Secure Machine Intelligence and Distributed Ledger

Dmitry Arseniev[1](✉), Dmitry Baskakov[2], and Vyacheslav Shkodyrev[2]

[1] Saint-Petersburg State University, Saint Petersburg, Russia
[2] Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia

Abstract. Modern machine and deep learning systems are becoming part of high-performance cloud services and technologies. It is extremely important to understand that in systems such as recommendation systems, data is stored on local machines, and the trained system (matrix) is located in the cloud vendor, for example, in AWS or Google Cloud. Data on local machines can be updated or deleted. Local machines are often networked, which requires the use of synchronization methods and specialized protocols for the exchange of such information. And the central server is used as a single computer center for machine learning tasks. At the same time, it is necessary to control both the integrity of local data and their relevance with respect to other local machines. An important aspect is that the data center in the cloud should not know about our data, that is, we must be able to transmit them in encrypted form. At the same time, the deep learning model should be able to work with such encrypted data and send us the answers in encrypted form too. All this should be calculated in polynomial time, that is, quickly enough. For encryption purposes, it is proposed to use homomorphic algorithms. This report attempts to combine two promising modern paradigms for solving similar problems: machine intelligence and distributed ledger. For the purposes of distributed deep learning in relation to recommender systems, this symbiosis shows very serious practical prospects.

Keywords: Deep learning · Differential privacy · Homomorphic encryption · Machine learning · Secure computation

1 Introduction

Machine learning techniques are widely used in practice to produce predictive models for use in medicine, banking, recommendation services, threat analysis, and authentication technologies. The popularity and relevance of cloud machine learning has grown significantly. Moreover, often in such projects, in our opinion, due attention is paid specifically to issues of confidentiality and data security. In this paper, we consider a problem in which some neural network is located in the provider's cloud. This network is trained on some of its data and provides a service, for example, classification on images or customer data. The purpose of this work is to show and identify possible problems of such interaction, as well as to demonstrate the possibility of building secure protocols that would provide the ability to obtain classification results by a client without revealing their data to a neural network.

In the past years, deep neural networks (DNNs) have achieved remarkable progress in various fields, such as computer vision, natural language processing, and medical images analysis [1]. Consider a distributed recommendation system. Several data sources, for example, medical, send data from each clinic to the cloud for automatic diagnosis. In the cloud is a decision-making system based on a deep learning system [2]. In the general case, the look of this service using the Amazon SageMaker implementation example is as follows (Fig. 1):

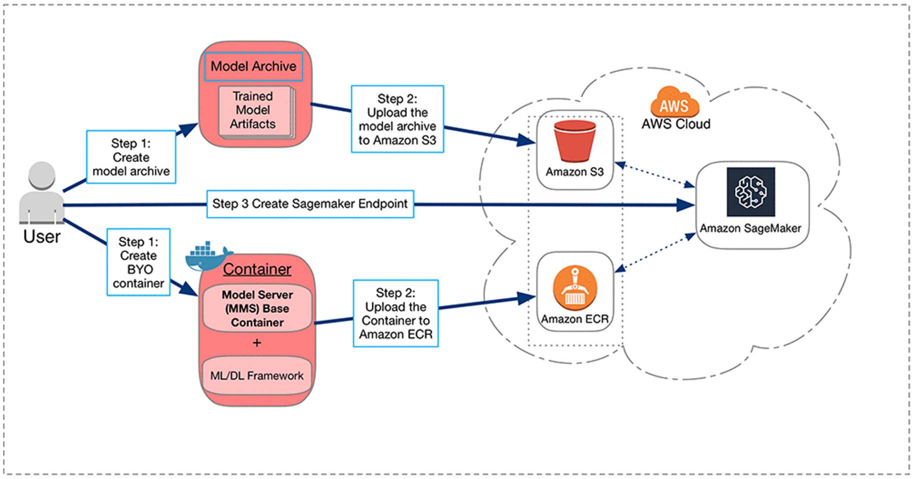
Model Archive
Trained Model Artifacts
Step 1: Create model archive
Step 2: Upload the model archive to Amazon S3
AWS
AWS Cloud
Amazon S3
User
Step 3 Create Sagemaker Endpoint
Amazon SageMaker
Step 1: Create BYO container
Container
Model Server (MMS) Base Container
+
ML/DL Framework
Step 2: Upload the Container to Amazon ECR
Amazon ECR

Fig. 1. Deep learning framework to Amazon SageMaker (https://aws.amazon.com/sagemaker/).

If this architecture is used, a problem arises when transferring data in *step 2* when loading the model or, much more often, just data in S3[1] or Amazon ECR[2]. Let's assume that it happens very often that we do not want to at least somehow provide the cloud service with access to our source data due to, for example, legislative restrictions. Suppose this is data both of a personal nature (personal) and commercial, which we cannot send in the usual way without encryption. In this case, we have the right to talk about *Private Distributed Recommender Systems* (businesses with proprietary consumer data would like to build recommender systems which can leverage data across all the businesses without compromising the privacy of any party's data) [3]. It should be understood that at this stage we continue to face all the features of machine learning that are traditionally inherent in this industry. Simple problem where standard deep earning either [4]:

[1] https://aws.amazon.com/s3/?nc=sn&loc=0.
[2] https://aws.amazon.com/ecr/

- Does not work well
  - Requires prior knowledge for better architectural/algorithmic choices
  - Requires other than gradient update rule
  - Requires to decompose the problem and add more supervision
  - Requires more data
- Does not work at all
  - No «local-search» algorithm can work
  - Even for «nice» distribution and well-specified models
  - Even with over-parameterization (a.k.a. improper learning)

In order to provide a secure exchange of data with cloud services, in addition to encryption, one or another protocol should be used that will make such data transactions more secure (Secure Deep Learning Inference, SDLI) [5]. It seems very promising to consider the possibility of using homomorphic encryption together with the use of blockchain technology to control both data integrity and to track the chain of possible data changes taking into account the distributed structure, which we will discuss later [6]. Deep learning as a service (DLaaS) has emerged as a promising to further enable the widespread use of DNNs in industry/daily-life. Google[3], Amazon[4], and IBM[5] have all launched DLaaS platforms in their cloud services. Using DLaaS, a client sends its private data to the cloud server. Then, the server is responsible for performing the DNN inference and sends the prediction results back to the client. Obviously, if the private client data, are not protected, using DLaaS will cause potential privacy issues. A curious server may collect sensitive information contained in the private data (i.e. client's privacy) [5].

To address this privacy issue, researchers have employed the homomorphic encryption to perform various DNN operators on encrypted client data [7]. As a result, the cloud server only serves as a computation platform but cannot access the raw data from clients. However, there exist two major obstacles in applying these approaches. First, some common non-linear activation functions, such as ReLU and Sigmoid, are not cryptographically computable [8]. Second, the inference processing efficiency is seriously degraded by thousands of times. To tackle these problems, a recent work proposes using an interactive paradigm. A DNN inference is partitioned into linear and non-linear computation parts. Then, only the linear computations are performed on the cloud server with encrypted data. The nonlinear computations are performed by the client with raw data. However, in such an interactive paradigm, the intermediate features extracted by the linear computations are directly exposed (sent back) to the client. Thus, a curious client can leverage these features to reconstruct the weights of the DNN model held by the cloud. This issue is called the leakage of server's privacy [5].

In fact, a practical solution for secure DNN inference should protect both client's privacy and server's privacy. In addition, it should support DNNs with all types of

[3] https://cloud.google.com/products/ai.

[4] https://aws.amazon.com/machine-learning/

[5] https://www.ibm.com/analytics/machine-learning.

non-linear activation functions. Unfortunately, there still lacks an effective approach in literature. Our key strategy is to combine deep learning, homomorphic encryption and distributed ledger.

The traditional approach of using distributed ledger involves the use of this technology in artificial intelligence systems, for example, in robotic systems. For example, cyberphysical systems (Robotics) exchange some information among themselves, the integrity of which should be controlled. Such systems learn, store this data, pass it to other nodes, and so on. There are even some implementations of such an approach and such solutions on the market [9]. There were other concepts for using distributed ledger in artificial intelligence. For example, there were prerequisites and ideas for building decentralized platforms that would allow the user to create, organize joint participation and monetize artificial intelligence systems using blockchain technologies [9]. And of course, the use of distributed ledger in conjunction with artificial intelligence systems was not at all limited to such solutions, of which there were, in fact, quite a lot. One of the curious examples is the use of machine learning for joint decision making or forecasting of certain processes [10]. The use of a distributed machine learning system along with blockchain technologies has been successful enough to predict the market prices of certain assets [10].

2 Materials and Methods

We introduce a few basic primitives that we will need for our solution, namely:

- Deep Neural Network (DNN), which is located at the cloud provider.
- Homomorphic encryption (HE). Additive homomorphic encryption (AHE).
- Secret Sharing (SS) and Garbled Circuit (GC).
- Oblivious ROM (OROM).
- Data Aggregation (DA).
- Differential Privacy (DP).

2.1 Deep Neural Network

DNN, for example, Convolutional neural network (CNN) in cloud provider. Suppose that a cloud provider or we, as the owners of this service, would not want to discover both the weights of this neural network, its hyperparameters, and the models that we used in the training process (Fig. 2):

In fact, in the provider's cloud, we store a matrix with the weights of a neural network of the form:

$$\begin{bmatrix} \omega_{1,1} & \cdots & \omega_{1n} \\ \vdots & \ddots & \vdots \\ \omega_{m,1} & \cdots & \omega_{m,n} \end{bmatrix} \tag{1}$$

That is, we enter a certain vector with signs at the input, it is multiplied by matrices and at the output we get again the response vector and belonging to some class (Fig. 3):

It is important that the neural network does not know about the input data and their structure, and, preferably, does not remember the output data or stores it in its memory.

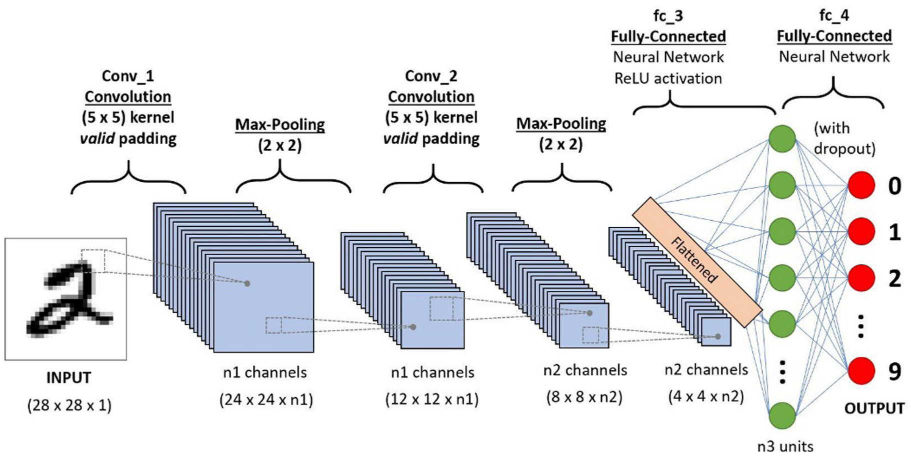

Fig. 2. Convolutional neural network (https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53).

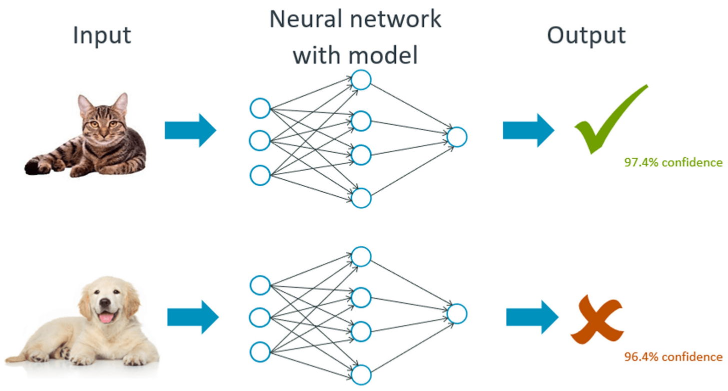

Fig. 3. Deep learning inference

2.2 Homomorphic Encryption (HE)

Homomorphic encryption is a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext. Homomorphic encryption can be used for privacy-preserving outsourced storage and computation. This allows data to be encrypted and out-sourced to commercial cloud environments for processing, all while encrypted [11]. Fully Homomorphic Encryption (FHE), is an encryption method that allows anyone to compute an arbitrary function f on an encryption of x, without decrypting it and without knowledge of the private key [12]. Using just the encryption of x, one can obtain an encryption of $f(x)$. The major bottleneck for these techniques, notwithstanding these recent developments, is their computational complexity. But recent efforts, both theory and in practice have given us large results in the performance of homomorphic scheme [13, 14] (Fig. 4)

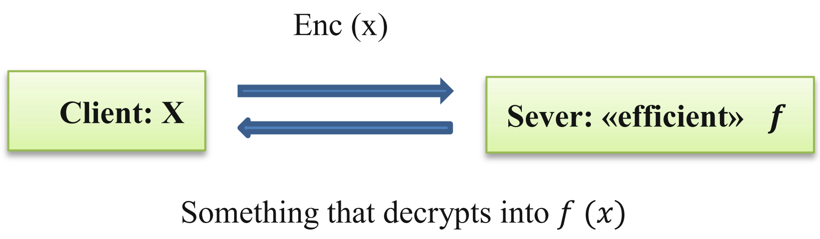

Fig. 4. Homomorphic encryption

It is very important to use an effective homomorphic encryption model. What is efficient?

- Small low-degree arithmetic circuit.
- Small Boolean circuit.

2.3 Additive Homomorphic Encryption (AHE)

A (private-key) additive homomorphic encryption (AHE) scheme is private-key encryption scheme with three additional algorithms Add; CAdd and CMult, which supports adding two ciphertexts, and addition/multiplication by constants. We require our AHE scheme to satisfy standard IND-CPA security and circuit privacy, which means that a ciphertext generated from Add, CAdd and CMult operations should not leak more information about the operations to the secret key owner, other than the decrypted message [15].

2.4 Secret Sharing and Garbled Circuit

Gabled circuit (GC) is a cryptographic protocol that enables two-party secure computation in which two mistrusting parties can jointly evaluate a function over their private inputs without the presence of a trusted third party. In the garbled circuit protocol, the function has to be described as a Boolean circuit. The invention of garbled circuit was credited to Andrew Yao, as Yao introduced the idea in the oral presentation of his paper [16]. Improvements to GC have been proposed in literature, for example, free-XOR and half-gates [17]. Using Advanced Encryption Standard (AES) as the block cipher, we leverage Intel AES instructions for faster garbling procedure [15]. Yao's garbled circuits [16] and the secret-sharing based Goldreich-Micali-Wigderson (GMW) protocol [18] are two leading methods for the task of two-party secure computation (2PC). Now, after three decades theoretical and applied work about improving and optimizing these protocols, we have modern and efficient implementations [19].

One of the main problems of all these protocols and techniques until recently was their communication complexity[6]. Indeed, three recent works followed the garbled circuits paradigm and designed systems for secure neural network inference: the Secure ML system [20], the Mini ONN system [21], the Deep Secure system [14, 22]. Actually, it is not so obvious to use only Secret sharing or Garbled circuit in machine learning security problems. Our vision is that compromises should be sought, in which case it is better to use both technologies depending on the tasks.

We can use Hybrid protocols:

- mix homomorphic encryption and garbled circuits via secret sharing
- Homomorphic Encryption for linear operations and Garbled Circuits for non-linear operations
- Homomorphic Encryption for fully-connected layers and Garbled Circuits for ReLu-activation [15].

3 Oblivious RAM

Consider the simplest model of our calculations and secure compute $a[i]$ (Fig. 5):

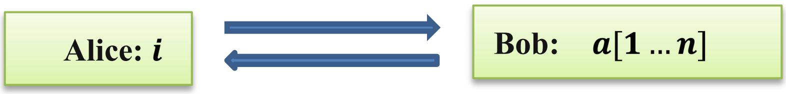

Fig. 5. Securely compute a.

The key problem is that the network often stores the data of our access to it, information about transactions, other data. Ideally, I would like the network to somehow know how to forget all this and not even store encrypted data accessing it. Oblivious RAM (ORAM) algorithms, first proposed by Goldreich and Ostrovsky [23], allow a client to conceal its access pattern to the remote storage by continuously shuffling and re-encrypting data as they are accessed. An adversary can observe the physical storage locations accessed, but the ORAM algorithm ensures that the adversary has negligible probability of learning anything about the true (logical) access pattern. Since its proposal, the research community has strived to find an ORAM scheme that is not only theoretically interesting, but also practical [24, 25].

In this case ORAM, distributed machine learning is characterized by the following types of possible threats and attacks (Fig. 6):

At the very key factor of all these attacks is the ability Data Aggregation.

4 Data Aggregation

Here, we introduce the most prominent data privacy preserving mechanisms. Not all these methods are applied to deep learning, but we briefly discuss them for the sake of comprehensiveness. These methods can be broadly divided into two groups of context-free

[6] https://en.wikipedia.org/wiki/Communication_complexity.

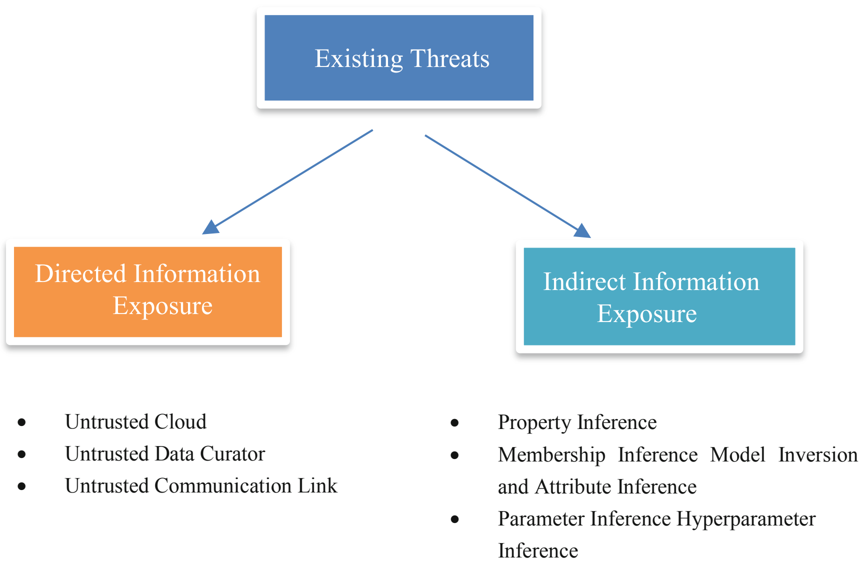
Existing Threats
Directed Information Exposure
Indirect Information Exposure
Untrusted Cloud
Untrusted Data Curator
Untrusted Communication Link
Property Inference
Membership Inference Model Inversion and Attribute Inference
Parameter Inference Hyperparameter Inference

Fig. 6. Existing threats deep learning.

privacy and context-aware. Context-free privacy solutions, such as differential privacy, are unaware of the specific context or the purpose that the data will be used for. Whereas context-aware privacy solutions, such as information-theoretic privacy, are aware of the context where the data is going to be used, and can achieve an improved privacy-utility tradeoff.

4.1 Naïve Data Anonymization

What we mean by naive anonymization in this survey is the removal of identifiers from data, such as the names, addresses, and full postcodes of the participants, to protect privacy. This method was used for protecting patients while processing medical data and has been shown to fail on many occasions [26].

4.2 K-Anonymity

A dataset has k-anonymity property if each participant's information cannot be distinguished from at least $k - 1$ other participants whose information is in the dataset. K-anonymity means that for any given combination of attributes that are available to the adversary (these attributes are called quasi-identifiers), there are at least k rows with the exact same set of attributes. K-anonymity has the objective of impeding re-identification [26].

4.3 Semantic Security and Encryption

Semantic security (computationally secure) is a standard privacy requirement of encryption schemes which states that the advantage (a measure of how successfully an adversary can attack a cryptographic algorithm) of an adversary with background information should be cryptographically small.

5 Distributed Ledger

Actually, there are not so many works and studies where qualitative integration of two such well-known paradigms as machinery intelligence and blockchain was given or offered one way or another. Existing works clearly, of course, solve certain problems facing the industry and researchers, but this is clearly not enough if we are talking about the integration of such important modern technologies [9, 27, 28]. In our approach to distributed machine learning as a service, the use of Blockchain technology is due to a number of important circumstances, namely:

- Oblivious RAM implies a limit on the number of transactions N, where N must necessarily be bounded above by some integer variable:

$$N \leq K \tag{2}$$

- All participants in the distributed computer network of machine learning, in the case of access to the cloud provider (AWS, Google Cloud), save the most important parameters of the access to the database in the blockchain, for example, the number of requests or transactions n_i, where:

$$N \approx \sum_{i=1}^{m} n_i \tag{3}$$

- When the counter of transactions or hits N becomes greater than or equal to K $N \geq K$, then controlled deletion of data from the neural network should take place, or, as an option, reconfiguration of its parameters, hyperparameters which in theory should entail the removal of data from the network [29].
- A temporary restriction $t \leq T$ on the use of this neural network if, for some reason, users have stopped changing the counter settings or even stopped accessing the services of a cloud provider.

Thus, the general algorithm of work will consist in the fact that each time the service provider is accessed, the client stores data on the number of transactions in the distributed ledger, after exceeding which the saved user data on the neural network are reset, as well as the possible setting of new parameters in accordance with the concept Oblivious ROM [30].

6 Differential Privacy

Definition 5.1. $\epsilon -$ ***Differential Privacy***$(\epsilon - $ ***DP***$)$**.** For $\epsilon \geq 0$, an algorithm A. satisfies $\epsilon - DP$ [31] if and only if for any pair of datasets D and D' that differ in only one element:

$$P[A(D) = t] \leq e^{\epsilon} P\left[A\left(D'\right) = t\right] \forall t \tag{4}$$

Where, $P[A(D) = t]$ denotes the probability that the algorithm A outputs t. In this setup, the quantity bel is named the *privacy loss:*

$$\ln \frac{P[A(D) = t]}{P[A(D') = t]} \tag{5}$$

DP tries to approximate the effect of an individual opting out of contributing to the dataset, by ensuring that any effect due to the inclusion of one's data is small. One of the widely used DP mechanisms when dealing with numerical data is the Laplace mechanism [32].

Definition 5.2. ***Laplace Mechanism.*** Given a target function f and fixed $\epsilon \geq 0$, the randomizing algorithm $A_f(D) + x$ where x is perturbation random variable drawn from Laplace distribution:

$$Lap\left(\mu, \frac{\Delta_f}{\epsilon}\right) \tag{6}$$

Is called the Laplace Mechanism and is $\epsilon - DP$. Here, Δ_f is called global sensitivity of function f, and is defined as:

$$\Delta_f = \sup\left|f(D) - f\left(D'\right)\right| \tag{7}$$

For all the dataset pair $\left(D, D'\right)$ that differ in only one element. Finding this sensitivity is not always trivial, specifically if the function f is a deep neural network, or even a number of layers of it [33]. Differential privacy satisfies a composition property that states when two mechanisms with privacy budgets ε_1 and ε_2 are applied to the same datasets, together they use a privacy budget $\varepsilon_1 + \varepsilon_2$. As such, composing multiple differentially private mechanisms consumes a linearly increasing privacy budget. It has been shown that tighter privacy bound for composition can be reached, so that the privacy budget decreases sub-linearly (Fig. 7):

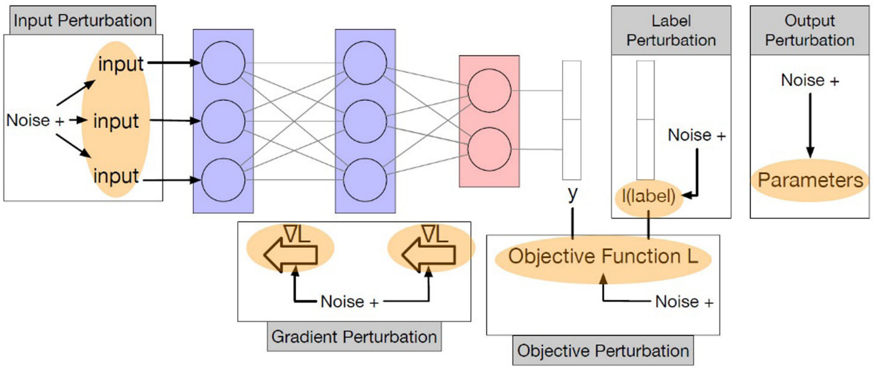

Fig. 7. How work differential privacy in deep learning [32]

7 Conclusion and Discussion

In this paper, we examined modern approaches and concepts that relate to distributed machine learning. We in no way claimed to be an exhaustive exposition of this area, which is impossible in principle. Nevertheless, we showed that it is extremely important to use a distributed ledger in order to control both the integrity of the data in the cloud, and forcibly delete data or reconfigure the parameters of the neural network. In the case of joint and remote operation of several nodes at once, in the case of joint and remote operation of several nodes at once, these requirements become extremely important.

References

1. TRADI: Tracking deep neural network weight distributions (2020). https://arxiv.org/pdf/1912.11316v3.pdf. Accessed 03 May 2020
2. Buniatyan, D.: Hyper: distributed cloud processing for large-scale deep learning tasks (2019). https://arxiv.org/pdf/1910.07172v1.pdf. Accessed 25 May 2020
3. Tsikhanovich, M., Magdon-Ismail, M., Ishaq, M., PD-ML-Lite: private distributed machine learning from lightweight cryptography (2019). https://arxiv.org/pdf/1901.07986v2.pdf. Accessed 27 Apr 2020
4. Failures of Deep Learning (2017). https://simons.berkeley.edu/talks/shai-shalev-shwartz-2017-3-28. Accessed 10 Apr 2020
5. BAYHENN: Combining Bayesian deep learning and homomorphic encryption for secure DNN inference (2019). https://arxiv.org/pdf/1906.00639v2.pdf. Accessed 01 May 2020
6. Boddeti, V.N.: Secure face matching using fully homomorphic encryption (2018). https://arxiv.org/pdf/1805.00577v2.pdf. Accessed 2020 Apr 15
7. Chialva, D., Dooms, A.: Conditionals in homomorphic encryption, and machine learning applications (2019). https://arxiv.org/pdf/1810.12380v2.pdf. Accessed 11 Mar 2020
8. Thaine, P., Gorbunov, S., Penn, G.: Efficient evaluation of activation functions over encrypted data (2020). http://www.cs.toronto.edu/~pthaine/EEAFED.pdf. Accessed 17 Mar 2020
9. Lopes, V., Alexandre, L.A.: An overview of blockchain integration with robotics and artificial intelligence (2018). https://arxiv.org/pdf/1810.00329.pdf. Accessed 24 Feb 2020
10. Craib, R., Bradway, G., Dunn, X., Krug, J.: White paper: Numeraire: a cryptographic token for coordinating machine intelligence and preventing overfitting (2017)

11. Halevi, S.: Homomorphic encryption (2017). https://shaih.github.io/pubs/he-chapter.pdf. Accessed 17 Apr 2020
12. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. Thesis, Stanford University (2009)
13. Palisade homomorphic encryption software library. https://palisade-crypto.org/software-library/
14. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: GAZELLE: A Low Latency Framework for Secure Neural Network Inference, Baltimore, MD, USA, 15–17 August 2018
15. Chen, H., Chillotti, I., Dong, Y., Poburinnaya, O.: SANNS: scaling up secure approximate K-nearest neighbors search (2020). https://arxiv.org/pdf/1904.02033.pdf. Accessed 04 Apr 2020
16. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science (SFCS 1986) (1986)
17. Zahur, S., Rosulek, M., Evans, D.: Two Halves make a whole. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 220–250. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_8
18. Goldreich, O., Micali, S., Wigderson, A.: How to plait any mental game (1987)
19. Demmler, D., Schneider, T., Zohner, M.: ABY – a framework for efficient mixed-protocol secure two-party computation. In: 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA (2015)
20. Mohassel, P., Zhang, Y.: SecureML: a system for scalable privacy-preserving machine learning. In: Conference: 2017 IEEE Symposium on Security and Privacy (SP) (2017)
21. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via minionn transformations. In: 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas (2017)
22. Ouhani, B.D., Riazi, M.S., Koushanfar, F.: Deepsecure: scalable provably-secure deep learning (2017). https://arxiv.org/ftp/arxiv/papers/1705/1705.08963.pdf. Accessed 14 Mar 2020
23. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. In: ACM (1996)
24. Stefanovy, E., van Dijkz, M., Shi, E.: Path ORAM: an extremely simple oblivious RAM protocol (2013). https://eprint.iacr.org/2013/280.pdf
25. Stefanovy, E., van Dijkz, M., Shi, E.: Path ORAM: an extremely simple oblivious RAM protocol (2013). https://eprint.iacr.org/2013/280.pdf
26. Sweeney, L.: “k-anonymity: A model for protecting. Int. J. Uncertain. Fuzz. Knowl. Based Syst. **10**, 557–570 (2002)
27. Chen, F., Wany, H., Caiz, H., Cheng, G.: Machine learning in/for blockchain: future and future and challenges (2020). https://arxiv.org/pdf/1909.06189v2.pdf. Accessed 07 May 2020
28. Zheng, Z., Dai, H.-N., Wu, J.: Blockchain intelligence: when blockchain meets artificial intelligence (2020). https://arxiv.org/pdf/1912.06485v3.pdf. Accessed 03 May 2020
29. Serizawaa, T., Fujita, H.: Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization (2020). https://arxiv.org/ftp/arxiv/papers/2001/2001.05670.pdf. Accessed 11 Mar 2020
30. Liu, J., Tai, X.-C., Luo, S.: Convex shape prior for deep neural convolution network based eye fundus images segmentation (2020). https://arxiv.org/pdf/2005.07476v1.pdf. Accessed 13 May 2020
31. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Third Conference, Berlin (2006)
32. Mireshghallah, F., Taram, M., Vepakomma, P., Singh, A., Raskar, R., Esmaeilzadeh, H.: Privacy in deep learning: a survey. https://arxiv.org/pdf/2004.12254v3.pdf. Accessed 11 May 2020

33. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., Jana, S.: Certified robustness to adversarial examples with differential privacy (2019). https://arxiv.org/abs/1802.03471. Accessed 07 Feb 2020
34. Lopes, V., Alexandre, L.A.: An overview of blockchain integration with robotics and artificial intelligence (2018). https://arxiv.org/pdf/1810.00329v1.pdf. Accessed 29 Apr 2020