Deep Learning on Private Data

M. Sadegh Riazi, Bita Darvish Rouhani, and Farinaz Koushanfar | University of California San Diego

Emerging complex deep neural networks require vast amounts of data to achieve high precision. However, the information is often collected from user logs and personal data. In this article, we summarize recent cryptographic methodologies for provably privacy-preserving deep learning and inference.

With the ever-increasing volume of data in our digital world, powerful machine-learning models are needed to efficiently process the vast amount of information in different applications and industries. Hierarchical deep neural networks (DNNs), also known as *deep learning* (*DL*) techniques, provide efficient modeling methodologies for automatic extraction of the underlying features in the content. In several important and frequent tasks, including visual classification, game playing, and speech recognition, DL achieves superior accuracy in comparison with human cognition.

In a supervised setting, DL involves two distinct phases: training and inference (prediction). Figure 1 illustrates a high-level block diagram of the learning process for devising a typical machine-learning system. The performance of a DL model is directly dependent on the volume of available training data. However, the training samples are usually collected from users' content stored on cloud servers—content that contains sensitive information, such as images, voice recordings, location logs, and medical records.

Users' privacy is of critical concern not only during training but also for inference. Internet companies are now providing machine learning as a service where users can send their queries to cloud servers and receive a prediction result by paying certain fees. These queries can include sensitive data, such as users' health records, sent to the remote DL servers for medical diagnosis.

One naïve solution to mitigate the risk of data leakage during DL inference is to have consumers download the model and run it on their own trusted platform. But this solution is not attainable in real-world settings because the DL model is learned by processing massive amounts of training data. As such, the trained artificial intelligence models are usually considered to be sensitive intellectual property.[1] Companies require confidentiality to preserve their competitive advantage.

In a nutshell, three main requirements should be considered in privacy-preserving DL.

1. During the training phase, users' sensitive data should not be revealed to the central server that is training the DL model.
2. During inference, users' queries should not be revealed to the server.
3. The server's proprietary DL model should not be revealed to the user.

Efficient methodologies for private inference and training are essential for future advances in deep learning. In theory, any function can be evaluated on private inputs from two or more distinct parties using secure function evaluation (SFE) protocols. However, a naïve transformation of a particular function using the generic

Digital Object Identifier 10.1109/MSEC.2019.2935666
Date of current version: 24 September 2019

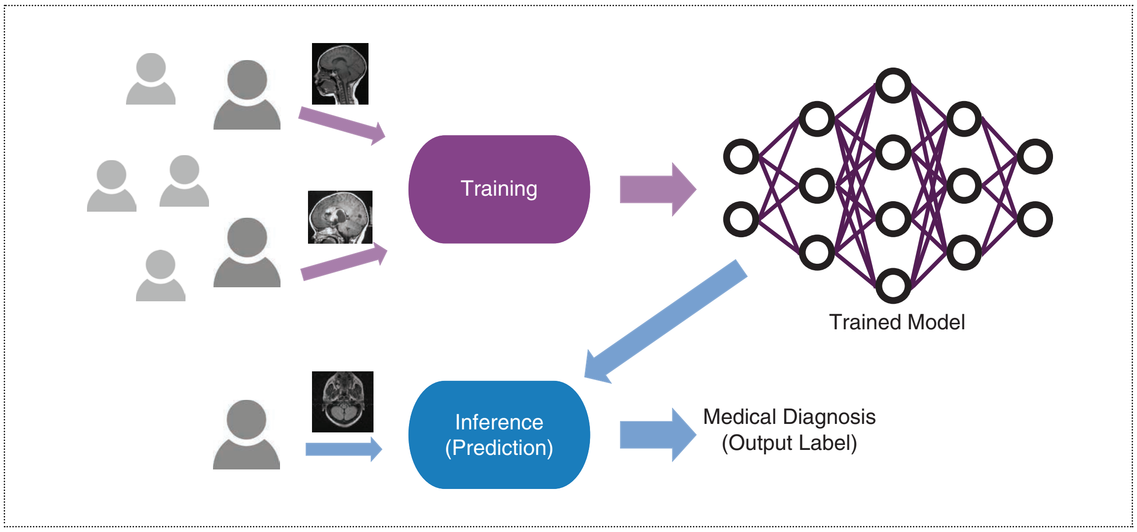

Figure 1. An overview of training and inference in DL.

SFE protocols incurs several orders of magnitude of overhead, both in terms of computation and communication. To empower privacy-preserving computation in the context of DL, it is crucially important to devise novel domain-specific customized techniques that can reduce the complexity of SFE protocol execution.

This article presents a systemization of knowledge of the most recent privacy-preserving frameworks for DL and provides a comprehensive comparison in terms of the unique properties and performances on standard benchmarks. We also include a summary of the recent attacks and illustrate how they are related to privacy-preserving frameworks. Pertinent terminology can be found in "Glossary of Low-Level Building Blocks Leveraged for Privacy-Preserving Deep Learning."

"Efficient methodologies for private inference and training are essential for future advances in deep learning."

DNNs

DL is a subclass of machine learning that resembles the functionality of a human brain. There are different variants of DL architectures. However, all of them have a layered structure, starting from an input layer and ending with an output layer. There can be one or more layers in between, which are called *hidden layers*. The output of a given layer is the input to the next layer. Once new raw data are given as input to the NN, the output of each layer is computed until reaching the final output layer. In most cases, the output values represent the probabilities of different classes that the input can belong to. Popular main categories of the neuron networks that are widely used in DL are DNNs, convolutional NNs (CNNs), and recurrent NNs (RNNs).

The DNN is a more generic architecture used in various applications. The most computationally expensive parts of a DNN involve matrix multiplication, also known as *fully connected layers*. In addition, DNNs have an activation layer, which is the only nonlinear part of their structure. Popular activation functions include the rectified linear unit (ReLU), logistic Sigmoid, and hyperbolic tangent. The output layer is usually a softmax layer that translates the output of the last hidden layer to a probability vector. In addition to the aforementioned layers, CNNs have

- one or more convolution layers, each of which is a weighted sum of different segments of the previous layer; these weights as well as the weights in the fully connected layer are learned in the DL training phase
- one or more pooling layers, each of which selects either the average (mean-pooling) or maximum (max-pooling) of different segments in the previous layer.

CNNs are specifically used where input data have spatial correlation, such as in facial recognition systems. In contrast to DNNs and CNNs, RNNs have an internal state. This characteristic makes RNNs suitable for

Glossary of Low-Level Building Blocks Leveraged for Privacy-Preserving Deep Learning

- *Garbled circuits (GC)*[S1]: This is one of the generic secure function evaluation (SFE) protocols introduced by Andrew Yao in 1986. It enables two parties, Alice and Bob, to jointly compute a function on their private inputs without revealing anything but the outcome of the function. The first step in utilizing this protocol is to describe the function as a Boolean circuit with two-input logic gates. Alice then starts assigning random keys to each wire in the circuit. For each gate, she encrypts the output keys of the gates using the corresponding input key pairs and forms garbled tables. She sends all of the garbled tables as well as the input keys associated with her input. Bob also acquires the correct keys associated with his input. After that, Bob decrypts each gate one by one until he finds the output keys. Finally, Alice provides the mapping between the output keys to the true semantic values to generate the plaintext output of the function. Note that the number of interactions between Alice and Bob is constant and independent of the function.
- *Goldreich–Micali–Wigderson (GMW)*[S2]: This is another generic SFE protocol. Similar to GC, GMW requires that the function be described as a Boolean circuit. However, unlike GC, two parties need to interact for every AND gate. Therefore, by paralleling all of the independent AND gates, the communication round complexity is *linear* with respect to the depth of the circuit. The GMW protocol requires only small communication for each gate compared to the GC protocol.
- *Secret sharing (SS)*: This is a method to distribute a secret to two or more parties where each share does not provide any information about the secret, but the secret can be reconstructed from the shares. One of the most popular SS variants is additive SS. In this case, a secret is shared by sampling random numbers and creating the last share such that the summation of all of the shares yields the secret value. The secret can be reconstructed by adding all of the shares.
- *Homomorphic encryption (HE)*: This is a cryptographic primitive that enables one party to encrypt data and send it to another party, who can then perform certain operations on the encrypted version of the data. Upon completion of the computation, the encrypted version of the result is sent back to the first party, who can then decrypt it and get the result in plaintext. For example, an additively HE, such as the Paillier cryptosystem,[S3] supports addition on the encrypted version of two numbers. In a fully HE,[S4] an arbitrary functional logic can be computed.
- *Differential privacy (DP)*[S5]: This is a metric that defines how much information about a single entry in a database is revealed upon a query to the database. To preserve the privacy of database entries, carefully chosen noise is added to the database, such that the statistical properties of the database are preserved while each data point is changed because of the added noise. Equivalently, DP can be seen as a way to reduce the dependency between the outcome of a computation and one of the data points in the database, thus reducing information leakage.

References

S1. A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. 27th IEEE Annu. Symp. Foundations Computer Science, (sfcs 1986)*, pp. 162–167.

S2. O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. 19th ACM Annu. Symp. Theory Computing*, 1987, pp. 218–229.

S3. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory and Applications Cryptographic Techniques*, 1999, pp. 223–238.

S4. C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. Annu. ACM Symp. Theory Computing*, vol. 9, 2009, pp. 169–178.

S5. C. Dwork, "Differential privacy," in *Encyclopedia of Cryptography and Security*. New York: Springer-Verlag, 2011, pp. 338–340.

applications where a sequence of input data needs to be processed. RNNs are widely used for speech recognition.

Delivering an efficient solution for privacy-preserving DL has been an active research area in recent years. The proposed frameworks can be categorized based on supporting training or inference. Training and inference are inherently different. In training, the data are usually distributed among many parties, and a centralized entity, e.g., an Internet company, may wish to learn the model on all of the data samples. Therefore, training the model without revealing each party's input is essentially a multiparty problem. In contrast, inference is a two-party problem in which one party holds an already trained model and another party has a private input.

Privacy-preserving DL frameworks can also be categorized based on the family of the utilized secure computation protocols, i.e., homomorphic encryption (HE), garbled circuits (GC), secret sharing (SS), and Goldreich–Micali–Wigderson (GMW). While all of these methods have provable privacy properties, each of them has specific characteristics and capabilities. In addition to secure computation approaches, differential privacy (DP) has been suggested as a solution for mitigating the information leakage when training a NN. However, unlike SFE protocols, DP does not provide a perfectly leakageproof solution.

Private Training

In what follows, we present a brief description of the systems and frameworks to train a DL model on a set of distributed data sets held by different individuals while preserving the data confidentiality and privacy of the data owners.

One approach, explored by Shokri and Shmatikov,[2] is to have each party locally train his or her version of the DL model and selectively share some of the updated parameters with a central server. The intuition is that optimization algorithms can be run in parallel by different individuals, and the results can be aggregated later on. To further mitigate the information leakage, the authors add specific noise to the model updates before sharing them instead of sending the raw values to the server. As a result, they introduce a tradeoff between the accuracy of the trained NN and the privacy of data. While this solution has a minimal computation overhead, it does not provide provable security guarantees in collaborative training. As we discuss later in this article, it has been shown that a malicious party can infer sensitive information about other participants' private data in this setting.

In light of designing a system with concrete privacy guarantees, Google has announced a cryptography-based approach for secure data aggregation in which updates from all clients are aggregated without revealing each individual update.[4] The protocol is based on Shamir's SS and is robust against clients exiting the protocol at any time.

The task of privacy-preserving training can be approached differently where the training data are secret -shared by each data owner to two (or more) non -colluding servers (SecureML and SecureNN). These servers execute specific secure computation protocols to train a DL model while keeping the training data private. All of the intermediate values are secret-shared among servers, and at the end of the protocol, servers hold a share of the trained model. The secret-shared model can be used directly in subsequent secure computation protocols to provide privacy-preserving inference, or the trained model can be reconstructed from the shares to achieve the model in cleartext. In the SecureML[3] system,

Table 1. The characteristics of privacy-preserving inference frameworks and their underlying cryptographic primitives.

Framework	Preserving accuracy	Constant number of interactions	DL model preprocessing	Scalability for large DL models	Nonpolynomial activation and max-pooling	Cryptographic protocol(s)
CryptoNets	✗	✓	✗	✗	✗	Leveled HE
SecureML	✗	✗	✗	✓	✗	Linearly HE, GC, SS
MiniONN (with Sqr Act.)	✗	✗	✗	✓	✗	Additively HE, GC, SS
MiniONN (with ReLU and pooling)	✓	✗	✗	✓	✓	
DeepSecure	✓	✓	✗	✓	✓	GC
DeepSecure (plus preprocessing)	✗	✓	✓	✓	✓	
Chameleon	✓	✗	✗	✓	✓	GMW, GC, SS
Gazelle	✓	✗	✗	✓	✓	Additively HE, GC, SS
XONN	✓	✓	✓	✓	✓	GC, SS

Conv: convolutional; Sqr Act: square activation function.

Table 2. A performance comparison of different frameworks for privacy-preserving inference (prediction) on the MNIST data set, given similar computational platforms.

Framework	Prediction timing (s)			Communication (MB)			Accuracy (%)	DNN architecture
	Offline	Online	Total	Offline	Online	Total		
CryptoNets	–	297.5	297.5	–	372.2	372.2	98.95	Conv-Sqr Act-MeanP-Sqr Act-FC
SecureML	4.7	0.18	4.88	–	–	–	93.1	FC-Sqr Act-FC-Sqr Act-FC
MiniONN (with Sqr Act.)	0.9	0.14	1.04	3.8	12	15.8	97.6	
DeepSecure	–	9.67	9.67	–	791	791	99	Conv-ReLU-FC-ReLU-FC
DeepSecure (plus preprocessing)	–	1.08	1.08	–	88.2	88.2	99	
Chameleon	1.34	1.36	2.7	7.8	5.1	12.9	99	
MiniONN	3.58	5.74	9.32	20.9	636.6	657.5	99	Conv-ReLU-MaxP-Conv-ReLU-MaxP-FC-ReLU-FC-ReLU
Gazelle	0.65	0.51	1.16	47.5	22.5	70.0	99	
XONN	–	0.15	0.15	–	32.13	32.13	99	

FC: fully connected layer; MaxP: max-pooling; MeanP: mean-pooling.

the authors propose a method to train and run DL models securely. The solution is based on HE, GC, and SS. Data owners secret-share their data to multiple noncolluding servers, which privately train the NN. SecureML uses a customized activation function that is more efficient for training a neural network using secure computation protocols. The SecureNN framework (SecureNN) proposes new protocols for training and inference on deep neural networks using three noncolluding servers. The training time and communication between servers can be significantly reduced in this computation model. Similarly, the ABY3 framework (aby3) introduces new protocols in the three-server setting where intermediate values are kept in arithmetic, binary, and Yao shares during the protocol execution. Compared to the two-party SecureML system, the training time is reduced by up to four orders of magnitude.

We do not quantitatively compare the solutions for privacy-preserving training since the methodologies have different security guarantees and computational models. Therefore, comparing the training times is not meaningful.

Private Inference

In the past few years, privacy-preserving inference for an already trained NN has been the main research focus. In this article, we discuss the suggested methods. The high-level comparison of the frameworks is provided in Table 1, while the quantitative comparison of their performance is shown in Table 2. All of the discussed frameworks are secure in the honest-but-curious (HbC) adversary model in which all parties are assumed to follow the protocol, although they might attempt to infer more information based on the data they send and receive. The HbC adversary model is a stepping stone to more powerful models, such as security against malicious adversaries, where any party can deviate from the protocol at any time.

Developed by Microsoft Research, CryptoNets[5] leverages leveled HE,[6] a variant of HE that supports a certain number of consecutive multiplications of a ciphertext. In HE, ciphertexts inherently have some noise. After each multiplication, the noise grows, and after many multiplications, the result is no longer correct. Therefore, the parameters of HE, e.g., the size of the ciphertext, can be adjusted based on the security threshold and the number of multiplications that the scheme has to support. The latter is derived from the NN architecture. Relying on leveled HE has the benefit that the client has to do significantly less computation compared to the server, which is usually computationally more powerful. However, popular nonpolynomial activation functions such as ReLU cannot be realized efficiently using HE. The authors propose to approximate the activation functions using low-degree polynomials. Therefore, the NN model should be retrained in plaintext using the same activation function to maintain high prediction accuracy.

Perhaps the most critical limitation of this approach is that increasing the number of layers in the NN model requires larger ciphertext size, which, in turn, significantly increases the computation and communication overhead. Researchers report experimental results on the Modified National Institute of Standards and Technology (MNIST) data set, which is a collection of handwritten digits, each represented as a 28 × 28 pixel image where each pixel takes a gray-scale value between zero and 255. To perform a single private inference, CryptoNets takes 298 seconds and requires 372 MB of communication.

In light of supporting nonpolynomial activation functions and pooling operations, the DeepSecure[8] framework has been introduced, which uses GC as its backbone cryptographic engine. DeepSecure does not require a retraining procedure and supports any activation and pooling functions. Additionally, DeepSecure introduces the idea of reducing the size of the data and the network by a preprocessing stage and before executing the GC protocol, thus compacting the computation and communication. The preprocessing stage is independent of the underlying cryptographic protocol and can be adopted by any other back-end engines.

The major downside of GC is the enormous communication overhead for multiplication, a ubiquitous operation in all NN models. To evaluate any function in GC, the functionality has to be described as a Boolean circuit. Since the number of Boolean gates in the multiplication circuit grows quadratically with respect to the bit width of operands, realizing secure multiplication in GC is inefficient.

To benefit from the best characteristic of different secure computation protocols, several frameworks are proposed that rely on mixed-protocol methodologies. The MiniONN framework[7] proposes to use GC to execute nonlinear activation functions and SS-based methods to execute linear operations. MiniONN has two main computation phases: an offline phase, based on additively HE to precompute random shares that are independent of the actual inputs, and an online phase, based on GC and SS.

The Chameleon framework[9] utilizes the GMW protocol for low-depth nonlinear activation functions and GC for more complicated nonlinear activation functions, e.g., Sigmoid, as well as for pooling layers. All arithmetical operations, such as addition and multiplication, are performed using SS. Similar to its concurrent work, MiniONN, Chameleon has offline and online computation phases. Most of the computation is shifted to the offline phase to provide a fast online prediction phase. The offline phase consists of creating the correlated randomness used in the online phase for all three protocols, i.e., GC, GMW, and SS. These random (but correlated) shares are independent of the actual inputs and functionality that are being evaluated; they have to be created by a third party.

Similar to SecureML, Chameleon requires the presence of two noncolluding servers. However, in contrast to SecureML, the third party is not involved in the online phase. Moreover, the functionality of the third party can be replaced using the Intel Software Guard Extension (SGX)[10] and achieve a two-party computation model. The GMW protocol in Chameleon supports single instruction, multiple data (SIMD), which creates a way to process the same operation on multiple data considerably faster.

Gazelle[15] is another mixed-protocol solution that suggests leveraging HE to perform linear operations and GC for nonlinear activation functions. The authors propose an efficient algorithm to realize convolutional layers using HE. As a result, Gazelle improves the runtime of private inference and reduces the communication between the client and the server.

While mixed-protocol solutions offer an interesting approach to reduce the private inference runtime, they require (at least) one round of interaction between client and server per each layer in the neural network. In Internet settings, this property can significantly deteriorate the performance and increase the execution time due to high communication delay.

The XONN system[16] introduces a different approach and suggests transforming the neural network such that the underlying operations are more compatible with secure computation protocols. The authors propose to binarize the neural network to avoid any multiplication during the private inference. In other words, all parameters and weights in the NN are restricted to take a binary value, zero or one. This, in turn, transforms the multiplication to a combination of XOR and bit-count operations. By relying on the GC protocol as the back-end cryptographic engine, the XOR operation can be executed with negligible computation and no communication.[19] As a result, XONN achieves state-of-the-art performance in private inference. In addition, XONN requires only a constant number of communication rounds regardless of the number of layers in the NN. Relying on standalone GC leads to another advantage, too: the solution can be upgraded using standard protocols to be secure against active adversaries that can deviate from the protocol at any time. The authors report experimental results on four medical data sets to enable secure and privacy-preserving medical diagnosis for breast cancer, diabetes, liver disease, and malaria infection, with inference time in the range of tens to a few hundred milliseconds. In the next section, we elaborate on the transformation procedure of XONN.

Preprocessing Neural Networks

In the XONN framework, a one-time preprocessing step is proposed to transform the neural network to a representation that is more efficient for secure computation protocols. The procedure starts by binarizing neural networks in which the DL parameters take only binary values. Given a binary NN, the costly matrix multiplication will be transformed into bitwise-XOR and bit-count operations. Due to the free-XOR optimization[19] proposed

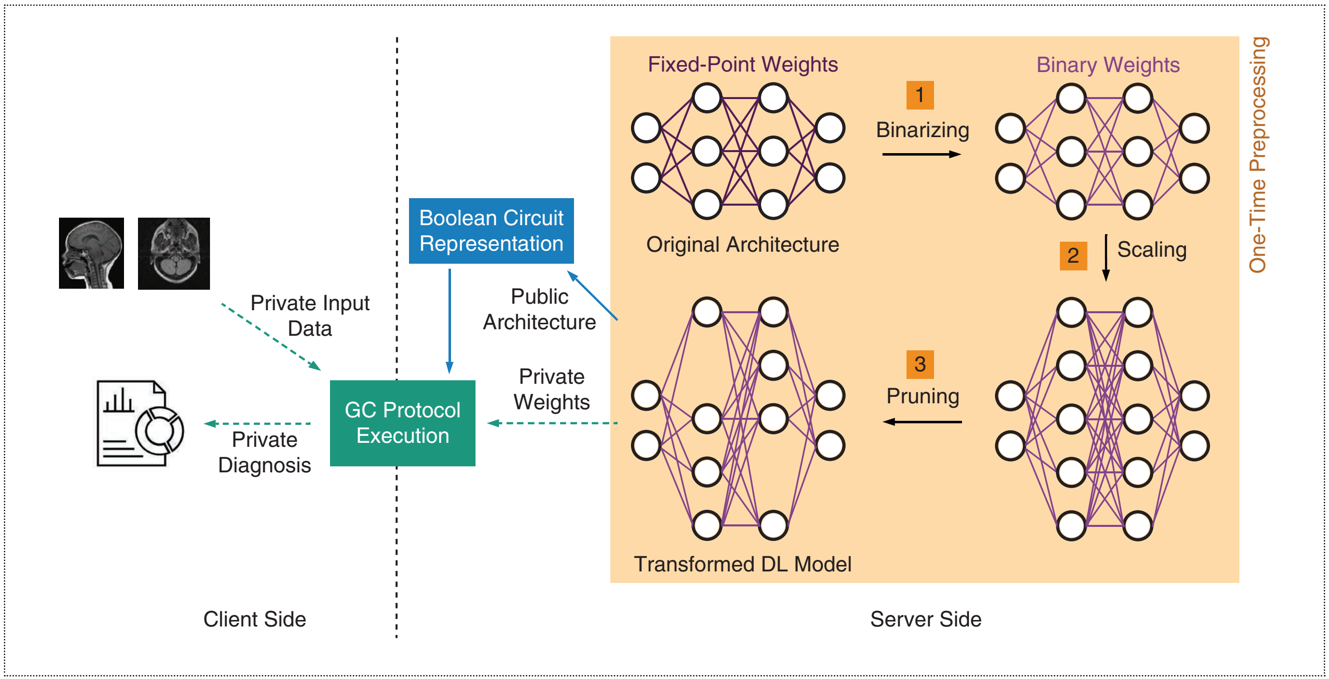

Figure 2. The internal architecture of the XONN framework comprising the core GC execution engine and the preprocessing steps.

for the GC protocol, the XOR operations can be evaluated with minimal computation and no communication. As a result, DL models can be preprocessed according to the secure computation protocols to render significantly faster inference. One major downside of binarizing neural networks is the reduced prediction accuracy. In XONN, a two-phase process is introduced to boost the accuracy back to the desired level. The two phases are called scaling and pruning. In the first phase, the number of neurons in each layer is scaled by a constant factor and the model is retrained. If the prediction accuracy is more than the desired level, the process stops; otherwise, the process is rerun with a higher scaling factor. Once the prediction accuracy passes the desired level, the pruning phase starts. In this phase, the XONN algorithm selects and removes neurons that contribute least to the final accuracy of the DL model. The algorithm selects these neurons based on the cost function tailored for the GC protocol. In the end, a trimmed DL model is created that incurs minimal computation and communication cost in the GC protocol with desired prediction accuracy. As input to the GC protocol, a Boolean circuit representation of the model is produced and passed to the protocol. Figure 2 illustrates an overview of the preprocessing step and the execution flow in XONN.

“DL models can be preprocessed according to the secure computation protocols to render significantly faster inference.”

Securely Outsourcing the Computation

In scenarios where the client (input holder) is greatly resource constrained, such as with mobile phones and embedded devices, it is favorable to outsource the computation to a secondary server. While there can be many solutions to this problem, a simple one is to use the XOR-sharing technique adopted by DeepSecure[8] and XONN.[16] In this method, a client who owns the input data only has to XOR his or her original input with a random binary vector of the same size as the input. The XORed result is then sent to one of the servers and the random vector to the secondary server. Both servers then execute the GC protocol and send back the shares of the result. The client has only to XOR the final shares to reconstruct the true prediction result.

In the outsourcing mode, the client needs only to perform the XOR operation, which has a negligible

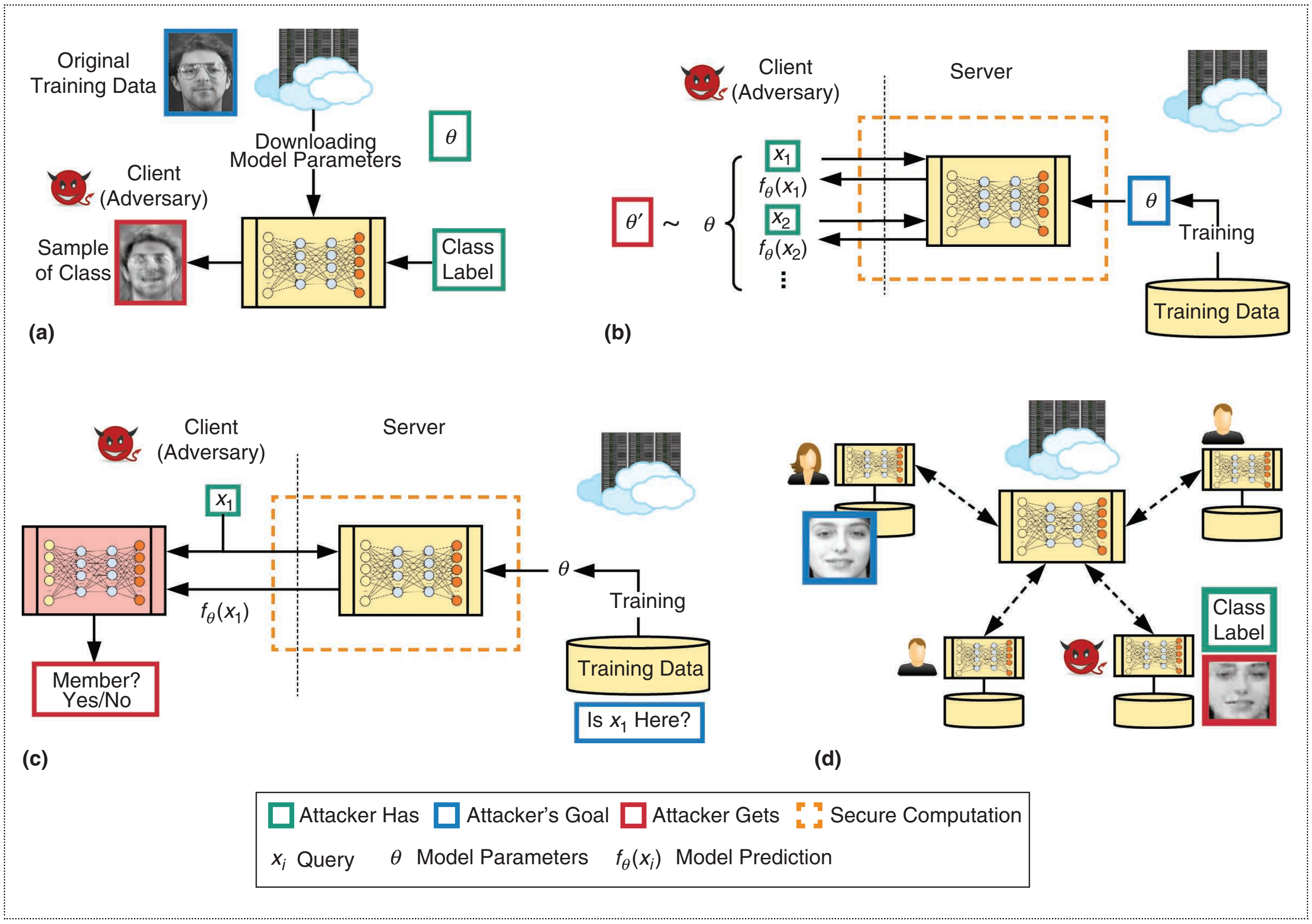

Figure 3. An overview of attacks on DL in terms of scenarios, adversary abilities, and goals: (a) a model inversion attack; (b) a model extraction attack; (c) a membership inference attack; and (d) a GAN-based attack.

computation cost. Outsourcing is well suited for scenarios where the client is resource constrained, e.g., the Internet of Things, in which the edge nodes usually have limited computational power. The security model is based on the noncollusion assumption of the servers. FHE does not support the XOR-sharing technique and incurs a higher computational cost at the client side.

Attacks on DNNs

We now survey four of most important attacks on DL. The first three are designed for the inference phase where there is a server holding the trained model and a client querying the model. Depending on the attack, the adversary may have either black- or white-box access to the model. In the black-box scenario, an adversarial client can only query the server and receive the prediction result. The prediction can consist of the label with or without the confidence values for each class. In the white-box scenario, the attacker is able to download the model and acquire the complete network parameters. The fourth attack is designed for the training phase where multiple parties want to jointly train a model on their private inputs. Figure 3 provides an overview of all four attacks.

Model Inversion

The model inversion attack[11] extracts information about the training data used for learning the model. For a given class, the attacker, with either black-box or white-box access to the model, attempts to create an input that maximizes the confidence score of that class using the gradient descent algorithm. Figure 3 details the attack for the white-box access scenario. The generated input is a prototypical sample of that class. For example, in the case of a facial recognition system, an attacker can produce an approximate image of one of the people whose image was used in the training phase by only having her name. The model inversion attack does not depend on how the network was trained and reveals information about the average features of a given class. To mitigate this attack in the black-box access scenario, it is suggested to round the confidence score before the prediction results are sent back to the client.

Model Extraction

The model extraction attack[12] attempts to learn an approximation of a server model by querying the model many times. In the client-server computation model, a malicious client creates its own version of the model, with the incentive of undermining the pay-per-prediction fees (stealing the model). Model extraction can also be used as a step toward model inversion: the attacker first learns an approximation of the model and then attempts to reconstruct the training data. However, similar to model inversion, this attack can be prevented by not providing the confidence vector and reporting only the final class label that has the highest confidence value.

Membership Inference

In the membership inference attack, [13] the goal is to identify whether a specific record has been used in the training phase or not. There are certain privacy concerns that arise from this attack. For example, a membership inference attack on a classifier that suggests different medications for a given disease can reveal that a person whose data was used in the training phase actually has that disease. The intuition behind the attack is that models can perform better on the data records used in the training phase compared with the data they have never seen. In this case, the model is overfitting on the training data. Such a difference in the behavior of the model enables a membership inference attack. There are two main mitigation strategies: 1) during the training phase, regularization techniques should be used to avoid overfitting and 2) during the inference phase, the complete confidence vector should not be revealed to the client.

The Hitaj et al. Attack

Hitaj et al.[14] crafted an attack based on generative adversarial networks (GANs) to infer sensitive information in a collaborative DL. More specifically, they showed that an adversarial client can learn the training samples held by other clients even in the privacy-preserving collaborative DL setting suggested by Shokri and Shmatikov.[2] GANs generate samples that look similar to the training data without having access to the training data themselves. A GAN produces samples by interacting only with a discriminative DNN. The goal of a GAN is to deceive the discriminative model into believing that the produced sample is real and that the goal of the discriminative model is to detect the synthetic samples. The process continues until the discriminative model cannot perform better than a random guess. In general, Hitaj et al. showed that applying record-level DP in collaborative DL is not effective. It is illustrated that more effective solutions based on secure multiparty computation and HE are needed to guarantee data owners' privacy.

Note that the goal of private inference frameworks is to preserve the privacy of inputs from both the client and the server. In model inversion attacks with a white-box access, the network parameters are downloaded, and, as a result, the attack is out of the scope of private inference frameworks. In the black-box setting, the attack can be mitigated by adding a layer inside the secure computation protocol to round the confidence scores before sending the result back to the client. Similarly, model extraction and membership inference attacks can be effective if the prediction confidence vector is sent to the client. Private inference frameworks that support nonlinear functionalists can output only the label of the class that has the highest confidence score. In this case, the confidence vector is not revealed to the client, and the attack will become infeasible.[11]

Privacy is one of the biggest concerns when learning (and using) the NN models on users' sensitive information. This article provides a systematic view and comparison of the latest efficient privacy-preserving solutions for DNN learning and inference. The methodologies rely on various building blocks, such as GC, HE, SS, and DP. We have also outlined various attacks on DL. Several of these attacks rely on the confidence vector provided by the cloud server. Hence, if only the output label is provided to the client, such attacks can be prevented. Such techniques as 1) devising mixed-protocol solutions,[7,9,15] 2) preprocessing the network and data, [8,16] and 3) using correlated randomness[9] can significantly reduce the overhead of secure computation protocols.

During the past few years, there has been more focus on private inference than private learning. To bridge the wide gap between machine learning on plaintext and encrypted data, more research for finding efficient customized privacy-preserving learning techniques is needed. ■

Acknowledgments

We thank Ari Juels and the referees for their comments on earlier versions of this article.

References

1. G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *Int. J. Security Netw.*, vol. 10, no. 3, pp. 137–150, 2015.

2. R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Computer and Communications Security*, 2015, pp. 1310–1321.
3. P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Security and Privacy*, 2017, pp. 19–38.
4. K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in Proc. ACM SIGSAC Conf. Computer and Communications Security, 2017, pp. 1175–1191
5. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Machine Learning*, 2016, pp. 201–210.
6. J.W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *Proc. IMA Int. Conf. Cryptography and Coding*. Berlin: Springer-Verlag, 2013, pp. 45–64.
7. J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MiniONN transformations," in *Proc. ACM SIGSAC Conf. Computer and Communications Security*, 2017, pp. 619–631.
8. B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: Scalable provably-secure deep learning," in *2018 55th ACM/ESDA/IEEE Design Automation Conf. (DAC)*, p. 2.
9. M. S Riazi et al., "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. 2018 Asia Conf. Computer and Communications Security*, 2018, pp. 707–721.
10. R. Bahmani et al., "Secure multiparty computation from SGX," *in Proc. Int. Conf. Financial Cryptography and Data Security,* Cham, Switzerland: Springer, 2017, pp. 477–497.
11. M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf.* Computer and Communications Security, 2015, pp. 1322–1333.
12. F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 601–618.
13. R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp.* Security *and Privacy*, 2017, pp. 3–18.
14. B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Computer and Communications Security*, 2017, pp. 603–618.
15. C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. "GAZELLE: A low latency framework for secure neural network inference," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 1651–1669.
16. Riazi, M. Sadegh, Mohammad Samragh, Hao Chen, Kim Laine, Kristin E. Lauter, and Farinaz Koushanfar. "XONN: XNOR-based Oblivious Deep Neural Network Inference," in *Proc. 28th USENIX Security Symp.,* 2019, pp. 1501–1518.
17. S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-Party secure computation for neural network training," *Proc. Privacy Enhancing Technologies*, vol. 1, no. 3, pp. 26–49, 2019.
18. P. Mohassel and P. Rindal, "ABY^3: A mixed protocol framework for machine learning," in *Proc. 2018 ACM SIGSAC Conf. Computer and Communications Security,* pp. 35–52.
19. V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *Proc. Int. Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science series, Berlin: Springer, 2008, pp. 486–498.

M. Sadegh Riazi is a Ph.D. candidate in the Department of Electrical and Computer Engineering at the University of California San Diego. His research interests include secure multiparty computation, privacy-preserving deep learning, large-scale machine learning, and high-performance computing platforms for secure computation. Riazi received a master's degree from Rice University, Houston, Texas, in 2016. Contact him at mriazi@ucsd.edu.

Bita Darvish Rouhani is a research scientist at Microsoft. Her research interests include deep learning, the safety of machine-learning models, and big data analysis. Rouhani received a Ph.D. in electrical and computer engineering from the University of California San Diego. Contact her at bita@ucsd.edu.

Farinaz Koushanfar is a professor and Henry Booker Faculty Scholar in the Department of Electrical and Computer Engineering at the University of California San Diego, where she directs the Adaptive Computing and Embedded Systems Lab. She is the cofounder and codirector of the University of California San Diego Center for Machine-Integrated Computing and Security. Koushanfar received a Ph.D. from the University of California, Berkeley. She is a fellow of the Kavli Foundation Frontiers of the National Academy of Engineering. She has received a number of awards and honors for her research, mentorship, teaching, and outreach activities, including the Presidential Early Career Award for Scientists and Engineers from U.S. President Obama, ACM SIGDA Outstanding New Faculty Award, Cisco IoT Security Grand Challenge Award, and MIT Technology Review TR-35 2008 (World's Top 35 Innovators Under 35), as well as Young Faculty/CAREER Awards from National Science Foundation, DARPA, U.S. Office of Naval Research, and U.S. Army Research Office. Contact her at farinaz@ucsd.edu.