Bident Structure for Neural Network Model Protection

Hsiao-Ying Lin, Chengfang Fang and Jie Shi
Huawei International, Singapore

Keywords: Machine Learning Security, Deep Neural Networks, Learning Model Protection, Model Confidentiality.

Abstract: Deep neural networks are widely deployed in a variety of application areas to provide real-time inference services, such as mobile phones, autonomous vehicles and industrial automation. Deploying trained models in end-user devices rises high demands on protecting models against model stealing attacks. To tackle this concern, applying cryptography algorithms and using trusted execution environments have been proposed. However, both approaches cause significant overhead on inference time. With the support of trusted execution environment, we propose bident-structure networks to protect the neural networks while maintaining inference efficiency. Our main idea is inspired by the secret-sharing concept from cryptography community, where we treat the neural network as the secret to be protected. We prove the feasibility of bident-structure methods by empirical experiments on MNIST. Experimental results also demonstrate that efficiency overhead can be reduced by compressing sub-networks running in trusted execution environments.

1 INTRODUCTION

Artificial neural networks based machine learning techniques have achieved great advancements in many tasks, such as image classification, language model and speech recognition. To provide a faster inference services, models are deployed in end-user devices for many services or applications. For instance, deep learning based object detection and classification techniques have been deployed in image-based advanced driver-assistance systems.

However, deploying models in devices brings new challenge on model protection. It takes considerable amount of time and effort to train a learning model to its peak accuracy, especially for collecting training data and sophisticated training processes. It is important to protect those trained model against model stealing attacks.

There are two types of model stealing attacks. The first type attempts to learn the network parameters and structure by probing the runtime environment in order to obtain the ability of the train model. The second type tries to construct an equivalent model by adaptively querying the model (Tramér et al., 2016). Here we address the first type of attacks.

Previous studies took cryptographic approaches that applying secure computation for secure training and inference of machine learning models while keeping parameters in encrypted format (Chen et al., 2018; Jiang et al., 2018; Juuti et al., 2019). Another approach is to use trusted execution environments, such as Inter SGX (Olga et al., 2016; Gu et al., 2018; Tramér & Boneh, 2019), or TEE (Volos et al., 2018; VanNostrand et al., 2019). Both approaches introduce significant performance overhead.

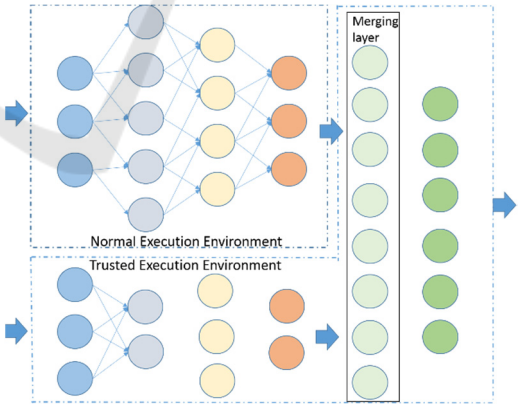

Figure 1: Bident structure based neural network.

We use a hybrid approach that combines the secret-sharing concept from cryptography community and trusted execution environments supported by commercial hardware, where the model is the secret to be protected. We propose the method of using bident structure based neural networks, bident networks in short, where inference can be

Lin, H., Fang, C. and Shi, J.
Bident Structure for Neural Network Model Protection.
DOI: 10.5220/0008923403770384
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 377-384
ISBN: 978-989-758-399-5; ISSN: 2184-4356

executed along two sub-networks in two separated environments with no intermediate interaction. As long as one cannot obtain both parts of the model, one is unable to reconstruct an effective model which has the same ability as the original one. An abstract network structure is illustrated in Figure 1. Each circle is a neural and each edge is a data flow path. The whole neural network consists of two sub-networks and they are glued by a merge layer where output from two sub-networks are merged. One part of the network is deployed in untrusted environment (also called normal execution environment.) Another part of the network including the final output is deployed in trusted execution environment.

To validate the feasibility of our proposed method, we conduct experiments on MNIST datasets by exploring several bident structure based neural networks. Experimental results not only prove the feasibility. They also show that the efficiency overhead can be reduced by compressing the sub-network in trusted execution environment while keeping the same accuracy.

Our main contribution is providing a model protection method by utilizing bident structures in neural networks and leveraging trusted execution environment while maintaining performance.

2 BACKGROUND

2.1 Deep Neural Network

For a classification service, at a high level, a deep neural network model takes an input, transforms it along the network layers of neurons and finally outputs the predicted class. It requires a training dataset and an initial deep neural network model to generate such a model. For each neuron, the computation is applying an activation function on the weighted summation of input. An example is shown in Figure 2. Two input a_1 and a_2 are summed up by weights w_1 and w_2 on the corresponding edges and then provided to activation function *f* to generate the output. Generating a neural network model is called training and using a trained model is call inference.

In this study, we focus on convolutional neural networks.

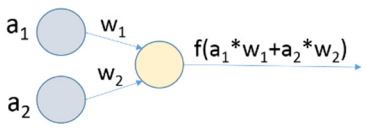

Figure 2: Computation of a neuron.

2.2 Trusted Execution Environment

A trusted execution environment is isolated from an untrusted execution environment. It is implemented by hardware-software co-design. The isolated environment guarantees integrity of the execution codes, privacy of the execution processes, and data confidentiality in secure memory. Available providers include Intel SGX, ARM TrustZone and Sanctum.

Operations in trusted execution environments are usually slower than ones in untrusted execution environment. Experimental results from (Tramér & Boneh, 2019) show that neural network operations in trusted execution environment is 50 times slower than in untrusted execution environment. Interactions between processes in two separated environment are also costly in terms of performance. Secure memory that can only be accessed from trusted execution environment is usually limited. For instance, the secure execution environment OP-TEE based on ARM TrustZone allocates 7MB secure memory while running neural networks requires much more memory (VanNostrand et al., 2019).

3 RELATED WORK

Previous studies took cryptographic approaches or used trusted execution environments.

Applying secure computation for secure training and inference of neural network models keeps the model in encrypted format (Chen et al., 2018; Jiang et al., 2018; Juuti et al, 2019). Specifically, weights of the model are encrypted and then inference is performed in encrypted format. Those cryptographic operations results in significant computation overhead.

Another approach is to use trusted execution environments. As described in previous section, trusted execution environments have performance constraints: less computation efficiency, limited memory space, and latency introduced by interaction between two environments.

To address the efficiency constraint, researchers divide the inference computation into two parts and outsource the part of matrix multiplications from trusted execution environment to untrusted one (Tramér & Boneh, 2019). Figure 3 illustrates the main idea. During executing the neural network, for each layer in the network, there is one matrix multiplication. Matrices are masked in trusted execution environment and then sent to untrusted execution environment where matrix multiplication is

done. The result is sent back and unmasked in trusted environment. This method reduced the cost of matrix multiplication in trusted execution environment. However, inference computation involves many interaction between untrusted and trusted execution environments. The interaction cost is linear in the number of neurons in the network.

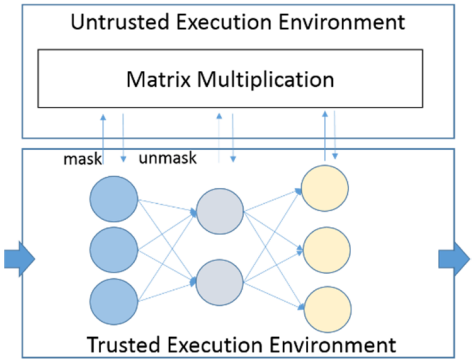

Figure 3: Outsource matrix multiplications.

To address the limited memory space constraint, researchers partition the neural network into several sub-networks and sequentially perform inference computation for each sub-networks in trusted execution environments (VanNostrand et al., 2019). Figure 4 illustrates the proposed partitioning method. The partition principle is that computation of each partition can be completed in trusted execution environment within the memory space limitation.

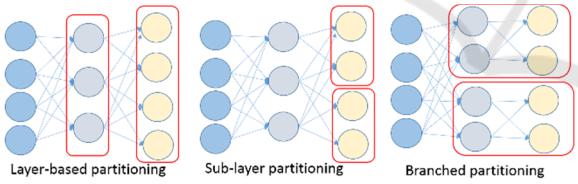

Figure 4: Partitioning types.

Above two methods address the model protection issue after the neural network model is made. We take another viewpoint to manage the model protection need. With the need in mind, a sophisticatedly designed neural network can be considered before the training phase, not after. It makes our method different from other similar approaches.

Bident network structures are not new in the areas of applying machine learning. They are applied to processing heterogeneous input for object detection in autonomous vehicles (Chen et al. 2017). However, it is the first time that it is used for protecting models.

4 BIDENT NETWORK

We assume that a trusted execution environment is available in deployed devices. Our idea is to build the model in a structure such that part of the model is running in trusted execution environment. Analog to secret sharing techniques in cryptography, the secret (i.e. the model) is firstly transformed into another format (i.e. bident network structure.) Later the transformed format is split into pieces and each piece is hold by a different entity. As long as over a certain number of pieces are available, the secret can be reconstructed back. Inspired by this concept, we propose bident network structure for designing the neural network. Figure 5 illustrates an example of symmetric bident network structure where the network consists of two sub-networks deployed in two environments separately. Both sub-network takes the same input and their output are merged by the merging layer.

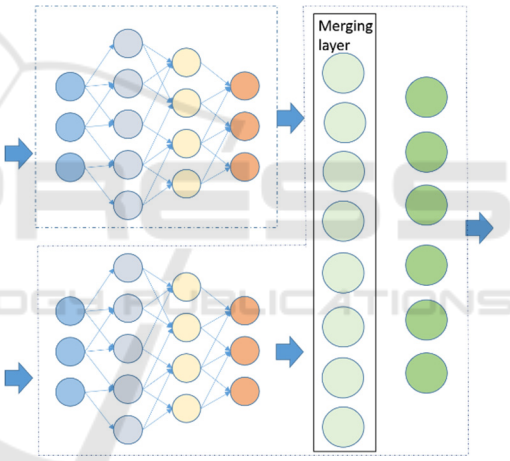

Figure 5: Symmetric bident network.

The generalized bident network structure takes input for two sub-networks without any limitation on how input are prepared. We focus on duplications here where input to sub-networks are identical.

By putting a sub-network in trusted execution environment, corresponding parameters are protected. By putting another sub-network in untrusted execution environment, computation cost is partially offloaded from the trusted execution environment. By the bident structure, sub-networks can be performed in parallel, so the interaction between two environments is massively reduced.

One may concern about the sub-network exposed in untrusted execution environment, the feasibility of the solution and the impact on the performance. We investigate them in the following.

With the bident network structure, when the sub-network in untrusted execution environment makes major influence on the result, it reveals too much information. To avoid this situation, we implement the merging layer by operations that both input make influence, such as multiplication, maximum and average operations.

To prove the feasibility of bident network structures, two issues need to be addressed. The first one is the potential overfitting issue since the network structure may be overkilled. We investigate this issue by conducting experiments where networks are fine-tuned.

The second issue is the offloading functionality. That is whether the computation in trusted execution environment can be reduced by the help of one in untrusted execution environment. We conducted experiments on asymmetric bident network to explore the opportunity. Figure 6 illustrates an example of asymmetric bident network structure. Our experiments are conducted to show that it is feasible to find a bident network structure where the resulting accuracy is above 99% and the sub-network in trusted execution environment can be slightly reduced.

To observe the impact on performance, we analyse experimental results of variant networks in the next section.

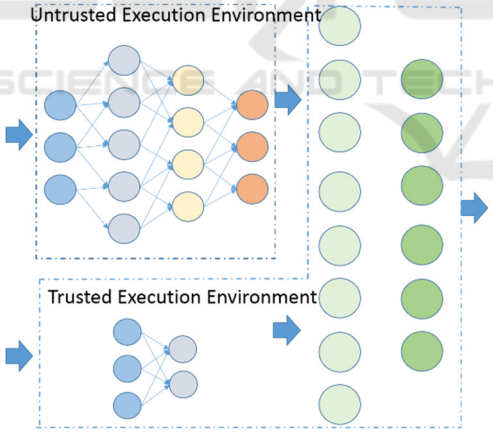

Figure 6: Asymmetric bident network.

5 EMPIRICAL EVALUATION

To evaluate the feasibility and performance, we conduct experiments on National Institute of Standards and Technology (MNIST) dataset, a handwritten classification dataset. Firstly, we train a base model with at least 99% accuracy as a baseline for the network configuration and the number of epochs.

We then evaluate a symmetric bident network structure with concatenation operation to test the water. Taking one more step, we evaluate symmetric bident network structures with other merging operations, i.e. multiplication, maximum and average. Finally, in order to see if a sub-network can be reduced, we evaluate asymmetric bident networks.

5.1 Setup

Our experiment dataset is MNIST, where the training and test datasets contain 60,000 and 10,000 samples respectively. The training dataset is divided into two subsets with 50,000 samples and 10,000 samples for training and validation.

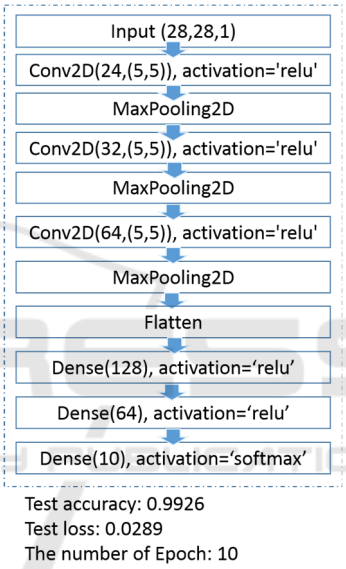

Figure 7: Base convolution neural network.

Each experiment has two major steps. The first one is to set the batch size as 32, train the model with 30 epochs, and find the suitable number of epochs. A suitable number of epochs is where the accuracy is over 99% and there is no overfitting sign. Here we consider the overfitting sign is that when the number of epoch is increasing, accuracy of training dataset gets better and one of validation dataset gets worse. The second step is to train a model based on the original training dataset of 60,000 samples with the selected number of epochs and test on test dataset of 10,000 samples.

Over all experiments, the optimizer is the Adam optimization algorithm and the loss function is categorical cross entropy. We start from a basic convolution neural network (CNN) shown in Figure 7. This neural network model is modified from

LeNet-5 (LeCun et al., 1998) by adding one more pooling layer, replacing activation functions as rectified linear unit (ReLU) and tuning detailed parameters. The number of epochs is set to 10 since it results in 99% accuracy without overfitting sign. Figure 8 shows how the number of epochs is selected. Dotted lines give the result of training and solid lines give the result of validation. After 10 epochs, validation loss and accuracy do not continue improving as training loss and accuracy do. It shows that more than 10 epochs may be overfitting.

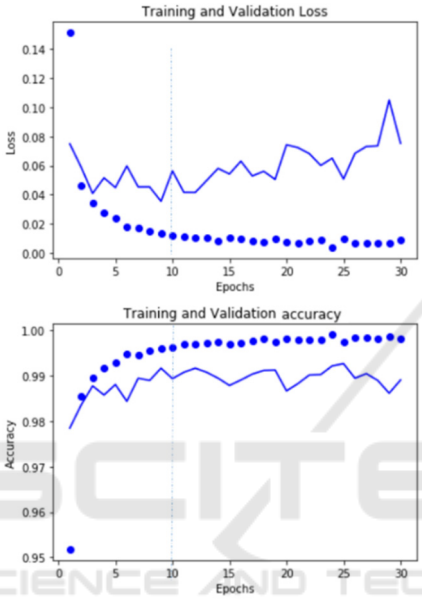

Figure 8: Training process of base CNN.

5.2 Experiments

We consider four different implementations for the merging layers. They are concatenation, multiplication, maximum and average. For each of them, we conduct a pair of experiments for symmetric and asymmetric bident networks. Among all networks in our experiments, activation functions are all rectifiers except for the final dense layer which uses softmax. For each experiment, we measure accuracy, loss and the required number of epochs for training.

The first pair of experiments use concatenation operation for the merging layer. Experimental results of symmetric and asymmetric networks are shown in Figure 9 and Figure 10, respectively. The left sub-network is in untrusted execution environment while the right one is in trusted execution environment. The colored boxes and numbers in bold font indicate major differences from the base model. Those differences are made in order to achieve 99% accuracy without overfitting sign.

There are several ways to mitigate overfitting, such as reducing the network size, using weight regularization and applying dropout technique. Here we take the approach of reducing the network size by using less layers and/or less neurons.

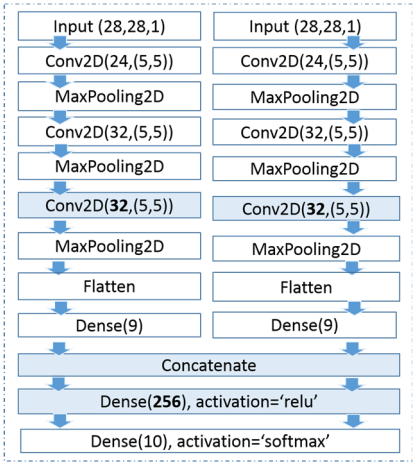

Figure 9: Symmetric bident network with concatenation.

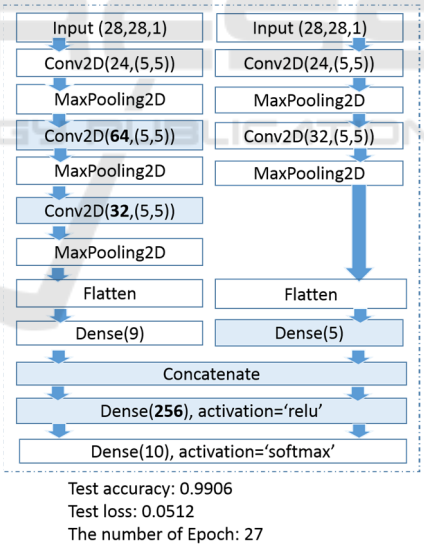

Figure 10: Asymmetric network with concatenation.

We then implement the merging layer by using multiplication. The results of symmetric and asymmetric bident networks are shown in Figure 11 and Figure 12, respectively.

Similarly, we implement the merging layer by using maximum and average operations for both symmetric and asymmetric networks, shown in Figure 13 and Figure 14, respectively.

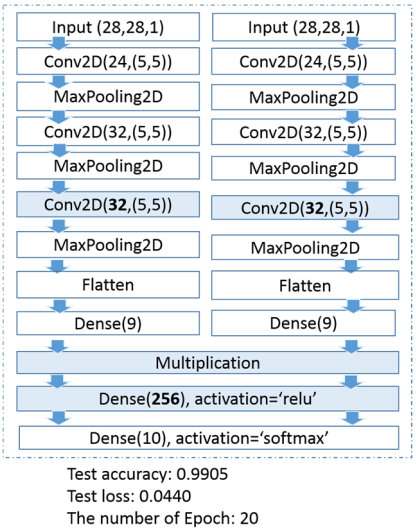

Figure 11: Symmetric bident network with multiplication.

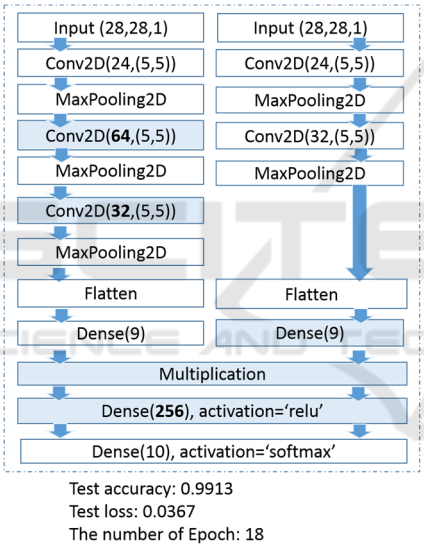

Figure 12: Asymmetric bident network with multiplication.

Our approach of making asymmetric bident networks is to reduce layers in trusted execution environment as the asymmetric one with concatenation. We tried to reduce more layers but the resulting accuracy cannot fulfil our requirement.

There are other ways of forcing both sub-networks jointly to decide the final result. For instance, weights on edges from sub-networks to the merging layer can be pre-configured significant enough.

5.3 Summary

We summarize experimental results of epochs in Table 1 where 99% accuracy is guaranteed.

The performance overhead of the training phase can be observed by the number of epochs. Table 1 shows training cost when using bident networks. The comparison between the base model and bident networks present the training overhead. In most cases, training overhead is significant. However, training is usually performed offline, we consider the overhead is acceptable in most scenarios.

The comparison between symmetric and asymmetric bident networks gives no tendency on which one having higher training overhead. Networks with concatenation and maximum have higher training cost in asymmetric structure. Yet networks with multiplication and average have higher training cost in symmetric structure.

The performance overhead of inference phase is well-maintained by the number of layers in the network.

Although the experiment scale is limited, results still provide an initial positive evidence on the feasibility.

Table 1: Required Epochs for Different Structures.

	Base	Concat	Multi	Max	Avg
Symmetric	10	10	20	10	25
Asymmetric	N/A	27	18	25	12

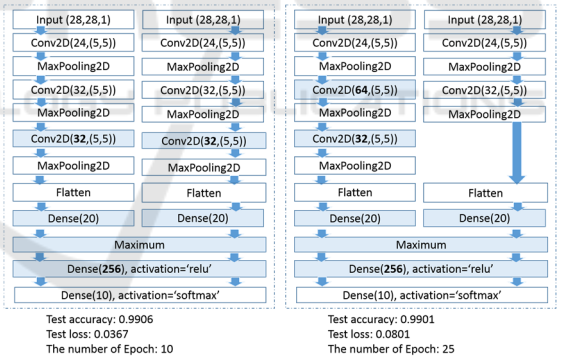

Figure 13: Bident networks with maximum.

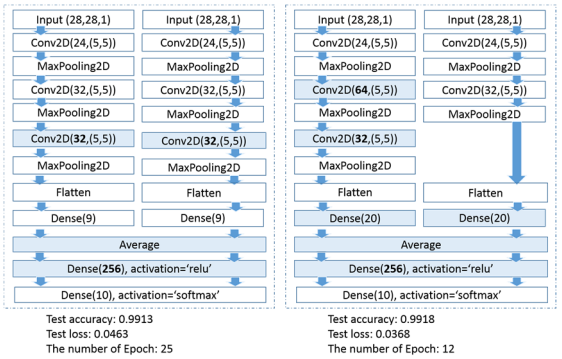

Figure 14: Bident networks with average.

6 DISCUSSION

From experimental results, we validate the feasibility of our idea on bident network structure. For each considered structure, the trained model can achieve at least 99% accuracy. In addition, results of using asymmetric network structures show that offloading computation from trusted execution environment to untrusted execution environment is doable.

However, the training overhead varies among different operations. We evaluate the performance overhead for the training phase and the inference phase by the required number of epochs and the layers in the model. We consider the overhead of the training phase is more acceptable than one of the inference phase. The required numbers of epochs indeed are increased, so it takes more time to accomplish the training processes. We do not change the total numbers of layers among models in experiments, so the real-time performance (the inference phase) remains.

Bident network structures protect the model by embedding it into two different environments. This method does not prevent the query-based model stealing attack. To fully protect the model, a complementary protection is preferred, such as limiting the query throughput or detecting query-based attacks.

7 CONCLUSION

As trained models are crucial intelligent properties for deep learning based applications, we propose the use of bident network structure to protect model confidentiality. By dividing the neural network into two sub-networks and minimizing their intermediate interaction, each sub-network is deployed in a different environment. As long as one cannot obtain parameters from both environments, one cannot reconstruct the model. Experimental results of difference bident network structures on MNIST dataset show the feasibility with low performance overhead of the inference phase.

We list some potential directions to explore more related to this research.

- To validate the feasibility of bident networks in general, more experiments are required on different datasets, different types of input data (such as texts instead of images), different ways of inputting data into sub-networks (such as dividing instead of replication), and different types of models (such as recurrent neural networks instead of CNN.)
- To quantify the impact on confidentiality, more investigations are required to analyze the information entropy on the sub-network.
- In addition to trusted execution environment in devices, bident network structures are potentially applicable to machine learning-as-a-service cloud with multi-server structures where each server holds a sub-network. As an extension, bident network structures can be expanded to trident-network, quadruplet-network, and more.

REFERENCES

Florian Tramér, Fan Zhang, Ari Juels, Michael K. Reiter and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs. In *Proceedings of the 25th USENIX Security Symposium, pp.*601-608

Xuhui Chen, Jinlong Ji, Lixing Yu, Changqing Luo and Pan Li. 2018. SecureNets: Secure Inference of Deep Neural Networks on an Untrusted Cloud. In *Proceedings of the 10th Asian Conference on Machine Learning 2018,* 646-661

Xiaoqian Jiang, Miran Kim, Kristin E. Lauter, and Yongsoo Song: 2018. Secure Outsourced Matrix Computation and Application to Neural Networks. In *Proceedings of 2018 ACM SIGSAC Conference on Computer and Communications Security*.1209-1222

Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan 2019. PRADA: Protecting Against DNN Model Stealing Attacks. *IEEE European Symposium on Security and Privacy 2019*

Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, Manuel Costa. 2016. Oblivious Multi-Party Machine Learning on Trusted Processors. *The USENIX Security Symposium 2016*: 619-636

Zhongshu Gu, Hani Jamjoom, Dong Su, Heqing Huang, Jialong Zhang, Tengfei Ma, Dimitrios Pendarakis, Ian Molloy. 2018. Reaching Data Confidentiality and Model Accountability on the CalTrain, *arXiv preprint arXiv:1812.03230*

Stavros Volos, Kapil Vaswani, Rodrigo Bruno. 2018. Graviton: Trusted Execution Environments on GPUs, *In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation*, 2018: 681-696

Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia.2017. Multi-view 3D Object Detection Network for Autonomous Driving. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017.* 6526-6534

Florian Tramèr, Dan Boneh. 2019. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. *ICLR 2019*

Peter M. VanNostrand, Ioannis Kyriazis, Michelle Cheng, Tian Guo, Robert J. Walls. 2019. Confidential Deep Learning: Executing Proprietary Models on Untrusted Devices. *arXiv preprint arXiv:1908.10730*

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner. 1988. Gradient-based learning applied to document recognition. *In Proceedings of the IEEE* 86 (11), 2278-2324.