HybridTEE: Secure Mobile DNN Execution Using Hybrid Trusted Execution Environment

Akshay Gangal, Mengmei Ye, and Sheng Wei
Department of Electrical and Computer Engineering
Rutgers University, Piscataway, NJ, USA
Email: {akshay.gangal, mengmei.ye, sheng.wei}@rutgers.edu

2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST) | 978-1-7281-8952-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/ASIANHOST51057.2020.9358260

***Abstract*—Deep neural networks (DNNs) have been increasingly adopted in many mobile applications involving security sensitive data and inference models. Therefore, there is an increasing demand for secure DNN execution on mobile devices. Catering to this demand, hardware-based trusted execution environments (TEEs), such as ARM TrustZone, have recently been considered for secure mobile DNN execution. However, it is challenging to run DNN models in TrustZone, due to the stringent resource and performance limitations posed by the mobile TEE. We develop *HybridTEE*, a novel hardware-based security framework to securely execute DNNs in the resource-constrained *local TEE* (i.e., ARM TrustZone), by offloading a part of the DNN model to a resource-rich remote TEE (i.e., Intel SGX). *HybridTEE* strategically divides the DNN model into privacy-aware local and remote partitions by employing two privacy-oriented metrics based on object recognition and Scale Invariant Feature Transform (SIFT). Also, it builds a trustworthy communication channel bridging TrustZone and SGX to enable secure offloading of the DNN model between the two TEEs. Our security and performance evaluations on real hardware systems show that *HybridTEE* can ensure the security and privacy of the DNN model with superior execution time compared to the non-TEE baseline.**

I. Introduction

Deep neural networks (DNNs) have been widely adopted in various mobile applications to accomplish critical inference tasks [1]. Many of these applications interact with sensitive data that require security/privacy protection, such as the biometrics used by an authentication app and the medical information involved in a healthcare app. In addition, the confidentiality and integrity of the mobile DNN model itself are subject to a variety of security threats [2], [3].

Recently, hardware-based trusted execution environments (TEEs), such as ARM TrustZone [4], have been developed to address mobile security challenges. TrustZone can create a hardware-isolated secure world to protect the sensitive data, which remains secure even if the operating system (OS) has been compromised by an attacker. Such exclusion of OS from the trusted computing base (TCB) makes TrustZone a strong hardware security primitive for securing mobile DNNs.

However, it is very challenging to execute DNNs in TrustZone due to the huge gap between the limited computing resources (e.g., memory space or hardware/software accelerators) in the TrustZone secure world and the high demand of the DNN models that are both data and computation intensive.

Although it is physically feasible to deploy more resources into the secure world, doing so is directly against the security principle of maintaining a small TCB and would result in an increased level of exploitable security vulnerabilities. The existing research efforts of executing DNN in TrustZone have not fully addressed this challenge. For example, [5] partitions the DNN model and runs all the partitions sequentially within the limited memory space of TrustZone. Although the approach could meet the desired security requirement, it results in nontrivial timing overhead in the DNN execution. Several other approaches [6], [7], [8], [9] partition the DNN model into a small secure component that runs in the protected TEE and a large non-secure component that runs outside the TEE. In this case, the non-secure component may leave traceable information for the attacker to compromise the DNN model.

To address the security and performance challenges in executing DNN in TrustZone, we propose to adhere to two design principles. First, the entire DNN model must remain in TEE to meet the security requirement. Second, for the performance challenge, a second heterogeneous TEE (e.g., Intel SGX [10]) that runs on high-performance server platforms can be leveraged to compensate for the limited computation resources in TrustZone. The two design principles combined lead to our proposed solution *HybridTEE*, which offloads a strategically partitioned part of the DNN to a remote Intel SGX enclave [10]. In this way, *HybridTEE* ensures the security of the DNN model as it is entirely executed in TEEs. Also, it has the potential of significantly improving the performance given the higher amount of computation resources and capacity in the remote SGX enclave. *HybridTEE* creates a new category of approach for secure mobile DNN execution, which bridges TrustZone and SGX to meet the high security requirement while achieving premium performance.

HybridTEE involves two key technical components with novel research contributions to realize the aforementioned design principles. *First*, we develop two privacy-aware partitioning algorithms to eliminate the potential privacy concerns while offloading the model from the local mobile device to the remote server. *Second*, we develop a *HybridTEE* framework by bridging the two singular TEEs, namely TrustZone and SGX, via a secure handshake, which ensures that no sensitive data stays in the clear during the secure offloading. Our evaluations on real hardware systems justify the effectiveness of *HybridTEE* in terms of security and performance.

II. Background: Trusted Execution Environment

The security community has developed and deployed hardware-based trusted execution environments (TEEs) to protect sensitive computations and data [4], [10]. A TEE provides isolated execution of the application to ensure its confidentiality and integrity, which cannot be exploited even if the OS has been compromised. This excludes the OS from the trusted computing base of the system and significantly reduces the possible attack surface. In our proposed *Hybrid-TEE* framework, we primarily focus on the combination of two TEEs: ARM TrustZone [4] for mobile devices and Intel SGX [10] for server/PC platforms. TrustZone secures sensitive data on ARM-based mobile devices through bus-level isolation between the trusted *secure world* and the untrusted *normal world*. SGX is a security extension to the Intel architecture, which enables secure *enclaves* to protect sensitive data.

III. Threat Models

Security Threat Model. We assume that the adversary attempts to compromise the confidentiality and integrity of the target DNN model, including (1) reconstructing the model by using confidential information obtained during the model execution; (2) uncovering the final inference results of the model; and (3) modifying the input or model proprietary information to alter the inference results. Note that in this work we do not aim to *enhance* the security of the TEEs themselves. Therefore, we do not consider side channel or hardware physical attacks that were not the original design objectives of TrustZone and SGX and thus could compromise the security of them.

Privacy Threat Model. We also consider a privacy threat model, where the users are concerned about their private information (e.g., the input images) being offloaded to a remote server and accessed by the (even non-malicious) service provider. This requires that the images that contain recognizable private information should not be offloaded to the remote SGX enclave, even if the security of the enclave is not compromised. This creates a significant challenge to the design of the offloading mechanism in *HybridTEE*, which we aim to address with the privacy-aware partitioning algorithm.

IV. Problem Definition: Mobile DNN Execution

A pre-trained DNN model contains proprietary information such as runtime configuration parameters and weights. Exposure of such information to an adversary could compromise the intellectual property of the model. Also, the input data and the final inference results are often security and privacy sensitive as well. In order to protect the model, input data, and inference results, it is desirable to execute the DNN model in a hardware-based TEE [5], [7].

However, the design principle of TEE is to protect the sensitive code/data with small TCB size [4], [10]. Such design principle minimizes the potential security vulnerabilities of the TEE, thus making it more challenging for attackers to compromise. This indicates that the amount of data or code that can be securely stored or executed in the TEE is limited. On the other hand, the size of a DNN model is typically huge containing 3 major components: neural network layers, weight vectors, and input data. In particular, the weight vectors of popular DNNs can be up to a few hundred megabytes [11]. As a comparison, the memory capacity of a typical TrustZone implementation, such as OP-TEE [12] on Raspberry Pi 3, is in the range of a few megabytes. The huge gap between the demand and supply creates the research problem we aim to address with the design of *HybridTEE*.

V. Privacy-Aware DNN Partitioning

In *HybridTEE*, we deploy a three-way sequential partitioning technique to address the aforementioned security and performance challenges. The DNN model is divided into 3 partitions, namely *LocalNet*, *RemoteNet*, and *PredNet*, as shown in Figure 1. First, *LocalNet* or the local partition contains the most sensitive layers of the neural network that are vulnerable to privacy exposure about the input data. The code and data of *LocalNet* and its feed forward function should reside in the *local TEE* (i.e., TrustZone) for privacy protection. Second, *RemoteNet* or the remote partition contains the next set of sequential layers in the neural network. It is stored and executed in the *remote TEE* (i.e., SGX). In order to maximize the overall performance of the application, *RemoteNet* should contain the most number of layers in the neural network, since the SGX server is a faster and more resource-rich platform. However, due to privacy consideration, there is critical information in the DNN model that must stay in *LocalNet*, and the *partition point* for the transition from *LocalNet* to *RemoteNet* is determined by our privacy-aware partitioning algorithm that will be discussed next. Finally, *PredNet* or the prediction partition contains the final layer of the DNN that performs the final inference. This layer runs in the *local TEE* (i.e., TrustZone), considering the privacy-sensitive nature of the inference results.

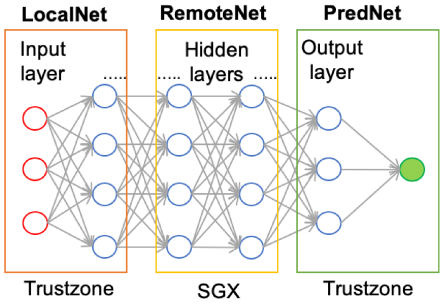

Fig. 1: DNN partitioning with *LocalNet* and *PredNet* running in ARM TrustZone and *RemoteNet* running in Intel SGX.

Consider an N-layer DNN model with feed forward function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$. The network partitions for *LocalNet*, *RemoteNet* and *PredNet* can be formulated as $f_\theta = f_{\theta_l} \oplus f_{\theta_r} \oplus f_{\theta_p}$, where $f_{\theta_l}, f_{\theta_r}$ and f_{θ_p} are the respective feed forward functions of *LocalNet, RemoteNet and PredNet*. *LocalNet* obtains the high dimensional input vector $\mathbb{R}^{w \times h \times c}$ and generates the partial outputs in the form of $f_{\theta_l}(p\_point)$, where p_point

represents the *partition point*. These outputs are then encrypted using symmetric encryption before being sent to *RemoteNet*.

$$f_{\theta_l} = f(\mathbb{R}^{w \times h \times c}) \tag{1}$$

$$f^*_{\theta_l} = Enc(f_{\theta_l}(p\_point)) \tag{2}$$

The encrypted outputs of *LocalNet* act as inputs to *RemoteNet*, which decrypts the inputs, followed by computing and encrypting the results up to the $(N-1)^{th}$ layer of the neural network.

$$f_{\theta_r} = Dec(f^*_{\theta_l}) \tag{3}$$

$$f^*_{\theta_r} = Enc(f_{\theta_r}(N-1)) \tag{4}$$

The encrypted outputs from *RemoteNet* act as inputs to *PredNet*, which decrypts the partial outputs and performs the final prediction $\mathcal{Y}$ based on the class labels.

$$f_{\theta_p} = Dec(f^*_{\theta_r}) \tag{5}$$

$$\mathcal{Y} = f_\theta = Pred_{labels}(f_{\theta_p}) \tag{6}$$

While designing the partitioning algorithm, our key idea is to automatically detect the privacy exposure in each layer of the DNN model and identify the first layer that no longer exposes privacy-sensitive information as the *partition point*. While there is no common standard of privacy defined for this domain-specific application, we consider the privacy of the user by regarding all the original input images as privacy-sensitive, and the intermediate inputs/outputs that could reveal the original input images should remain in *LocalNet* without being offloaded. Following this privacy model, we develop two systematic methods to find the optimal *partition point*. First, we employ an auxiliary DNN to conduct object detection at the intermediate layers and identify potential exposure of meaningful, identifiable information of the original input images (discussed in Section V.A). Second, we employ the Scale Invariant Feature Transform (SIFT) method to detect localized keypoints in the intermediate inputs/outputs and quantify the potential privacy exposure (discussed in Section V.B). In both methods, the optimal *partition point* can be determined as the DNN layer that results in significantly low exposure of the input images.

A. Object Detection Using an Auxiliary DNN

The objective of the partitioning algorithm is to find the last layer in the DNN model that is still susceptible to information exposure about the original input images. This is based on the assumption that the adversary is able to identify the objects that are distinctly visible in the input/output images at an intermediate layer of the DNN model. We develop an algorithm to determine such *partition point* by employing an auxiliary DNN to detect the objects in the intermediate input/output images. The auxiliary DNN acts as an adversary attempting to classify the key objects in the intermediate images. If the auxiliary DNN is not able to classify an images with a certain degree of confidence, the image is then considered as free of privacy exposure for offloading.

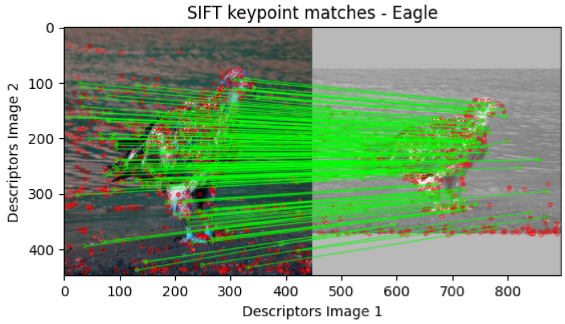

Fig. 2: SIFT keypoint matching between two images.

In particular, we use YOLO [11] as the auxiliary DNN model, which is a state-of-the-art, real-time object detection mechanism. The YOLO model predicts the bounding boxes using dimension clusters, with 4 coordinates for each box. Then, each box predicts the classes that are related to the objects present in the box with a confidence score. We run the target DNN model offline and save the images of each channel in every layer. Then, we feed each image into the YOLO model to calculate its confidence score of object detection. We select the maximum confidence score among all the channels as the confidence score in an individual layer, and we repeat this process over all the layers. Finally, we select the *partition point* based on one of the two criteria: (1) the layer achieves sufficiently low confidence score, or (2) the layer reaches the cumulative memory requirement for the *local TEE*. The first layer that fulfills either standard is then treated as the *partition point* for the DNN model.

Let f_θ be the DNN model function under consideration. Let $\{l_i\}_{i=1}^N$ be the layers of the DNN and $\{c_i\}_{i=1}^C$ be the number of channels in each layer. Therefore, $\{\{X_{cl}\}_{c=1}^C\}_{l=1}^N$ are the images of respective layers and channels. Let $\{\phi\}_{l=1}^N$ be the maximum confidence scores generated at each layer and l_{TCB} be the cutoff layer with memory usage below TCB threshold. With $g_{\theta_l}(c) = f_\theta(l)(c)$ for all $1 \leq c \leq C$, we obtain the maximum confidence score of object detection among all channels in every layer i.e. ϕ_i.

$$\{\phi_i\}_{i=1}^N = \max_{1 \leq c \leq C} g_{\theta_l}(X_c) \tag{7}$$

Then, we find the layer with the minimum confidence score among all the layers, and compare this value to the TCB cutoff layer l_{TCB}. We then select the minimum value between the two as the *partition point* p_point.

$$p\_point = \min(\min_{1 \leq i \leq N} \phi_i, l_{TCB}) \tag{8}$$

B. Scale Invariant Feature Transform

In addition to object detection, we also employ SIFT [13] as another means of identifying the privacy exposure at intermediate DNN layers. SIFT is a feature detection algorithm to detect local features or keypoints in images, which goes through four key steps: *feature detection, feature matching and indexing, cluster identification*, and *model verification*. In *feature detection*, an image is transformed into a large

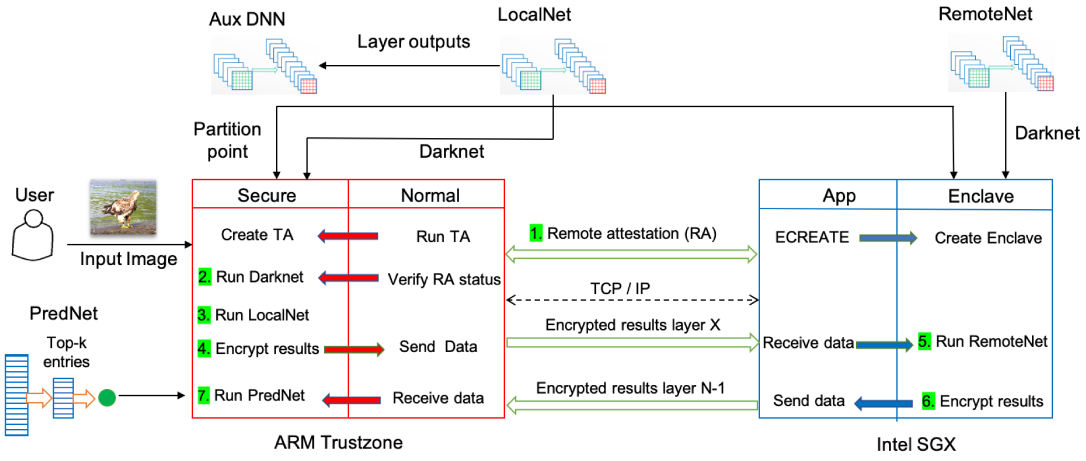

Fig. 3: System architecture and workflow of *HybridTEE*.

collection of feature vectors that are agnostic to image scaling or rotation. The features are detected using a staged filtering process to identify stable points. Each point is used to generate a feature vector that describes the local image region [13]. In *feature matching and indexing*, the SIFT keys for similar images are stored in a database to index a new object in the image. In *cluster identification*, an object is recognized in the new image by comparing to the features in the original database. Finally, *model verification* uses least-squares method to determine the closest parameters from the projected model locations to the corresponding image locations [13].

In order to determine the similarity between two images, the keypoints of both images are computed using a keypoint descriptor. We remove the unnecessary or approximate matches using the ratio test given by [13] and only select the matches with distance less than 0.75x of the feature distance in the original image. The number of matches obtained indicates the degree of similarity between the images. Figure 2 demonstrates the keypoints matched between the input image in the left half of the figure and the output generated at layer 1 channel 8 in the right half. The red dots indicate the features in the respective images, and the green lines indicate the matches detected.

Let $\mathbb{R}1$ and $\mathbb{R}2$ be the two input images represented by high dimensional vectors. Let d_θ be the function to compute and detect the keypoints in the image, and $\mathcal{Z}$ be the keypoint descriptor. Let g_θ be the function to detect the number of matches between the images using k-nearest neighbours, and *len* represents the length of the vector. We compute the keypoint descriptors of the input image $\mathcal{Z}1$ and the layer output image $\mathcal{Z}2$ using the SIFT keypoint detection function.

$$\mathcal{Z}1 = d_\theta(\mathbb{R}1) \tag{9}$$

$$\mathcal{Z}2 = d_\theta(\mathbb{R}2) \tag{10}$$

Then, we determine the similarity between features by comparing the k-nearest neighbours of $\mathcal{Z}1$ and $\mathcal{Z}2$.

$$\mathcal{Z}^* = g_\theta(\mathcal{Z}1, \mathcal{Z}2, k) \tag{11}$$

Finally, we use the ratio test to remove approximate matches and determine the number of features matched, which represents the degree of similarity ϵ.

$$\epsilon = len(\mathcal{Z}^*) \tag{12}$$

In order to determine the *partition point*, we set a threshold on the degree of similarity $\mathcal{T}$. The layer at which $\epsilon_l < \mathcal{T}$ is selected as the *partition point* p_point.

VI. *HybridTEE* System Design and Implementation

The key goal in the design of *HybridTEE* is to bridge the two TEEs and jointly accomplish the offloading of DNN execution in a secure manner. Figure 3 shows the architecture and system workflow of *HybridTEE*, which consists of the *Local TEE* (TrustZone) and the *Remote TEE* (SGX) with a secure communication channel. We adopt two solutions, namely *remote attestation* and *symmetric encryption* to conduct a secure handshake between the local and remote TEEs and establish the secure communication channel for DNN offloading. First, we employ *remote attestation* [10] to authenticate the SGX enclave to eliminate the possibility of *local TEE* communicating with a fake enclave staged by an adversary. The offloading of the DNN execution can be initiated if and only if the remote attestation procedure has been completed successfully. Also, all the communications shared between the two TEEs are encrypted to ensure confidentiality.

In our *HybridTEE* prototype, we adopt OP-TEE [12] as the *Local TEE*. OP-TEE is an ARM TrustZone-based open source TEE that is designed as companion to a non-secure Linux kernel [12]. It implements the TEE Internal Core API that acts as the secure world for trusted applications, and the TEE client API that acts as the secure monitor. We use Darknet [14] as the DNN implementation. We create a new trusted application (TA) with a universally unique identifier (UUID) for the Darknet code/data. The normal world OS can only access the application using the UUID of the TA, which is equivalent to the NS (non-secure) bit in TrustZone [4]. The entire layer creation and computation code resides in the TA, and it is encrypted before being shared with the rich OS.

We use an open source cryptographic library for AES-GCM encryption with tag validation [15] and a 128-bit symmetric key. The key has been shared between the two TEEs using SIGMA [16] key exchange protocol before the offloading session begins. Once shared, the key is hard coded in the secure world of TrustZone and the enclave of SGX, which cannot be accessed by attackers. Also, we implement a lightweight attestation procedure in our prototype considering the limited computation resources in TrustZone. In the remote attestation, an enclave ID is hardcoded in the application running in both TrustZone and SGX. A pseudorandom token is generated by TrustZone and sent to SGX. SGX appends its enclave ID to the token, encrypts the combined string using the previously shared key, and sends it back to TrustZone. TrustZone decrypts the data and verifies the token and Enclave ID. The remote attestation process is completed if the verification is successful.

VII. Experimental Results

We use a Raspberry Pi 3 Model B board with Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1 GB RAM as the mobile device. It runs OP-TEE [12] with a Linux OS as the *local TEE*. Also, the *remote TEE* is deployed on a server with 4-core Intel Core i7-6700 3.4GHz CPU under the SGX hardware mode. We conduct experiments using 4 pre-trained DNN models (i.e., Darknet19, VGG-16, Resnet152, and GoogLeNet) from [14] and 5 test images (i.e., Eagle, Dog, Cat, Horse, and Giraffe) from [14] and [17].

A. Security and Privacy Evaluation

In *HybridTEE*, the model configuration and weight vectors are stored in the local and remote TEEs and, therefore, the hardware-based isolation provided by TEEs ensures that the attacker does not have access to such information to reconstruct the neural network. From the privacy perspective, the partitioning algorithms in *HybridTEE* ensure that the input image sent to the *remote TEE* has minimum information exposure about the privacy-sensitive input data from the user. We define the degree of information exposure based on (1) the relative ability of the auxiliary DNN model to detect the object in the data sent to the server; and (2) the degree of similarity between the input image and the layer output image, determined by the number of SIFT features matched between these images.

TABLE I: Confidence scores in the auxiliary DNN-based partitioning for the first 8 layers of the Darknet19 model.

Image	L1	L2	L3	L4	L5	L6	L7	L8
Eagle	0.9989	0.9860	0.9716	0	0	0	0	0
Dog	0.9976	0.9976	0	0	0	0	0	0
Cat	0.9965	0.9963	0.9951	0	0	0	0	0
Horses	0.8172	0.8362	0.6105	0	0	0	0	0
Giraffe	0.9993	0	0	0	0	0	0	0

In the auxiliary DNN method, we assume that with a non-zero confidence score the adversary can recover the original input using this model. In other words, we set the confidence score threshold at 0 for partitioning, to provide the strongest privacy guarantee. Table I shows the confidence score for the Darknet19 model with input image size 448×448×3, which drops to 0 at the output of layer 4 for all the images. It indicates that the object detection algorithm is not able to detect any image from all channels of layer 4 for all the 5 images under evaluation. Therefore, we select layer 4 as the *partition point*. Furthermore, we observe in the experiments with smaller image sizes (e.g., 128×128×3) that the object detection algorithm is not able to detect the output images in most of the layers of the 4 models. Therefore, in order to corroborate our observation in this auxiliary DNN method, we further employ the SIFT-based method to find the *partition points* for these models.

Figure 4 shows the percentage of matched SIFT features between the input image (with size 448×448×3) and the intermediate images of the 4 DNN models. Let $good\_features_{layer\_img}$ be the features of the layer output image with distance less than 0.75x of the feature distance in the original image, and $features_{input\_img}$ be the total number of features in the input image. The percentage value is

$$percent\_match \leftarrow \frac{length(good\_features_{layer\_img})}{length(features_{input\_img})} \times 100$$

Based on the different thresholds of feature matched values, we could select different layers as the *partition point*. As shown in Figure 4, we select layer 3, layer 2, layer 1, and layer 2 as the *partition points* for the 4 models when the threshold is 5%. In our further experiments with smaller-size images (e.g., 128×128×3), the SIFT percentage values in most layers are less than 1%, which is consistent with what we have observed in the auxiliary DNN method. Therefore, we suggest employing the larger-size images (e.g., 448×448×3) to determine the *partition points* in both partitioning methods for stronger privacy protection.

B. Performance Evaluation

Table II shows the execution time (in seconds) of *HybridTEE* by applying the *partition points* based on the SIFT results in Figure 4. The execution time consists of the total model execution time and the communication time between the *local TEE* and the *remote TEE*. We observe that *HybridTEE* achieves speedup (1.06x to 6.20x) compared to the non-TEE baseline (i.e., model execution on the local mobile device without the protection of TEE) for Darknet19 and GoogLeNet. The image size we use in the timing evaluations is 128×128×3 due to the resource limitations of the mobile device. Note that we are not able to run the baseline versions of VGG-16 and Resnet152 due to their high memory requirements, which further justifies our idea of using the *HybridTEE* approach.

VIII. Related Work

The recent developments of heterogeneous TEEs [18], [19], [20], [21] have enabled the opportunity of secure offloading the computation-intensive workload to accelerators (e.g., FPGA

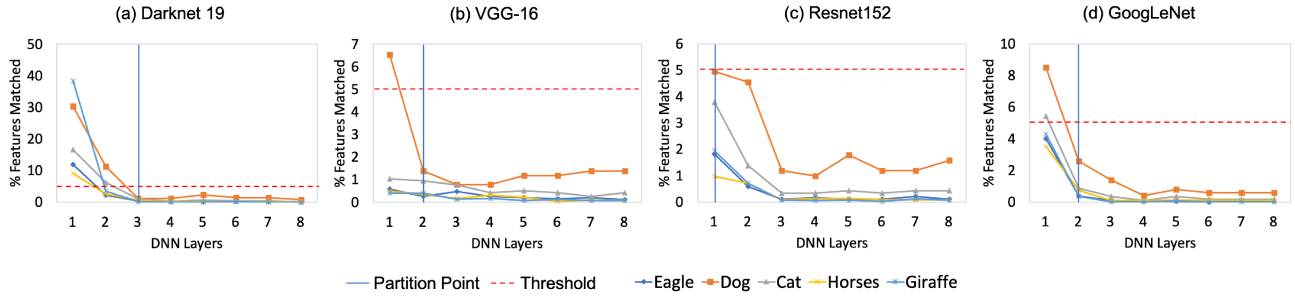

Fig. 4: Percentage of matched SIFT features between the input image and the images generated at layers 1 to 8 of the 4 DNN models.

TABLE II: Performance evaluation for *HybridTEE*.

DNN model	Baseline (sec)	HybridTEE (sec)	Speedup
Darknet19	20.85	19.64	1.06x
GoogLeNet	30.94	4.99	6.20x

and GPU) that are still under the protection of TEE. However, the size of TCB in the heterogeneous TEEs must be reduced to avoid potential security vulnerabilities. To minimize the workload of TEEs, researchers have developed several TEE-oriented partitioning techniques [6], [7], [8], [9] to partition and deploy the security-sensitive and non-sensitive portions of the workload inside and outside the TEE, respectively. However, they pose a high security requirement on the partitioning algorithm, which may result in enlarged attack surface in the partial workload deployed outside the TEE. Different from the prior works, *HybridTEE* for the first time bridges two TEEs to reduce the workload of the local mobile TEE for secure DNN execution, together with the privacy consideration.

IX. Conclusion

We have developed *HybridTEE*, a novel hardware-based security framework with a combination of TrustZone and SGX TEE platforms to securely execute DNNs. We studied the layer-wise privacy exposure of the DNNs regarding the input data and devised an offline partitioning strategy based on object and feature detections. Our experiments on real hardware systems demonstrated that *HybridTEE* ensured the security and privacy of the DNN models, while speeding up the execution time compared to the unprotected mobile DNN execution. We have released the source code of *HybridTEE* [22] to motivate further research in the community.

Acknowledgment

We appreciate the constructive reviews provided by the anonymous reviewers. This work was supported in part by the National Science Foundation under Award CNS-1912593.

References

[1] K. Ota, M. S. Dao, V. Mezaris, and F. G. B. De Natale, "Deep learning for mobile multimedia: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 3s, pp. 34:1–34:22, 2017.

[2] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security Symposium (Security)*, 2016, pp. 601–618.

[3] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Network and Distributed System Security Symposium (NDSS)*, 2018, pp. 1–15.

[4] "ARM security technology: Building a secure system using TrustZone technology," 2009.

[5] P. M. VanNostrand, I. Kyriazis, M. Cheng, T. Guo, and R. J. Walls, "Confidential deep learning: Executing proprietary models on untrusted devices," in *arXiv:1908.10730*, 2019.

[6] Z. Gu, H. Huang, J. Zhang, D. Su, H. Jamjoom, A. Lamba, D. Pendarakis, and I. Molloy, "Confidential inference via ternary model partitioning," in *arXiv:1807.00969*, 2020.

[7] F. Mo, A. S. Shamsabadi, K. Katevas, S. Demetriou, I. Leontiadis, A. Cavallaro, and H. Haddadi, "DarkneTZ: Towards model privacy at the edge using trusted execution environments," in *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2020, p. 161–174.

[8] F. Tramèr and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *International Conference on Learning Representations (ICLR)*, 2019, pp. 1–19.

[9] M. Ye, J. Sherman, W. Srisa-An, and S. Wei, "TZSlicer: Security-aware dynamic program slicing for hardware isolation," in *International Symposium on Hardware Oriented Security and Trust (HOST)*, 2018, pp. 17–24.

[10] "Intel software guard extensions (SGX)," 2020, https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[12] "OP-TEE," https://www.op-tee.org.

[13] D. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision (ICCV)*, 1999.

[14] J. Redmon, "Darknet: Open source neural networks in C," http://pjreddie.com/darknet/, 2013–2016.

[15] M. Clark, "AES-GCM," https://github.com/michaeljclark/aes-gcm, 2016.

[16] H. Krawczyk, "SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols," in *International Cryptology Conference (CRYPTO)*, 2003, pp. 400–425.

[17] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[18] J. Zhu, R. Hou, X. Wang, W. Wang, J. Cao, L. Zhao, F. Yuan, P. Li, Z. Wang, B. Zhao, L. Zhang, and D. Meng, "Enabling privacy-preserving, compute- and data-intensive computing using heterogeneous trusted execution environment," in *arXiv:1904.04782*, 2019.

[19] M. Ye, X. Feng, and S. Wei, "HISA: hardware isolation-based secure architecture for CPU-FPGA embedded systems," in *International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.

[20] S. Volos, K. Vaswani, and R. Bruno, "Graviton: Trusted execution environments on GPUs," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018, pp. 681–696.

[21] R. Bahmani, F. Brasser, G. Dessouky, P. Jauernig, M. Klimmek, A.-R. Sadeghi, and E. Stapf, "CURE: A security architecture with customizable and resilient enclaves," *arXiv preprint arXiv:2010.15866*, 2020.

[22] "HybridTEE," https://github.com/hwsel/HybridTEE.