Privacy-Preserving Intrusion Detection using Convolutional Neural Networks

Martin Kodyš
ST Engineering
Singapore

Zhongmin Dai
ST Engineering
Singapore

Vrizlynn L. L. Thing
ST Engineering
Singapore

***Abstract*—Privacy-preserving analytics is designed to protect valuable assets. Commonly, data from the client is inputted into the model on the analyst's side. Privacy protection on both sides is fuelled by legal obligations and intellectual property concerns. We explore the use case of a model owner's analytic service for customer's private data. The data must not be revealed to the analyst and the model must not leak to the customer. Current methods involve costs: accuracy deterioration and computational complexity. The complexity translates as a longer processing time, more computing resources, or data communication between the client and the server. In this work, we show the scenario's feasibility on an example of an attack detection system based on deep Convolutional Neural Networks that we augment with privacy-preserving technology based on Function Secret Sharing. We share insights into this implementation.**

***Index Terms*—Internet of Things, Convolutional Neural Networks, Privacy-Preserving Analysis, Function Secret Sharing**

I. Introduction

Privacy-preserving technologies gain popularity as the value of both, the know-how and the data, increases. An interest around the protection of Artificial Intelligence models concerns multiple points: data used for training, model created from the data (both, the intellectual property and the traces of the training data), and data used for inference.

In such a setup, we can discern three *roles*: Training Data Owner, Model Owner, Inference Data Owner.

In case of a provision of the model as a service, the Model Owner must ensure the protection of the assets of Data Owners against leaks, as well as defend their own model against exfiltration.

In particular, intrusion detection can benefit from privacy-preservation mechanisms. This paper's aim is to illustrate the feasibility of this connection. We augment the Deep Learning inference [1] with privacy-preservation techniques according to the PriMIA framework [2].

We point out important factors for the optimisation of hyper-parameters, especially the fixed fractional precision that steers the accuracy of the system compared to unprotected version.

The paper's section II provides background in Intrusion Detection on IoT data, and privacy preservation; III compares current research to our work; IV explains the design of the solution; V summarises experiments and results, followed by VI discussing possible evolutions and applications, thereafter concluded in VII.

II. Background

Two topics intersect in this paper: intrusion detection in the Internet of Things (IoT) based on the actual data readings, and privacy-preserving technologies. We present the perspective of a privacy-conscious intrusion detection service provider. Our goal is to provide a (1) reliable service (2) unaware of the actual client's data, while (3) keeping our model secret.

A. Intrusion Detection in IoT

The task of intrusion detection is a prediction of a system status based on a set of observations in time.

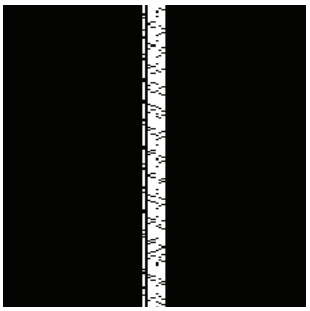

Fig. 1. Example of an encoded $224 \times 224 \times 3$-channel picture representing a 224 items long history (vertically) of 17 different sub-sensors (horizontally) as a sensor provides one or more sub-sensor readings.

In our case, the intrusion detection follows [1]: The analysis runs Convolutional Neural Networks (CNN) architecture. Its input is a 2-dimensional picture (shown in Fig. 1) made of pre-processed IoT sensor readings. It is a slit-view of chronological sensor readings. The output is one of the 8 classes – a benign state `normal`, or one of 7 different attacks: `backdoor`, `ddos`, `injection`, `password`, `ransomware`, `scanning`, or `xss`. These classes are defined in the TON_IoT "TrainTest" dataset [3]. The TrainTest was pre-processed using `TT500n` aggregation strategy described in [1]: `TT` (standing for "TrainTest") was partitioned in sequences of `500` readings, with `no` aggregation by time. To conserve temporal patterns, each 500-item long sequence as a whole went into train or test set. Within their respective sets, these sequences were concatenated and a sliding window produced the data for our architecture, i.e., 224 readings.

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00205

The architecture selected for privacy-preserving analytics was *ResNet50* [4]. In the attack classification task, the performance of *ResNet50* and *EfficientNet-B0* was relatively close [1]. Between them, we chose pragmatically the one sharing building blocks with the already implemented *ResNet18* architecture within PriMIA framework [2]. The required modification is shown in Fig. 3.

The adjustment of the PriMIA framework to our use case is illustrated in Fig. 2. In simple terms, the TON_IoT dataset was converted into RGB images. The hard-coded 1-channel greyscale image input was extended to allow for 3-channel inputs. We completed the implementation of ResNet50 that reused the blocks available in ResNet18. The number of output classes was extended from 3 to 8. More details are shared in section IV.

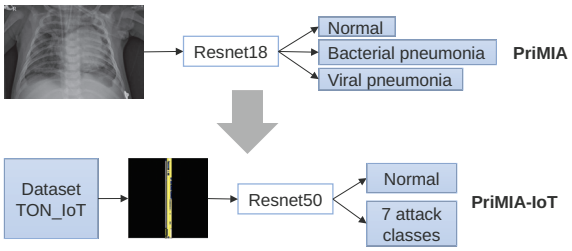

Fig. 2. Adjustments of PriMIA to process data from IoT detection use case.

```
def resnet50(pretrained=False, progress=True, in_channels=3, pooling="avg", **kwargs):
    r"""ResNet-50 model from
    `"Deep Residual Learning for Image Recognition" <https://arxiv.org/pdf/1512.03385.pdf>`_

    Args:
    pretrained (bool): If True, returns a model pre-trained on ImageNet
    progress (bool): If True, displays a progress bar of the download to stderr
    """
    return _resnet(
        "resnet50",
        Bottleneck,
        [3, 4, 6, 3],
        pretrained,
        progress,
        in_channels=in_channels,
        pooling=pooling,
        **kwargs
    )
```

Conv 1x1
Batch normalization
Conv 3x3
Batch normalization
Conv1x1
Batch normalization
+= input (downsample if required)
ReLU

Fig. 3. Extension of PriMIA to implement ResNet50. Bottleneck had already been defined as specified in the inner frame. This implementation corresponds to [4]. The array [3, 4, 6, 3] defines the number of successive Bottleneck blocks before a change of dimensions.

Given 17 sub-sensor readings, there are 17 columns placed in the middle and padded with value `0` on the left and right to fill 224 columns (translated to pixels).

In this work, we focused on a simple and radical strategy `miss3` that replaces actual values by `0` and missing ones by value `1`. As indicated by number "3" in its name, the method is used to fill all three channels equally to obtain an RGB picture. An example of a resulting picture after normalisation is presented in Fig. 1.

B. Privacy-Preserving Technologies

Many ways are being explored [5] in search of privacy-preserving technology with negligible overhead and optimal security adapted to requirements. The research falls into the field of Secure Multi-Party Computation (MPC). And from our standpoint, we will focus on protecting the assets against other parties of the computation.

Although the Homomorphic Encryption is a promising technology that is being actively researched to achieve high performance [6], Fully Homomorphic Encryption solutions still incur an unaffordable computational cost, especially in the case of large neural network architectures. Even VGG9 with only 6 convolutional and 3 fully connected layers, a single inference takes 30 minutes [7]. This may be improved by deploying specialised hardware.

Hybrid approaches delegating expensive tasks to the Data Owner [8] generate high communication costs, undermining the relevance of the server side altogether.

A common practice is to defend against an *honest but curious* adversary (also called *semi-honest* in [9]). It is the use case of PriMIA, where several medical establishments collaborate on a common model, and then share it safely with a non-participating medical establishment to apply the know-how but without giving it away. An *honest but curious* adversary does not attempt to break the protocol, although they might extract any information from the available data. In practice, the common model shared among all participating medical establishments could be subjected to an inversion attack. To prevent it, PriMIA employs an optional Differential Privacy technique.

The FSS paper [10] mentions the query quota as the only mitigation for model inversion attacks to limit the amount of information the attacker might obtain to reconstruct the model. The more complex neural network, the slower the evaluation, which naturally, provides a quota-like protection.

We can distinguish the MLaaS with training and without training: We situate our research in the case where we provide a secure access to a privately trained model without sharing it. The training is excluded from the problem this paper addresses.

III. Related Works

In our specific use-case of Machine Learning as a Service (MLaaS), we discern several aspects that have been addressed in the literature. Generally, privacy preserving technologies protect data in different stages of the computation but also the computation process itself. We will focus on technologies that do not rely on the trust amongst different parties of the computation as defined in section II.

Machine Learning, and in particular, Deep Learning were objects of privacy-preservation efforts. The most straightforward way to protect the Training Data Owner is the use of techniques of input data perturbation while conserving useful features or properties. In particular, the concept of Differential Privacy (DP) [11] is becoming popular as the perturbation is quantified and provides a framework for expressing the privacy cost in terms of accuracy. DP can improve privacy for Training Data Owner while producing a model, where it effectively hides the details from the Model Owner. However, it does

not provide any advantage to Inference Data Owner, as the predictions must not be biased.

In order to protect the IDO's assets, computation using homomorphic encryption is gaining interest. However, the current solutions are mostly relevant for simpler Machine Learning algorithms. For example, CryptoNets [12] operate on MNIST with dimensions of 28×28 pixels and uses client-side evaluation of activation functions.

Falcon [13] offers security against a malicious adversary in a 3-party setup. However, it requires an honest majority, which is not trivial to achieve in a two parties setup. Any third party would present a risk of collusion and, thus, breaking down the trust in the system.

Additionally, methods of *model tuning* based on user feedback [14] address privacy concerns. They have complementary goals to our research: Updating the model with private data without disclosing it. These methods might be used to further extend our current research where we focus only on the inference provision.

In use cases similar to ours, the service may be Cloud-like [8], where the service provider offers their infrastructure and guarantees the privacy of its usage. The client would privately input data for the training, chooses an architecture, train it, and obtains inference – all without letting the external observer or service provider learn anything about the data and the model. The described mode of operation is suitable for infrastructure service providers (e.g., Cloud) and may find a practical use in systems relying on a public blockchain. However, we focus on providing a private access to the inference service. The Model Owner will not have any access to the data provided by the Inference Data Owner. Moreover, the Inference Data Owner will not be able to reconstruct the model. Thus, we protect not only the intellectual property of the Model Owner but also the Training Data Owners against inversion attacks.

IV. Design

The PriMIA framework extends PySyft framework of its version 0.2.9, in particular its PyGrid component for remote execution of Machine Learning tasks. We adapt the solution of intrusion detection [1] to be compatible with PriMIA.

An implementation of ResNet50 is effortless as the basic building blocks were ready in PriMIA's implementation of ResNet18.

A. PriMIA framework

Used as a basis for our contribution, we focus on specific aspects of the PriMIA framework. The framework is open source, is interpreted in Python 3.7, and requires CUDA 10.x to enable GPU acceleration for training although in our experimentation, the GPU was not involved during inference. In this paper we will look closer at the inference process.

PriMIA's input data has a format of $224 \times 224 \times 1$ tensor, i.e., a 1-channel (usually representing a greyscale) square image with the dimensions 224×224.

The inference scenario is depicted in Fig. 4 relying on *Function Secret Sharing (FSS)* [10]. The client prepares their raw data, encrypts it using the trusted source of correlated randomness. Meanwhile, the server prepares shares of the function that computes the model. This concludes the offline phase not requiring any communication between parties. In the online phase, the masked shares are then exchanged to obtain the encrypted result. The client then uses their secret key to get the cleartext result.

The correlated randomness is particularly important in the Secure Multi-Party Computation. PriMIA falls into the category of MPC with pre-processing according to definitions in [15]. This means that a speed-up is achieved by generating randomness in parallel, independently of actual encryptions. To secure the computation, additional checks on the quality of randomness can be performed.

B. Fixed Fractional Precision

Shared secret computation is enabled for integers because they can be masked with the random strings to obtain additive secret shares. Not all representations of numbers allow the exploitation of this property. In particular, a common floating point implementation breaks these properties. The original PriMIA implementation uses *fixed fractional precision* parameter to handle the conversion between a decimal value and its expression in memory as an integer with a given length, multiples of a pre-defined factor. This conserves additive share properties equally for very large and small values.

PriMIA, through the underlying PySyft library[1], uses integers 64 or 32 bits long. An integer m represents a decimal value z, following the equation: $m = z \times b^p$, where base $b = 10$ and precision $p \in [\![1, 16]\!]$. Therefore, this system can represent decimal numbers up to p decimal places within the interval $[-\frac{2^{s-1}}{b^p}; \frac{2^{s-1}-1}{b^p}]$, which for `long` of size $s = 64$, base $b = 10$, and precision $p = 4$ provides an approximate range of $[-9.22 \times 10^{14}; 9.22 \times 10^{14}]$, while a precision $p = 16$ represents only values between -922 and 922.

These approximate ranges provide an insight why a computation with high (or low) precision might fail - due to a value overflow (or underflow). It is especially critical for model training to avoid exploding (or vanishing) gradients. In our use case, the inference on the model is also affected. The unwanted overflow and underflow interfere with additive shared secrets as well, where an "overflow" is part of the design of the encryption/decryption as the addition is performed given a specific modulo.

In order to achieve optimal results (best matching to the original results on non-encrypted model), we perform an exhaustive search on the hyperparameter of the *fixed fractional precision*. The results are presented in Fig. 5. Note that each of the three tests (t_1, t_2, t_3) was a single-item inference whose ground truth is presented in the bottom of the table. Even though we can see a clear pattern of which values of fixed fractional precision are optimal for this network and dataset, it might not be the case for all models and all data.

[1]PySyft v0.2.9: https://github.com/OpenMined/PySyft/tree/syft_0.2.x

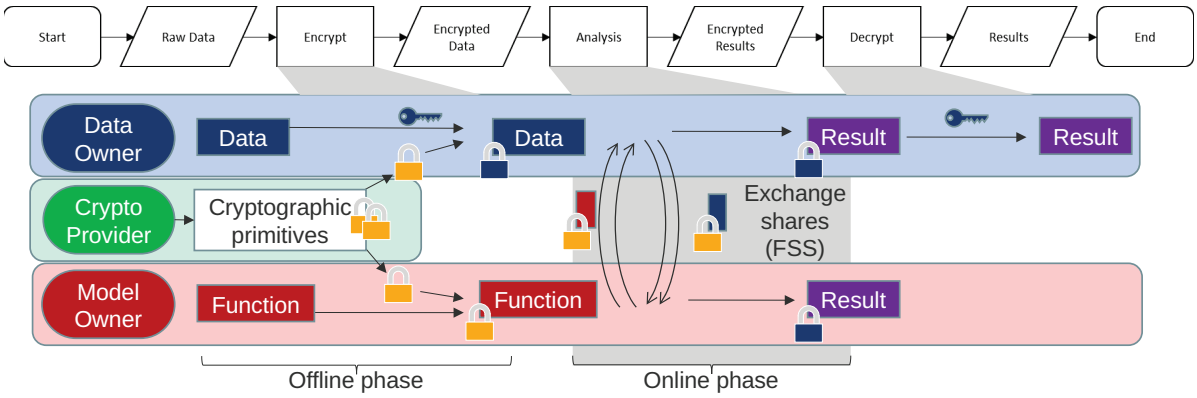

Fig. 4. Mapping of PriMIA framework to the use case of privacy-preserving analytics

Inference Mode	Fixed Precision	Tests t_1	t_2	t_3	
Encrypted	2	2	2	1	Deviation from unencrypted
Encrypted	**4**	**2**	**1**	**1**	Same as unencrypted
Encrypted	**6**	**2**	**1**	**1**	
Encrypted	8	2	2	0	Deviation from unencrypted
Encrypted	10	1	0	1	
Encrypted	12	0	0	0	
Encrypted	14	0	0	0	
Encrypted	16	0	0	0	
Unencrypted		**2**	**1**	**1**	
Ground Truth		**2**	**1**	**0**	

Classification result:
0: bacterial pneumonia
1: normal
2: viral pneumonia

Fig. 5. Fixed fractional precision hyperparameter search. In the original PriMIA setting, there are three classes (0, 1, 2). Changing the fixed fractional precision changes how well the encrypted inference matches the unencrypted inference results. This table provides evidence that this parameter is crucial for a reliable inference.

C. Function Secret Sharing

PriMIA's code[2] claims that there are differences between the Function Secret Sharing defined in the paper [10], although they are not specified and a detailed analysis might be required to establish the differences.

The FSS implementation in PriMIA requires three parties, from which only two are needed when dealing with the actual inference. The third party, a crypto-provider is the source of the correlated randomness. It is used to generate masking keys for the exchanges between the two active parties: the Inference Data Owner (*data owner* in PriMIA), and the Model Owner (also *model owner* in PriMIA).

The original FSS scheme and the improvements [10] were evaluated under the assumption of a semi-honest [9] adversary that is honest but curious. That is, an adversary that has the interest to follow the protocol but is interested in exploiting possible leaks to gain an insight in the other party's data. In order to defend against a malicious adversary, the authors suggest an extension by simple authentication methods.

The FSS is also *perfect secret sharing* scheme that does not leak any information about the plaintext from the ciphertext. The encryption is based on the principle of a one-time pad. Additive secret sharing protocol generates correlated randomness that serves to hide the plaintext.

[2]https://github.com/gkaissis/PriMIA

V. Experiments

Experiments were conducted to assess the overhead of the solution compared to an unencrypted computation. The inference stage is the focus of our use case of MLaaS.

We monitored the time, memory resources, and exactness with respect to the unencrypted inference.

A. Configuration

The results were achieved using two configurations M_1 for GPU-enabled training and M_2 for inference task:

- M_1: Ubuntu 20.04.1 LTS focal: Linux 5.4, 64 GB RAM - **32 cores** - Intel® Xeon® Silver 4215 CPU **2.50Ghz** - NVIDIA-driver 470.161.03; CUDA 10.1.243
- M_2: Ubuntu 22.04.1 LTS jammy: Linux 5.15, 64 GB RAM - **16 cores** - Intel® Core™ i7-10700K CPU **3.80GHz** - GPU not in use in the inference task

B. Methodology

The departure point was to reproduce the results from [1] using Deep Learning framework of PyTorch instead of Keras models that were used in the aforementioned work. ResNet50 model pre-trained on ImageNet was used for the initial training on M_1: 20 epochs on 14,995 images took 1 h 07 min.

Its performance is similar to the original work [1]. Their comparison is available in Table I and their respective confusion matrices in Fig. 6. The differences between the inference results of the two Deep Learning frameworks are sufficiently small to be equivalent for the purpose of this study.

The reason to consider a 5% subset is to obtain results of the encrypted inference within a reasonable time, given that

the inference on the full dataset would take approximately 42 days.

Previous works: Kodyš et al. 2021:

True \ Predicted	b	d	i	n	p	r	s	x
backdoor b	213	3	0	**106**	1	1	0	1
ddos d	18	273	4	**0**	0	20	0	0
injection i	0	0	521	**9**	2	1	2	0
normal n	**8**	**0**	**1**	**3179**	**0**	**0**	**0**	**1**
password p	0	8	7	**19**	364	1	5	4
ransomware r	0	0	22	**0**	7	141	0	5
scanning s	3	0	2	**2**	15	0	20	0
xss x	0	0	0	**4**	1	0	0	45

PriMIA adapted for IoT:

True \ Predicted	b	d	i	n	p	r	s	x
backdoor b	259	0	0	**65**	0	1	0	0
ddos d	0	312	3	**0**	0	0	0	0
injection i	0	0	528	**3**	0	0	4	0
normal n	**108**	**0**	**0**	**3081**	**0**	**0**	**0**	**0**
password p	0	8	0	**13**	387	0	0	0
ransomware r	0	0	0	**0**	0	170	0	5
scanning s	1	0	0	**0**	17	0	24	0
xss x	0	0	0	**1**	1	0	0	48

Fig. 6. Confusion matrices of two models: upper table is from [1]; lower table represents this work.
Both inferences were performed on TON_IoT dataset `TT500n`, with `miss3` imputation strategy that fills in missing values with `1` and actual values are replaced with `0`.
More significant differences between *normal* and *attack* classes are highlighted in bold red font.

We possess an unencrypted model that is the property of the Model Owner and will be protected by the applied technology according to our use case.

The performance of encrypted inference is measured on M_2, and compared to unencrypted inference. No GPU accelerated implementation of FSS was available as of June 2023.

C. Results

The presented results were obtained on ResNet50 architecture implemented in PyTorch. We chose to use `TT500n` dataset for comparison to the original TON_IoT sensor readings dataset. The imputation strategy `miss3` was chosen for its simplicity and results on par with more complex imputation strategies. Input for our ResNet50 architecture is an image $224 \times 224 \times 3$.

Table I compares the binary classification metrics for the reference model trained in [1] and our re-implementation. Additionally, we test the same model on a subset of 5% of the `TT500n_miss3` dataset. We note that an incidental improvement of FPR to 0% and a deterioration of TPR are visible but irrelevant - as they are due to an implicit bias of the random sample.

The comparison of unencrypted inference and its encrypted versions are shown in Table II, and further details of the inference on 251 items of the 5% `TT500n_miss3` dataset in form of confusion matrices Table III and Table IV. We verified item by item that the encircled values are the only change in

TABLE I
BINARY CLASSIFICATION METRICS

	[1]: Keras	This paper: PyTorch	
Dataset	TT500n_miss3	TT500n_miss3	5% TT500n_miss3*
Cardinality	5,039	5,039	251
Accuracy	94.4%	96.2%	97.6%
TPR	92.4%	95.6%	93.5%
FPR	0.3%	3.4%	0%

* Both, encrypted and unencrypted cases yield the same values.

TABLE II
DURATION AND MATCHING OF ENCRYPTED INFERENCE

	Unencrypted	Encrypted	
	CPU	Local	HTTP
Inference duration per item	0.15 s	677.31 s = 11 min 17 s	1,356.15 s = 22 min 36 s
Matching	100%	99.6% = 250/251	*

* Same as the Local Encrypted computation because only network stack differs, the computation algorithm is the same.

predictions: a single was classified as `ransomware` (an *attack* class) instead of originally correct `backdoor` (also an attack class in the binary classification). We note that the binary classification metrics for unencrypted and encrypted inference have the exact same values and correspond to those reported in the last column of Table I. The error in encrypted inference does not translate into any change of binary classification metrics as it remains within the set of *attack* classes.

VI. DISCUSSION

During the experimentation with PriMIA framework, the key parameter to have an immense influence was the *fixed fractional precision*. Its adjustment decided whether the re-

TABLE III
UNENCRYPTED INFERENCE – CONFUSION MATRIX

True Label	Predicted Label							
	b	d	i	n	p	r	s	x
backdoor b	(11)	0	0	5	0	(0)	0	0
ddos d	0	15	0	0	0	0	0	0
injection i	0	0	25	0	0	0	1	0
normal n	0	0	0	159	0	0	0	0
password p	0	0	0	1	19	0	0	0
ransomware r	0	0	0	0	0	9	0	0
scanning s	0	0	0	0	2	0	1	0
xss x	0	0	0	0	0	0	0	3

TABLE IV
ENCRYPTED INFERENCE – CONFUSION MATRIX

True Label	Predicted Label							
	b	d	i	n	p	r	s	x
backdoor b	(10)	0	0	5	0	(1)	0	0
ddos d	0	15	0	0	0	0	0	0
injection i	0	0	25	0	0	0	1	0
normal n	0	0	0	159	0	0	0	0
password p	0	0	0	1	19	0	0	0
ransomware r	0	0	0	0	0	9	0	0
scanning s	0	0	0	0	2	0	1	0
xss x	0	0	0	0	0	0	0	3

sulting output ended up the same value or it followed the unencrypted model. Either underflow or overflow degraded the performance. An extension of this work can focus on automation of the optimisation of this process beyond the naive method of enumeration search that we used in this paper. Further research can lead to a more comprehensive optimisation techniques.

Further research can be conducted to compare this approach to other privacy-preserving technologies. Especially, Fully Homomorphic Encryption shows substantial improvements in recent years and the efforts to improve the efficiency using GPU are under way.

Other Secure Multi-Party Computation protocols (e.g., Falcon [13]) are secure against a minority of malicious parties. Their use in 2-party setting is problematic because both parties are *de facto* required to be honest. It is also to be noted that the FSS implementation has an implicit third party, a trusted dealer – the source of correlated randomness. The original paper on secure computation via FSS [10] mentions the possibility of extension to malicious security model referring to authentication techniques of [15], [16], the issue can be mitigated by including tests in the protocol and abort if the tests show a deviation from the protocol. More insights about correlated randomness, on which many protocols rely, can be found in [15].

VII. Conclusion

This work applies privacy-preserving technology designed for medical imaging to intrusion detection.

Through the use case of providing a secured service of inference on customer's private data and ensuring the privacy of our own model, we illustrated different issues and possible ways forward to solve them. It includes a calibration of fixed precision to ensure the accuracy.

Following the current state of the art, we show the results of an application to ResNet50 allow for an inference through secure channel. The current speed of the inference is suitable for demonstration purposes using private data rather than a deployable Machine Learning as a Service solution.

There are remaining issues to be addressed in future research. An automated setting of the fixed precision would improve developer experience. More deep learning architectures should be implemented. Continuous improvements in underlying tools should also propagated in the future versions.

References

[1] M. Kodyš, Z. Lu, K. W. Fok, and V. L. L. Thing, "Intrusion detection in internet of things using convolutional neural networks," in *2021 18th International Conference on Privacy, Security and Trust (PST)*, 2021, pp. 1–10.

[2] G. Kaissis, A. Ziller, J. Passerat-Palmbach, T. Ryffel, D. Usynin, A. Trask, I. Lima, J. Mancuso, F. Jungmann, M.-M. Steinborn, A. Saleh, M. Makowski, D. Rueckert, and R. Braren, "End-to-end privacy preserving deep learning on multi-institutional medical imaging," *Nature Machine Intelligence*, vol. 3, no. 66, p. 473–484, Jun 2021.

[3] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT telemetry dataset: a new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016, p. 770–778.

[5] J. Domingo-Ferrer and A. Blanco-Justicia, "Privacy-preserving technologies," *The Ethics of Cybersecurity*, pp. 279–297, 2020.

[6] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, p. 114–148, Aug 2021.

[7] A. Stoian, J. Frery, R. Bredehoft, L. Montero, C. Kherfallah, and B. Chevallier-Mames, "Deep neural networks for encrypted inference with tfhe," Cryptology ePrint Archive, Paper 2023/257, 2023, https://eprint.iacr.org/2023/257. [Online]. Available: https://eprint.iacr.org/2023/257

[8] H. Tian, C. Zeng, Z. Ren, D. Chai, J. Zhang, K. Chen, and Q. Yang, "Sphinx: Enabling privacy-preserving online learning over the cloud," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 2487–2501.

[9] G. Oded, *Foundations of Cryptography: Volume 2, Basic Applications*, 1st ed. USA: Cambridge University Press, Aug 2009.

[10] E. Boyle, N. Gilboa, and Y. Ishai, "Secure computation with preprocessing via function secret sharing," 2019, report Number: 1095. [Online]. Available: https://eprint.iacr.org/2019/1095

[11] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, Oct 2016, p. 308–318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[12] 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."

[13] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: Honest-majority maliciously secure framework for private deep learning," *Proceedings on Privacy Enhancing Technologies*, 2021. [Online]. Available: https://petsymposium.org/popets/2021/popets-2021-0011.php

[14] J. Li, Y. Pan, Y. Lyu, Y. Yao, Y. Sui, and I. W. Tsang, "Earning extra performance from restrictive feedbacks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 10, pp. 11 753–11 765, 2023.

[15] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky, "On the power of correlated randomness in secure computation," in *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*. Springer, 2013, pp. 600–620.

[16] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias, "Semi-homomorphic encryption and multiparty computation," in *Advances in Cryptology – EUROCRYPT 2011*, ser. Lecture Notes in Computer Science, K. G. Paterson, Ed. Springer, pp. 169–188.