Privacy-Preserving Machine Learning with Differential Privacy & Trusted Execution Environments

Akhil Sai Peddireddy, Ali Ahad
University of Virginia
ap3ub@virginia.edu

December 10, 2020

Abstract

The privacy preserving techniques coupled with trust computing platforms can offer end-to-end private and secure Machine Learning. Myelin is a state-of-the-art framework which allows the user to do this, with a combination of many techniques such as differential privacy, TEE, TVM & data oblivious algorithms. However, using this in real life applications is not trivial and we believe one-size-fits-all is not a desired approach as privacy often comes at a cost of speed or accuracy. In this work, we aim to do a fine grained analysis on the effect of each of the components of Myelin on accuracy, speed, and privacy and provide detailed insights on them. The goal is to offer a guide to make an informed decision regarding the choice of selections of environments, based on specific individual needs. Based on our results, we recommend that users concerned about privacy should port their models to TEEs with the help of TVM and suggest they consider adding differential privacy where protecting the privacy of training data is more important.

1 Introduction

Machine learning and Deep learning have become increasingly popular in this decade. These models are being widely used for many applications including those which are security and privacy sensitive [23]. This raises the importance of having robust and privacy aware machine learning pipelines to safely and securely use the models by avoiding the potential risks. These potential risks are becoming more significant due to the shift in the Machine learning processes from a static training and inference into a more streamlined way that includes directly gathering the data from multiple untrusted sources and performing the training and deployment on untrusted clouds. Several attacks are prevalent such as Training / Input / Output data being stolen or exploited [26], or Models (architecture/weights) being tampered or stolen from memory [15].

One of the ways to tackle these issues is to use Trusted Execution Environments (TEEs) such as Intel SGX [22]. SGX acts as a shield around memory and processing, which consists of trusted hardware enclaves that seperate the general application code with the security-sensitive code through an enclave boundary. This helps to allow secure computations and data access & storage on an untrusted system even while the non-SGX specific code remains virtually unchanged. However, using the Trusted Execution Environments alone is not sufficient as they are vulnerable to Model Inversion attacks [6], other side channel attacks that exploits the Memory Access Patterns [16].

Hence Trusted Execution Environments alone do not satisfy the privacy expectations for an End-to-end Privacy-preserving Machine Learning that safeguards data, code and computations. The solution is to combine trusted execution environments with techniques such as differential privacy, in order to mitigate these attacks. Myelin [10] is a state-of-the-art and first of its kind framework that combines differential privacy, trusted execution environments, TVM [4] and data oblivious algorithms [17], and supports end-to-end fully private training on multi sourced data from mutually distrusting parties.

The evaluation of Myelin however is limited and it does not provide an ablation study to understand the effects of each of the components on privacy, accuracy and speed. This is very important since incorporating privacy preserving pipelines is not trivial in real life scenarios due to the fact that privacy comes at a cost of either speed or accuracy. The users should be able to strike a clear balance of each of these tradeoffs, according to their own unique needs. We believe there is no gold standard in the selection of environments in this case, rather it is to be customized on a case by case basis. This motivated us to do a fine grained analysis on the effects of the individual components of Myelin and provide a guide along with insights and recommendations.

The remainder of this paper proceeds as follows: Section 2 gives the background, an overview of the results of Myelin and a discussion on the conditions that are to be studied further. Section 3 describes the design of our approach to obtain the results for these conditions. Section 4 evaluates the results and provides the insights. Section 5 discusses the challenges, limitations and other additional topics. Section 6 describes the related work in privacy-preserving ML and Section 7 concludes our work.

2 Background & Overview

2.1 Background

2.1.1 Privacy Expectations in ML

The privacy expectations associated with the Machine learning models can be broadly categorized into 4 types: First, Training data privacy - training data collector and aggregator adheres to the privacy expectancy. Second, Input data privacy - making sure the model input data remains private. Third, Model weights privacy - ensuring that the model weights are not tampered with. Finally, Output data privacy - preserving privacy of the sensitive output data.

2.1.2 Differential Privacy

Differential Privacy is a algorithmic framework for guaranteeing the privacy of input data through adding Gaussian noise at some point during training[2]. In Myelin [10], this noise is added to the gradient before back-propagation and in our testing environment the noise is added directly to the input data before the model is trained on it.

2.1.3 TVM

Apache TVM [4] is an open source deep learning compiler stack for CPUs, GPUs, and specialized accelerators. Myelin uses TVM for performance improvement in terms of efficiency. It aims to run models on any hardware backend (including SGX). Uses Relay Intermediate Representations (IR) to optimize and run computations efficiently.

2.1.4 Myelin

Myelin [10] is an integration of many means for achieving end-to-end privacy-preserving Machine Learning. For the privacy of data and computations, it uses Trusted Execution Environments. For the privacy of the training data and to avoid the exploits through Memory Access Patterns, differential privacy is used. These techniques however severely affect the speed and performance. The data should be transferred back and forth to the secure enclaves that increases the time required for the computations and differential privacy also adds an overhead for adding the noise to the data or the model. Hence, for the sake of efficiency, Myelin uses modular, optimized numerical libraries for machine learning computations, with the help of the Apache TVM framework. Another advantage of TVM is that it helps to directly enable the interactions with the hardware backend of Intel SGX. Myelin is observed to be 3x faster than cryptographic methods.

Hence, the Machine learning pipeline in the case of Myelin (Figure 1) will be as follows: The Model code written by the ML user will be converted into the Relay Intermediate Representations of TVM and added upon with the other techniques of the Myelin framework, it will then be moved to a trusted model training enclave. This trusted model along with the data from multiple providers is trained in an enclave-enabled untrusted cloud, thereby finally producing the privacy-preserving trained model.

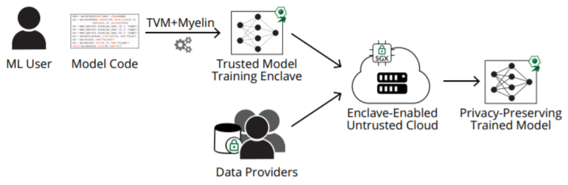
ML User
Model Code
TVM+Myelin
Trusted Model Training Enclave
SGX
Enclave-Enabled Untrusted Cloud
Privacy-Preserving Trained Model
Data Providers

Figure 1: Myelin in action

2.2 Overview

Since Myelin [10] is a first of its kind work, there is no benchmark or baseline to compare with. Hence the results of Myelinated models trained without differential privacy were compared against the other works, in this area on trusted hardware but without Differential Privacy, like Gazelle [11], Chiron [9] and Slalom [24]. This also includes the compact models such as MobileNets [8] but were done only for inference and not training. Another comparison presents the results of a fully private model of Myelin against non-private CPU baseline. We are specifically interested in the second comparison and we believe there is a lot more to study in depth over here.

The results available in the Myelin paper, were only for baseline with no SGX or TVM or DP, compared against fully private Myelinated models. The first comparison gives us the results of Myelinated models without the presence of Differential Privacy. We rearranged the results and organized them into a more structured form with a focus on the aspects of privacy, accuracy and speed; and covering all possible combinations of the environments (see Table 1). Those having both SGX and DP are termed to be fully private which those having only DP are DP-private and only SGX are SGX-private. The speed is calculated in terms of both training and inference where training speed is measured in minutes/75 epochs and inference is in images/sec.

These results are, however, not extensive and miss out on many combinations which are needed in order to allow for a clear overview for understanding and balancing the security-usability, accuracy-security and accuracy-usability trade-offs. Some of the results in Table 1, which we intended to investigate and involve SGX, are not covered in the paper, however, we were unable to reproduce them due to some issues with availability of SGX as highlighted in the challenges in Section 5.

Hence the goal of this work is to analyze and evaluate the Myelin framework to determine its bottlenecks & shortcomings in terms of ASP (accuracy, speed and privacy), for the combinations of cases that does not involve SGX, and summarize our findings as a whole in comparison to some of the results presented in Myelin that involve SGX. The combinations of cases for which we seek to get results, are available in the 'setting' column of Table 2.

3 Design

The experiment setting will make an impact, especially on the speed. Different processors might contribute to different speeds. Hence we had to present results in two different experiment settings, one from the results of the Myelin work and the other that corresponds to our results. We however reproduced the results of the baseline in our setting, so as to allow an easy comparison across the two experiment settings in the two tables.

The experiment setting 1, from the Myelin paper, is as follows:
Model: Resnet-32 (epochs:75)
Task: CIFAR-10 image recognition (50k train images + 10k test images)
Processor: Intel Xeon E5-2690 v4 CPU
DP setting: ϵ = 4 , failure probability δ = 10-5 , lot size L = 1024, added noise in model directly

The experiment Setting 2, from our work, is as follows:
Model: Resnet-32 (epochs:75)
Task: CIFAR-10 image recognition (50k train images + 10k test images)
DP setting: $\epsilon = 4$, failure probability $\delta = $ 10-5 , sensitivity $= 1$, added noise to input data
Processor: Intel® Core™ i7-9700K CPU @ 3.60GHz × 8 GPU: GeForce GTX 1070/PCIe/SSE2; Memory: 31.3 GiB

In our experiment setting, the noise was added to the input data through the Laplace mechanism for (ϵ, δ) - differential privacy as proposed by Holohan et al. [7]. Since it is not trivial to operate on the intermediate representations of TVM to perform such operations, we avoided the addition of noise in the model itself directly. We also faced some issues with training in the presence of TVM, which are highlighted in the challenges in Section 5, hence, we left out some of the values for training speed for TVM cases as shown in Table 2. This overhead of addition of noise to the input data affects only the overall training speed and not the speed of inference. The detailed interpretation of the results is presented in Section 4.

4 Evaluation

4.1 Accuracy

From Table 1, it can be seen that the accuracy will only be affected for the cases that involve differential privacy, since noise is added to the input data or model, while for other cases it remains the same, i.e., presence or absence of TVM or SGX does not have an impact on accuracy. In the experiment it can be observed that the accuracy drops from 92.4% to 90.8% with an overall decrease of 1.7%. Although this percentage is small, it still holds the claim that the accuracy decreases with Differential Privacy. With this, further research needs to be done to find out the impact of Differential Privacy on a range of models to better understand the introduction of Differential Privacy to different models.

4.2 Privacy

In order to introduce any privacy, data obfuscation needs to come into play through differential privacy and/or a trusted environment needs to be introduced to avoid any memory alterations or side-channel attacks. Myelin successfully takes into account both the privacy issues and incorporates both a secure isolated environment and training data protection through differential privacy. The only privacy issue that could be posed on the system would be if a weak differential privacy algorithm is adopted by the user. Other than that, the system was thought to be highly robust. But in conjunction with other trade-offs, one can make a choice if they specifically need fully private algorithm or any of DP-private or SGX-private would suffice. Applications that are not highly privacy sensitive can choose to go with SGX-privacy if speed or accuracy are rather more important.

4.3 Performance

The performance from our experiments varies over the range of settings. As it can be seen that the baseline models in both of the tables (Table 1 and 2) take 12.3 and 18.125 minutes per 75 epochs to train on the data. The issue arises however with the introduction of Differential Privacy which in experiment 2 almost doubles the time to

Experiment Setting 1 (Myelin)				
Setting	Accuracy	Speed (train) - min/75 epochs	Speed (infer) - img/s	Privacy
Baseline	92.4%	12.3	476	No
base + TVM + SGX + DP (myelin)	90.8%	12.9	same as base+TVM+SGX	Fully-private
base + DP + SGX	90.8%	-	same as base+SGX	Fully-private
base + TVM + SGX	92.4%	11.4	-	SGX-private
base + SGX	92.4%	-	-	SGX-private

Table 1: Results for experimental setting 1 (Myelin) reorganized in ASP perspective

Experiment Setting 2 (Our)				
Setting	Accuracy	Speed (train) - min/75 epochs	Speed (infer) - img/s	Privacy
Baseline	92.4%	18.125	327	No
base + DP + TVM	90.8%	base+TVM + 12.525	338	DP-private
base + DP	90.8%	30.65	327	DP-private
base + TVM	92.4%	-	338	No

Table 2: Results for experimental setting 2

train for the training considering the time added to make the input data differentially private.

Furthermore, the TVM framework makes the inferential speed fast (11 img/s - 3.2%) and so the framework itself would create a performance improvement for inference, but with an additional implementation overhead. However, TVM provides easy access to TEEs like SGX through readily available libraries, so given the advantages with TVM, it is always suggested to use it if SGX is required at all, but rather choose to skip it if differential privacy (sensitive training data) and ease of implementation are the prime focus.

However, by introducing any low-powered trusted environment, we can expect high performance overhead both in training and inference, due to the need for multiple data transfers with the trusted enclaves. Trusted Execution Environments also comes at a cost of limited memory. The inference speed for Myelin in table 1 is missing since the paper did not report it and so we cannot make any direct implications on the performance evaluation in cases that involve TEE.

4.4 Summary

From the results, with regards to the Privacy-Speed-Accuracy tradeoff, it is apparent that the cost to speed for Differential Privacy and SGX in both training and testing is somewhat trivial and therefore should generally pose a rather negligible hurdle whenever time is not an exceedingly limited resource. In terms of accuracy, however, the decrease when using Differential Privacy is somewhat more apparent, leaving it much more reasonable to omit if the guarantee of the privacy of training data is less important than avoiding this drop-off in accuracy. Because of these results, we recommend that if possible, users should port their machine learning architectures to TEEs whenever there is any concern about privacy, and should consider including Differential Privacy if the cost of slightly lower accuracy is acceptable and training data privacy is valued. The side channel attacks in TEEs through Memory Access Patterns are rather rare and can be a valid trade-off based on the requirements. TVM is highly recommended to use in conjunction with TEEs, however, not with Differential Privacy.

5 Discussion

5.1 Challenges

When compiling the results in Tables 1 & 2 we were unable to successfully port any Machine learning architectures to Intel SGX and did not have values from paper [10] for certain combinations of Myelin features. This is essentially due to the issues in setup and access of SGX. Some of the issues are: it requires enabling SGX in BIOS, supports only Intel based processors, has a very limited memory, and most virtual machines cannot directly access the SGX portion of the memory. We believe these challenges are a reason for the slow adoption of SGX in the machine learning community.

We were also not able to reproduce some of the non-SGX results ourselves due to the various issues faced with

TVM. TVM offers only pre-trained models out of the box that means the training performance under TVM framework required implementation of models from scratch which is especially challenging due to its nature of having an intermediate representation that performs the computations. Implementing Differential Privacy & Data Oblivious algorithms with Relay IR is hence challenging to implement in TVM and so transitively in Myelin as well. The TVM framework is actively under development and we hope it will be robust enough soon, which will help to implement Myelin on TVM easily. TVM also requires several steps of configuration which are very time consuming and requires expertise in systems engineering.

Furthermore, there is no publicly available implementation of Myelin in its full form that is robust enough. For instance properties like differential privacy are not incorporated in the demo code provided by the authors which forced us to use the technique of the addition of noise to the input data instead of the model directly, further adding to the issues of flexibility in TVM. This resulted in a huge increase in the time needed for training, which would not be the case if it is done on the model directly.

5.2 Limitations

Due to the challenges noted above, we were unable to fully analyze all combinations of Differential Privacy, TVM, and SGX on a single experiment setting, to achieve a full picture of the Privacy-Speed-Accuracy tradeoff for each aspect. The Differential Privacy implemented in Myelin [10] was internal to the architecture while the Differential Privacy in our experiment setting was added through inserting noise directly into the training data. Our work only articulates limited information for looking at the union of the results due to the differences introduced by multiple hardware environments as well as the difference in the implementation of Differential Privacy. We also missed out on some of the results due to the challenges faced with the lack of ease of access to SGX, TVM and a robust implementation of Myelin.

Because of these reasons, while we were able to generally describe the effects of these features on speed and accuracy, we remain unable to fully quantify them in a reliable way. One other limitation of Myelin is that it did not explore the usage of compact or optimized Machine Learning models that can indeed be private and also be used on the trusted hardware platforms. It is also clear that while there are combinations of privacy preserving components of machine learning that can combine to create fully private models at limited cost to accuracy and speed, the implementation of such components proves prohibitively onerous.

5.3 Future Work

The immediate future work is to fill out the results that were left in our experiments along with those in Table 1 related to SGX, due to the issues faced as noted above. Incorporating differential privacy to light-weight models like MobileNets and TinyML is a way to increase the speed while ensuring privacy, if there is a flexibility in the choice of the model selection for the task intended. This is a potential area of future work: to build compact or light-weight models such as MobileNets [8] and TinyML [25] in a fully private mode and also make sure that the impact on accuracy due to these lightweight models is negligible. There can also be more exploration in using any better optimization techniques other than TVM, with ideas taken from algorithms or other numerical optimizations.

In light of the challenges faced with the implementation, it becomes apparent that future work into privacy preserving machine learning can focus on improving the approachability of tasks including porting machine learning architectures into trusted execution environments and incorporating the option for adding Differential Privacy into architectures within programming suites like Pytorch [20] and TensorFlow [1]. The first of these tasks could be approached via either improving the support for multiple operating systems in TVM or through developing a more ad-hoc software package specifically designed to port machine learning architectures to platforms like Intel SGX [22], Arm TrustZone [14], or other similar TEEs.

6 Related Work

Several cryptographic methods are leveraged to tackle this issue, such as Multiparty Computation [5], Homomorphic Encryption [21] and Zero Knowledge Proofs, however these methods suffer from slow execution and delayed processing times, and adversely affect the performance. Although these cryptographic methods provide strong theoretical guarantees, this disadvantage of speed, especially in computationally intensive deep learning models, will not make it scalable thereby hampering its adoption. Hence, the non-cryptographic methods like Trsuted Execution Environments and Differential Privacy are more realistic in the context of end-to-end privacy preserving machine learning that requires heavy computations.

There are other works in this area on trusted hardware but without Differential Privacy, like Gazelle [11], Chiron [9] and Slalom [24]. In the other related works, Differential Privacy is proposed as a low cost method for training a model without leaking sensitive information about inputs [2]. Differential Privacy is further explored and quantified in this work [3]. TensorSCONE [12] looks into the integration of TEEs with TensorFlow to enable and allow for privacy preserving training of Machine Learning models on untrusted hardware [13].

The work of Nicolas .et.al. [19] articulates a comprehensive threat model for ML, and categorizes attacks and defenses within an adversarial framework. Their work however does not highlight key takeaways of Privacy-Speed-Accuracy tradeoff for machine learning models under different settings especially under TEEs. Moreover, another work [18] comprised research in exploration of machine learning models under different environments while taking into consideration tensions between model complexity, accuracy, and resilience. This work, however, unlike ours, did not consider performance as a factor.

7 Conclusion

In this work we attempted to study and analyse privacy-preserving machine learning with trust computing platforms through the lens of trade-offs between Accuracy, Speed and Privacy. We explained the need for such a study to equip users with detailed insights that can help them make decisions on the choice of environments based on their own needs, in contrast to a one-size-fits-all approach. We also recommend that developers of these frameworks allow fine grained choice of features for implementation, rather than giving out only the fully private stack. Our work also highlights the fact that this space of privacy in ML is very nascent and there needs to be a lot of software engineering effort to allow users to adopt them easily without additional implementation overhead. While the results and insights are limited due to multiple challenges faced, we hope this will act as a first step towards a robust and reliable ASP study in privacy-preserving ML.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, and et.al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2016.

[3] J. Allen, B. Ding, J. Kulkarni, H. Nori, O. Ohrimenko, and S. Yekhanin. An algorithmic framework for differentially private data analysis on trusted processors, 2019.

[4] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy. Tvm: An automated end-to-end optimizing compiler for deep learning, 2018.

[5] A. R. Choudhuri, M. Ciampi, V. Goyal, A. Jain, and R. Ostrovsky. On round optimal secure multiparty computation from minimal assumptions. *IACR Cryptol. ePrint Arch.*, 2019:216, 2019.

[6] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS '15*, 2015.

[7] N. Holohan, D. J. Leith, and O. Mason. Differential privacy in metric spaces: Numerical, categorical and functional data under the one roof. *CoRR*, abs/1402.6124, 2014.

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[9] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel. Chiron: Privacy-preserving machine learning as a service, 2018.

[10] N. Hynes, R. Cheng, and D. Song. Efficient deep learning on multi-source private data, 2018.

[11] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. Gazelle: A low latency framework for secure neural network inference, 2018.

[12] R. Kunkel, D. L. Quoc, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer. Tensorscone: A secure tensorflow framework using intel sgx, 2019.

[13] R. Kunkel, D. L. Quoc, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer. Tensorscone: A secure tensorflow framework using intel SGX. *CoRR*, abs/1902.04413, 2019.

[14] W. Li, Y. Xia, and H. Chen. Research on arm trustzone. *GetMobile: Mobile Comp. and Comm.*, 22(3):17–22, Jan. 2019.

[15] P. Mukherjee. Protecting cryptographic memory against tampering attack. 2015.

[16] Y. Nakano, S. Kiyomoto, and Y. Miyake. Evaluation of memory access pattern protection in a practical setting. 2014.

[17] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa. Oblivious multi-party machine learning on trusted processors.

In *25th USENIX Security Symposium (USENIX Security 16)*, pages 619–636, Austin, TX, Aug. 2016. USENIX Association.

[18] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman. Towards the science of security and privacy in machine learning. *ArXiv*, abs/1611.03814, 2016.

[19] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman. Sok: Security and privacy in machine learning. *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414, 2018.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and et. al. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[21] L. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13:1333–1345, 2018.

[22] M. Schunter. Intel software guard extensions: Introduction and open research challenges. In *Proceedings of the 2016 ACM Workshop on Software PROtection*, SPRO '16, page 1, New York, NY, USA, 2016. Association for Computing Machinery.

[23] R. Shokri. Trusting machine learning: Privacy, robustness, and transparency challenges. *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, 2019.

[24] F. Tramèr and D. Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *ArXiv*, abs/1806.03287, 2019.

[25] P. Warden and D. Situnayake. Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers. 2019.

[26] N. Xu, Q. Liu, T. Liu, Z. Liu, X. Guo, and W. Wen. Stealing your data from compressed machine learning models. *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.