Trusted-DNN: A TrustZone-based Adaptive Isolation Strategy for Deep Neural Networks

Zhuang Liu
Nankai University
Tianjin, China
frdm@foxmail.com

Ye Lu*
Nankai University
Tianjin, China
luye@nankai.edu.cn

Xueshuo Xie
Nankai University
Tianjin, China
xueshuoxie@mail.nankai.edu.cn

Yaozheng Fang
Nankai University
Tianjin, China
fyz@mail.nankai.edu.cn

Zhaolong Jian
Nankai University
Tianjin, China
jianzhaolong@mail.nankai.edu.cn

Tao Li
Nankai University
Tianjin, China
litao@nankai.edu.cn

ABSTRACT

Deep neural network (DNN) models have been widely deployed on embedded and mobile devices in lots of application fields such as health care, face recognition, driver assistance, etc. These applications usually require privacy or trusted computing protection. However, diverse hardware resources, various transport protocols, and limited computation and storage capacity make it challenging for traditional embedded systems to provide complex security protection mechanism oriented DNN models. To meet the challenges, we propose Trusted-DNN, a TrustZone-based adaptive isolation strategy for DNN models. We first design a normal pattern to exploit TrustZone technology to provide overall protection for running DNNs. To deploy arbitrary DNN models into TrustZone, we then develop a dynamic model partition method, which makes our strategy easily adaptive to various DNN models and devices. Finally, we employ several optimization techniques to reduce the inference latency of Trusted-DNN models. We perform AlexNet on OP-TEE, which is a TrustZone-based secure operating system, based on a Raspberry Pi 3B+ board. The extensive experimental results highlight that the optimized Trusted-DNN can reduce memory footprint by up to 98% compared with the ordinary program and Trusted-DNN only increase inference latency by 22.8%. Our code is available at https://gitee.com/PaintZero/alexnet-tee.

CCS CONCEPTS

• **Security and privacy** → *Embedded systems security*; *Hardware-based security protocols*; *Trusted computing*; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Deep neural network, ARM TrustZone, Hardware security, Embedded devices, Performance optimization

ACM TURC, July 30-August 1, 2021, Hefei, China

ACM ISBN 978-1-4503-8567-1/21/07…$15.00
https://doi.org/10.1145/3472634.3472652

ACM Reference Format:
Zhuang Liu, Ye Lu, Xueshuo Xie, Yaozheng Fang, Zhaolong Jian, and Tao Li. 2021. Trusted-DNN: A TrustZone-based Adaptive Isolation Strategy for Deep Neural Networks. In *ACM Turing Award Celebration Conference - China (ACM TURC 2021) (ACM TURC), July 30-August 1, 2021, Hefei, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3472634.3472652

1 INTRODUCTION

Deep neural networks (DNNs) have been widely applied in various mobile or embedded applications to accomplish inference tasks, such as object detection and recognition, health care, speech recognition and so on[10]. Many of these applications contain sensitive data that require security or privacy protection. For example, spiteful attackers may steal users' privacy input data contained in a medical device, or attackers may manipulate the inference result of a DNN model on cars to trigger traffic crashes. However, with limited computation and storage capacity, the embedded or mobile devices cannot effectively utilize complex security algorithms to ensure the security of DNN models.

Hardware-based trusted execution environments (TEEs), such as ARM TrustZone[1], are considered as a promising solution for embedded security issues. TrustZone can create a secure hardware-isolated world from the normal world. Programs executed in secure world cannot be manipulated by any other software in normal world, including the operating system. By deploying DNN models into secure world of TrustZone, we can thoroughly protect both the model privacy and integrity.

However, there are several challenges to implement DNN model deployment into TEEs. First, the memory of TEEs usually are limited to maintain a small attack surface. For instance, the secure world operating system we used, OP-TEE[8], has only 8 MB of memory, which makes it hard to run large DNN models, such as AlexNet with about 240 MB of parameters[7]. Second, there are lots of differences in both the secure world memory space of different devices and the structures of different models. It is hard for the deploying methods to be competitive when facing new devices or models. Third, normal world can only communicate with secure world via shared memory, and the context switching between the two worlds has high time costs. These factors make it difficult to maintain desired performance of DNN models in TEE.

To this end, in this paper, we propose an adaptive isolation strategy to provide effective security protection based on TrustZone for

various DNN models and devices. Our experiment results prove the effectiveness of the strategy. There are fourfold main contributions as follows:

- We design an overall model protection strategy based on TrustZone and encryption algorithms oriented the security issues of embedded devices. The strategy protects the privacy and integrity of user input data, DNN model weights and the output data, and all computations in Trusted-DNN are guaranteed.
- We propose a dynamic model partition method to meet the memory challenge. This method fits the available memory space by cutting the model weights into segments. The memory footprint of AlexNet through the method can be reduced by up to 98%.
- We develop a special computation method for convolutional layers in order to overcome the adaptivity challenge, so that our partition method can be independent both on the DNN models and the memory of devices.
- we employ several optimization techniques such as model compression and strategy adjustments to further improve Trusted-DNN performance. The inference latency of Trusted-DNN is significantly reduced by 70% approximately after using these optimization.

2 BACKGROUND AND RELATED WORK

ARM TrustZone. TrustZone[1] is a kind of TEE technology and another is Intel SGX. TrustZone mainly aims for embedded and mobile devices and it has been widely utilized in these devices. By extending a secure bus to processors, TrustZone can run two independent execution environments, the normal world and secure world, on the same processor through time division, and all memory and peripherals are divided between the two environments. Software in normal world can only access to a subset of memory and peripherals while secure world have full access including secure world memory. These two environments are securely context switched by TrustZone hardware mechanisms, thus programs executed in secure world can be well protected.

Deep neural networks. DNN is a machine learning method by simulating the structure and function of human brain. DNNs have lots of neurons organized in layers and there are connections between layers. By multiplying all input neurons values by weights and adding them to neurons in the next layer, the network propagates forward and finally returns an inference result. In a popular type of DNNs called convolution neural networks (CNNs), there are two main kinds of layers, fully connected layers and convolutional layers. Each neuron in a fully connected layer is connected to all the neurons in the last layer and each connection has a weight. While in convolutional layers, convolution kernels are used to reduce the number of parameters. For an m×m size input, X, and an n×n kernel, K, the output can be calculated by the formula: $z_{u,v} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_{u+i,v+j} \times k_{i,j} + b \quad (0 \le u, v \le m - n + 1,\ no\ padding,\ strike = 1)$.

Related work. There have been efforts to protect DNN models based on TEE. VanNostrand et al.[13] try solving the memory-limited challenge with simple DNN-layer-based partition. However, their partition methods heavily depend on the structure of the model and they did NOT experiment on real devices. Tramèr et al.[12] outsources all linear layers in DNNs to an untrusted GPU and check the results in TEE by Freivalds' algorithm[2] so that the integrity or privacy are guaranteed. And Gangal et al.[3] combine TrustZone with Intel SGX which has more computing resources to get better performance. However, these methods are specifically designed for Intel SGX on cloud servers rather than embedded devices, and they all require additional hardware devices, which enlarges the attack surface of TEE.

3 THE PROPOSED STRATEGY

3.1 Threaten model

In this paper, we focus on a pre-trained DNN inference model deployed on an untrusted device. We assume that adversaries on the device has full control authority over the rich operating system and total access to the device's cache, memory and storage. One of the goals of adversaries is to steal the model's input and output data which contains users' privacy information. Another is to steal the weight data of a proprietary pre-trained model. Moreover, adversaries will try modifying data or computing results to give users wrong inference results.

Meanwhile, we also assume that there is a trusted execution environment supported by TrustZone on the device, which is called secure world. All the data and code in secure world won't be accessed or affected by any software in normal world. However, if any data leave secure world without encryption, they'll be exposed to adversaries at once. To provide a realistic context, we considered an existing secure world operating system, OP-TEE[8].

Our strategy aims to use such a trusted environment like OP-TEE to provide the DNN model with thorough protection including preventing adversaries from stealing input, output or weight data, and also from manipulating any computing value or code of the DNN model. To simplify our design, we reasonably assume that the DNN model provider has deployed the model onto the device in a secure way, and the model users have secure approaches to collect and process the input data. In short, our strategy's protection works from the input data's arrival until the output result leaves TEE.

3.2 Overall model protection strategy

According to our threaten model above, there are four main parts that need protecting in DNN: model code, input/output data, and weight data. In this subsection, we propose an overall model protection strategy to protect all these parts together.

Firstly, the DNN model code is encapsulated as a trusted application (TA) running in secure world. TAs will be signed and optionally encrypted with the RSA key from the build of the original OP-TEE core blob[8]. All signature and encryption operations are finished before TAs are installed on the device. When loading a TA, OP-TEE will check the signature and load it only if it passes the verification, which avoids adversaries' modifying. TAs' code will be executed in isolation from all software in normal world. It is TrustZone's trusted firmware that ensures that no adversaries could access or modify any data or code of TAs while executing.

Secondly, our strategy uses RSA encryption algorithm to protect a client's input data and the output data. Before invoking the Trusted-DNN application to make an inference, both the client

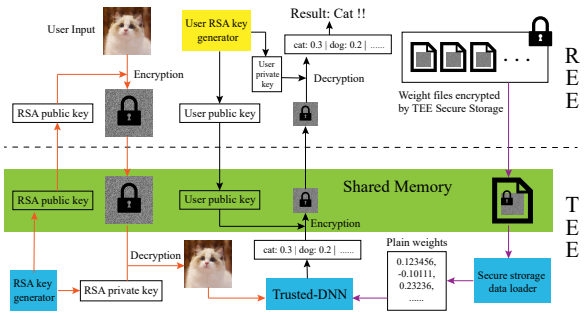


Figure 1: DNN inference procedure under overall model protection strategy, using image recognition model as an instance.

application and the TA generate a pair of RSA keys and send the public keys to each other. The client utilizes his own private key to sign the input data and utilizes TA's public key to encrypt. The cipher data will be later decrypted with TA's private key and verified with client's public key by TA. After the inference, there is a symmetrical process when sending back the output data to the client application.

Lastly, the weight data are protected by the secure storage mechanism provided by TrustZone. TrustZone has a multi-level key management unit based on a hardware unique key (HUK) which varies in different devices and can only be read by TEE. TEE will utilize HUK to generate a trusted applicant storage key (TSK) for every single TA. The DNN model's weight file is encrypted with a randomly generated AES key and the AES key will be encrypted with TSK so that only the specific TA can read the file.

Base on TrustZone and encryption algorithm, Trusted-DNN ensures that adversaries access no data stored in plain text and the computation won't give an inconsistent result. Thus, it can provide a thorough protection for all parts in DNN models.

3.3 Dynamic model partition method

In this section, we propose a dynamic model partition method which is independent of the structure of DNN models to address the memory-limited issue in TrustZone.

In fact, it's a pretty common phenomenon of shortage of memory resource in computer science. Modern computers usually use multi-level storage and memory paging to conquer it. So when it comes to TEE, we naturally try alleviating the shortage of secure world memory by utilizing secondary storage with much larger space, such as flash or disk. However, there are still problems when implementing this. First, operations on files in secure world are not as flexible as those in normal world. OP-TEE maintains a hash tree to handle data encryption and decryption of a secure storage file, but the tree itself takes lots of secure world memory space, which makes OP-TEE unable to handle files larger than 2 MB. Second, there is a time overhead to decrypt the secure storage files. We'll have to try to read the files as infrequently as possible to reduce the loss of performance.

For the first problem mentioned above, we partition the encrypted weight file into certain-size segments and the size can be

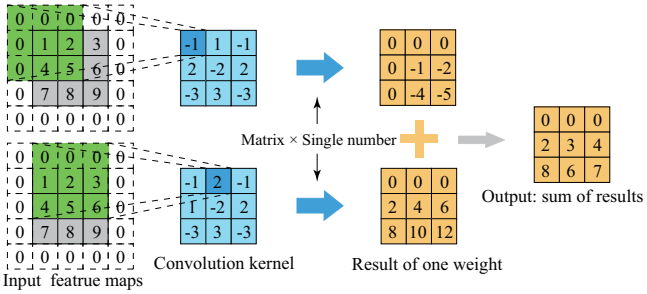


Figure 2: The first two steps of convolution in the dynamic model partition method. (3×3 input feature map, 3×3 weight kernel, padding = 1 and stroke = 1)

adjusted to meet the memory limitation before each run. At one time, only one segment will be loaded into a memory buffer of the same size. And the buffer changes its content when weight parameters in another segment are needed during the computation.

Meanwhile, for the second problem, we developed a special calculation method of convolutional layers in DNN to make it possible that we can complete an inference by traversing the weight file only once from front to back.

Instead of computing with a convolution kernel, we separately calculate all results related to one parameter in a convolution kernel and accumulate the results to the corresponding position in the output feature map one by one. Figure 2 demonstrates how our method works during a two-dimension convolution. As for the full connected layers and biases, the computation of parameters has been already separated. Thus, by means of this convolution method, we can utilize all weight parameters one by one and no parameters need a twice visiting. Benefiting from this, the file segments can be loaded only once by turn and we don't need a complex file map to handle them. Besides, the method has the same time complexity as the original convolution method when computing with only CPUs.

By combining dynamic model partition method with the overall model protection strategy, we develop an adaptive isolation strategy, which enables us to load any DNN model into TEE. And the strategy can make full use of the available memory space by properly adjusting the parameter buffer size to improve the computing performance. In other words, the strategy has very good adaptivity for both various DNN models and devices.

4 PERFORMANCE OPTIMIZATION

DNN models under the adaptive isolation strategy's protection usually take much longer time to make an inference than ordinary models, which maybe unsatisfactory for the users. To address the issue, in this section, we talk about two effective ways to enhance the inference performance of Trusted-DNN models.

4.1 Model compression

Model compression has been widely utilized when porting DNN models to embedded devices. There are different methods to compress DNN models, such as weights pruning, model quantification, knowledge distillation and low-rank decomposition. These methods can significantly reduce the amounts of a DNN model's weights and computation, and thus have quite good effects on performance

optimization. In this paper, we make a profound study on a binary quantification compression method[4].

The parameters of DNN models are usually stored as 32-bit floating point numbers. The quantification compression method simply saves all non-negative numbers as 1 and the others as 0 so that we can use just 1 bit to replace a 32-bit floating point number. In this way, the whole weight file shrinks to 1/32 of its original size and the time cost on encrypted file reading will be much less.

Model compression methods won't affect the security of Trusted-DNN. However, these methods usually have an influence on the inference accuracy. Therefore, user may need to retrain the model after quantification to fine tune the parameters for higher accuracy.

4.2 Strategy adjustments

As we have discussed in Section 3, our adaptive isolation strategy provides an all-around protection. However, in some realistic scenarios, maybe not all parts of DNN models are concerned. In this situation, we can adjust our strategy to exchange better performance with the security of some unconcerned parts.

As a common situation, when utilizing an open model with pre-trained weight parameters from the Internet, actually there is no need for encryption of the weight file. At this time, we can save all weights in a plain text file and optionally sign it with a key generates by TEE. Without decryption operations before data reading, the inference delay time can be greatly reduced.

Furthermore, there is a phenomenon that the intermediate values generated from different layers in a DNN model expose different amount of information. Gu et al.[5] analysed the input information leakages of every layer in a DNN model, and discovered that the front layers leak more information, while the back layers' leakages are much less. Similarly, Mo et al.[9] made a research on the leakages related to training data to prevent membership inference attacks, and their results show that the back layers contain most of the effective information. The degree of correlation between intermediate values and private data is called the sensitivity of the layer. Based on the sensitivity analyse methods propose by Gu, Mo, or others, we can partition a DNN model into two or more parts according to each layer's sensitivity. If one layer's sensitivity is larger than the threshold we set, it is deployed in secure world. Otherwise, normal world will handle it.

These adjustments have nothing to do with the inference accuracy. However, there may be a small amount of information leakage when utilizing the sensitivity based partition method and adversaries will be able to manipulate some computing processes to give a misleading result. So these adjustments can only be used when when the user confirms that he can bear the relevant security risks, if not, model compression will probably be a better choice.

5 EXPERIMENTAL EVALUATION

5.1 Experiment setup

Hardware and operating systems. On the first, we port OP-TEE 3.8.0 onto a Raspberry Pi 3B+ board. The development board provides us TrustZone hardware support and OP-TEE provides a secure isolated execution environment based on TrustZone. Our board has 1 GB memory in total and 8 MB of it are the secure world memory which can be used by TEE. For the normal world side, there is a Linux 4.14 system on the board and it can use the whole memory except for the secure part.

DNN model and dataset. We select a typical DNN model, AlexNet[7], and make two programs in C for Linux and OP-TEE respectively. And we utilized pre-trained weights of AlexNet from Caffe[6]. As for the dataset, we choose Tiny ImageNet, a subset from ImageNet[11], which has 200 categories.

Evaluation metrics. We use the following three metrics to evaluate the proposed strategy and its improvements.

- **I/O time**: time spent on decrypting and loading weight files.
- **Computing time**: time spent on other operations, obtained by subtracting the I/O time from the total delay time.
- **RAM usage**: minimum size of memory space required to run the program.

It needs to be claimed that **our strategy itself does NOT harm the models' accuracy**. Each calculation process in secure world should have the same results with normal world, unless there are programming mistakes. Specially, when optimizing with model compression, the accuracy depends on the compression method, which is not a main topic of this article. Therefore, we exclude accuracy from our evaluation metrics.

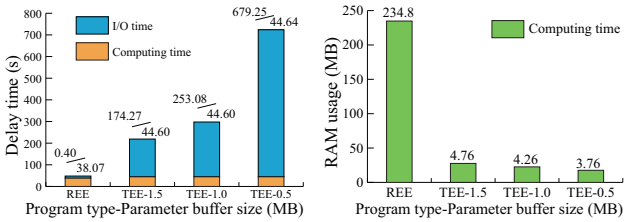
5.2 Performance of unoptimized Trusted-DNN

We've implemented an ordinary AlexNet application running in normal world, as well as a corresponding TA in OP-TEE. By running these two program on the same device, we compare the metrics to evaluate the performance of our strategy. In addition, to eliminate irrelevant interference factors, we do NOT use multi-threading or GPU heterogeneous computing methods.

Figure 3(a) indicates that the Trusted-DNN model takes much longer delay time to make an inference than the ordinary model. The total delay time of the Trusted-DNN model depends on the parameter buffer size. The smaller the buffer is, the longer delay time our strategy takes.

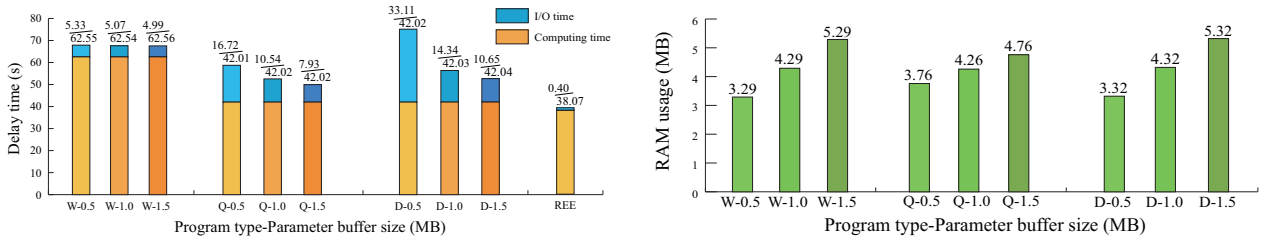
Under the best conditions we can achieve, the latency of Trusted-DNN is about 4.7 times more than ordinary programs. The computing time of the Trusted-DNN model has been very close to that of the ordinary program and it keeps stable, while the I/O time is much longer and it increases sharply as the buffer size decreases. This phenomenon is caused by a large number of decryption operations when reading weight files.

As for the memory, Figure 3(b) shows that our strategy can significantly reduce the RAM usage to less than 2% by means of



(a) Delay time comparison (b) RAM usage comparison

Figure 3: Inference performance comparison of AlexNet in different environments



(a) Delay time comparison

(b) RAM usage comparison

Figure 4: Inference performance comparison of AlexNet in secure world optimized by different methods

the dynamic model partition method. By the way, this method can also be utilized in some memory-limited scenarios in normal world to optimize ordinary DNN models.

5.3 Results after optimization

Long I/O time leads to long inference time of the Trusted-DNN model. To address this issue, we've proposed several optimization methods in Section 4. We apply three optimization methods in AlexNet TAs and compare them with the ordinary program to observe the optimization effects. For convenience, we use three letters to represent these three methods:

Method W: saving all weight parameters in a plain text file.

Method Q: binary quantification of the weights.

Method D: partition based on the sensitivity of DNN layers.

Please note that we don't analyse the layer sensitivity by ourselves for method D, but refer to the research results of Gu et al.[5]. We compute the last three fully connected layers of AlexNet in normal world and the others in secure world.

According to Figure 4(a), all the three methods can greatly reduce the delay time, especially the I/O time of the AlexNet trusted application. Compared to the original Trusted-DNN model with an 1.5 MB buffer size, method W takes 30.9% delay time, and method Q takes 22.8%, and 24.1% taken by method D. Furthermore, the delay time after optimization has been much closer to that in normal world. Compared with the ordinary program, the three methods with three different sizes of parameter buffer on average take 53.02% more delay time and at least 29.84% delay time overhead is taken when utilizing method Q with a 1.5 MB buffer. As for the RAM usages, the three methods won't affect the dynamic partition in our strategy, so the RAM usages still depend on the buffer size, while the difference between different optimization methods is unapparent.

After optimization, the performance of Trusted-DNN models becomes close to ordinary models, which allows us to provide DNN models TrustZone-based protection at a low cost.

6 CONCLUSION

We designed an adaptive isolation strategy isolation strategy based on TrustZone oriented DNNs, which is adaptive for various models or devices. We optimized the strategy respectively by binary quantification method, saving weights in plain text, and sensitivity-based partition. Experiment results indicate that after optimization, the strategy can provide protection with at least only 29.84% inference latency overhead compared to the ordinary AlexNet.

ACKNOWLEDGMENTS

This work is partially supported by the National Key Research and Development Program of China (2018YFB2100300), the National Natural Science Foundation (62002175), the Natural Science Foundation of Tianjin (19JCQNJC00600, 20JCZDJC00610), the Open Project Fund of State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences (CARCHB202016, CARCH201905), the Innovation Fund of Chinese Universities Industry-University-Research, Heterogeneous Intelligent Computing Project(2020HYA01003), and Sponsored by Zhejiang Lab (2021KF0AB04).

REFERENCES

[1] Tiago Alves and Felton Don. 2004. Trustzone: Integrated hardware and software security. *ARM White paper* 3, 4 (2004), 18–24.

[2] Rusins Freivalds. 1977. Probabilistic Machines Can Use Less Running Time.. In *IFIP congress*, Vol. 839. 842.

[3] Akshay Gangal, Mengmei Ye, and Sheng Wei. 2020. HybridTEE: Secure Mobile DNN Execution Using Hybrid Trusted Execution Environment. In *2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 1–6.

[4] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing Deep Convolutional Networks using Vector Quantization. *arXiv e-prints*, Article arXiv:1412.6115 (Dec. 2014), arXiv:1412.6115 pages. arXiv:1412.6115 [cs.CV]

[5] Zhongshu Gu, Heqing Huang, Jialong Zhang, Dong Su, Hani Jamjoom, Ankita Lamba, Dimitrios Pendarakis, and Ian Molloy. 2018. Confidential Inference via Ternary Model Partitioning. *arXiv e-prints*, Article arXiv:1807.00969 (July 2018), arXiv:1807.00969 pages. arXiv:1807.00969 [cs.CR]

[6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* (2014).

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012), 1097–1105.

[8] Linaro. Accessed on 2021. Open portable trusted execution environment. https://www.op-tee.org.

[9] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. 2020. Darknetz: towards model privacy at the edge using trusted execution environments. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 161–174.

[10] Kaoru Ota, Minh Son Dao, Vasileios Mezaris, and Francesco G. B. De Natale. 2017. Deep Learning for Mobile Multimedia: A Survey. *ACM Trans. Multimedia Comput. Commun. Appl.* 13, 3s, Article 34 (June 2017), 22 pages. https://doi.org/10.1145/3092831

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[12] Florian Tramèr and Dan Boneh. 2018. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. *arXiv e-prints*, Article arXiv:1806.03287 (June 2018), arXiv:1806.03287 pages. arXiv:1806.03287 [stat.ML]

[13] Peter M. VanNostrand, Ioannis Kyriazis, Michelle Cheng, Tian Guo, and Robert J. Walls. 2019. Confidential Deep Learning: Executing Proprietary Models on Untrusted Devices. *arXiv e-prints*, Article arXiv:1908.10730 (Aug. 2019), arXiv:1908.10730 pages. arXiv:1908.10730 [cs.CR]