www.czasopisma.pan.pl PAN www.journals.pan.pl

jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2022, VOL. 68, NO. 4, PP. 741–747
Manuscript received September 2, 2022; revised November, 2022. DOI: 10.24425/ijet.2022.143880

The High-Level Practical Overview of Open-Source Privacy-Preserving Machine Learning Solutions

Konrad Kuźniewski, Krystian Matusiewicz, Piotr Sapiecha

***Abstract*—This paper aims to provide a high-level overview of practical approaches to machine-learning respecting the privacy and confidentiality of customer information, which is called *Privacy-Preserving Machine Learning*. First, the security approaches in offline-learning privacy methods are assessed. Those focused on modern cryptographic methods, such as *Homomorphic Encryption* and *Secure Multi-Party Computation*, as well as on dedicated combined hardware and software platforms like *Trusted Execution Environment - Intel® Software Guard Extensions (Intel® SGX)*. Combining the security approaches with different machine learning architectures leads to our *Proof of Concept* in which the accuracy and speed of the security solutions will be examined. The next step was exploring and comparing the *Open-Source Python*-based solutions for *PPML*. Four solutions were selected from almost 40 separate, state-of-the-art systems: *SyMPC*, *TF-Encrypted*, *TenSEAL*, and *Gramine*. Three different *Neural Network* architectures were designed to show different libraries' capabilities. The *POC* solves the image classification problem based on the *MNIST* dataset. As the computational results show, the accuracy of all considered secure approaches is similar. The maximum difference between non-secure and secure flow does not exceed 1.2%. In terms of secure computations, the most effective *Privacy-Preserving Machine Learning* library is based on *Trusted Execution Environment*, followed by *Secure Multi-Party Computation* and *Homomorphic Encryption*. However, most of those are at least 1000 times slower than the non-secure evaluation. Unfortunately, it is not acceptable for a real-world scenario. Future work could combine different security approaches, explore other new and existing state-of-the-art libraries or implement support for hardware-accelerated secure computation.**

***Keywords*—Privacy-Preserving Machine Learning, Homomorphic Encryption, Secure Multi Party Computation, Trusted Execution Environment**

I. Introduction

RECENT advances in *Machine Learning (ML)* or *Deep Learning (DL)* techniques have demonstrated outstanding performance on various tasks, including organ recognition from medical images, classification of interstitial lung diseases, detection of lung nodules, medical image reconstruction, and segmentation of brain tumors. The advantage of *DL* models over humans has resulted in the development of computer-aided diagnosis systems - for example, the United States Food and Drug Administration recently announced the approval of an intelligent diagnosis system for medical images that do not require human intervention [1], [2].

Konrad Kuźniewski, Krystian Matusiewicz, Piotr Sapiecha are with Intel, the IPAS division (e-mail: {konrad.kuzniewski, krystian.matusiewicz}@intel.com, piotr.sapiecha@intel.com).

Nowadays, deep model training and evaluation are frequently outsourced to clouds, referred to in the literature as *Machine Learning as a Service (MLaaS)*. Cloud providers like *Google*, *Microsoft Azure*, or *Amazon Web Services* offer these services. Despite the impressive performance of *DL* algorithms, numerous recent studies have raised concerns about the security and robustness of machine learning models [3]–[5]. Moreover, the security of such algorithms' execution environments is being questioned [6]–[8]. The realization that *DL* models are neither safe nor resilient considerably complicates their practical implementation in security-critical applications such as predictive healthcare, which is basically life-critical. As a result, maintaining the integrity and security of deep learning models and health data is critical to the industry's wider adoption of *ML* and *DL*. However, significant research has been conducted recently to resolve this challenge using different cryptography techniques. *CryptoNets* [9], which showed *Privacy-Preserving Machine Learning (PPML)* prediction using *Homomorphic Encryption (HE)* cryptography in 2016 was one of the first publicly publicized benefits. As shown in the 2021 publications [10], [11], the use of multi-party computing techniques demonstrates breakthroughs in the privacy-preserving analysis of large amounts of medical data. While *MLaaS* is utilized in an insecure cloud environment, hardware-accelerated alternatives such as *Trusted Execution Environment (TEE)* or *Field Programmable Gate Arrays (FPGA)* accelerated computations are also used.

II. Security overview for Machine Learning

An *Artificial Intelligence (AI)* system's *ML* components include data, models, and methods for training, testing, and validation. In general, *ML* data-driven approach poses additional security risks throughout the training and inference phases of *ML* operations. These security concerns include the possibility of malicious manipulation of training data and adversarial exploitation of model sensitivities to degrade *ML* classification and regression performance. *Adversarial Machine Learning (AML)* is focused on the development of secure *ML* algorithms, the analysis of attacker capabilities, and the comprehension of attack repercussions. Malicious adversaries launch attacks and *ML* security refers to protections designed to avoid or mitigate the results of such attacks. *The NIST NISTIR 8269* [12] draft is one of the publications that summarize the vocabulary and security-related concepts associated with *AML*. It discusses the various kinds of attacks, their defenses, and their associated

implications in terms of *AML*. The privacy of *ML* solutions is violated if an adversary obtains personal information about one or more individual and legitimate model inputs, either included in the training data or not. An example would be when an adversary acquires or extracts an individual's medical records in violation of privacy policies. One of the defenses against testing (inference) attacks can be based on applying the following security models: *Homomorphic Encryption*, *Secure Multi-Party Computation*, use of *Trusted Execution Environment*.

HE is a type of encryption that enables the computation over encrypted data. Homomorphism is a mathematical notion that refers to preserving structure and correctness across a computation. The most important scheme supports both additions and multiplications of the ciphertext data which are called *Fully Homomorphic Encryption - FHE*. Their security relies on the hard mathematical problems based on number lattices like *Shortest Vector Problem*. One widely adopted schemes are *BGV*, *BFV* and *CKKS*.

Secure Multi-Party Computation (SMPC) computing may be extended to several parties with *SMPC*, in which processing is carried out on encrypted data shares that are shared among the parties so that no single party can recover the complete data on their own. The outcome of the calculation can be published without any party ever seeing the data, which would be retrieved only by agreement. A conceptual illustration of *SMPC* is a ballot, in which the outcome is required, but the individual voters' choices are not. The basic protocol for secure two-party computation is Yao's garbled circuit protocol and its underlying security is based on *1-out-of-2 Oblivious Transfer* protocol. Other than that, we can find more modern and complex approaches including three or more parties, other possibilities to share the data (arithmetic or Boolean sharing), and problem-specific optimizations.

TEE is a tamper-resistant processing environment that runs on a dedicated, hardened subsystem. It guarantees the authenticity of the executed code, the integrity of the runtime states (e.g., central processing unit registers, memory, and sensitive *I/O*), and the confidentiality of its code, data and runtime states stored on persistent memory. In addition, it should be able to provide a remote attestation that proves its trustworthiness for third parties. One of the examples of *TEE* technology is *Intel® Software Guard Extensions (Intel® SGX)*.

III. Problem Statement

This paper aims to provide a high-level overview of practical approaches to machine-learning respecting the privacy and confidentiality of customer information, in short - *Privacy-Preserving Machine Learning*. In order to achieve it, we need to check and verify different security approaches as well as their implementation to meet privacy preserving *ML* prediction's goal. As one of the *ML* tasks, we consider image recognition problem using *Modified National Institute of Standards and Technology (MNIST)* [13] dataset. Having this *ML* problem set, we will use different libraries with their security approaches to solve it and compare its accuracy and efficiency.

In other words, we need to ensure *Data Confidentiality* in *MLaaS Cloud Computational Environment*:

- Input: Datasets for digits image *MNIST* and recognition issue using different *Neural Network (NN)* architectures;
- Constrains: *Data Confidentiality* in the *Cloud Computational Environment*;
- Output: *PPML* predictions for a dataset;
- Security approaches: *HE*, *SMPC*, *TEE*.

As a part of problem definition, we define three research questions as follows:

- How secure environment affect the *accuracy* of computation?
- What overhead is associated with ensuring security in terms of neural network prediction *time*?
- What are efficient ways to *transform* neural networks from the non-secure computation to a secure one?

Tool selection criteria

In our solution, we mainly focused on two important criteria for tool selection:

- Several solutions will be compared. As a result, the solution should be *Open-Source*, with the source code accessible to the *public*.
- The majority of *ML* solutions and libraries are written in *Python*. To facilitate library efficient adoption, it is noted whether a particular solution is available as a *Python* package that includes support for either *Tensorflow* or *PyTorch* - the two most popular *ML* libraries available in *Python*.

Tool selection procedure

We compared tools that meet our security approaches requirement, those should be based on *HE*, *SMPC* or *TEE*.

Homomorphic Encryption Based Tools

Regarding *HE* libraries - see Table I, the most promising ones are *nGraph-HE2* [14] (an extension of *nGraph-HE* [15]) and *TenSEAL* [16].

TABLE I
THE COMPARISON OF PRIVACY-PRESERVING *HE* LIBRARIES

Library	*Tensorflow* or *PyTorch* support	Open Source
nGraph-HE2 [14]	✓	✓
nGraph-HE [15]	✓	✓
TenSEAL [16]	✓	✓
CryptoNets [9]	✗	✓
Cingulata [17]	✗	✓
TFHE [18]	✗	✓
MLwithHE [19]	✗	✓
CHET [20]	✗	✗
CryptoDL [21]	✗	✗
Chimera [22]	✗	✗
Glyph [23]	✗	✗

The *nGraph-HE2* has the interface to support *Tensorflow*, *TenSEAL* is supported via *PyTorch*. However, using *TenSEAL* is less complex to use. The other promising solutions, such as *CryptoNets* [9], *Cingulata* [17], *TFHE* [18], *MLwithHE* [19] were indeed open-source but written in *C* or *C++*. Other alternatives like *CHET* [20], *CryptoDL* [21], *Chimera* [22], *Glyph* [23] that have been developed are not open source and do not support *Tensorflow* or *PyTorch*.

TenSEAL [24] is the *HE* library that makes use of the *CKKS* technique. It employs relinearization, rescaling, and modulus flipping by default. The polynomial modulus degree is set at 8192 for the $\lambda = 128$ bit security level, with primes scaled to 26 bits. As it's core implementation it utilizes the *Microsoft SEAL* library.

Despite the continued study, however, the *HE* cryptosystems do not provide direct division and maximum operations [25], [26]. As a result, the use of contemporary *NN* topologies is constrained. For instance, the activation function of the *ReLU* [27] is approximated by $f(x) = x^2$ [9]. The *TenSEAL* is distributed under *Apache Licence 2.0*.

Secure Multi-Party Computation Based Tools

Dalskov [28] is the *SMPC* system that satisfies the requirements (see Table II).

TABLE II
THE COMPARISON OF PRIVACY-PRESERVING *SMPC* LIBRARIES

Library	*Tensorflow* or *PyTorch* support	Open Source
Dalskov [28]	✓	✓
SyMPC [29]	✓	✓
CrypTFlow [30]	✓	✓
CrypTFlow2 [31]	✓	✓
SIRNN [32]	✓	✓
Crypten [33]	✓	✓
TF-Encrypted [34]	✓	✓
TASTY [35]	✗	✓
ABY3 [36]	✗	✓
SecureNN [37]	✗	✓
Cerebro [38]	✗	✓
FALCON [39]	✗	✓
XONN [40]	✗	✗
Chameleon [41]	✗	✗
Sadeghi [42]	✗	✗
Barni [43]	✗	✗
SecureML [44]	✗	✗
Tetrad [45]	✗	✗
BLAZE [46]	✗	✗
SWIFT [47]	✗	✗

However, the solution was designed in a way that does not support installation via *Python* package. *CrypTFlow* [30], *CrypTFlow2* [31], and *SIRNN* [32] are all components of the broader *EzPC* [48] library. While some components of the system support *Tensorflow*, integration with the standard machine learning *Python* flow is quite challenging due to its complexity.

SyMPC [29], which is included in *PySyft* [49] can run low-level protocols such as *ABY3* [36] or *FALCON* [39] and supports *PyTorch*. *TF-Encrypted* [34] could also be used with the *Tensorflow* libraries, and it's working with the *SecureNN* [37] protocol.

Other libraries did not conform to the specified requirements. However, it is worth mentioning current findings in *SMPC* subjects such as *Tetrad* [45], *BLAZE* [46], and *SWIFT* [47].

SyMPC [29] is a solution that uses the *AriaNN* [50] protocol for semi-honest two-party computing. Due to the library's ability to utilize the *ReLU* activation function, multi-layer perceptron *NN* architectures are feasible. However, convolution *NNs* are limited in their use since they only support the *MaxPool* building component [27]. Additionally, *Dropout* does not work as a construction block [27]. The *SyMPC* is distributed under *MIT Licence*.

TF-Encrypted [51] is a *SMPC* that is based on the *SecureNN* [37] - three-party malicious-aware computation protocol. Consequently, even if one of the parties is a malicious actor, the computation may still succeed. The *TF-Encrypted* is distributed under *Apache Licence 2.0*.

Trusted Execution Environment Based Tools

Three solutions (see Table III) that match the required criteria are *Gramine*, [52], *TF-Trusted* [53] and *SLALOM* [54].

TABLE III
THE COMPARISON OF PRIVACY-PRESERVING *TEE* LIBRARIES

Library	*Tensorflow* or *PyTorch* support	Open Source
Gramine [52]	✓	✓
TF-Trusted [53]	✓	✓
SLALOM [54]	✓	✓
PPFL [55]	✗	✓
DarkneTZ [56]	✗	✓
BigDL PPML [57]	✗	✓
Eleos [58]	✗	✓
TensorSCONE [59]	✓	✗
PERUN [60]	✓	✗
Flatee [61]	✗	✗

However, *SLALOM* [54], like *Dalskov* [28], the solution was designed in a way that does not support installation via *Python* package. On the other hand, *TF-Trusted* [53] is based on an older *Intel® Software Guard Extensions SDK for Linux* OS (Intel® SGX SDK for Linux* OS)* version which is nearly impossible to run on more recent versions. The other alternatives are either proprietary or do not support *Tensorflow* or *PyTorch*.

Gramine [52] solution allows running unmodified *Linux* binaries on *Intel® SGX*. The *Gramine* is distributed under *GNU GPLv3*.

Comparison Conclusion

HE, *SMPC* and *TEE* were addressed throughout the solution search. Only a few of the nearly 40 privacy-preserving technologies examined are open-source and could be effectively

integrated into the *Python ML* code base. Those are: *SyMPC* [29], *TF-Encrypted* [51], *TenSEAL* [24] and *Gramine* [52].

IV. System Architecture

The following modules were created as part of the implementation:

- *Training Module* - optimizes a given *NN* architecture in terms of weights for a specific dataset classification or regression issue - see Figure 1.
- *Plaintext Module* - this module computes the solution to a given issue using a trained *NN* model and a provided dataset - see Figure 2.
- *Encrypted Module* - with a given dataset and trained *NN*, utilizes a *PPML* library and conducts the encrypted computation - see Figure 3.
- *Benchmark Module* - this module provides the average time required to compute the prediction for a given computation in either the *Plaintext* or *Encrypted Modules* - see Figure 4.

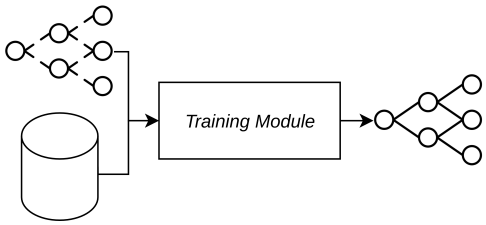

Fig. 1. *Training Module* in implemented solution. For given *NN* architecture and dataset it outputs trained *NN* for a given issue.

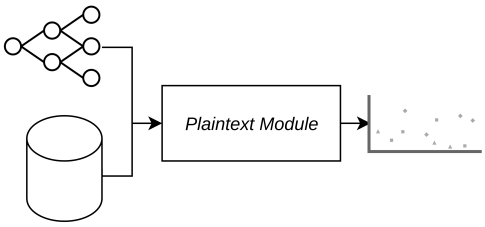

Fig. 2. *Plaintext Module* in implemented solution. For a given trained *NN* model and dataset it outputs the network evaluation.

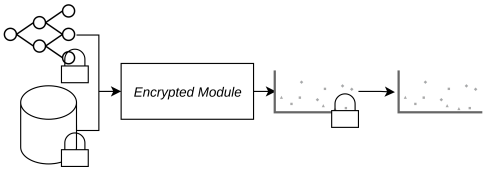

Fig. 3. *Encrypted Module* in implemented solution. For the given *PPML* solution it encrypts the trained *NN* model and dataset. It outputs the encrypted network evaluation which is decrypted for plaintext output.

For a given dataset and *NN* architecture model, the application flow goes as follows:

- Using the *Training Module*, the *NN* model is trained for a given training dataset and issue type. The training parameters may be tailored to a particular issue. The module, in particular, enables the user to choose the training criterion and optimizer parameters such as learning rate, number of epochs, and batch size.
- Then the *Plaintext Module* verifies the model training metrics for a test dataset. Those are specific to the issue type but could be parametrized within the same problem type. For a classification problem, it computes the *Accuracy*.
- The *Encrypted Module* uses one of the *PPML* libraries to perform the encrypted computation with the same parameters as the *Plaintext Module*. After the evaluation, the same metric as for the *Plaintext Module* shows to compare specific library quality.
- The *Benchmark Module* is used in conjunction with the *Plaintext* or *Encrypted Modules*. It shows the elapsed time for the one instance of the test dataset to perform its computation. By default, an average of 20 runs is returned in this module.

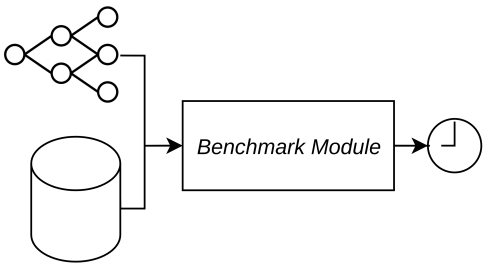

Fig. 4. *Benchmark Module* in implemented solution. The trained *NN* model and dataset outputs the average evaluation time for a sample record. It can be used together with either *Plaintext* or *Encrypted* module.

All application modules are specific to a given library implementation. As stated in previous chapters, the application will use the following libraries: *TenSEAL* [24], *SyMPC* [29], *TF-Encrypted* [51], *Gramine* [52]. From the implementation point of view, the *TenSEAL* and *SyMPC* are *PyTorch*-based library but *TF-Encrypted* and *Gramine* are *Tensorflow* based. However, from the security point of view, *TenSEAL* is *HE* based, *SyMPC* and *TF-Encrypted* are *SMPC* based and *Gramine* is *Intel® SGX* based.

TF-Encrypted and *Gramine* should support even complex *NN* architectures including modern convolution building blocks such as *Conv2D* layers, *Average* and *Max Pooling* [27].

V. Neural Network Architecture Models

After reviewing the features of each library, let us discuss the *NN* architectural models that were constructed. All training was carried out with the *Adam* optimizer, and the learning phase consists of 30 epochs with a learning rate of 0.001 using momentum 0.9 and 10^{-7} epsilon. The *CrossEntropy* criterion was used to address the multi-class classification challenge.

Model A this model was built primarily to showcase the *TenSEAL* library's capabilities. It consists of two fully connected layers with 128 neurons each. $f(x) = x^2$ was used to activate the function. All libraries support this network architecture.

Model B is the multi-layer perceptron model too. It comprises two fully linked layers of 128 neurons that activate

TABLE IV
RESULTS FOR THE PERFORMANCE TEST SCENARIO. PT - REFERS TO THE TIME IN SECONDS THAT WAS PROVIDED FROM THE BENCHMARK MODULE FROM THE PLAINTEXT MODULE. ET - REFERS TO THE TIME IN SECONDS THAT WAS PROVIDED FROM THE BENCHMARK MODULE FROM THE ENCRYPTED MODULE

	TenSEAL		*SyMPC*		*TF-Encrypted*		*Gramine*	
	FHE		*SMPC*		*SMPC*		*TEE - SGX*	
Model Type	*PT*	*ET*	*PT*	*ET*	*PT*	*ET*	*PT*	*ET*
Model *A*	0.0019	1.5187	0.0011	1.3249	0.0005	0.1361	0.0002	0.0012
Model *B*	✗	✗	0.0025	10.006	0.0004	0.2611	0.0003	0.0012
Model *C*	✗	✗	0.0051	156.51	0.0011	2.8737	0.0018	0.0697

using the *ReLU* function. The final activation function varies according to the kind of problem. The *Softmax* function was utilized for multiclass classification, and the *Sigmoid* function was employed for binary classification. The *Identity* function was used to define the regression type. *TenSEAL* does not support this type of architecture since it requires using *ReLU*, *Softmax*, and *Sigmoid* functions.

Model C this model was used to validate the convolutional *NN's* performance on the image classification task. Its architecture is similar to *LeNet* [62]: *Conv2D* 5x5, *Max Pooling* 2x2, *Conv2D* 5x5, *Max Pooling* 2x2, *Fully connected layer* 120, *Fully connected layer* 84, *Fully connected layer* 10, *Softmax* activation.

Max Pooling was employed to enable the inclusion of the *SyMPC* library, as it does not support other pooling layers. Additionally, the design may be executed in *TF-Encrypted*, or *Gramine*.

VI. COMPUTATION ENVIRONMENT

The computation environment consists of:

- *Intel® Xeon® Platinum 8358* 2 CPU 2.60GHz processors with 128 threads; 512GB RAM
- *Linux OS* - *Ubuntu* `20.04.03` (kernel `5.16.5`)
- *TenSEAL* `0.3.6` with *Microsoft SEAL* support for *Intel® HEXL*
- *SyMPC* commit hash `634396`
- *TF-Encrypted* version `0.5.9`
- *Gramine* version `1.1` with *Intel® SGX SDK for Linux* OS* `2.11` version

For all libraries, *Python* `3.7` was used. Because of specific implementation for serialization and deserialization in chosen libraries, each has its instance of *NN* architecture model.

VII. EXECUTION & TEST PLAN

The test plan consists of two activities to answer questions in our problem statement. For the correctness question, for each library and its trained model, the accuracy will be compared for secure and non-secure flow. Accuracy is a measure of correct decisions normalized by the size of the whole space of decisions. For the performance question, the arithmetic mean of 20 runs for one prediction will be compared for secure and non-secure flow (measured execution time will be in seconds). the computation is based on *MNIST* dataset. It is a collection of handwritten digits as 28x28 pixels black-white image divided into ten classes. It contains 60000 training examples and 10000 test instances. In terms of pre-processing the dataset, all pixels were normalized.

VIII. SYSTEM EVALUATION

To conclude, the difference between non-secure and secure accuracy for *TenSEAL* was 0.63%, but the biggest noted was for the *SyMPC*. For *Architecture A* it gave: 1.14% difference and for *Architecture B*: 1.01% and *C*: 1.1% respectively. *TF-Encrypted* noted the maximum difference of 0.03%. *Gramine* gave the same results. The performance results are summarized in Table IV. As the results are shown for the first test, the accuracy of the computation is preserved. The maximum difference between non-secure and secure solutions is no more than 1.2%. However, are many differences in execution time. *TenSEAL* appeared to secure evaluation *NN* the slowest.

Due to its security model, we cannot design architectures other than *Architecture A*. In *FHE* systems we can define only functions based on the polynomials. That is why we mathematically cannot express any other activation function like *ReLU* or *Softmax*. It is the most significant limitation of the *FHE* solutions.

Other libraries are based on *SMPC*. One of them, *SyMPC*, still computes *Architecture A* faster than the *FHE* one. However, it fails to compute other architectures efficiently. This may be connected to the fact that the library's implementation uses only two threads for computation (in contrast to other solutions where all 128 threads are used, except for *Intel® SGX* where 20 threads are used). Other *SMPC* solutions such as *TF-Encrypted* were much more effective in their computations. It takes under a second to compute simple architecture networks. However, it is quite a challenge for convolutional network evaluation - for *TF-Encrypted* it lasts less than 3 seconds. *TF-Encrypted* is *Tensorflow* based, and its runtime configuration heavily uses the *Tensorflow* server's configuration. The best results showed *TEE* solutions and *Gramine*. For simple architectures, it is only 10 times slower compared to non-secure computation, and for the convolutional network - it is 30 times slower. However, when we do not have access to *TEE* the computation is at least 1000 times slower. On the other hand, as is shown by the results, all libraries that evaluate *CNN* architectures are significantly slower than *MLP* (network complexity).

IX. SUMMARY

In overall conclusion, the computational results show that the most effective *PPML* library is based on *TEE*. However, when one does not access such platforms, *SMPC* solutions give a similar accuracy ratio but with reduced performance. The worst performance results were achieved for the *HE* solution. To emphasize this fact, as demonstrated by the results, the

security costs in terms of evaluation time are very high. Most of the encrypted computations are 1000 times slower than the plaintext evaluation. It is a costly operation, but if we want to achieve an adequate security level, there is no other option. Unfortunately, it is not yet acceptable for a real-world scenario. Multiple future research directions are possible from this point. First, one could try combining multiple security models, for instance, combining operations using *SMPC* or *FHE* and offloading more complex ones using *TEE*. The second one is to have a broader library comparison and include support for libraries written in *C/C++*. It would be more complex to adapt to real-world scenarios, but it will be closer to the original implementation of underlying protocols, thus achieving better performance. The other future research could be improving *SMPC* protocols implementation using hardware accelerators. Similarly to what has been achieved with *FHE TenSEAL* library and *Intel HEXL* hardware accelerator.

X. Notice & Disclamers

We want to sincerely and deeply thank our colleagues Marcin Kolasiński and Grzegorz Gerka for their help and support during our computation execution.

References

[1] F. Newsroom. (2018) Fda permits marketing of artificial intelligence-based device to detect certain diabetes-related eye problems. [Online]. Available: https://www.fda.gov/news-events/press-announcements/fda-permits-marketing-artificial-intelligence-based-device-detect-certain-diabetes-related-eye

[2] FDA. Artificial intelligence and machine learning (ai/ml)-enabled medical devices. [Online]. Available: https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-aiml-enabled-medical-devices

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[4] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 6106–6116. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/22722a343513ed45f14905eb07621686-Abstract.html

[5] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, 2019. [Online]. Available: https://doi.org/10.1109/TNNLS.2018.2886017

[6] D. M. Bamasoud, A. S. Al-Dossary, N. M. Al-Harthy, R. A. Al-Shomrany, G. S. Alghamdi, and R. O. Algahmdi, "Privacy and security issues in cloud computing: A survey paper," in *International Conference on Information Technology, ICIT 2021, Amman, Jordan, July 14-15, 2021*. IEEE, 2021, pp. 387–392. [Online]. Available: https://doi.org/10.1109/ICIT52682.2021.9491632

[7] Y. Zhang and R. Sion, "Speculative execution attacks and cloud security," in *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, CCSW@CCS 2019, London, UK, November 11, 2019*, R. Sion and C. Papamanthou, Eds. ACM, 2019, p. 201. [Online]. Available: https://doi.org/10.1145/3338466.3360287

[8] Y. Alghofaili, A. Albattah, N. Alrajeh, M. A. Rassam, and B. A. S. Al-rimy, "Secure cloud infrastructure: A survey on issues, current solutions, and open challenges," *Applied Sciences*, vol. 11, no. 19, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/19/9005

[9] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," Tech. Rep. MSR-TR-2016-3, 2016. [Online]. Available: https://www.microsoft.com/en-us/research/publication/cryptonets-applying-neural-networks-to-encrypted-data-with-high-throughput-and-accuracy/

[10] J. Alvarez-Valle, P. Bhatu, N. Chandran, D. Gupta, A. Nori, A. Rastogi, M. Rathee, R. Sharma, and S. Ugare, "Secure medical image analysis with cryptflow," 2020.

[11] A. Soin, P. Bhatu, R. Takhar, N. Chandran, D. Gupta, J. Alvarez-Valle, R. Sharma, V. Mahajan, and M. P. Lungren, "Multi-institution encrypted medical imaging ai validation without data sharing," 2021.

[12] M. H. M. Elham Tabassi (NIST), Kevin Burns (MITRE). A taxonomy and terminology of adversarial machine learning. [Online]. Available: https://csrc.nist.gov/publications/detail/nistir/8269/draft

[13] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[14] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski, "ngraph-he2: A high-throughput framework for neural network inference on encrypted data," 2019.

[15] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, "ngraph-he: A graph compiler for deep learning on homomorphically encrypted data," 2019.

[16] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," 2021.

[17] S. Carpov, P. Dubrulle, and R. Sirdey, "Armadillo: A compilation chain for privacy preserving applications," in *Proceedings of the 3rd International Workshop on Security in Cloud Computing*, ser. SCC '15. Association for Computing Machinery, 2015, p. 13–19. [Online]. Available: https://doi.org/10.1145/2732516.2732520

[18] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," Cryptology ePrint Archive, Report 2016/870, 2016, https://ia.cr/2016/870.

[19] S. S. Magara, C. Yildirim, F. Yaman, B. Dilekoglu, F. R. Tutas, E. Öztürk, K. Kaya, Ö. Tastan, and E. Savas, "Ml with he: Privacy preserving machine learning inferences for genome studies," 2021.

[20] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, "Chet: an optimizing compiler for fully-homomorphic neural-network inferencing," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019, pp. 142–156.

[21] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 3, pp. 123–142, 2018. [Online]. Available: https://doi.org/10.1515/popets-2018-0024

[22] C. Boura, N. Gama, M. Georgieva, and D. Jetchev, "Chimera: Combining ring-lwe-based fully homomorphic encryption schemes," Cryptology ePrint Archive, Report 2018/758, 2018, https://ia.cr/2018/758.

[23] Q. Lou, B. Feng, G. C. Fox, and L. Jiang, "Glyph: Fast and accurately training deep neural networks on encrypted data," 2020.

[24] OpenMined. (2021) Tenseal library. [Online]. Available: https://github.com/OpenMined/TenSEAL

[25] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, "Numerical method for comparison on homomorphically encrypted numbers," Cryptology ePrint Archive, Report 2019/417, 2019, https://ia.cr/2019/417.

[26] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," Cryptology ePrint Archive, Report 2019/1234, 2019, https://ia.cr/2019/1234.

[27] U. Michelucci, *Advanced applied deep learning : convolutional neural networks and object detection*. Apress, 2019.

[28] A. Dalskov, D. Escudero, and M. Keller, "Secure evaluation of quantized neural networks," Cryptology ePrint Archive, Report 2019/131, 2019, https://ia.cr/2019/131.

[29] OpenMined. (2021) Sympc library. [Online]. Available: https://github.com/OpenMined/SyMPC

[30] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," 2020.

[31] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow2: Practical 2-party secure inference," *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. [Online]. Available: http://dx.doi.org/10.1145/3372297.3417274

[32] D. Rathee, M. Rathee, R. K. K. Goli, D. Gupta, R. Sharma, N. Chandran, and A. Rastogi, "Sirnn: A math library for secure rnn inference," Cryptology ePrint Archive, Report 2021/459, 2021, https://ia.cr/2021/459.

[33] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," in *arXiv 2109.00984*, 2021.

[34] M. Dahl, J. Mancuso, Y. Dupis, B. Decoste, M. Giraud, I. Livingstone, J. Patriquin, and G. Uhma, "Private machine learning in tensorflow using secure computation," 2018.

[35] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Tasty: Tool for automating secure two-party computations," Cryptology ePrint Archive, Report 2010/365, 2010, https://ia.cr/2010/365.

[36] P. Mohassel and P. Rindal, "Aby¡sup¿3¡/sup¿: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. Association for Computing Machinery, 2018, p. 35–52. [Online]. Available: https://doi.org/10.1145/3243734.3243760

[37] S. Wagh, D. Gupta, and N. Chandran, "Securenn: Efficient and private neural network training," Cryptology ePrint Archive, Report 2018/442, 2018, https://ia.cr/2018/442.

[38] W. Zheng, R. Deng, W. Chen, R. A. Popa, A. Panda, and I. Stoica, "Cerebro: A platform for Multi-Party cryptographic collaborative learning," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2021, pp. 2723–2740. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/zheng

[39] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," 2020.

[40] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "Xonn: Xnor-based oblivious deep neural network inference," 2019.

[41] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," 2018.

[42] A.-R. Sadeghi and T. Schneider, "Generalized universal circuits for secure evaluation of private functions with application to data classification," Cryptology ePrint Archive, Report 2008/453, 2008, https://ia.cr/2008/453.

[43] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-preserving ecg classification with branching programs and neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 452–468, 2011.

[44] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[45] N. Koti, A. Patra, R. Rachuri, and A. Suresh, "Tetrad: Actively secure 4pc for secure training and inference," Cryptology ePrint Archive, Report 2021/755, 2021, https://ia.cr/2021/755.

[46] A. Patra and A. Suresh, "Blaze: Blazing fast privacy-preserving machine learning," *Proceedings 2020 Network and Distributed System Security Symposium*, 2020. [Online]. Available: http://dx.doi.org/10.14722/ndss.2020.24202

[47] N. Koti, M. Pancholi, A. Patra, and A. Suresh, "Swift: Super-fast and robust privacy-preserving machine learning," 2021.

[48] EzPC. (2021) Ezpc. [Online]. Available: https://github.com/mpc-msri/EzPC

[49] PySyft. (2021) Pysyft. [Online]. Available: https://github.com/OpenMined/PySyft

[50] T. Ryffel, P. Tholoniat, D. Pointcheval, and F. Bach, "Ariann: Low-interaction privacy-preserving deep learning via function secret sharing," 2021.

[51] D. Labs. (2021) tf-encrypted library. [Online]. Available: https://github.com/tf-encrypted/tf-encrypted

[52] gramine. (2021) gramine, library. [Online]. Available: https://github.com/gramineproject/gramine

[53] D. Labs. (2021) tf-trusted, library. [Online]. Available: https://github.com/capeprivacy/tf-trusted

[54] F. Tramèr and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," 2019.

[55] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "Ppfl: Privacy-preserving federated learning with trusted execution environments," 2021.

[56] F. Mo, A. S. Shamsabadi, K. Katevas, S. Demetriou, I. Leontiadis, A. Cavallaro, and H. Haddadi, "Darknetz," *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, 2020. [Online]. Available: http://dx.doi.org/10.1145/3386901.3388946

[57] J. J. Dai, Y. Wang, X. Qiu, D. Ding, Y. Zhang, Y. Wang, X. Jia, L. C. Zhang, Y. Wan, Z. Li, J. Wang, S. Huang, Z. Wu, Y. Wang, Y. Yang, B. She, D. Shi, Q. Lu, K. Huang, and G. Song, "Bigdl: A distributed deep learning framework for big data," in *Proceedings of the ACM Symposium on Cloud Computing*, ser. SoCC'19. Association for Computing Machinery, 2019, pp. 50–60. [Online]. Available: https://arxiv.org/pdf/1804.05839.pdf

[58] M. Orenbach, P. Lifshits, M. Minkin, and M. Silberstein, "Eleos: Exitless os services for sgx enclaves," in *Proceedings of the Twelfth European Conference on Computer Systems*, ser. EuroSys '17. Association for Computing Machinery, 2017, p. 238–253. [Online]. Available: https://doi.org/10.1145/3064176.3064219

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

[60] W. Ozga, D. L. Quoc, and C. Fetzer, "Perun: Secure multi-stakeholder machine learning framework with gpu support," 2021.

[61] A. Mondal, Y. More, R. H. Rooparaghunath, and D. Gupta, "Flatee: Federated learning across trusted execution environments," 2021.

[62] LeNET. (2021) Lenet. [Online]. Available: https://en.wikipedia.org/wiki/LeNet