Poster: Towards Characterizing and Limiting Information Exposure in DNN Layers

Fan Mo
f.mo18@imperial.ac.uk
Imperial College London

Ali Shahin Shamsabadi
a.shahinshamsabadi@qmul.ac.uk
Queen Mary University of London

Kleomenis Katevas
k.katevas@imperial.ac.uk
Imperial College London

Andrea Cavallaro
a.cavallaro@qmul.ac.uk
Queen Mary University of London

Hamed Haddadi
h.haddadi@imperial.ac.uk
Imperial College London

ABSTRACT

Pre-trained Deep Neural Network (DNN) models are increasingly used in smartphones and other user devices to enable prediction services, leading to potential disclosures of (sensitive) information from training data captured inside these models. Based on the concept of generalization error, we propose a framework to measure the amount of sensitive information memorized in each layer of a DNN. Our results show that, when considered individually, the last layers encode a larger amount of information from the training data compared to the first layers. We find that the same DNN architecture trained with different datasets has similar exposure per layer. We evaluate an architecture to protect the most sensitive layers within an on-device Trusted Execution Environment (TEE) against potential white-box membership inference attacks without the significant computational overhead.

CCS CONCEPTS

• **Computing methodologies** → **Distributed artificial intelligence**; • **Security and privacy** → *Distributed systems security.*

KEYWORDS

deep learning, privacy, training data, sensitive information exposure, trusted execution environment

1 INTRODUCTION

On-device DNNs have achieved impressive performance on a broad spectrum of services such as face recognition for authentication and speech recognition for interaction. However, DNNs memorize in their parameters information from the training data [8]. Thus, keeping DNNs accessible in user devices leads to privacy concerns when training data contains sensitive information.

Previous works have shown that a reconstruction of the original input data is easier from a DNN when using the layer's output (activation) for inference [1]. In addition to that, a speech or face recognition model deployed on devices can be attacked by membership inference attacks (MIA) [7]. This leaks the information about whether one audio or picture has involved in pre-training the model, which could further lead to other serious privacy issues.

We hypothesize that the memorization of sensitive information from training data differs across the layers of a DNN. We present an approach and show that each layer behaves differently on the data they were trained on compared to the data seen for the first time, by quantifying the generalization error (i.e. the expected distance between prediction accuracy of training data and test data [4]). We further quantify the risk of sensitive information exposure of each layer as a function of generalization error. The larger the generalization error, the easier it is to infer sensitive information from the training set. Our results show that last layers memorize more sensitive information about training data, and the risk of information exposure of a layer is independent of the dataset.

To protect the most sensitive layers from potential white-box attacks [2, 3], we leverage on-device TEE unit (Arm's TrustZone (TZ)), as an example of protection mechanism. Experiments are conducted by training the last layers in the TrustZone and the first layers outside the TrustZone. Results show that the overhead in memory and execution time is minor, thus making it an affordable solution to protect a model from potential attacks.

2 MEASURING INFORMATION EXPOSURE

2.1 Information Exposure Metric

Based on MIA, we define the exposure of private information of an algorithm as the difference between the results obtained on a database with and without the presence of one data record [5, 7]. Because of its similarity with the generalization error [4, 7], we can then apply the generalization error $\mathcal{E}^A$ to measure the exposure of private information of an algorithm A:

$$\mathcal{E}^A = \mathbb{E}_{z\in T}[\ell(A_S, z)] - \mathbb{E}_{z\in S}[\ell(A_S, z)]. \tag{1}$$

S and T are training dataset and testing dataset respectively. Let DNN models be A, and A_S be the model trained with S. $\ell()$ is the lost function. $z = (x, y)$ refers to data points. Each layer's output is the input of next layer until the end of the model, so $A(x) = \theta_L(...\theta_l(...\theta_1(x)...)...)$. θ_i is the layers.

To remove the private information in θ_l, we create a model $\mathbf{M_r}$ by fine-tuning θ_l as θ_{p_l} on S and T and by freezing the parameters of the other layers of A. During fine-tuning,

$$\theta_l^{(X)} \leftarrow \theta_l^{(X)} - \eta\delta_l^{(X)}. \tag{2}$$

CCS '19, November 11–15, 2019, London, United Kingdom
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6747-9/19/11.
https://doi.org/10.1145/3319535.3363279

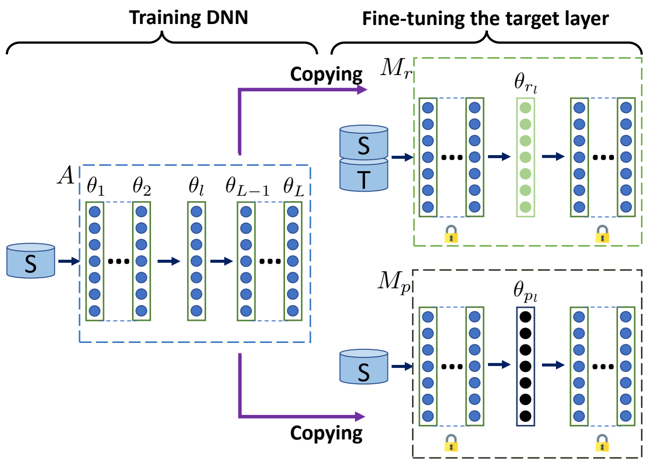

Figure 1: The proposed framework for measuring the risk of sensitive information exposure in a deep neural network A trained on a private dataset S. M_r and M_p are obtained by fine-tuning a target layer l.

$X \in \{S, T\}$. $\delta_l^{(X)}$ is the back-propagated error calculated using θ of the whole model and inputs. Therefore, $\delta_l^{(X)}$ is dependent on $\theta_{1:l-1}, \theta_l, \theta_{l+1:L}, X$. By learning on X, the $\theta_l^{(X)}$ memorises both S and T, which means that it is generalized and can not be utilized by MIA for distinguishing S and T.

To remove the θ_l which contains private information, we simply consider that after training for a considerable number of epochs (instead of resetting as random values), if training accuracy does not significantly increase, the initial parameters of $\theta_l^{(X)}$ only have a slight influence on its final parameters.

$\theta_{1:l-1}$ and $\theta_{l+1:L}$ are frozen during fine-tuning $\theta_l^{(X)}$. Let us define the functional relationship $\mathcal{E}^{\theta_l} = R_l(\theta_l)$. After fine-tuning, the $\theta_l^{(X)}$ can be presented as

$$\theta_l^{(X)} = v(R_{1:l-1}^{-1}(\mathcal{E}^{\theta_{1:l-1}}), R_{i+1:L}^{-1}(\mathcal{E}^{\theta_{l+1:L}}), X). \tag{3}$$

Equation 3 shows that the $\theta_l^{(X)}$ include information of X. However, private information of other layers $\mathcal{E}^{\theta_{1:l-1}}$ and $\mathcal{E}^{\theta_{l+1:L}}$ are also passed to $\theta_l^{(X)}$.

To exclude these passed private information from other layers, we create another model $\mathbf{M_p}$ by fine-tuning θ_l as θ_{p_l} using dataset S and by freezing all other layers of A. This lead to $\theta_l^{(X)}$ keep learning from S. We simply assume that the overfitting effect only sightly increase after training for a considerable number of epochs and the training accuracy does not significantly increase. Therefore, by comparing $\mathbf{M_r}$ and $\mathbf{M_p}$, we can remove passed information from $\theta_{1:l-1}^{(X)}$ and $\theta_{i+1:L}^{(X)}$ and obtain the exposure of private information of θ_l as

$$\mathcal{E}^{\theta_l} = \frac{\mathcal{E}^{M_p} - \mathcal{E}^{M_r}}{\mathcal{E}^{M_p}}. \tag{4}$$

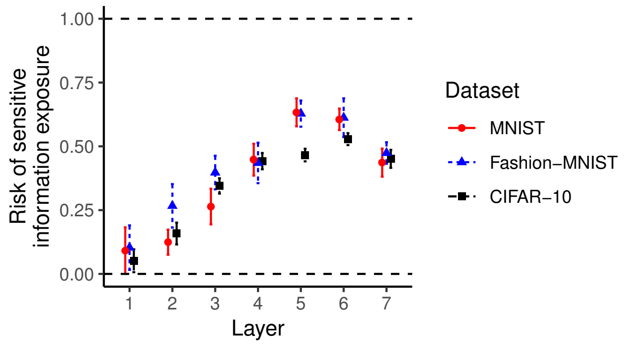

Figure 2: The risk of sensitive information exposure of VGG-7 per layer on MNIST, Fashion-MNIST and CIFAR-10. Error bars represent 95% CI.

2.2 Model and Datasets

We use VGG-7 [6] as the DNN A, which has six convolutional layers followed by one fully connected layer (16C3-16C3-MP-32C3-32C3-MP-32C3-32C3-MP-64FC-10SM). Each layer is followed by ReLU activation function.

We use three datasets: MNIST, Fashion-MNIST, and CIFAR-10. MNIST and Fashion-MNIST include 60k training images of 28×28×1 of 10 classes of handwritten digits and clothing respectively. CIFAR-10 includes 50k training $32 \times 32 \times 3$ images of 10 classes. We split each set into set S and set T. We use 20 epochs for MNIST, 40 epochs for Fashion-MNIST, and 60 epochs for CIFAR-10.

2.3 Results and Discussion

Figure 2 shows the risk of sensitive information exposure for each layer of VGG-7 on all three datasets. The first layer has the lowest risk, with the risk increasing as we go through the layers, with the last convolutional layer having the highest sensitive information exposure (i.e. 0.63 for both MNIST and Fashion-MNIST and 0.5 for CIFAR-10). The last layer is a fully-connected layer which has a lower exposure risk than its previous convolutional layer. In addition, the order of layers in terms of sensitive information exposure is almost the same across all three datasets.

3 ON-DEVICE TEE PROTECTION

3.1 Setup

In this section, we develop an implementation and evaluate the cost of protecting the last layers of an on-device DNN during fine-tuning by deploying them in the TrustZone of a device (see Figure 3). TrustZone establishes a private region on the main processor. Both hardware and software approaches isolate this region to allow trusted execution. We only protect the most sensitive layers of the model because the TrustZone's secure memory is limited and use the normal execution environment for the other layers.

We use Darknet [1] as DNN library and Open Portable TEE [2] as the framework for TrustZone on a Raspberry Pi 3 Model B. This device runs TrustZone with 16 mebibytes (MiB) secure memory.

[1]https://pjreddie.com/darknet
[2]https://www.op-tee.org

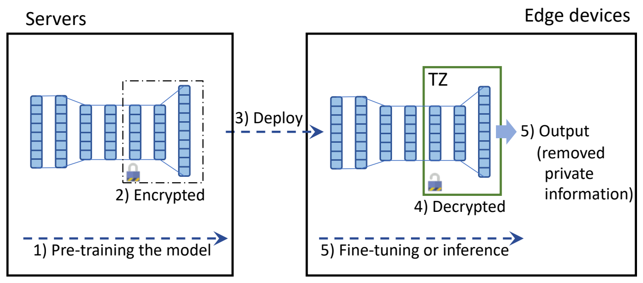
Servers
Edge devices
2) Encrypted
1) Pre-training the model
3) Deploy
TZ
4) Decrypted
5) Output (removed private information)
5) Fine-tuning or inference

Figure 3: Proposed protection for sensitive layers (last layers) of an on-device deep neural network using TrustZone.

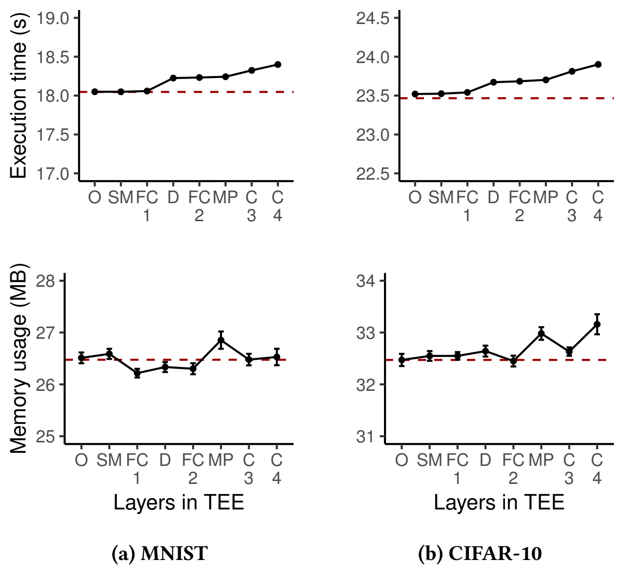

Figure 4: Execution time and memory usage for protecting layers of VGG-7 using the TrustZone. The x-axis corresponds to several last layers included in the TrustZone. *O*, *SM*, *FC*, *D*, *MP*, and *C* refer to the cost, softmax, fully connected, dropout, maxpooling, convolutional layers of VGG-7. Number of layers with trainable parameters in the TrustZone are 1, 2, 3, and 4. The dash line represent the baseline, which runs all the layers outside the TrustZone.

The choice of Darknet is due to its high performance and small dependencies. The developed implementation in our evaluation is available online [3]. We fine-tune the pre-trained VGG-7 with MNIST and CIFAR-10, respectively. Several layers are deployed in the TrustZone from the end, including both layers with (i.e. the convolutional and fully connected layer) and without (i.e. the dropout and maxpooling layer) trainable parameters.

3.2 Results and Discussion

Figure 4 shows the execution time (in seconds) and memory usage (in MB) of our implementation when securing a part of the DNN in the TrustZone, starting from the last layer, and continuing adding layers until the maximum number of layers the zone can hold. The resulting execution times are MNIST: $F_{(7,232)} = 3658, p < 0.001$; CIFAR-10: $F_{(7,232)} = 2396, p < 0.001$ and memory usage is MNIST: $F_{(7,232)} = 11.62, p < 0.001$; CIFAR-10: $F_{(7,232)} = 20.01, p < 0.001$. The increase however is small compared to the baseline (Execution time: 1.94% for MNIST and 1.62% for CIFAR-10; Memory usage: 2.43% for MNIST and 2.19% for CIFAR-10).

Specifically, deploying the dropout layer and the maxpooling layer in the TrustZone increases both the execution time and memory usage. The reason is that these two layer types have no trainable parameters, and for Darknet, the dropout and maxpooling are directly operated based on trainable parameters of their front layer. Therefore, to run these two types of layers in the TrustZone, their front layer (i.e. fully connected/convolutional layers) needs to be copied into the TrustZone, which increases the cost. For layers with parameters that we aim to protect (1, 2, 3, and 4 in Figure 4), deploying fully connected layers (i.e. 1, 2) in the TrustZone does not increase the execution time accumulated on the first layers, as well as the total memory usage. Deploying convolutional layers (i.e. 3 and 4) also leads to an increase of execution time. However, exhausting most of the available memory of the TrustZone can also cause an increase in overhead. Overall, for our implementation, protecting fully connected and convolutional layers have lower costs than other layers without trainable parameters with the TrustZone.

4 CONCLUSION

We proposed a method to measure the exposure of sensitive information in each layer of a pre-trained DNN model. We showed that the closer the layer is to the output, the higher the likelihood that sensitive information of training data is exposed, which is opposite to the exposure risk of layers' activation from test data [1]. We evaluated the use of TrustZone to protect individual sensitive layers (i.e. the last layers) of a deployed DNN. The results show that TrustZone has a promising performance at low cost.

Future work includes investigating the advantages of protecting the later layers of a DNN against, among others, white-box membership inference attacks [3].

REFERENCES

[1] Zhongshu Gu, Heqing Huang, Jialong Zhang, Dong Su, Hani Jamjoom, Ankita Lamba, Dimitrios Pendarakis, and Ian Molloy. 2018. YerbaBuena: Securing Deep Learning Inference Data via Enclave-based Ternary Model Partitioning. *arXiv preprint arXiv:1807.00969* (2018).
[2] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. IEEE.
[3] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Comprehensive Privacy Analysis of Deep Learning: Stand-alone and Federated Learning under Passive and Active White-box Inference Attacks. *arXiv preprint arXiv:1812.00910* (2018).
[4] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. 2010. Learnability, stability and uniform convergence. *Journal of Machine Learning Research* 11, Oct (2010), 2635–2670.
[5] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.
[6] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[7] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 268–282.
[8] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. France.

[3]https://github.com/mofanv/darknetp