Privacy-Preserving Deep Learning and Inference (Invited Paper)

M. Sadegh Riazi, and Farinaz Koushanfar
Electrical and Computer Engineering Department
University of California San Diego
mriazi@ucsd.edu, farinaz@ucsd.edu

ABSTRACT

We provide a systemization of knowledge of the recent progress made in addressing the crucial problem of deep learning on encrypted data. The problem is important due to the prevalence of deep learning models across various applications, and privacy concerns over the exposure of deep learning IP and user's data. Our focus is on provably secure methodologies that rely on cryptographic primitives and not trusted third parties/platforms. Computational intensity of the learning models, together with the complexity of realization of the cryptography algorithms hinder the practical implementation a challenge. We provide a summary of the state-of-the-art, comparison of the existing solutions, as well as future challenges and opportunities.

KEYWORDS

Artificial Intelligence, Machine Learning, Deep Learning, Privacy, Security, Privacy-Preserving Deep Learning, Secure Function Evaluation, Homomorphic Encryption, Secret Sharing

1 INTRODUCTION

Contemporary datasets are quickly growing in terms of dimension, size and sophistication, and the valuable trends learned from the extensive amount of data holds promise for a significant paradigm shift in several important applications. While many algorithms for learning from the massive datasets are available, deep learning-based algorithms have been the most popular option in the past five years due to their increased accuracy. Deep learning (DL) relies on multiple layers of sophisticated neural network architecture and intensive computing to deliver its promise. As a result, cloud platforms and big data servers are the most prominent platform of choice for big data processing and deep learning. A prevalent emerging business model is offering deep learning as a service, where the Intellectual Property (IP) of the service provider is the wealth of its data/resources utilized for building the sophisticated learning models executed on user's data. However, in this business model and with restrictive new privacy laws (e.g., in Europe and California), concerns over user privacy and cloud IP piracy present major roadblocks to the future adaptation of these statistical models.

One naive solution to mitigate the risk of data leakage is to have the consumers acquire the model and execute it on their own trusted platform. This solution is not practicable in real-world settings due to (at least) two main reasons: (i) The DL model is learned by processing massive amounts of training data held by big technology companies. As such, the trained AI models are usually considered an IP; companies require confidentiality to preserve their competitive advantage and ensure receiving continuous query requests by the users. (ii) DL models can reveal information about the data on which they were trained, violating the privacy of users whose data was utilized during the training phase.

This paper aims to provide a systemization of knowledge of the active field of privacy-preserving deep learning based on provably secure primitives. We summarize the research in the two categories of training and execution separately. During DL training, the focus is to fine-tune the DL parameters such that the accuracy of the model is maximized. Whereas, the DL execution attempts to find the corresponding inference label for newly arrived data samples using the trained model. Training methodologies that work on private data typically assume that the content is arriving from multiple distributed parties respecting the privacy of each individual party. The objective is different for various application scenarios; for example, if there is one party or a server building the final model, they get the full information about the model. In another scenario, various parties may own pieces of the model and then combine them to construct the final model. DL execution on private data commonly assumes two parties: a model owner (e.g., server) and a client with data who needs to execute the model on its content.

Depending on the privacy-preserving objective and scenarios, different cryptographic primitives have been used. These range from secret sharing to Homomorphic Encryption [2, 3], Garbled Circuits [21], and Goldreich-Micali-Wigderson protocols [8]. While each of these methods provides a number of advantages, none of them possesses all the desired properties for the wide-range of privacy-preserving applications of interest. Therefore, a number of recent private machine learning methodologies are built upon a hybrid combination of the basic protocols that aim to bring the best characteristics to each custom application. This paper covers both basic and hybrid protocols that are provably secure. Note that a number of recent solutions for DL on encrypted data based on a trusted third party are also being developed [15]. This paper does not include such trusted third party solutions such as those built on trusted platforms, e.g., SGX, where the key assumption is that the keys are held by Intel [12].

ICCAD '18, November 5–8, 2018, San Diego, CA, USA

ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3274560

The remainder of the paper is organized as follows. Section 2 provides the necessary background in terms of deep neural networks and various cryptographic primitives. In Section 3 we discuss the generic framework for privacy-preserving deep learning. Private training is outlined in Section 4. Execution of deep learning on private data is described in Section 5. The paper concludes in Section 6 while citing the present challenges and future research directions.

2 BACKGROUND

We provide a short background on deep neural networks as well as the cryptographic primitives leveraged in the privacy-preserving frameworks.

2.1 Deep Neural Networks

Deep Neural Networks (DNN) have demonstrated a break-through classification accuracy that can even outperform human-level accuracy in many applications such as speech/visual recognition and natural language processing. DNN has a hierarchical structure where each layer is stacked on top of the previous layer. The first layer accepts the input data and the output layer produces the confidence values corresponding to each possible class. There are one (or more) "hidden" layers in between. Layers are comprised of set of neurons. Each layer receives the result from the previous layer, performs a specific operation, and passes the result to the next layer. For example, in a *fully connected* layer, the value of each neuron is computed as a linear combination of neurons in the previous layer followed by a non-linear *activation* function. Widely adopted activation functions include Rectifier Linear Unit (ReLu) $f(x) = max(0, x)$, hyperbolic tangent $f(x) = \frac{e^{2x}-1}{e^{2x}+1}$, and Sigmoid $f(x) = (1 + e^{-x})^{-1}$. Convolutional is another popular type of layer that operates on two (or higher) dimensional inputs by moving a window on the input and computing a linear combination of selected values. Convolutional layers are extensively used in visual imagery applications.

2.2 Cryptographic Primitives

In this section, we briefly review the cryptographic primitives that are used in the privacy-preserving frameworks for training and execution (inference) of deep neural networks.

Homomorphic Encryption (HE) is an encryption method that enables computation on the encrypted version of the data [3, 7]. For example, in an *additive* homomorphic encryption scheme, given encryption of two numbers a and b as $\mathcal{E}(a)$ and $\mathcal{E}(b)$, $\mathcal{E}(a+b)$ can be computed as $\mathcal{E}(a) \odot \mathcal{E}(b)$ where $\odot$ is a scheme-specific operation. A HE-scheme is called *fully* homomorphic if it can support arbitrary computation on the encrypted form of data.

Secret Sharing (SS) is a method to distribute a secret s, e.g., an integer value, to many untrusted parties such that each share does not reveal any information about the secret but the combination of all (or a subset) of shares can reconstruct the secret. For example, additive secret sharing can produce two shares as $sh_1 = r$ and $sh_2 = s - r$ where r is a randomly generated number and $s - r$ is the subtraction over a finite field. While both sh_1 and sh_2 do not reveal anything about s, the secret can be reconstructed as $s = sh_1 + sh_2$.

Garbled Circuits (GC) is a generic secure function evaluation protocol introduced by Andrew Yao which enables two parties to jointly evaluate a function $f(.)$ on their inputs without disclosing anything but the outcome of the computation [21]. In this protocol, the function $f(.)$ has to be described as a Boolean circuit with two-input gates. The protocol runs in a constant number of interactions regardless of the number of gates or the depth of the circuit.

Goldreich-Micali-Wigderson (GMW) is another generic secure function evaluation protocol. Similar to GC, it requires the function to be described as a Boolean circuit [8]. However, unlike GC, it requires one round of interaction for evaluating each layer of Boolean gates. Compared to GC, it requires less data communication but the round complexity is linear with respect to the depth of the circuit.

3 PRIVACY-PRESERVING FRAMEWORKS

During the past few years, there has been an extensive research on how to perform privacy-preserving deep learning and inference. In this article, we briefly summarize this effort. We categorize the frameworks into private training and private inference frameworks since the computation and security model is drastically different. In the first group, the goal is to *train* a model where the training data is distributed among many (usually more than two) parties and each party wants to keep her data private. At the end of the protocol, the model can be either held in plaintext by one party or be secret-shared among many parties. In the second category of frameworks, the goal is to perform private *execution* (inference) where one party has an already trained model and the other party has an input to the model for which she is interested to receive the outcome. The inference is essentially a two-party problem. Figure 1 illustrates the process of training and inference in deep learning.

In what follows, we briefly review the state-of-the-art frameworks in each category. The security model considered in all of the following frameworks is Honest-but-Curious (HbC) adversary model where participants are assumed to follow the protocol but they can attempt to infer information based on the data they send and receive. There exist more powerful attack models such as security against malicious adversaries in which an adversary can deviate from the protocol at any time. However, in practice, HbC adversary model provides reasonable security guarantees for data owners. Moreover, such solutions can be used as a building block for stronger security models.

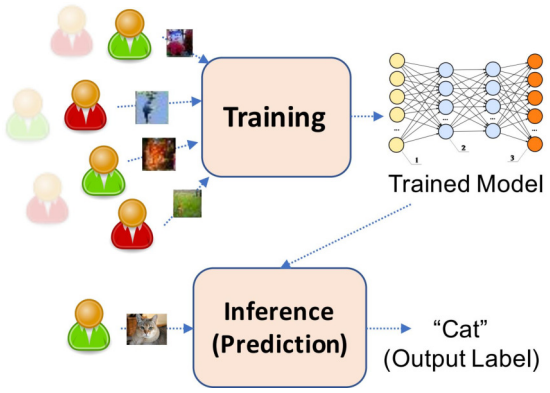

Figure 1: Overview of training and inference in DL.

Table 1: High-level characteristics of different privacy-preserving inference frameworks.

Framework	Preserving Accuracy	Independence of Secondary Server	Data&Network Embedding	Data Batching	Scalability for very Deep models	Fixed-Point Operations	Non-Linear Activation	Cryptographic Primitive
CryptoNets	N	Y	N	Y	N	N	N	HE
SecureML	N	N	N	N	Y	Y	Y	HE, GC, SS
DeepSecure	Y	Y	Y	N	Y	Y	Y	GC
MiniONN	Y	Y	N	N	Y	Y	Y	HE, GC, SS
Chameleon	Y	N	N	N	Y	Y	Y	GC, SS, GMW

4 PRIVATE TRAINING

One of the early solutions for privacy-preserving deep learning is suggested by Shokri and Shmatikov [18]. They propose a non-cryptographic solution in which each data owner trains her private model and selectively shares the updates of DL model parameters to other parties. Each data owner can then update the local model with the updates she has received. To further mitigate the information leakage, authors propose that each party adds a carefully chosen noise to the update before sending it to the other parties.

Google has introduced a secure data aggregation protocol that can compute the sum of high-dimensional vectors held by different parties [1]. The protocol is based on Shamir's secret sharing and is robust to parties dropping in the middle of the protocol. The secure data aggregation protocol can be used in *federated* learning in which many clients privately train and hold their models but they contribute to improving a model held by a central agency. The model updates from all parties are aggregated in a secure protocol and then the result is revealed to the central server. The server leverages the sum of updates to improve the accuracy of its model.

Hitaj et al. have demonstrated a powerful attack against privacy-preserving federated and decentralized deep learning [9]. The attack is based on Generative Adversarial Network (GAN) that can produce a prototypical sample of the target class, revealing the private data of another honest participant. More generally, authors show that applying record-level differential privacy in collaborative learning is not effective and they suggest utilizing solutions based on secure computation protocols.

In this direction, SecureML [14] framework is proposed that utilizes cryptographic primitives such as GC, HE, and secret sharing. Data owners secret share their data and send it to two non-colluding servers who train the model on the private data. The learned model is also secret-shared between the two servers. Authors have studied the problem of training models for linear regression, logistic regression, and deep neural networks. The framework also supports private inference after the model is trained.

5 PRIVATE INFERENCE

In this section, we review the privacy-preserving frameworks proposed for neural network inference in the chronological order.

CryptoNets [4] is a framework developed by Microsoft Research that is based on Fully Homomorphic Encryption (FHE). In CryptoNets, non-linear activation functions are approximated by low-degree polynomials such that they can efficiently be evaluated using FHE. As a result, the neural network has to be (re)trained with the suggested activation functions in order to maintain the prediction accuracy. Each level of multiplication in the FHE scheme increases the *noise* within the ciphertext. Therefore, given a certain set of parameters for the FHE scheme, a limited number of consecutive multiplications can be performed, thus, the notion of *leveled* FHE. Increasing the multiplication depth drastically increases the computation complexity. FHE scheme utilized in CryptoNets has the property that multiple plaintext values can be embedded inside a single ciphertext which in turn amortizes the computation and communication costs over many neural network inferences.

DeepSecure [17] is a framework based on the GC protocol that addresses two of the most important shortcomings of the CryptoNets: (i) since DeepSecure is based on GC, it can support any non-linear activation function such as ReLu; and (ii) increasing the number of layers in the DL model does not blow up the overall computation and communication because the overall costs only depends on the number of gates in the function. DeepSecure also introduces *data and network preprocessing* step in order to reduce the overhead of the GC protocol. This step transforms the input to an ensemble of lower dimensional sub-spaces and compacts the DL model [5, 13]. The preprocessing step is performed by two parties independently in plaintext and has significantly less computational overhead compared to the GC protocol.

The main computational overhead of DeepSecure is due to the multiplication operations, i.e., fully-connected and convolutional layers. Securely computing multiplication using the GC protocol is costly since the circuit has *quadratic* size as a function of input bits. Utilizing GC for multiplication also results in high communication overhead. In order to overcome these limitations, two frameworks are proposed that suggest performing multiplication based on secret-sharing: MiniONN [11] and Chameleon [16]. The multiplication can then be performed by precomputing "multiplication triplets". The MiniONN framework suggests precomputing multiplication triplets using homomorphic encryption.

Chameleon framework, on the other hand, proposes the idea of utilizing *correlated randomness* [10] generated by a semi-honest third party. Leveraging correlated randomness can significantly reduce the computation and communication overhead of the secure computation. Chameleon comprises two phases: (i) an offline phase in which the correlated shares are generated and distributed to the two parties (who possess the private data); and (ii) an online phase in which two parties execute the secure computation protocol. Similar to the SecureML framework, Chameleon requires an additional non-colluding party. However, in contrast to SecureML, the additional party is not present in the online phase. In Chameleon, low-depth activation functions are computed using the GMW protocol.

Table 1 summarizes the high-level characteristic of different private inference frameworks.

6 CONCLUSIONS AND FUTURE DIRECTIONS

During the past few years, there has been an extensive research on developing privacy-preserving frameworks for neural network training and inference. These frameworks have different high-level characteristics and utilize different cryptographic primitives.

It is worth-mentioning that private inference frameworks inherently cannot solve all security and privacy challenges related to deep learning. For example, several attacks have been proposed that exploit the result of the DL model to extract more information about the model parameters or the training data: (i) Model Inversion [6] attack infers information about the training data used to train the model, (ii) Model Extraction [20] attack recovers some information about the model parameters by sending many queries to the model and analyzing the result, and (iii) Membership Inference [19] attack attempts to find out whether a specific sample data has been used during training the neural network model. All of these attacks can extract more information only based on the input data and the output result. As such, they only access the model as a black-box and are considered out-of-scope of private inference frameworks. The information leakage can significantly be mitigated by omitting the confidence scores and only sending the outcome class label. Figure 2 illustrates a high-level description of model extraction and membership inference attacks. The scope of secure computation is also outlined as the dashed orange box.

Privacy-preserving training of deep neural networks has received less attention compared to private inference. Private training inherently has higher computation overhead. More research in this direction is needed to find efficient solutions that can preserve the privacy of data owners while building more powerful models. As a future direction, it is necessary to investigate the efficient integration of different cryptographic solutions. Mixed-protocol solutions have illustrated a break-through in reducing the computation and communication overhead of privacy-preserving solutions [11, 16]. Another promising paradigm is to utilize correlated randomness generated by a third-party dealer. Such random (but correlated) shares are independent of the private data and the computation task and can be generated beforehand. Customizing secure computation protocols for deep learning (and vice versa) is another viable research direction and opens the door for more efficient solutions.

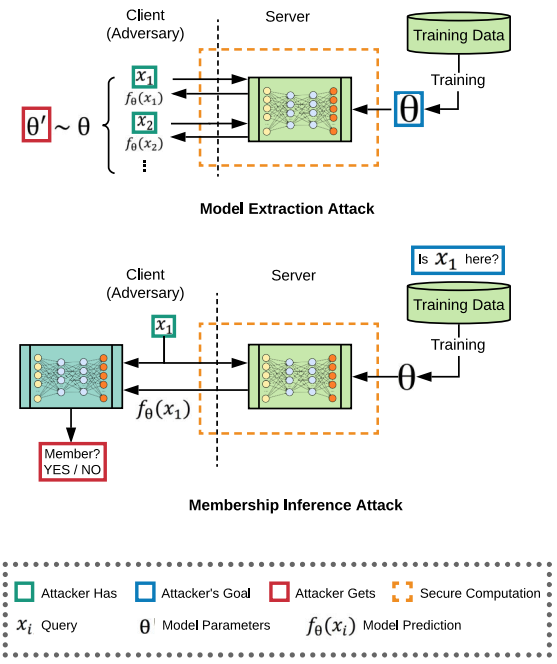

Figure 2: Overview of attacks on DL models.

ACKNOWLEDGMENTS

This paper was supported, in parts, by the following grants: NSF Trust-Hub CNS-1649423, NSF GC@Scale CNS-1619261, MURI-SUB 141414 (FA9550-14-1-0351), and ONR-N00014-17-1-2500.

REFERENCES

[1] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*. ACM.

[2] Joppe W Bos, Kristin E Lauter, Jake Loftus, and Michael Naehrig. 2013. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme.. In *IMA*. Springer.

[3] Zvika Brakerski and Vinod Vaikuntanathan. 2014. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.* 43, 2 (2014), 831–871.

[4] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*.

[5] Eva L Dyer, Aswin C Sankaranarayanan, and Richard G Baraniuk. 2013. Greedy feature selection for subspace clustering. *The Journal of Machine Learning Research* 14, 1 (2013), 2487–2517.

[6] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*. ACM.

[7] Craig Gentry et al. 2009. Fully homomorphic encryption using ideal lattices.. In *STOC*, Vol. 9. 169–178.

[8] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to play any mental game. In *STOC*.

[9] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *CCS*. ACM.

[10] Yan Huang. 2012. *Practical secure two-party computation.* Ph.D. Dissertation. Citeseer.

[11] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. 2017. Oblivious Neural Network Predictions via MiniONN transformations. In *CCS*. ACM.

[12] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. 2013. Innovative instructions and software model for isolated execution. *HASP@ ISCA* 10 (2013).

[13] Azalia Mirhoseini, Bita Darvish Rouhani, Ebrahim M Songhori, and Farinaz Koushanfar. 2016. Perform-ml: Performance optimized machine learning by platform and content aware customization. In *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*. IEEE, 1–6.

[14] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning.. In *IEEE S&P*.

[15] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. 2016. Oblivious Multi-Party Machine Learning on Trusted Processors.. In *USENIX Security Symposium*. 619–636.

[16] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. In *ASIACCS*. ACM.

[17] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. 2018. DeepSecure: Scalable Provably-Secure Deep Learning. In *DAC*. IEEE.

[18] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 1310–1321.

[19] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *S&P*. IEEE.

[20] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs.. In *USENIX Security*.

[21] Andrew Yao. 1986. How to generate and exchange secrets. In *FOCS*.