Securing Deep Neural Networks on Edge from Membership Inference Attacks Using Trusted Execution Environments

Cheng-Yun Yang
yang2316@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Gowri Ramshankar
gramshan@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Nicholas Eliopoulos
neliopou@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Purvish Jajal
pjajal@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Sudarshan Nambiar
nambias@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Evan Miller
mill3131@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Xun Zhang
zhan4273@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Dave (Jing) Tian
daveti@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Shuo-Han Chen
shch@nycu.edu.tw
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

Chiy-Ferng Perng
Alex_Perng@wistron.com
Wistron
Taipei, Taiwan

Yung-Hsiang Lu
yunglu@purdue.edu
Purdue University
West Lafayette, Indiana, USA

ABSTRACT

Privacy concerns arise from malicious attacks on Deep Neural Network (DNN) applications during sensitive data inference on edge devices. Membership Inference Attack (MIA) is developed by adversaries to determine whether sensitive data is used to train the DNN applications. Prior work uses Trusted Execution Environments (TEEs) to hide DNN model inference from adversaries on edge devices. Unfortunately, existing methods have two major problems. First, due to the restricted memory of TEEs, prior work cannot secure large-size DNNs from gradient-based MIAs. Second, prior work is ineffective on output-based MIAs. To mitigate the problems, we present a depth-wise layer partitioning method to run large sensitive layers inside TEEs. We further propose a model quantization strategy to improve the defense capability of DNNs against output-based MIAs and accelerate the computation. We also automate the process of securing PyTorch-based DNN models inside TEEs. Experiments on Raspberry Pi 3B+ show that our method can reduce the accuracy of gradient-based MIAs on AlexNet, VGG-16, and ResNet-20 evaluated on the CIFAR-100 dataset by 28.8%, 11%, and 35.3%. The accuracy of output-based MIAs on the three models is also reduced by 18.5%, 13.4%, and 29.6%, respectively.

ISLPED '24, August 5–7, 2024, Newport Beach, CA, USA

ACM ISBN 979-8-4007-0688-2/24/08.
https://doi.org/10.1145/3665314.3670821

CCS CONCEPTS

• **Security and privacy** → **Embedded systems security**; • **Computing methodologies** → *Object recognition*; *Neural networks*.

KEYWORDS

Membership Inference Attack, ARM TrustZone, Trusted Execution Environment, Model Partitioning, Model Quantization

ACM Reference Format:

Cheng-Yun Yang, Gowri Ramshankar, Nicholas Eliopoulos, Purvish Jajal, Sudarshan Nambiar, Evan Miller, Xun Zhang, Dave (Jing) Tian, Shuo-Han Chen, Chiy-Ferng Perng, and Yung-Hsiang Lu. 2024. Securing Deep Neural Networks on Edge from Membership Inference Attacks Using Trusted Execution Environments . In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '24), August 5–7, 2024, Newport Beach, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3665314.3670821

1 INTRODUCTION

Deploying Deep Neural Networks (DNNs) on edge devices, such as smartphones, wearables, and IoT devices has become popular in recent years [1–3]. Compared with sending data to the cloud through wireless networks, processing the data on edge has several advantages such as low (and predictable) latency, low bandwidth usage, and better protection of private data. Though DNNs deployed on edge devices can directly process the data captured by sensors such as microphones or cameras, private information can leak if the model inference process is not well-secured.

Membership inference attacks (MIAs) are malicious attacks that discover whether a particular piece of data is used for training. The attacks pose privacy threats to DNN applications on edge devices that use private data for training or fine-tuning the DNN models. For example, the user identity of a smartphone can be inferred by

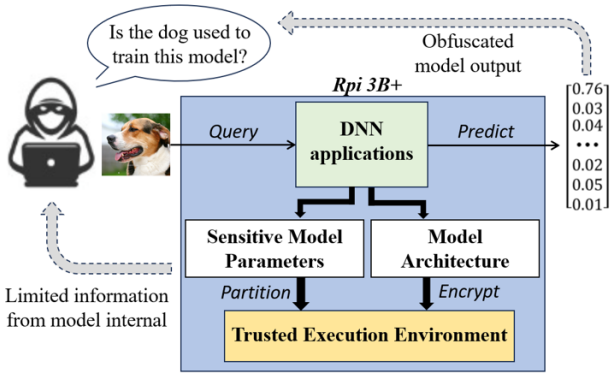

Figure 1: The proposed method on protecting DNN applications from gradient-based and output-based MIAs on Raspberry Pi 3B+ by limiting the access to the DNN internal and obfuscating the prediction.

launching MIAs on the built-in face recognition system. To secure DNN applications on edge devices, recent studies propose to run model inference inside a Trusted Execution Environment (TEE) [4]. TEE is a hardware feature that provides a secure execution environment for sensitive applications and data. For edge devices, Arm TrustZone [5] is the most commonly used TEE implementation.

Existing methods utilizing TEEs to secure DNN applications on edge devices encounter two main challenges that impede their real-world adoption. (1) They cannot secure large DNN models due to the limited memory of the TEE operation system (OS). For example, a popular TEE OS, OP-TEE, has a maximum of 2MB stack size and 16MB heap size. Restricted by the memory size, DarkneTZ [6] can run only the last layer of VGG-7 inside a TEE. T-slices [7] partition DNN models into smaller pieces but still fail to run the entire VGG-16 model inside a TEE due to the high peak memory of around 923MB. (2) Existing methods are effective on defending only gradient-based MIAs by hiding the intermediate DNN inference process. The DNN models protected by these methods still suffer from output-based MIAs that require only the model output.

The proposed solution alleviates the above two problems in prior work. (1) We present a depth-wise layer partitioning method to better utilize the memory of TEE to run computationally expensive DNN layers. The inference of sensitive DNN layers is decomposed into several matrix multiplications between each feature map and kernel weight. This allows more DNN layers to be run inside TEEs. (2) This paper proposes a MIA-aware model quantization strategy to increase the difficulty of determining data membership solely by the model output. It is done by finding the quantization scale that leads to the lowest accuracy of the MIA conducted by the defender. The DNN models after quantization are more robust to output-based MIAs while also become lighter to be deployed on edge devices. Figure 1 describes the two parts of our method to prevent leaking sensitive information to membership inference attackers.

Existing methods require that developers train DNN models from scratch using the Darknet framework or convert the models from other frameworks. Significant effort is needed if they want to protect customized DNN models with TEE [6, 7]. Our method can automatically convert and encrypt the PyTorch-based DNN model weights. This reduces the development effort to implement secure inference on board using a TEE. For users who want to protect their DNN models, the model weight saved in PyTorch format is the only thing they need to prepare.

This paper has three major contributions to secure DNN model inference system:

- With the proposed depth-wise model partitioning method, the gradient-based MIA accuracy for a set of pre-trained models is reduced by 8.1% to 35.3% across the Tiny ImageNet and CIFAR-100 datasets.
- By protecting the model architecture information with TEE and applying the proposed model quantization strategy, the output-based MIA accuracy is reduced by 11.7% to 29.6% on CIFAR-100 dataset.
- The proposed method supports PyTorch models. Users do not need to convert and encrypt the DNN models manually. Besides, the hyperparameters for building the system are chosen based on the protected DNN models automatically.

2 BACKGROUND AND RELATED WORK

2.1 Membership Inference Attacks on DNNs

Membership Inference Attacks (MIAs) use information from DNNs to determine whether a piece of data is used for training. From the literature, we identify two common types of MIAs: (1) gradient-based and (2) output-based MIAs. Gradient-based MIAs employ backpropagation to calculate the gradients of intermediate layers and utilize them for membership prediction. In general, training data exhibits low gradients as a well-trained DNN model typically converges to a low training loss, consequently resulting in low gradients for each training sample [8]. Therefore, gradients from member and non-member data can be used to train a binary classifier as an attack model to predict membership of unknown data. Output-based MIAs use the output from DNNs to predict membership of data. For an image classification task, the output is a confidence vector comprised of the score of each possible class. Member data usually has high score of the correct class and low scores of the other classes while non-member exhibits a more uniform distribution of scores across all classes. A binary detector for membership can thus be trained with confidence vectors from member and no-member data. Both types of MIAs can be more threatening if the model architecture is known to an adversary [9].

2.2 Secure DNNs on Edge from MIAs

Arm TrustZone is a security extension technology that separates the execution environments on a single System-on-Chip into: (1) a Trusted Execution Environments (TEE) for secure execution and (2) a Rich Execution Environment (REE) for non-secure execution. Arm TrustZone is widely adopted to provide security on edge devices due to its robust hardware and software support. OP-TEE is a popular OS of edge devices that employ Arm TrustZone to ensure the confidentiality and integrity of security-sensitive computation and data inside the TEEs. Previous work applies TEEs to secure DNN computation on Arm-based edge devices [6, 7]. However, these methods have no support for PyTorch models, making protecting modern DNN models using TEE challenging.

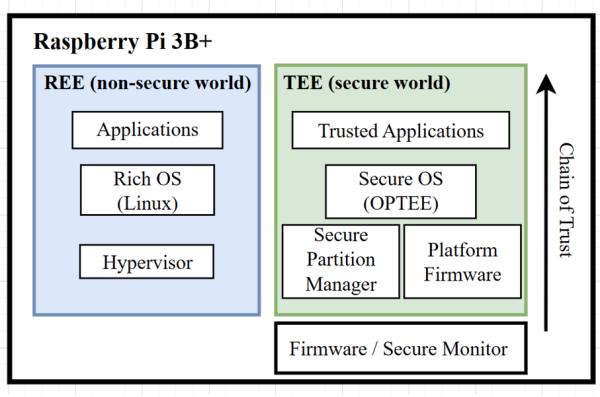

Figure 2: Software architecture of a TEE deployed on Raspberry Pi 3B+. A chain of trust is established starting from the boot ROM stage and each layer of software is loaded and checked for integrity by the previous layer.

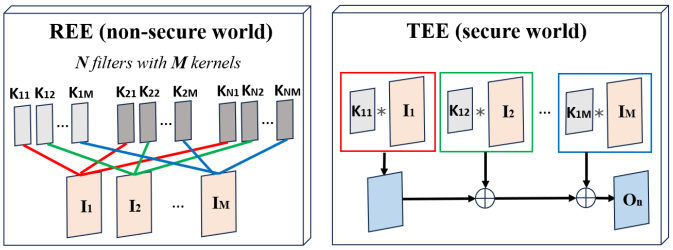

Figure 3: Inference of a single DNN layer when the proposed depth-wise partitioning method is applied. K_{nm} denotes the mth kernel of the nth filter. I_n and O_n denote the nth channel of the input and output feature map. This figure shows how the first channel of the output feature map is calculated. The calculation is done subsequently from O_1 to O_N.

DNN models protected by TEEs are still under the threat of output-based MIAs. Most prior work tries to defend such attacks by closing the generalization gap between the training and validation data during the training process. Li et al. [10] employ a regularizer using the Maximum Mean Discrepancy between the softmax output empirical distributions. Yuan et al. [11] present an optimization goal that minimizes the Kullback–Leibler divergence loss between training and validation data pairs. Different from the prior work, we propose a post-training quantization method that eliminates the need for retraining the original model. The proposed method facilitates the updating of DNN applications on edge devices and benefits the defense against gradient-based MIAs since the DNN models are quantized to have more layers run inside TEEs.

3 SECURE TRAINING DATA PRIVACY ON EDGE

3.1 Threat Model

Following the assumption of TrustZone [5], we consider the adversary to have full access to REEs and has no control or knowledge of applications running inside TEEs. The adversary can get the gradients during the model inference process in REEs by memory invasion attack [12]. The files stored in REE are disclosed to the

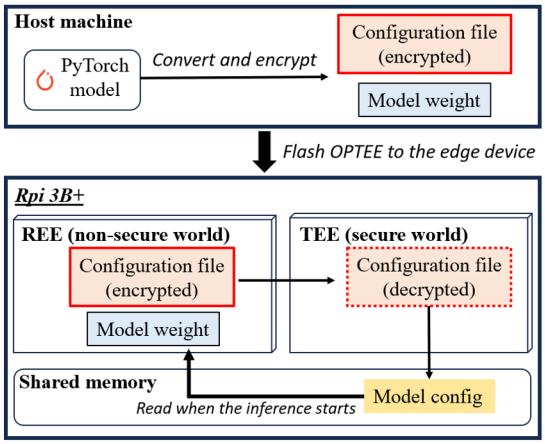

Figure 4: The framework of protecting the configuration file of the DNN deployed on the edge device using TEE. The file is kept encrypted until the DNN model inference starts.

adversary. Besides, we assume that the DNN applications return the confidence score vector as output. For an image classification task, the output is the scores of all possible classes. We assume that the adversary can only interact with the DNN model deployed on the device remotely i.e., the adversary does not have physical access to the device. Therefore, hardware-based attacks like side-channel attacks are out of scope for this work. We assume that all firmware and Trusted Applications (TAs) developed on TEE are manually verified and compiled, meaning we trust the authors of all the software developed for the TEE. The attestation mechanisms of TEE automatically check TAs for integrity as shown in Figure 2.

3.2 Depth-wise Model Partitioning

Since the model inference in TEE is time-consuming, we only run the most sensitive part of DNN inside a TEE. Previous work has shown that the gradients from the layers closer to the output layer contribute more to conduct successful gradient-based MIAs [6].Thus, we run the inference of the last few DNN layers inside a TEE so that the adversary has no access to the weights to calculate gradients. This is done by sending the input feature maps and model weights to TEE by invoking TAs. The TAs then use a key that is securely stored in the TEE to decrypt the model weights and run the computation in the secure memory of TEE. The number of layers being protected is configurable.

The peak memory usage of a single layer from DNNs can reach up to several hundred megabytes, beyond the size of 16 MB in practice [7]. Therefore, we propose a depth-wise model partitioning method to better integrate DNN applications with TEEs. The convolution operation of DNN model inference is split into several iterations to execute. For each iteration, we send the weights of a kernel and channels of feature maps into the secure world for convolution operation. We choose the maximum number of feature maps that can be multiplied with the weights within the trusted memory capacity, i,e., the maximum number of $K_{nm} * I_m$ that can be executed inside the secure world, until all the feature maps are multiplied. As shown in Figure 3, the execution order of deriving the first channel of output feature maps in the secure world follows

$K_{11} * I_1$, $K_{12} * I_2$, ..., and $K_{1M} * I_M$, where $*$ denotes a convolution operation. This saves the memory usage of directly running all convolution operations inside the secure world. In addition, we quantize the values of weights and feature maps from 32-bit floating points to 8-bit integers to reduce the memory requirement of *im2col* transformation. The detail of the applied quantization method will be further elucidated in Section 3.3.

3.3 MIA-aware Model Quantization

Hiding the model architectures can increase the defense capability of DNN models to gradient-based MIAs. However, the DNN models are still vulnerable to output-based MIAs that only require the output from the models. Therefore, we propose a post-training model quantization technique to further mitigate the threat posed by output-based MIAs. For a DNN model with N layers, we quantize the weights of the first $N-1$ layers by searching for the scale factor Δ_w that minimizes the *l2 norm* distance of the output before and after quantization:

$$\min_{\Delta_w} \sqrt{\sum_{i=1}^{k} (O_i - \hat{O}_i)^2}, \tag{1}$$

where O_i and $\hat{O}_i$ are the $i-$th entries of the output vector before and after quantization, and Δ_w is the scale factor used to quantize the model weight. We search the scale factor over 100 points between 0.1 and 2. Traditional quantization methods apply Formula 1 as the searching goal to all layers to maintain the prediction accuracy of DNN model after quantization. We keep the same searching goal for the $N-1$ layers but change the one for the last layer to:

$$\min_{\Delta_w} (\sqrt{\sum_{i=1}^{k} (O_i - \hat{O}_i)^2} + \alpha \cdot Acc_{\text{MIA}}), \tag{2}$$

where Acc_{MIA} is the accuracy of simulated MIAs done by the defender itself and α is a weighting factor empirically determined to balance the two terms. The search goal is designed to make the model output less informative regarding the membership of the input data. As the defenders who have the ground truth of the training data membership, we can train a simulated attack model by taking the output vectors of the member data as positive samples and those of the non-member data as negative samples. The simulated attack model can be simply done by training a multi-layer perceptron. After that, we find the quantization scale by Formula 2 that minimizes the accuracy of the simulated attack model while maintaining the model accuracy. We only apply Formula 2 to quantize the last layer of the victim DNN model because it is the closest one to have an impact on the output confidence vectors. Besides, quantizing a single layer has a negligible effect on the model accuracy. The experiments show that the proposed model quantization strategy can reduce the accuracy of output-based MIAs on the DNN model by 10% on average with a trade-off of only around 1% model accuracy drop. In addition, the DNN model is quantized from FP32 to INT8, which provides around four times reduction on the model size and thereby benefits the inference speed on edge devices.

3.4 Model Configuration Protection Framework

Prior work utilizing TEEs for protecting data privacy on edge devices is susceptible to model architecture leakage if the model configuration file is compromised [6]. This file stores the model architecture, allowing the application to load the model weights. MIA will be more threatening if the model architecture is known to an adversary. Therefore, we propose a protection framework that utilizes TEEs to prevent the model configuration file from being accessed. We encrypt the model configuration file and load it onto the non-secure world of the edge device using OP-TEE. When the DNN application is triggered, a TA is invoked to transfer the encrypted file into the secure world. Inside the secure world, the file is decrypted and the decrypted data is stored in shared memory for the DNN application to access. This approach prevents an adversary from directly fetching the model configuration from either the non-secure world or the memory location where it resides. We choose to implement the Tiny Encryption Algorithm [13] due to its minimal execution time overhead. The process of securing the DNN configuration file is illustrated in Figure 4.

3.5 Secure PyTorch Models at the Edge

Figure 5 shows how the proposed method described in this section defend against MIAs compared with that of DarkneTZ [6]. Our system hides the inference of the last three layers and the model architecture in the TEE. It prevents the adversary from leveraging the gradients from these layers to train a powerful attack model. Furthermore, the proposed MIA-aware model quantization makes the model output less informative for the adversary to predict the data membership. Another feature of the system is that it can directly take PyTorch models as input. Prior work [6, 7] only supports to read DNN models built with Darknet, which is a neural network framework rarely used for training DNN models. Our system automatically converts the PyTorch model to the format that can work with OP-TEE and encrypt the architecture and weight of the model. On the whole, the proposed secure system can (1) protect any customized PyTorch-based DNN models without using Darknet to train the models again and (2) improve the defense capability of DNN models against MIAs compared with the existing methods.

4 EVALUATION AND RESULTS

4.1 Experimental Setup

We evaluate our method on protecting four classic DNN models including AlexNet, VGG-11, VGG-16, and ResNet-20 against MIAs. We target image classification task and use CIFAR-100 and Tiny ImageNet as the benchmarks. The experiments are done on Raspberry Pi 3B+ flashed with OP-TEE. We measure the efficiency of our method by CPU execution time because the GPU computation is not supported by OP-TEE. All the models are built with PyTorch and trained with one NVIDIA A100 SXM GPU.

4.2 Privacy Measurement

We validate our method by measuring the accuracy of the attack model. The attack model is a binary classifier applied by the adversary to predict the membership of the test data. If the accuracy is close to 100%, it means that the defense method is not effective. Conversely, if the attack accuracy goes down to 50%, it becomes a random guess, meaning that the DNN model under attack is robust to the MIAs. The experiments involve three steps. The first step is to train the victim model and the shadow model. The second step involves training the attack model and the final step is assessing

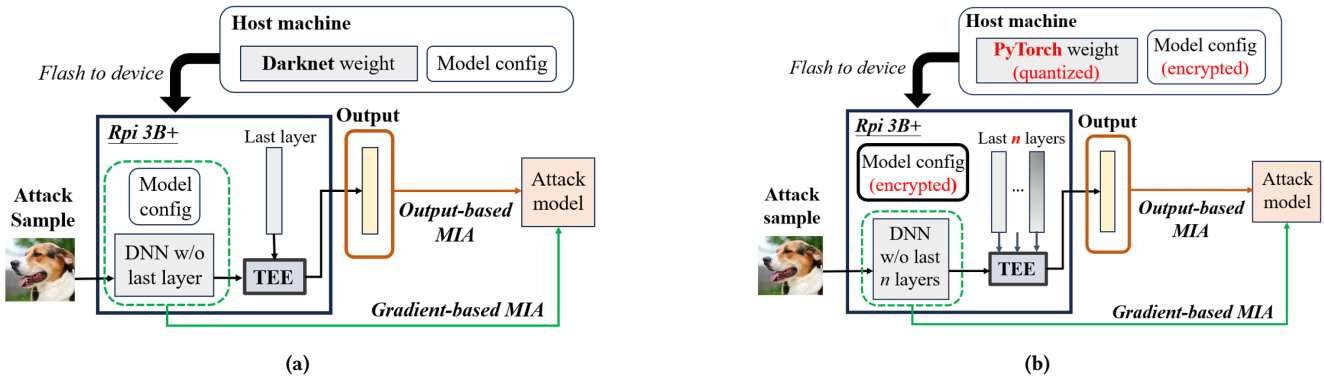

Figure 5: The illustration of how the adversary conducts gradient-based and output-based MIAs on the DNN applications protected by (a) Mo et al. [6] and (b) the proposed method. The green and brown boxes denote the information available for gradient-based and output-based MIAs, respectively.

the privacy of the victim model by leveraging the attack model. The three steps are explained further in the following paragraphs.

First, we use half of the training and validation data to build a victim model, which serves as the model adversaries want to attack. Then we use the other half of the training and validation data to build a shadow model, which serves as the surrogate of the victim model. We assume the adversary knows which data is used to train the shadow model. Therefore, the shadow model can be used to guide the learning of the attack model with the training data labeled as members and the validation data labeled as non-members.

Second, we train the attack model, a binary classifier that predicts the membership of the victim models using the knowledge learned from the shadow models. For gradient-based MIAs, we extract the gradients from shadow models and concatenate them into a feature vector, which serves as the input to the attack model. Because the real membership of the shadow models is disclosed, the adversary can optimize the attack model to learn the relationship between the feature vector and the data membership. The process of building an attack model is the same for output-based MIAs except that the feature vector is changed to the output of the shadow model.

Finally, we assess the privacy of the victim model by leveraging the attack model to predict the membership of test data. We first send the test data to the victim model to collect the feature vector. Then the feature vector is fed to the attack model to predict the possible membership of the data. The prediction of the attack model is either *in* or *out* if the model predicts the test data to be a member or non-member of the training data, respectively. We measure the prediction accuracy of the attack model under three conditions: without any defense methods, protected by Mo et al. [6], and protected by our method. We run the last three layers of the test models inside a TEE to achieve the best trade-off between MIA accuracy and execution time on Raspberry-Pi 3B+. The result of conducting gradient-based MIAs and output-based MIAs on the DNN models is shown in Table 1 and Table 2, respectively.

Given the resource constrained environment of TEE, it is important to quantify the efficiency of our method and understand the feasibility for deployment on edge devices. Table 3 compares the inference time and memory usage of the DNN models and their quantized versions under two scenarios: full execution in REE and execution with the proposed method.

Table 1: Gradient-based MIA accuracy on AlexNet, VGG-16, and ResNet-20 when the models are under no protection, protected by Mo et al. [6], and our method. The goal of defenders is to reduce the attack accuracy to 50% (random guess).

Dataset	Model	MIA Accuracy		
		Vanilla	Mo et al. [6]	Ours
CIFAR-100	AlexNet	88.0%	65.1%	59.2%
	VGG-16	77.9%	72.4%	66.9%
	ResNet-20	85.0%	60.6%	49.7%
Tiny ImageNet	AlexNet	94.1%	70.7%	59.1%
	VGG-16	75.5%	62.4%	60.0%
	ResNet-20	65.1%	61.0%	57.0%

Table 2: Output-based MIA accuracy on victim DNN models when various shadow models are used for training attack models. Model(Q) represents the model that is quantized by our method. The experiments are done on CIFAR-100 dataset.

Victim model	Shadow Model			
	AlexNet	VGG-11	VGG-16	ResNet-20
AlexNet	87.7%	82.3%	86.7%	78.1%
AlexNet(Q)	69.9%	68.5%	69.2%	61.8%
VGG-11	88.9%	87.9%	88.4%	79.2%
VGG-11(Q)	60.1%	77.2%	75.9%	64.0%
VGG-16	78.5%	68.2%	87.4%	77.6%
VGG-16(Q)	74.0%	63.1%	84.5%	71.5%
ResNet-20	71.7%	82.8%	63.4%	94.4%
ResNet-20(Q)	64.8%	62.5%	62.9%	88.9%

4.3 Results and Discussion

As shown in Table 1, our method reduces the accuracy of the gradient-based MIAs on AlexNet, VGG-16, and ResNet-20 by 28.8%, 11%, and 35.3% when tested on CIFAR-100 dataset. For the Tiny ImageNet dataset, the accuracy of gradient-based MIAs is reduced by 35%, 15.5%, and 8.1% on AlexNet, VGG-16, and ResNet-20, respectively. The proposed method can effectively reduce the accuracy

Table 3: Comparison of memory size and execution time per image with and without the proposed method for AlexNet, VGG-11, VGG-16, and ResNet-20 on Raspberry Pi 3B+.

Victim model	Memory size (MB)	Execution Time (s)	
		Vanilla	Ours
AlexNet	9.76	4.65	13.06
AlexNet(Q)	2.44	1.88	6.62
VGG-11	36.23	3.97	10.89
VGG-11(Q)	9.06	1.61	5.53
VGG-16	57.69	4.65	11.41
VGG-16(Q)	14.42	1.88	5.80
ResNet-20	1.14	4.19	5.30
ResNet-20(Q)	0.28	1.69	2.27

of gradient-based MIAs than Mo et al. [6] does across all test DNN models and datasets. We choose to hide the inference of the last three layers inside a TEE. This is because we observed that hiding more than three layers does not decrease the effectiveness of gradient-based MIAs on the models employed in our experiments. However, the number of layers being protected by our method is a configurable parameter. Users can tailor this parameter to fit the hardware and the models they want to protect.

Table 2 shows the accuracy of output-based MIAs on the four models using CIFAR-100 dataset. We observe that if the adversary knows the exact model architecture of the victim model, the output-based MIAs will be more effective. For example, using AlexNet architecture as the shadow model to attack the victim AlexNet model achieves 87.7% accuracy, which is higher than using other model architectures as shadow models. Therefore, it is necessary to apply the model architecture protection framework described in Section 3.4. We also observe that applying a shadow model that has a similar model architecture to that of the victim model can attain high attack accuracy. For instance, the attack accuracy of using VGG-16 as the shadow model to attack VGG-11 (88.4%) is even higher than that of using VGG-11 to attack itself (87.9%). It is reasonable because VGG-16 is the deeper version of VGG-11.

The accuracy of output-based MIAs is further reduced by the proposed model quantization method as mentioned in Section 3.3. As shown in Table 2, quantization reduces the accuracy of MIAs on all the victim models. The threat of using a similar DNN model as the shadow model to conduct output-based MIAs is also mitigated. For the case we studied earlier when using VGG-16 as the shadow model to attack, the accuracy of the same attack on VGG-11 drops from 88.4% to 75.9% after the proposed quantization method is applied. In addition, even if the adversary can guess the right model architecture of the victim model, the attack accuracy on the quantized version of each victim model is constantly lower than the vanilla model for the chosen four DNN models. Overall, the model configuration protection framework increases the difficulty of finding appropriate shadow models to conduct output-based MIAs. The proposed model quantization method further obfuscates the distribution of model output to be more challenging for the adversary to train effective attack models for output-based MIAs.

From Table 3, we observe that the memory reduction is roughly four times since the model is quantized from FP32 to INT8. The execution time overhead of protecting AlexNet in our system is 8.41 seconds, which is the highest among all tested models because the last three layers of AlexNet require the most iterations to complete the inference. On the contrary, ResNet-20 has the lowest execution overhead of 1.11 seconds due to the light weight of its last layers. Deployment of the model quantization can further reduce the execution time overhead by around 50%. Applying MIA-aware model quantization also brings a minor reduction in the test accuracy of the victim models. The accuracy drop is 0.91%, 1.33%, 0.53%, and 0.87% for AlexNet, VGG-11, VGG-16, and ResNet-20, respectively, which is a decent trade-off for training data privacy improvement.

5 CONCLUSION

In this paper, we present a secure method to protect DNN applications on edge from the privacy threat of MIAs. With the employment of a depth-wise model partitioning method, more sensitive DNN layers are secured in TEE during model inference to mitigate the threat of gradient-based MIAs. This system also features functionality to safeguard the DNN model architecture. Besides, a novel and simple enough model quantization method is applied to enhance the defense capability of DNN models to output-based MIAs. Our method can be applied to any PyTorch-based DNN models with acceptable inference time overhead and minor model accuracy drop. For a ResNet-20 model, the system can decrease the accuracy of gradient-based MIA by 35.3% and the output-based MIA by 29.6% with only 0.87% drop on the image classification accuracy. We believe our method offers a simple yet effective way for engineers to secure their DNN models on edge using TEEs.

ACKNOWLEDGMENT

This project is supported in part by NSF IIS-2229876, ONR N00014-23-1-2157, and Wistron. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Sebastian P Bayerl et al. "Offline model guard: Secure and private ML on mobile devices". *Design, Automation & Test in Europe Conference & Exhibition*. 2020.
[2] Han Cai et al. "Enable Deep Learning on Mobile Devices: Methods, Systems, and Applications". *ACM Transactions on Design Automation of Electronic System* 27.3 (2022).
[3] Gianmarco Cerutti et al. "Sound event detection with binary neural networks on tightly power-constrained IoT devices". *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*. 2020.
[4] Nicholas Carlini et al. "Membership inference attacks from first principles". *IEEE Symposium on Security and Privacy*. 2022.
[5] *ARM Security Technology Building a Secure System using TrustZone Technology*.
[6] Fan Mo et al. "DarkneTZ: towards model privacy at the edge using trusted execution environments". *International Conference on Mobile Systems, Applications, and Services*. 2020.
[7] Md Shihabul Islam et al. "Confidential Execution of Deep Learning Inference at the Untrusted Edge with ARM TrustZone". *ACM Conference on Data and Application Security and Privacy*. 2023.
[8] Milad Nasr et al. "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning". *IEEE Symposium on Security and Privacy*. 2019.
[9] Reza Shokri et al. "Membership inference attacks against machine learning models". *IEEE Symposium on Security and Privacy*. 2017.
[10] Jiacheng Li et al. "Membership Inference Attacks and Defenses in Classification Models". *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*. 2021.
[11] Xiaoyong Yuan et al. "Membership Inference Attacks and Defenses in Neural Network Pruning". *31st USENIX Security Symposium*. 2022.
[12] Fan Yao et al. "DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips". *29th USENIX Security Symposium*. 2020.
[13] Simon J. Shepherd. "The Tiny Encryption Algorithm". *Cryptologia* 31.3 (2007).