obtain the entire neural network (NN) model by physically snooping the memory bus that connects the accelerator chip with DRAM memory. Therefore, memory encryption becomes important for DL accelerators to improve their security. Nevertheless, we observe that traditional memory encryption techniques that have been efficiently used in CPU systems cause significant performance degradation when directly used in DL accelerators, due to the big bandwidth gap between the memory bus and the encryption engine. To address this problem, our paper proposes SEAL, a Secure and Efficient Accelerator scheme for deep Learning to enhance the performance of encrypted DL accelerators by improving the data access bandwidth. Specifically, SEAL leverages a criticality-aware smart encryption scheme that identifies partial data having no impact on the security of NN models and allows them to bypass the encryption engine, thus reducing the amount of data to be encrypted without affecting security. Extensive experimental results demonstrate that, compared with existing memory encryption techniques, SEAL achieves $1.34-1.4\times$ overall performance improvement.**

I. Introduction

With the increase of computing performance and storage capacity of edge devices, DL systems are increasingly expanded and used from cloud to edge devices, such as self-driving cars and Internet-of-things devices. By employing DL accelerators, e.g., GPU and NPU, edge devices are able to carry out real-time local inferences based on current environments without a connection with a remote control center with high latency.

In DL accelerators, neural network (NN) models are confidential information. Because NN models represent the Intellectual Property (IP) of model owners, which should be protected to preserve their competitive advantages. More importantly, the knowledge of NN models can facilitate an adversary to carry out more powerful adversarial attacks [5]. In adversarial attacks, an adversary is able to intentionally affect the outcome of the DL inference by modifying the input data with a slight perturbation that is imperceptible to humans. In general, if the adversary does not know NN models, the success rate of the adversarial attack is low. With the knowledge of NN models, the success rate is significantly improved [4].

However, DL accelerators deployed on edge devices are easier to be physically accessed, thus being vulnerable to physical access based attacks. The accelerator chip and DRAM themselves are usually well packaged and hence secure to physical access, but the memory bus connecting accelerator and DRAM is not secure, due to being vulnerable to bus snooping attacks [9], [10], [24]. Since the DL accelerator has to access the NN model stored in the DRAM memory through the memory bus during the inference, an adversary can easily obtain the entire NN model by inserting a bus snooper on the memory bus to intercept the data communicated between the DL accelerator chip and DRAM. Therefore, memory encryption for encrypting the data transmission through the memory bus is important.

Although there already exist mature memory encryption techniques successfully used in secure CPU systems, we observe employing them in DL accelerators significantly decreases the performance. The overall performance of DL accelerators is reduced by over 50% after using memory encryption, as evaluated in Section II-B. Such a significant performance decrease is unacceptable for the latency-sensitive DL accelerators on edge devices that must carry out real-time inferences based on current environments, e.g., self-driving cars. We obtain the insight that its main reason comes from the big bandwidth gap between the memory bus of DL accelerators and the encryption engine. For DL accelerators, e.g., GPUs, their performance is highly bandwidth-bounded and hence they generally use the high-bandwidth memory, e.g., GDDR. The bandwidth of their memory bus is generally higher than 160GB/s [18]. However, the state-of-the-art encryption engine with hardware implementation achieves only about 8GB/s of bandwidth on average [14], [15], [21]. Even though we deploy one encryption engine in every memory controller (six encryption engines totally), the big bandwidth gap remains. As a result, the high bandwidth of the GDDR memory bus is under-utilized and the encryption engine becomes the bandwidth bottleneck in secure DL accelerators.

To address these problems, our paper proposes SEAL, a Secure and Efficient Accelerator scheme for deep Learning to enhance the security of DL accelerators on edge devices while delivering a high performance. SEAL reduces the performance overhead of encryption by using a criticality-aware smart encryption (SE) scheme. Specifically, the SE scheme identifies partial data having no impact on the security of NN models and allow them to bypass the encryption engine, lowering the amount of data to be encrypted without any loss of security. The idea of the SE scheme is to measure the relative importance of weight parameters in the NN model. Based on the relative importance, the SE scheme does not encrypt these weight parameters with the lowest importance, and thus it is unnecessary to encrypt their corresponding channels in the input or output feature maps. Based on the quantitative security evaluation in terms of both IP protection and adversarial attacks [5], we determine the percentage of encrypted data with which the SE scheme achieves the same security level as full encryption.

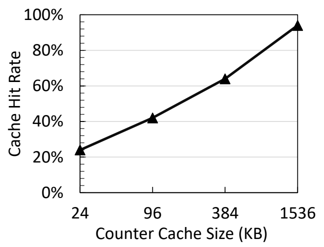
We have implemented SEAL in GPGPU-Sim [1] and evaluated it using three classical CNN models including VGG-16 [22], ResNet-18 [8], and ResNet-34 [8]. Experimental results show that, compared with existing memory encryption techniques, SEAL achieves $1.34-1.4\times$ overall performance improvement and reduces the inference latency by $26\%-28\%$.

II. Background and Motivation

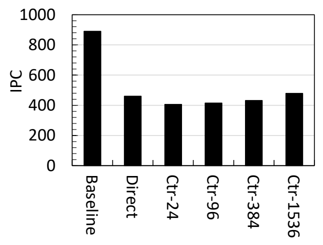
A. Threat Model and Purposes

A generic hardware architecture for DL accelerators (GPU, FPGA, and ASIC) consists of an array of processing elements (PEs, or called

[✉]Corresponding author: Yu Hua (csyhua@hust.edu.cn)



(a) Instruction per cycle (IPC)



(b) Counter cache hit rate

Fig. 1: The IPCs of GPUs with two straightforward memory encryption solutions.

cores in GPUs) and a data cache (or called global buffer) on chip. As the size of the on-chip data cache is limited, the entire NN model and the intermediate data produced during DL inference are stored in the off-chip DRAM memory with large capacity. The accelerator accesses the DRAM through a high-bandwidth memory bus.

Threat Model: Like existing threat models for hardware attacks on CPUs [24] and accelerators [9], [10], we consider on-chip components of accelerators and DRAM are secure. However, an adversary can insert a bus snooper or a memory scanner on the memory bus to obtain the data communicated between the accelerator chip and off-chip DRAM and further steals the entire NN model [9], [10].

Threat Purposes: We consider the following threat purposes that an adversary obtains NN models via bus snooping.

1) IP Stealing. NN models are considered as the IP of model owners [10]. Model owners may consume a large amount of financial and material resources to train a sophisticated NN model. The adversary may be a business competitor of model owners. The leakage of NN models incurs the property loss of model owners and reduces their competitive advantages.

2) Adversarial Attacks. The exposion of an NN model can significantly increase the risk that the NN model is attacked by adversarial attacks. In adversarial attacks, an adversary aims to apply an imperceptible non-random perturbation on the input data to change the prediction results of NN models [5]. The perturbed input data are termed as adversarial examples. If the adversary does not know the NN model, the adversarial attack is called *black-box* attack. If the adversary knows the entire NN model, the adversarial attack is called *white-box* attack. In the black-box attacks, the attack success rate is low. In the white-box attacks, the attack success rate significantly increases since the adversary can generate high-quality adversarial examples by using the known model information [4].

In order to protect the NN models in DL accelerators from bus snooping attacks, encrypting the data transmitted through the memory bus is important. Existing memory encryption techniques including direct encryption and counter mode encryption [24] are widely used in secure CPU systems to enable secure data transmission through the DDR bus of CPU memory. However, data security on the high-bandwidth memory bus for DL accelerators are rarely touched by existing work.

B. Straightforward Solutions for Securing DL Accelerators

We consider two straightforward solutions, i.e., simply employing existing direct encryption and counter mode encryption techniques in DL accelerators, to improve the security of NN models. Without loss of generality, in the rest of this paper, we analyze GPU as a representative example of DL accelerators. However, the problems, insights, and solutions that we develop are also applicable to other DL accelerators.

TABLE I: Performance comparisons of different AES encryption engine implementations (counter mode).

	Area (mm^2)	Power (mW)	Latency (cycle)	Throughput (GB/s)
Morioka et al. [16]	N/A	1920	10	1.5
Mathew et al. [15]	1.1	125	20	6.6
Ensilica [3]	1.4	N/A	11	8
Sayilar et al. [21]	6.3	6207	20	16
Liu et al. [14]	6.6	1580	152	19

We implement the two straightforward solutions in GPGPU-Sim [1]. Since the encryption engine increases the chip area and energy overhead that also affects the chip cooling [19], each memory controller generally includes one encryption engine [24]. Thus the six memory controllers in the modeled GPU include six encryption engines. For the counter mode encryption, we add an on-chip counter cache to buffer recently used counters. We use the encrypted GPUs to execute matrix multiplication computation that is the most common operation in DL algorithms. We evaluate the IPCs of GPUs with different encryption schemes and compare them with a baseline GPU without using memory encryption (`Baseline`), as shown in Figure 1.

First, we observe encrypted GPUs are significantly less efficient than the unencrypted one. Memory encryption decreases the GPU IPC by $45\% - 54\%$ for the matrix multiplication computation. Second, using counter mode encryption does not deliver higher performance compared to using direct encryption on GPU, due to incurring extra memory accesses from counters.

The reason why memory encryption significantly reduces the GPU performance is the big bandwidth gap between the GDDR memory bus and the encryption engine. In CPU systems, memory encryption works well [24], since the AES encryption engine has a similar bandwidth to the DDR memory bus of CPU. However, in GPU systems, the GDDR memory is designed for GPUs to achieve high memory access bandwidth, whose bus bandwidth is generally more than 160GB/s [18]. Moreover, the state-of-the-art pipelined AES engine with hardware implementation achieves only about 8GB/s of bandwidth on average [15]. Even though we deploy one encryption engine in every memory controller, the total encryption bandwidth is 48 GB/s. As a result, the high bandwidth of the memory bus is under-utilized and the AES engine becomes the bandwidth bottleneck in secure GPUs.

A single AES engine usually occupies over 1 mm^2 on-die area and has hundreds or thousands of mW power, as shown in Table I. As resources on the microprocessor die are very scarce, it is ruinously costly to integrate more encryption engines into memory controllers on the GPU die [6]. Even though a GPU/CPU die usually has an area of $90 - 600$ mm^2, most area is occupied by cores and on-die memory and only less than 10% area is left to memory controllers. This is also the reason why Intel carefully designs the AES hardware implementation to reduce area and energy overheads for SGX [6]. Like the design principle of Intel's SGX [6] and many previous works [24], the goal of this paper is also to improve the hardware security while having low on-die overheads.

III. The SEAL Design

A. Criticality-aware Smart Encryption

SEAL leverages a criticality-aware smart encryption (SE) scheme to reduce the amount of encrypted data while improving the NN model security. The SE scheme quantitatively measures the relative importance of weight parameters in each layer by calculating the sum of their absolute weights, i.e., ℓ_1-norm. The weight parameters with the smallest absolute values in each layer are considered to be least important and hence are not encrypted. Thus it is unnecessary to encrypt the corresponding channels in the input or output feature maps of unencrypted weight parameters. As a result, the amount of data to be encrypted is significantly reduced. The percentage of un-encrypted weight parameters is determined based on the quantitative security evaluation in Section III-B to obtain maximum performance benefit and highest security level.

In deep neural networks, we consider using the SE scheme in the convolution (CONV) layers since most layers in a CNN model are CONV layers, e.g., 13/16 for VGG-16, 17/18 for ResNet-18, and 33/34 for ResNet-34. The computation process of a CONV layer is shown in Figure 2. Weight parameters in a CONV layer are organized as a convolutional kernel matrix, and each convolutional kernel is a weight matrix, e.g., 3×3. The computation of a CONV layer transforms the input feature maps with the convolutional kernel matrix to the output feature maps. The convolutional kernel matrix has n_x kernel rows and n_y kernel columns. n_x is equal to the number of channels in the input feature maps. Each kernel row in the kernel matrix corresponds to a single input channel in the input feature maps and this input channel does not involve the convolution computation with other kernel rows, as shown in Figure 2. Similarly, n_y is equal to the number of channels in the output feature maps. Each kernel column in the kernel matrix corresponds to a single output channel in the output feature maps.

Relative Importance Measurement. We measure the relative importance of a kernel row in each layer by calculating the sum of its absolute weights, i.e., ℓ_1-norm. The sum of absolute weights in a row also represents the average magnitude of the kernel weights which gives an expectation of the magnitude of the output feature map. Thus kernel rows with smaller sums of absolute weights tend to produce feature maps with weak activations, compared with the other kernel rows in the same layer [13]. Hence, these rows with small absolute-value sums have a lower impact on the output of the entire NN model compared with the rows with large absolute-value sums. Existing work [13] on pruning NN models demonstrate that, even after completely eliminating the convolution computation that uses these weight parameters with small absolute values, the original accuracy of the NN model can be regained by retraining the networks. This observation indicates that these weight parameters with small absolute values are less important to the NN model and thus rarely affect the security of the NN model. We have confirmed this conjecture by performing IP protection and adversarial attack tests as presented in Section III-B, whose results motivate us to propose the smart encryption (SE) scheme to reduce the encryption overhead in DL accelerators by only encrypting the weight parameters with large absolute values.

Smart Encryption. After computing the sum of absolute weights in each row, the SE scheme sorts the kernel rows based on their sums, and then encrypts partial kernel rows with the largest sums. The percentage of the encrypted kernel rows is determined by our quantitative security analysis as shown in Section III-B. However, the encrypted weight parameters in the SE scheme can be easily figured out if the input and output feature maps of this CONV layer are unencrypted. Therefore, for each encrypted row, the SE scheme also encrypts one input channel in the input feature maps corresponding to the encrypted row, since each kernel row corresponds to a single input channel and does not involve the convolution computation with other input channels, as shown in Figure 2. In this way, the encrypted weight parameters cannot be figured out. For example, for the matrix multiplication $Y = X\omega$, the input channel X and the weights ω are encrypted. ω cannot be figured out even though the adversary knows Y. The data in the input channel X is encrypted once being produced by the previous CONV layer. Hence, the plaintext in the encrypted channel X is never exposed

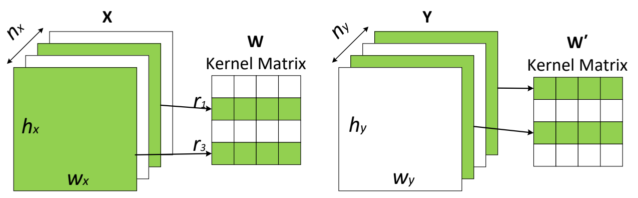


Fig. 2: An example for SEAL encryption. *(Green areas: encrypted data. Each grid in the kernel matrix is a kernel.)*

to the memory bus.

Moreover, when considering unencrypted data among multiple layers, the encrypted channels and weights cannot be figured out and hence also secure. To prove this, we use a simple example with two sequential CONV layers, i.e., $Y = X\omega$ and $Z = Y\omega^{'}$.

$$X = \begin{bmatrix} \boldsymbol{X_0} & X_1 \end{bmatrix}, \omega = \begin{bmatrix} \boldsymbol{\omega_{r0}} \\ \omega_{r1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega_{00}} & \boldsymbol{\omega_{01}} \\ \omega_{10} & \omega_{11} \end{bmatrix}, Y = \begin{bmatrix} Y_0 & \boldsymbol{Y_1} \end{bmatrix},$$
$$\omega^{'} = \begin{bmatrix} \omega^{'}_{r0} \\ \boldsymbol{\omega^{'}_{r1}} \end{bmatrix} = \begin{bmatrix} \omega^{'}_{00} & \omega^{'}_{01} \\ \boldsymbol{\omega^{'}_{10}} & \boldsymbol{\omega^{'}_{11}} \end{bmatrix}, Z = \begin{bmatrix} \boldsymbol{Z_0} & Z_1 \end{bmatrix} \tag{1}$$

The feature maps X, Y, and Z have 2 channels. Since there are 2 input and output channels, kernel matrixes ω and $\omega^{'}$ have 2 rows and 2 columns. With a 50% encryption ratio, we assume the first row ω_{r0} in ω is encrypted, and the second row $\omega^{'}_{r1}$ in $\omega^{'}$ is encrypted. Based on the SE scheme, we should encrypt the first channel X_0 in X and the second channel Y_1 in Y. Moreover, we assume Z_0 is encrypted in Z. Thus for the two sequential CONV layers, we can have the following equations (In Equations 1, 2 and 3, the bold fonts mean encrypted data):

$$\begin{cases} \boldsymbol{X_0} * \boldsymbol{\omega_{00}} + X_1 * \omega_{10} = Y_0 \\ \boldsymbol{X_0} * \boldsymbol{\omega_{01}} + X_1 * \omega_{11} = \boldsymbol{Y_1} \end{cases} \tag{2}$$

$$\begin{cases} Y_0 * \omega^{'}_{00} + \boldsymbol{Y_1} * \boldsymbol{\omega^{'}_{10}} = \boldsymbol{Z_0} \\ Y_0 * \omega^{'}_{01} + \boldsymbol{Y_1} * \boldsymbol{\omega^{'}_{11}} = Z_1 \end{cases} \tag{3}$$

We observe encrypted input channels are never multiplied with unencrypted weight rows, and unencrypted input channels are never multiplied with encrypted weight rows. Thus we can only obtain the product of two encrypted matrixes, e.g., $X_0 * \omega_{00}$, but cannot figure out any single encrypted matrix from Equations 2 and 3. Therefore, the data in encrypted channels and weights are secure even considering data among multiple layers.

In fact, the SE scheme can also be applied to full-connected (FC) layers since each FC layer also includes a kernel matrix like the CONV layer. Therefore, the proposed SE scheme can be applied to other deep neural networks, e.g., recurrent neural networks, that are composed of many FC layers.

To support the proposed SE, we expose a new programming primitive, `emalloc()`, to the high-level program in order to allow programmers to leverage the benefits of SEAL. The memory space allocated by `emalloc()` needs to be encrypted. The memory space allocated by existing `malloc()` in current programming languages does not need to be encrypted.

B. Security Analysis

For the security analysis, we first discuss the case where an adversary does not know what NN architecture is used in the target DL accelerator. In this case, even though some NN model data are obtained by the bus snooping attack, the adversary is difficult to distinguish which data are used for a particular layer. In our proposed SE scheme, some data are encrypted and hence it is more difficult for the adversary to

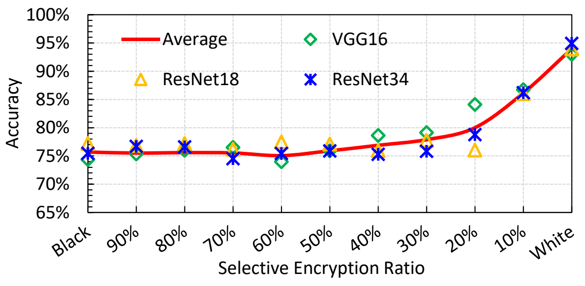


Fig. 3: The inference accuracy of substitute models.

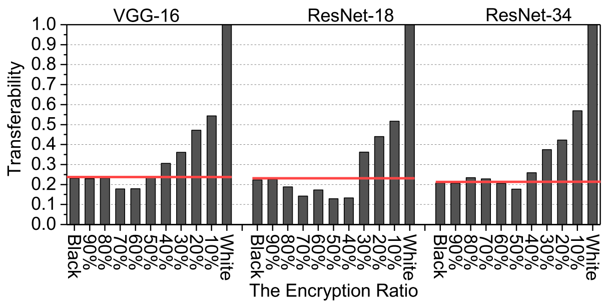


Fig. 4: The transferability of different substitute models.

recover the NN model. Therefore, we consider a strong attack model in which an adversary is able to figure out the NN architecture in the DL accelerator via side channel information [9], [10], e.g., memory access patterns obtained from the memory bus, or device specifications. In this case, the adversary can distinguish the data from different layers and know the locations in the NN model where the encrypted and unencrypted data correspond to. Under the strong attack model, we below present the security analysis. The security of NN models involves two aspects including IP stealing and adversarial attacks, as presented in Section II-A.

1) Substitute Model Generation: In the security evaluation tests, we use three classical CNN models including VGG-16, ResNet-18, and ResNet-34 and train them on the widely used CIFAR-10 dataset [11]. The NN model stored in the target DL accelerator is called *victim model*, and the NN model that the adversary extracts from the accelerator by using bus-snooping attacks is called *substitute model*. Based on the fact that the adversary does not know the training dataset of the victim model, we isolate 90% of training samples (45,000 images) in CIFAR-10 as the training dataset of the victim model [20]. The remaining 10% of training samples (5,000 images) are used by the adversary. Based on the 5,000 images, the adversary uses Jacobian-based dataset augmentation [20] to generate additional 40,000 images and then query them in the target accelerator to obtain their corresponding labels. The generated image-label pairs are used as the training dataset of the adversary's substitute models. The adversary may obtain three kinds of substitute models as follows.

- *White-box model.* If a DL accelerator does not equip memory encryption, the adversary can know the entire victim model including all weight parameters and the NN architecture. Thus we consider an NN model that is the same as the victim model as the white-box substitute model.
- *Black-box model.* If we encrypt all the victim model data and intermediate data, the adversary knows the NN architecture but does not know any weight parameters. However, the adversary can feed his/her own images into the target DL accelerator and obtain the output label. By using the image-label pairs, the adversary is able to retrain an NN model with the same architecture as the victim model. We consider the retrained NN model as the black-box substitute model.
- *SEAL models.* SEAL selectively encrypts partial data that are critical and thus the adversary knows the NN architecture and partial weight parameters that are unencrypted. We perform full encryption on the first two CONV layers, the last one CONV layer, and the last FC layers of a CNN model to prevent the adversary from calculating the weight parameters via input and output layers, and perform the SE scheme on the remaining weight layers. However, by using inputs and outputs of the target DL accelerator, the adversary is able to supplement the unknown part of weight parameters via retraining the NN. Specifically, the adversary initializes an NN model with known weight parameters and fills random numbers following a standard normal distribution for unknown weight parameters [7]. The adversary then keeps the known weight parameters unchanged and fine-tunes unknown weight parameters by retraining the NN using inputs and outputs of the target DL accelerator. Note that the attacker can know the information that the sums of unknown weight rows must be larger than those of known weight rows and then leverage this information during fine-tuning. However, in our experiments, we observe the generated substitute models leveraging the information do not perform better, since limiting the sums of unknown weight rows may destroy efficient parameter fine-tuning.

2) Security on IP Stealing: One of the attack purposes is to steal the IP of NN models. The efficiency of the stolen attacks depends on the inference accuracy of the extracted substitute models. In the stolen attack tests, we first generate the three kinds of substitute models including white-box, black-box, and SEAL models that the adversary may obtain as mentioned above. For SEAL models, we vary the encryption ratio from 90% to 10%. The encryption ratio is defined as the ratio of encrypted weight parameters to all weight parameters in each layer. The encrypted weights have the largest absolute weight values in each layer. We evaluate the inference accuracy of these substitute models using test samples of the victim model.

Figure 3 shows their inference accuracy. We observe that the while-box model has a very high accuracy, i.e., about 94%, due to being the same as the victim model. The black-box model significantly reduces the accuracy from 94% to 75%. This is because the adversary does not know any weights and training samples in the victim model, and the black-box model can only be trained from a blank model by using the adversary's training dataset. For SEAL models, when the encryption ratio is only 20%, the accuracy significantly decreases by 14% on average (from 94% to 80%), since the weight parameters with the largest absolutes are encrypted. When the encryption ratios $\geq$ 40%, the accuracy is almost the same as that of the black-box model. It means the SEAL with a $\geq$ 40% encryption ratio achieves the same security level as the black-box model for IP protection.

3) Security on Adversarial Attacks: If the purpose is to attack the victim model, the adversary can use the extracted NN models to generate adversarial examples and then use the adversarial examples to perform adversarial attacks. In adversarial attacks, the adversary aims to add the minimum perturbation on the input to mislead the victim model to produce a pre-assigned incorrect output [12]. In the adversarial attack tests, we use the three kinds of substitute models including white-box, black-box, and SEAL models to respectively generate 1,000 adversarial examples via the I-FGSM method [12]. Each batch of 1,000 adversarial examples have a 100% attack success rate to attack their corresponding substitute models. We then use these adversarial examples to attack the victim model and evaluate the transferability of adversarial examples. The transferability is defined as the ratio of the adversarial examples that successfully attack the victim

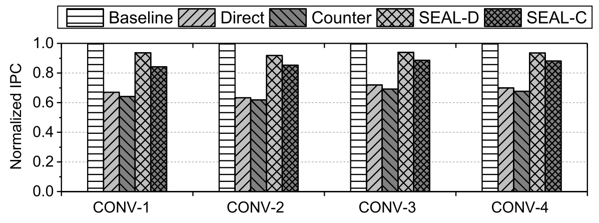


Fig. 5: IPCs normalized to that of Baseline for CONV layers.

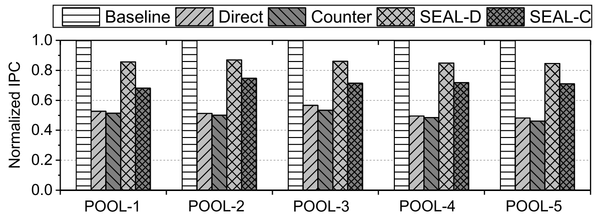


Fig. 6: IPCs normalized to that of Baseline for POOL layers.

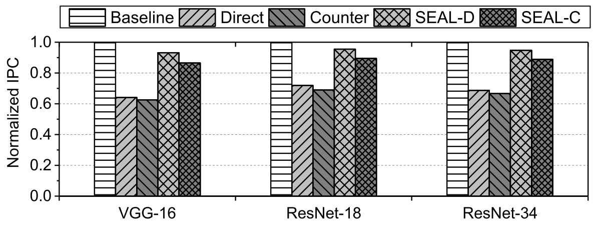


Fig. 7: Overall IPCs normalized to that of Baseline.

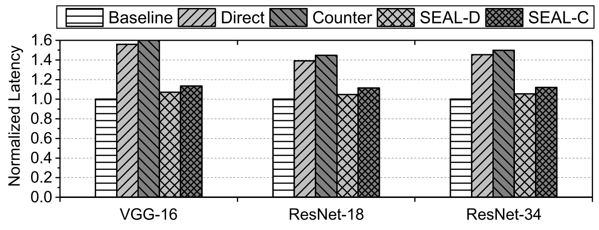


Fig. 8: The inference latency normalized to that of Baseline.

model to all adversarial examples, which is a widely used metric to evaluate the efficiency of substitute models for adversarial attacks [4]. Figure 4 shows the transferability of adversarial examples generated by different substitute models.

We observe that black-box models have much low transferability (about 20%) for the three CNN models compared with while-box models, since the adversary with black-box models does not know any weight parameters and training samples of the victim model. For SEAL models, when the encryption ratios $\geq$ 50% for the three CNN models, the transferability is close to, and even smaller than those of black-box models. The reason is that the unencrypted weight parameters in SEAL are relatively un-important because they have the smallest absolute weights in each layer. If the adversary keeps the unencrypted weight parameters unchanged and fine-tunes the remaining weight parameters, the unchanged, un-important weight parameters may disturb the retrained model, producing smaller attack success rates than the black-box model. When the encryption ratios $<$ 40%, the transferability rapidly increases since some important weight parameters with large absolutes are exposed to the adversary. Based on the above results, we set the encryption ratio of SEAL to 50%, which obtains the maximum performance benefit when achieving the same security level as the black-box models.

IV. Performance Evaluation

A. Methodology

We evaluate the performance of SEAL using the GPGPU-Sim v3.2.2 [1], a cycle-level simulator for contemporary GPUs. We model the microarchitecture for NVIDIA GeForce GTX480 GPU [18] with 15 streaming multiprocessors, one of the default GPUs in GPGPU-Sim. The GPU has a GDDR5 memory bus with 1848 MHz, 384-bit bus bandwidth, and 6 channels. To implement SEAL, we add an AES encryption engine in every memory controller of the simulated GPU. We model a pipeline AES encryption engine with 128-bit block [15], in which the overall AES encryption latency for a cache line is 20 cycles and the bandwidth of each AES engine is 8GB/s.

We use three classical CNN models including VGG-16 [22], ResNet-18 [8], and ResNet-34 [8] to evaluate the performance of different encryption schemes. For comparisons, we evaluate an insecure GPU without memory encryption as the baseline (`Baseline`), traditional direct encryption (`Direct`) and counter mode encryption (`Counter`). `SEAL-D` and `SEAL-C` mean our SEAL scheme with direct encryption and counter mode encryption respectively.

B. Experimental Results

1) Performance of Different Layers: We perform SEAL on CONV layers whose input and output feature maps are also the input and output of POOL layers. The default encryption ratio is 50% as presented in Section III-B. We evaluate four typical CONV layers in VGG, in which the number of input and output channels is 64/128/256/512, and the five different POOL layers.

Figure 5 shows the relative IPCs when computing these CONV layers. We observe that Direct and Counter reduce the GPU IPC by up to 40% compared with Baseline. The reason is that memory encryption significantly reduces the data access bandwidth in GPUs. By comparing the performance between Direct/Counter and SEA-D/SEAL-C, our proposed SEAL improves the performance by 39% and 33% respectively, due to reducing the amount of the encrypted data to improve the data access bandwidth without compromising security.

Figure 6 shows the relative IPCs of different encryption schemes when computing POOL layers. We observe Direct and Counter reduce the IPC by up to 50%, and perform worse in comparison to computing CONV layers since the computation of POOL layers is more bandwidth-bounded than that of CONV layers. SEAL-D and SEAL-C improve the performance by 66% and 44%, compared with Direct and Counter respectively.

2) Overall IPC: We evaluate overall IPCs with different encryption schemes when executing the NN inference using VGG-16, ResNet-18, and ResNet-34, as shown in Figure 7. Direct and Counter reduce the GPU IPC for executing NN inference by $30\% - 38\%$, compared with Baseline. Moreover, Direct and Counter deliver higher performance in ResNets than those in VGG. The reason is that the amounts of computation and data accesses to memory in VGG are much larger than those in ResNets and thus VGG requires higher data access bandwidth. By using SEAL to allow some data to bypass the AES engine, SEAL-D and SEAL-C significantly improve the IPC by 1.4$\times$ and 1.34$\times$ respectively, compared with Direct and Counter.

3) Inference Latency: We investigate the impact of different encryption schemes on the inference latency, as shown in Figure 8. Direct and Counter increase the inference latency by $39\% - 60\%$, compared to Baseline. By using SEAL, SEAL-D and SEAL-C reduce the inference latency by 28% and 26% on average, compared with Direct and Counter respectively.

V. Related Work

Model Extraction Attacks in Architecture Layer. Existing works exploit the information of the operating system and architecture layers to speculate the NN model related information. Naghibijouybari et al. [17] exploit the side channel information in the operating system, such as memory allocation APIs, GPU performance counters, and timing measurement, to speculate the NN model related information, e.g., the number of neurons. Hua et al. [10] and Hu et al. [9] exploit the side channel information in the DL accelerator architecture, e.g., the memory access pattern, to speculate the NN architecture related information.

The model extraction attacks mentioned above can obtain only a small part of the NN model related information. Compared with these model extraction attacks, bus snooping attacks for DL accelerators that our paper focuses on are much more dangerous. This is because an adversary can obtain all data of the entire NN model including weight parameters in each layer by the bus snooping attacks. Our paper proposes a secure and efficient solution, SEAL, to defend against the bus snooping attacks.

Memory Encryption. Obviously, software memory encryption, e.g., Graviton [23], cannot adequately defend against physical access based attacks, since the programs of encryption software themselves can be stored in the memory. Hardware memory encryption has been widely used in secure CPU systems to defend against physical access based attacks by adding the hardware encryption engine on the CPU chip [24], [25]. However, memory encryption significantly decreases the performance of DL accelerators, e.g., GPUs, due to the big bandwidth gap between the GDDR memory bus and encryption engine. Our proposed SEAL efficiently addresses this problem. Moreover, Cai et al. [2] focus on the NN model security issue in computing-in-memory systems based on non-volatile memory, which is orthogonal to our paper focusing on general accelerators.

VI. Conclusion

Memory encryption becomes important to guarantee the security of DL accelerators, which however causes significant performance degradation. This paper present the insights that the big bandwidth gap between the memory bus of DL accelerators and the encryption engine is the main reason of causing performance degradation. To address this problem, we propose SEAL to improve the data access bandwidth of DL accelerators by identifying partial data that have no impact on the security of NN models and allowing them to bypass the encryption engine without affecting the security. Our experimental results show that, compared with existing memory encryption solutions, SEAL achieves $1.34 - 1.4\times$ performance improvement on average.

Acknowledgement

This work was supported in part by National Natural Science Foundation of China (NSFC) under Grants No. 61772212, No. 62002338, and Key Laboratory of Information Storage System, Ministry of Education of China.

References

[1] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proceedings of the 2009 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2009.

[2] Y. Cai, X. Chen, L. Tian, Y. Wang, and H. Yang, "Enabling secure nvm-based in-memory neural network computing by sparse fast gradient encryption," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1596–1610, 2020.

[3] Ensilica, "Advanced encryption standard cryptographic ip," 2020, https://www.ensilica.com/ip/esi-crypto/aes/.

[4] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems (NeurIPS)*, 2014.

[6] S. Gueron, "A memory encryption engine suitable for general purpose processors," Cryptology ePrint Archive, Report 2016/204, 2016, https://eprint.iacr.org/2016/204.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2015.

[8] 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)*, 2016.

[9] X. Hu, L. Liang, S. Li, L. Deng, P. Zuo, Y. Ji, X. Xie, Y. Ding, C. Liu, T. Sherwood, and Y. Xie, "Deepsniffer: a dnn model extraction framework based on learning architectural hints," in *in Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[10] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proceedings of the 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018.

[11] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[12] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[14] B. Liu and B. M. Baas, "Parallel aes encryption engines for many-core processor arrays," *IEEE transactions on computers*, vol. 62, no. 3, pp. 536–547, 2011.

[15] S. Mathew, F. Sheikh, A. Agarwal, M. Kounavis, S. Hsu, H. Kaul, M. Anders, and R. Krishnamurthy, "53Gbps native GF $(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45nm high-performance microprocessors," in *Proceedings of the 2010 IEEE Symposium on VLSI Circuits (VLSIC)*, 2010.

[16] S. Morioka and A. Satoh, "A 10-gbps full-aes crypto design with a twisted bdd s-box architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 7, pp. 686–691, 2004.

[17] H. Naghibijouybari, A. Neupane, Z. Qian, and N. Abu-Ghazaleh, "Rendered insecure: Gpu side channel attacks are practical," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018.

[18] Nvidia Corporation, "NVIDIA GeForce GTX 480," https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-480/specifications, 2012.

[19] OpenCores, "Tiny AES," http://opencores.org/project/, 2012.

[20] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS)*, 2017.

[21] G. Sayilar and D. Chiou, "Cryptoraptor: High throughput reconfigurable cryptographic processor," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2014, pp. 155–161.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[23] S. Volos, K. Vaswani, and R. Bruno, "Graviton: Trusted execution environments on GPUs," in *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.

[24] C. Yan, D. Englender, M. Prvulovic, B. Rogers, and Y. Solihin, "Improving cost, performance, and security of memory encryption and authentication," in *Proceedings of the 33rd Annual International Symposium on Computer Architecture (ISCA)*, 2006.

[25] P. Zuo, Y. Hua, and Y. Xie, "Supermem: Enabling application-transparent secure persistent memory with low overheads," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019.