electronics

MDPI

Article

PFDLIS: Privacy-Preserving and Fair Deep Learning Inference Service under Publicly Verifiable Covert Security Setting

Fengyi Tang *, Jialu Hao, Jian Liu, Huimei Wang and Ming Xian

College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China; haojialu.nudt@gmail.com (J.H.); ljabc730@gmail.com (J.L.); freshcdwhm@163.com (H.W.); qwertmingx@sina.com (M.X.)

* Correspondence: tangfengyi10@nudt.edu.cn; Tel.: +86-13298651514

Received: 18 November 2019; Accepted: 1 December 2019; Published: 6 December 2019

Abstract: The recent popularity and widespread use of deep learning heralds an era of artificial intelligence. Thanks to the emergence of a deep learning inference service, non-professional clients can enjoy the improvements and profits brought by artificial intelligence as well. However, the input data of the client may be sensitive so that the client does not want to send its input data to the server. Similarly, the pre-trained model of the server is valuable and the server is unwilling to make the model parameters public. Therefore, we propose a privacy-preserving and fair scheme for a deep learning inference service based on secure three-party computation and making commitments under the publicly verifiable covert security setting. We demonstrate that our scheme has the following desirable security properties—input data privacy, model privacy and defamation freeness. Finally, we conduct extensive experiments to evaluate the performance of our scheme on MNIST dataset. The experimental results verify that our scheme can achieve the same prediction accuracy as the pre-trained model with acceptable extra computational cost.

Keywords: artificial intelligence; commitment; data privacy; data security; secure three-party computation

1. Introduction

In recent years, artificial intelligence (AI) has been applied to more and more fields, such as agriculture [1], industrial control [2], smart home [3] and, most popularly, medical treatment [4–7]. As one of the most representative technologies of AI, deep learning has been widely studied in terms of model accuracy improvement [8,9] and computational cost cutting [10,11]. The accuracy and cost are key to the availability of deep learning but the security might be the foundation of a wide acceptance of deep learning. However, research on ensuring the security of deep learning are not enough or satisfactory. As is stated in Reference [12], deep learning mainly faces challenges in generalizing unseen examples. Furthermore, the data hungriness can be seen as an extension of this problem. In fact, data are generated anytime and anywhere. It is the improper approaches to data collection and data utilization that cause data hungriness in deep learning. To be specific, designers of existing deep learning schemes rarely consider the plight of data contributors or data owners. They just stand at the perspectives of the users of deep models (model accuracy improvement) or the deep network trainers (computational cost cutting). From the data owner's point of view, it is not secure to give out personal data to a server (network trainer) without any commitment. Therefore it is reasonable that the data owners are unwilling to contribute their data to deep learning even when they intend to make inference on their data, which is the source of data hungriness. In other words, the data hungriness in deep learning reflects the lack of secure and satisfactory deep learning scheme for data owners.

The emerging Deep Learning Inference Service (DLIS) [13], where clients can enjoy a deep learning inference service on a pay-per-use basis provided by servers who pre-train deep learning models, requires attention to the privacy of the sensitive input data of clients as well as the models of servers. Not all types of data on which an inference is to be made are sensitive and must be private but in some scenarios, such as medicine and finance, the privacy of clients' data should be protected [14]. On the other hand, model parameters of the models pre-trained by servers should be secure as well because training models costs servers time and money (for collecting training data or buying infrastructure).

However, existing privacy-preserving deep learning schemes do not satisfy the security requirements of DLIS very well. These schemes are mainly based on two techniques: homomorphic encryption (HE) and secure multi-party computation (SMPC). As for HE-based schemes, computational complexity is a major challenge when inference is made on encrypted data. Another challenging aspect is that servers cannot inspect the data flowing in the trained models to check and find mislabelled items. Thus HE-based schemes do not seem to be feasible in DLIS. With regard to SMPC-based schemes, the efficiency declines dramatically when the number of parties grows to three or above.

Our scheme tackles this problem in the context of a deep learning inference service wherein a server has a convolutional neural network (CNN) trained on its private data and a client wishes to get classification results on its private images. Furthermore, there are three workers that help finish the model inference based on secure three-party computation.

1.1. Our Contribution

We design and implement a general scheme for privacy-preserving and fair deep learning inference service in the three-worker model under the setting of publicly verifiable covert security. Our contribution is that our scheme enjoys the following properties of security, accuracy and efficiency.

- **Security:** *Our scheme protects the private input data of the client as well as the pre-trained model of the server. Also, the workers do not need to worry about being framed. Moreover, we do not need to assume that the server would not collude with any one of the workers or the client would not collude with any one of the workers. Cheating behaviours will be caught and be publicly verifiable so that our scheme can act as a deterrent to misbehaviours.*
- **Accuracy:** *Our scheme provides identical inference accuracy to the pre-trained deep learning model. That is to say, after secure sharing the pre-trained model and implementing inference based on secure three-party computation in our scheme, the model accuracy does not decrease.*
- **Efficiency:** *Since we just utilize simple and technically mature hash function and RSA signature, the extra computational cost of our scheme is acceptable compared to the existing SMPC-based scheme.*

1.2. Organization of This Paper

The rest of the paper is organized as follows. We introduce the recent works related to privacy-preserving deep learning in Section 2. The preliminaries and problem definition are given in Section 3. Our innovative privacy-preserving and fair deep learning inference scheme is proposed in Section 4. The security analysis of our scheme is given in Section 5. We analyze the performance of our scheme from theoretical and experimental respects respectively in Section 6. Lastly, we conclude the paper in Section 7.

2. Related Work

In this section, we review recent representative research related to privacy-preserving deep learning. There are two main aspects: model training and inference.

2.1. Model Training

In order to protect the privacy of data owner in model training process, a line of work uses HE and another line of work takes advantage of SMPC.

2.1.1. HE-Based Methods

Le Trieu Phong et al. [15] build a privacy-preserving deep learning system using HE for neural network weights encryption in which multiple learning participants perform neural network training cooperatively, without actually revealing their local data to a central server. However, the system is based on the assumption that the server and any learning participant do not collude. Once they collude, the server could decrypt and get data from all learning participants [16]. So there are potential risks for the privacy of data owners.

Ehsan Hesamifard et al. [17] adopt convolutional neural networks (CNNs) within the practical limitation of current homomorphic encryption schemes. More specifically, they first design methods to approximate the activation functions commonly used in CNNs such as ReLU, Sigmoid and Tanh, via low degree polynomials, which replace original activation functions in CNNs training. Then CNNs could be implemented over encrypted data using HE. However, using HE for model training which relies on large amount of training data, computational complexity is a major challenge.

The above works leave much room for improvement in security or efficiency.

2.1.2. SMPC-Based Methods

A privacy-preserving neural networks (NNs) training system is proposed and implemented in Reference [18]. The authors consider the setting that there are two non-colluding servers and multiple distributed data owners. Using the secure two-party computation (2PC) technique, they develop techniques to support secure arithmetic operations on shared decimal numbers and propose SMPC-friendly alternatives to non-linear functions (i.e., sigmoid and softmax).

Bita Darvish Rouhani et al. [19] design a framework called DeepSecure, which enables privacy-preserving execution of deep learning models. However, besides using Yao's Garbled Circuit (GC) protocol, a set of pre-processing techniques are needed for further reducing the GC runtime in this solution.

Payman Mohassel and Peter Rindal [20] propose a mixed protocol framework for privacy-preserving machine learning (ML) model training which supports linear regression, logistic regression and neural network. The framework is based on secure three-party computation (3PC). It yields the state-of-the-art performance for privacy-preserving model training. However, the framework does not realize convolutional neural network training for ease of implementation. The convolutional kernel is replaced with a fully connected layer.

2.2. Inference

Applying learned neural networks to encrypted sensitive data is considered by Nathan Dowlin et al. [21]. By using homomorphic encryption, they present CryptoNets, which transfers trained neural network and supports inference on encrypted sensitive data, such as financial and medical data. The neural network is first trained in the cloud server, so that Machine Learning as a Service (MLaaS) can be provided. Then a data owner sends encrypted data to the cloud server to get corresponding encrypted prediction labels. However, further progress is needed for the CryptoNets, such as more efficient encoding schemes and faster homomorphic computation [22].

Fabian Boemer et al. [23] present nGraph-HE, Intel's deep learning graph compiler, which enables deploying trained models with popular frameworks (i.e., TensorFlow) by treating homomorphic encryption as another hardware target. It helps data scientists to benchmark deep learning models with less overhead. But extra hardware is needed.

The problem of privacy-preserving inference is explore by Jian Liu et al. [24] as well. They propose MiniONN, which transforms a trained neural network to an oblivious one to support privacy-preserving inference. However, oblivious transformations are needed in the whole execution of inference, which incurs considerable efficiency overheads and degrades the model accuracy inevitably.

Chiraag Juvekar et al. [25] introduce Gazelle, a system for secure neural network inference, which integrates homomorphic encryption and secure two-party computation. They design a homomorphic encryption library and homomorphic linear algebra kernels. Moreover, encryption switching protocols which seamlessly convert between homomorphic and garbled circuit encodings are presented to enable implementation of neural network inference. Such hybrid of homomorphic encryption and secure two-party computation delivers faster performance at the expense of much higher communicational costs.

2.3. Summary

We summarize the contributions of above related works and compare them with our work, which can be seen in Table 1.

Table 1. Related works comparison.

Works	Privacy Properties	Techniques	Shortages	Application Scenarios
Le [15]	Data privacy	HE	Collusion risk	NNs training
Ehsan [17]	Data privacy	HE	High computational overhead	CNNs training
Secureml [18]	Data privacy	2PC	Collusion risk	NNs training
DeepSecure [19]	Data privacy, model privacy	Yao's GC	Need much preprocessing	Deep learning
ABY3 [20]	Data privacy	3PC	Not support CNN	ML
CryptoNets [21]	Data privacy	HE	Need efficiency improvement	MLaaS
nGraph-HE [23]	Data privacy	HE	Need extra hardware	Benchmark DL models
MiniONN [24]	Data privacy	SMPC	Need oblivious transformations	NN Inference
Gazelle [25]	Data privacy	HE and SMPC	High communicational costs	NN Inference
Our scheme	Data privacy, model privacy, defamation freeness	3PC	Need expand to large scale CNNs and various networks	MLaaS

As listed in Table 1, we compare our scheme proposed in this paper with eight state-of-the-art literatures on privacy properties, techniques, shortages and application scenarios. It is shown that our scheme has the most privacy properties. Most of the related works only focus on protecting the data privacy of the data owners. But our scheme not only protects the data privacy of the data owners but also protects the model privacy of the server and the fame of honest workers.

3. Preliminaries and Problem Definition

We present the notations used in our description in Table A1 in Appendix A.

3.1. Preliminaries

3.1.1. Publicly Verifiable Covert (PVC) security

PVC security for secure two-party computation is proposed in Reference [26]. Under the setting of PVC security, the honest party could detect cheating behaviour with reasonable probability. Furthermore, if cheating behaviour is found, the honest party could generate a publicly verifiable certificate of that misbehaviour. Parties such as commercial cloud servers care about their reputation seriously and of course avoid being caught cheating. Therefore PVC security can have an obvious deterrent effect to cheating behaviours.

PVC security is deemed as a compromise between semi-honest and malicious security. In other words, PVC security provides more security guarantees than that of semi-honest security but less than that of malicious security. However, the overhead that it requires is less than that of malicious security but more than that of semi-honest security.

In this paper, we extend the PVC security for secure two-party computation to one for secure three-party computation, so that our scheme based on secure three-party computation is privacy-preserving and fair under the setting of PVC security.

3.1.2. Secure Multi-Party Computation

SMPC has been applied to privacy-preserving machine learning recently to relieve worries about privacy leakage. Generally SMPC, especially secure 3PC, is based on secret sharing [27], arithmetic sharing [28], binary sharing [29] and Yao sharing [30] protocols.

However, these protocols usually work over low-level circuits (either arithmetic or boolean). So for computationally expensive tasks, such as deep neural network training and prediction, SMPC-based protocols are so highly inefficient that these protocols can take forever to execute.

In these paper, we utilize a generic framework for privacy-preserving and federated deep learning called PySyft [31], which is based on SPDZ [32] and SMPC, to realize our scheme efficiently. The replicated secret sharing technique [29] is used in our scheme. There are two reasons why we choose three workers to hold different secret shares $(x_1, x_2), (x_2, x_3), (x_3, x_1)$ in our scheme. The first reason is about security. Any two out of the three workers can reconstruct the secret value x by addition: $x = x_1 + x_2 + x_3$. That is to say, our protocol based on secure 3PC can tolerate up to a single corruption, which cannot be achieved in 2PC-based protocols. The other reason we choose three workers is that 3PC is more efficient than 2PC according to existing studies [20].

3.1.3. Deep Learning Inference Service

In deep learning training service, the users need to not only provide their training data but also care about the model architecture, model initial weights and so on. It may be a difficult task for some non-professional users who just want to enjoy the inference service but not the customized service from model training to model inference. Different from deep learning training, deep learning inference only requires the users to input their data on which they want to infer.

In fact, inference is not possible to happen without training. Therefore nowadays, many famous technology companies provide DLIS, such as Microsoft Azure [33], Amazon Elastic Inference and NVIDIA. The reason for DLIS becoming popular is that DLIS is convenient, economic and user-friendly. More specifically, the cloud servers of the companies are responsible for computationally intensive training and the clients of DLIS enjoy the service on a pay-per-use basis.

Training a large and deep neural network model usually requires not only several terabytes of training data but also dozens of exaflops of compute, such as Baidu's Chinese speech recognition model [34]. After training is completed, the trained model is deployed to inference using what it has learned. In other words, clients of DLIS do not need all the storage and computing infrastructure of the entire training cycle. Therefore, DLIS is a speedier and easier way of benefiting from deep learning for general clients.

3.2. Problem Definition

The goal of our scheme is that both the cloud server that holds the model parameters and the client who owns the input data can protect their own information in DLIS, as well as obtaining correct inference results for the client. Besides, the scheme should be fair for workers so that they can be free of being framed.

Since SMPC for more than two parties is expensive for clients, it is not practical to perform SMPC among them [18]. Therefore we consider a worker-aided setting where the client outsources the computation to three untrusted workers. According to the definition of SMPC, the sensitive data leaks only if all the three workers are corrupted.

So in our scheme, in order to protect data privacy and model privacy, the client secretly shares its input data among the workers; and the trained model provided by the server is secretly shared among workers, too. Then the workers can compute the inference results cooperatively. Finally, the inference

results are sent to the client for output recovery. Moreover, to achieve fairness and correct results, all the parties in our scheme generate publicly verifiable commitments via hash and signature operations [35] for their computation.

Threat Model: We use the similar security model as that in Reference [26]. But we extend it to the three-party case. That is to say, there are three workers who are responsible for computation. They are supposed to be in the publicly verifiable covert security setting, which is a compromise between malicious security and semi-honest security. And we assume that a malicious adversary Ad could corrupt at most one of the three workers. Also, the client may be malicious and want to steal the model of server. Similarly, the server might be interested in the input data of the client. That is to say, we suppose that at most one worker, or the client, or the server would be corrupted by Ad.

Application Scenario: One application of our scheme is that a client (such as a patient) can get private inference results based on its sensitive data (such as the biomedical data of the patient collected by mobile telemonitoring system) without worrying about the privacy and accuracy in DLIS. Meanwhile, the workers responsible for inference computation are exempt from being framed. What's more, the model which belongs to the server is free from being stolen. That is to say, our scheme is quite fair for all parties.

We justify the proposed scheme is fair for the reason that our scheme is not partial to anyone and each of roles enjoys the rights they are entitled only if it obey the procedure of our scheme. For example, the honest client can enjoy deep learning inference service without the risk of data privacy leakage; the honest server can provide deep learning inference service without the risk of model privacy leakage; and the honest workers can execute model inference without the risk of being framed.

4. Our Scheme

In this section, we design a privacy-preserving and fair scheme for performing deep learning inference on private input data based on secure three-party computation under the setting of publicly verifiable covert security.

4.1. System Architecture

We consider a system that there are three workers (computation implementers), one client (input data provider as well as service consumer) and one server (model provider) involved in our scheme. The client C wants to get confidential inference results (such as prediction values or classification labels) on her own private data. And the server S first trains a deep learning model with its collected data and then provides DLIS. Three workers denoted as W_1, W_2, W_3 take responsibility for performing deep learning inference using a pre-trained and shared model. The system architecture is depicted in Figure 1.

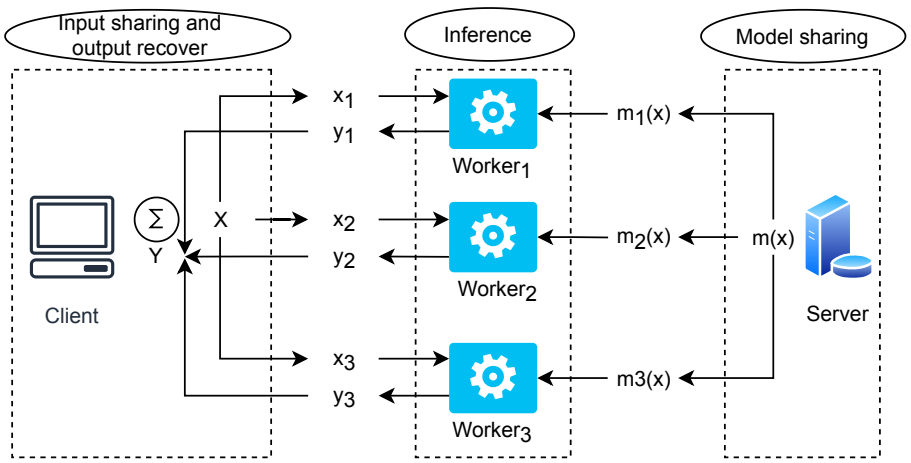

Figure 1. System architecture.

The relationship between the three roles of our scheme can be summary as follows: the client shares her private input data among three workers and the server shares her pre-trained model among three workers as well. Then the workers perform secure three-party computation to get inference results. Finally, the inference results are sent to the client to recover the output.

4.2. Scheme Description

In order to achieve publicly verifiable covert security in our "server–three workers–client" system, we design a privacy-preserving and fair scheme for realizing deep learning inference service against a malicious adversary who may corrupt a single worker, or the client or the server. Here, we present the main steps of our scheme as follows.

4.2.1. Input Data Preparation

The client C holds input data set $\langle X \rangle$ which include k test cases $\langle x \rangle$. And the labels of $\langle x \rangle$ are $\langle y \rangle$, which are only known to C at first. No one knows which part of input data are the test cases except C. Then C secretly shares the input data among the three workers evenly and randomly.

Meanwhile, C generates the commitment about the shared input data and makes the commitment public to all of the parties in the scheme. More specifically, if the number of data items that C shares among workers is m and the indexes of k test cases (denoted as I) as well as their corresponding labels (denoted as L) are $(I_1, L_1), (I_2, L_2), ..., (I_k, L_k)$, then the data commitment is computed as $share\_data\_com = Sign_C(hash(k||(I_1, L_1)||(I_2, L_2)||...||(I_k, L_k)||m))$, where $Sign_C(input)$ means computing client's signature of the input, $hash(input)$ denotes calculating the hash value of the input and $||$ indicates the operation of concatenate strings.

4.2.2. Model preparation

The server S first trains a model and determines the model accuracy threshold λ. Specifically, the model could be trained on the dataset which is bought or collected by the server or donated by some organizations without worrying about data privacy. Considering the randomness of test results, in order to decrease the possibility of getting one wrong, here we set the minimal test accuracy as the model accuracy threshold λ according to multiple experimental results after model training.

Then S secretly shares the model among the three workers. For the sake of fairness, S provides the commitment about the shared model for public verification as well. The commitment of the model is computed as $share\_model\_com = Sign_S(hash(model\_params))$, where $model\_params$ denotes the shared parameter values of the pre-trained model.

4.2.3. Inference Based on 3PC

Three workers perform the deep leaning inference cooperatively using the shared input data and the shared model based on secure three-party computation (3PC). Finally, each worker W sends its computation result to the client C to recover the output. Concretely, W_i obtains result $\langle Y \rangle_i$ and sends it to C $(i = 1, 2, 3)$. Simultaneously, in order to supervise and urge the workers to honestly compute and comply with the scheme, each worker must generates commitment for every result it obtains. For example, if worker W_i get the shared input data $\langle X \rangle_i$ at the beginning and obtains result $\langle Y \rangle_i$ after computation, then it makes a public commitment

$$result\_com = Sign_{W_i}(hash(i, \langle X \rangle_i, \langle Y \rangle_i)). \tag{1}$$

And the intermediate results should be recorded, too. That is to say, before sending the intermediate result y_{in} which W_i computes to another worker for multi-party computation, W_i makes a commitment for y_{in}:

$$inter\_result\_com = Sign_{W_i}(hash(count, y_{in})), \tag{2}$$

where *count* is the order of the intermediate result. Finally, all the commitments are public after multi-party computation.

4.2.4. Output Recovery and Check

After receiving inference results and corresponding commitments from three workers, the client C recovers final output via addition:

$$\langle Y \rangle = \sum_{i=1}^{3} (\langle Y \rangle_i). \tag{3}$$

When C obtains final output, she can check the labels of the test cases and calculate accuracy of the inference on test cases. For instance, if n test data items are judged correctly in the total k test cases, then the inference accuracy is $\frac{n}{k}$. If the accuracy is lower than the model accuracy threshold λ, we deem the inference results questionable, which means one of the workers is cheating (here we suppose only one worker might cheat). Otherwise, the inference output based on 3PC is achieved now.

4.2.5. Cheat Forensics and Punishment

In case cheat is found in the previous step, the client C needs to obtain evidence of cheating. Most importantly, which worker cheats should be found out and publicly verified. Here we make full use of the commitments made in above steps. The method is to check the workers one by one. Take checking W_i for example. Let the server perform the same execution as W_i from the beginning. In other words, the shared input data and model which were assigned to W_i before are sent to the server now, who performs deep learning inference with other two workers cooperatively based on 3PC. The key is that the server should generate and disclose the commitments for the final result and intermediate results:

$$result\_com = Sign_S(hash(i, \langle X \rangle_i, \langle Y \rangle_i)) \tag{4}$$

$$inter\_result\_com = Sign_S(hash(count, y_{in})) \tag{5}$$

After decrypting the commitments in Equations (1), (2), (4) and (5) with public keys of W_i and the server S respectively, two $hash(i, \langle X \rangle_i, \langle Y \rangle_i)$ and $hash(count, y_{in})$ can be obtained and then compared by everyone. If they are not the same, then W_i is verified cheating. Otherwise, check another worker.

What's more, the cheating one should be punished so that our scheme have a deterrent effect. Then the workers would be afraid to cheat. The punishment can be adding the cheating one to a blacklist and announce its cheating behaviour to the public.

5. Security Analysis

The security properties of our scheme can be summarized as follows: input data privacy, model privacy and defamation freeness.

First, we summarize the different attack scenarios considered in Figure 2.

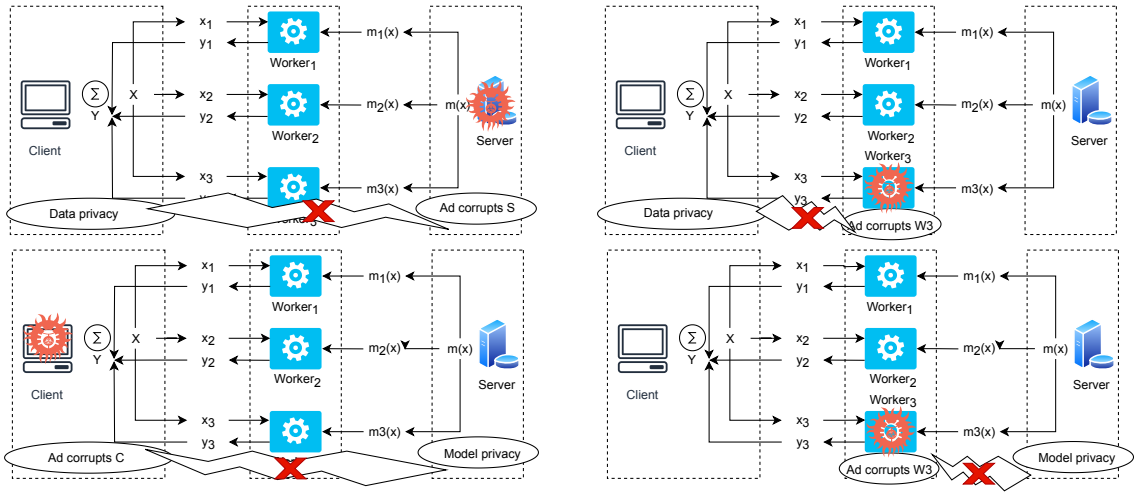

Figure 2. Attack scenarios.

5.1. Input Data Privacy

The client's sensitive input data and corresponding inference outputs are only revealed to herself, even in the presence of a malicious adversary.

Proof Sketch. A malicious adversary *Ad*, who wants to get the private input data of the client *C*, can corrupt the server *S* or any one of the workers in our system. Given that our scheme is symmetric with respect to the three workers, we just need to consider the scenario where *Ad* corrupts *S* or worker W_3. A simulator *Sim* is set to simulate *Ad* in the ideal world.

5.1.1. *Ad* Corrupts Server *S*

First, we consider the scenario that *Ad* corrupts the server *S*. *Sim* sends the corrupted server's model and accuracy threshold to *Ad*. Then *Sim* runs *Ad*. On behalf of *S*, *Sim* shares the model among three workers randomly and generates commitment of the model with the private key of *S*. If the model is wrong or the accuracy threshold is too high, the commitment will be an evidence of the misbehaviour of the server. So *Sim* would like to share the model honestly. Because we suppose all parties except for the server are honest in this scenario, there is no need to perform workers cheating forensics. Therefore this is the only execution where *S* is involved. It is obvious that *Ad*'*s* views in the ideal and real worlds are indistinguishable. And *Ad* cannot obtain any private information of the input data.

5.1.2. *Ad* Corrupts Worker W_3

Second, let us consider the other scenario that *Ad* corrupts worker W_3. *Sim* sends the shared input data and shared model which the corrupted worker received to *Ad*. Then *Sim* runs W_3. On behalf of W_3, *Sim* performs three-party computation in interactions with other two honest workers. After computation, *Sim* sends intermediate and final inference results to *Ad* and generates public commitments with the private key of W_3. Finally, *Sim* sends final inference result to the client *C*, as well as opens its commitments.

We argue that the views of *Ad* in both ideal and real worlds are indistinguishable. And the input data and corresponding inference results cannot be retrieved from the shared information *Ad* has. This is because the security of the arithmetic secret sharing and secure multi-party computation. If W_3 does not obey the steps of our scheme and cheats, the commitments will be the evidence of cheating and W_3 will be blacklisted. So the worker would be afraid to cheat and behave honestly.

To sum up, the input data privacy and output privacy of the client can be achieved in our scheme even in the presence of a malicious adversary that could corrupts the server or one of the workers.

5.2. Model Privacy

The model which is pre-trained by the server remains private to the server in our scheme, without the risk of being stolen by a malicious adversary who could corrupt the client or one of the workers.

Proof Sketch. A malicious adversary Ad, who wants to get the private model parameters of the server S, can corrupt the client C or any one of the workers in our model. Since our scheme is symmetric with respect to the three workers, we only consider the scenario where Ad corrupts C or worker W_3. A simulator Sim is set to simulate Ad in the ideal world.

5.2.1. Ad Corrupts Client C

First, we consider the scenario that Ad corrupts the client C. Sim sends the corrupted client's input data to Ad. Then Sim runs Ad. On behalf of C, Sim shares the input data among three workers randomly and generates commitment of the shared data with the private key of C. After the inference computation, Sim receives the final results from three workers and recovers the output. Finally, Sim sends the output to Ad. It is obvious that $Ad's$ views in the ideal and real worlds are indistinguishable. And Ad cannot obtain any private information of the model.

5.2.2. Ad Corrupts Worker W_3

Second, let us consider the other scenario that Ad corrupts worker W_3. Sim sends the shared input data and shared model which the corrupted worker received to Ad. Then Sim runs W_3. On behalf of W_3, Sim performs three-party computation in interactions with other two honest workers. After computation, Sim sends intermediate and final inference results to Ad and generates public commitments with the private key of W_3. Finally, Sim sends final inference result to the client C, as well as opens its commitments.

We argue that the views of Ad in both ideal and real worlds are indistinguishable. And the model parameters cannot be retrieved from the shared information Ad has. This is because the security of the arithmetic secret sharing and secure multi-party computation. If W_3 does not obey the steps of our scheme and cheat, the commitments will be the evidence of cheating and W_3 will be blacklisted. So the worker would be afraid to cheat and behave honestly.

To sum up, the model privacy of the server can be achieved in our scheme even in the presence of a malicious adversary that could corrupts the client or one of the workers.

5.3. Defamation Freeness

Our scheme introduce making commitments to realize the function of anti-being framed and non-repudiation. Each worker will first verify the correctness of the public commitment of the received model parameters shared by the server. If the verification of a commitment fails, the protocol aborts. And the server is deemed to frame the workers (or one worker). Hence the server will be likely to share her model honestly.

If the client wants to frame an honest worker, it will fail when she performs the cheat forensics. Because the commitments of the intermediate results and final results computed by the honest worker can pass the verification performed by the server and other two workers. And after checking three worker, the client may be deemed to frame the workers (or one worker) if all the verifications succeed. Thus the client will obey to the scheme and share her input data honestly.

Besides, the verification is simple. For instance, if worker W_3 receives shared model parameters $model\_params$ and its corresponding commitment $share\_model\_com = Sign_S(hash(model\_params))$ from S, then she decrypts $share\_model\_com$ using the public key of S and compares the result with $hash(model\_params)$ which she computes herself. If they are equal, the verification succeeds. Otherwise, it fails and W_3 aborts the protocol and reports that S cheats (here we suppose there is not transmission error). Similarly, the workers and server can verify the correctness of commitments in cheat forensics.

As long as the workers compute honestly, they will get the correct results based on the correct input data and model. Therefore, they will pass the checking of test cases. To sum up, our scheme protects the workers from malicious defamation.

In conclusion, our scheme is privacy-preserving and fair, which not only preserves the privacy of both the client and the server but also prevents them framing the workers up.

6. Performance Evaluations

The performance of our privacy-preserving and fair scheme for deep learning prediction service under publicly verifiable covert security setting is evaluated in this section.

Evaluation metric: There are three main metrics in our evaluation. First, the number of desirable security properties should be more than that of the existing schemes. Second, model prediction accuracy should not be decreased apparently after sharing the model based on SMPC. Last but not least, the extra computational cost of our scheme should be acceptable.

6.1. Theoretical Evaluations

Under publicly verifiable covert security setting, here we make a comparison of security properties between our scheme (here denoted as *Scheme 1*) and one recent SMPC-based scheme [31] (here denoted as *Scheme 2*).

As described in Section 5, our scheme not only preserve input data privacy of the client but also guard model privacy of the server. Additionally, our scheme is fair because it protects workers from malicious defamation. The comparison of desirable security properties between *Scheme 1* and *Scheme 2* [31] is listed in Table 2.

Table 2. Security properties comparison.

Security Property	Data Privacy	Model Privacy	Defamation Freeness
Scheme 1	Yes	Yes	Yes
Scheme 2	Yes	Yes	No

When it comes to extra computational cost of our scheme, we combine technically mature hash function and RSA signature to make commitments. Therefore the introduced extra computational cost is quite low compared to *Scheme 2* [31] without making commitments. The detailed analysis of extra computational cost of our scheme can be seen in Table 3.

Table 3. Theoretical analysis of extra computational cost.

Participant	Server	Client	Worker 1	Worker 2	Worker 3
Hash Function	3	1	m+2	m+2	m+2
RSA Signature	3	1	m+2	m+2	m+2

As described in Section 4, the server in our scheme needs to compute three commitments of three shares of model parameters. And computing one commitment means calculating one time hash function and RSA signature. So the server needs to calculate three times hash function and RSA signature. Similarly, the client just needs to calculate one time hash function and RSA signature for her shared input data. As for the workers, each worker needs to calculate a commitment for every intermediate result and the final result, so it is $m + 1$ times, where m is the number of shared input data. Furthermore, every worker needs to verify the commitment of the shared model parameters she received. And verifying the commitment needs decrypt the commitment first, which is similar to signing in RSA signature. Then hash function needs to be calculated, too. Therefore it is $m + 2$ times hash function and RSA signature calculation totally for each worker.

Here we do not consider the computational cost of commitments verification in the cheat forensics procedure because it depends on the behaviour of scheme participants (and we suppose they all behave honestly due to the deterrent of our scheme). For example, if one of the scheme participants cheats, the verification computation will increase. So we just give an approximate evaluation of times that each scheme participant need to calculate hash function and RSA signature at least here. In the next part, we implement experiments to run time statistics of our scheme to evaluate the computational cost.

6.2. Empirical Evaluations

Here we describe the experimental settings and results.

6.2.1. Experimental Settings

Our scheme is implemented on a computer with Intel Core i7-8700K CPU 3.70GHz×12 and 62.8GB RAM running on Ubuntu 18.04 LTS, python 3.7. All the five virtual machines representing three workers, one client and one server are in the same physical machine in our implementation. We have not specified or measured the communication bandwidth. But the communication time have been included in running time in our experimental results.

Implementation: The implementation of our scheme is based on the python libraries PySyft and PyTorch to realize the secure three-party computation and deep learning. As for hash function (we use SHA-256 here) and RSA signature computation, we utilize another python library PyCrypto. SHA-256 is one kind of hash function usually used for generating certificates to make sure network security. Moreover, the length of output of SHA-256 is always 256 bits, which is convenient for storage and comparison in our scheme.

In PySyft, using a list of PointerTensors properly can split and send the shares. Actually, we need not care much about the basic operations of MPC but just need focus on the implementation of steps of our protocol. It is the advantage that the python library brings us, which avoids repetitive labour. The major difference between PySyft and MiniONN is that MiniONN needs transform an existing neural network to an oblivious neural network. However, PySyft leverages the PointerTensors to split and share data and model.

We mainly compare the computational cost of *Scheme 1* with that of *Scheme 2*. Furthermore, we explore the influence of pre-trained model accuracy, the ratio of test cases to the data items in client input dataset and the key length of RSA signature used in our scheme.

Dataset: Our experiments are conducted on the popular MNIST dataset, which consists of grayscale images of hand written digits ($0 \sim 9$). It contains 60,000 data items for training and 10,000 data items for testing. In our experiment, the training dataset is used for pre-training the model before the server provides deep model prediction service. And the data items in test dataset are used as the input data of clients in our experiment. Furthermore, a part of data items in test dataset are used as test cases of our scheme. In our experiment, in order to explore effects of the number of test cases, we respectively take 100 (1%) and 200 (2%) test cases from the 10,000 data items in MNIST test dataset randomly and evenly.

Network architecture: The network architecture used in our experiment is listed in Table 4.

Table 4. Network architecture.

Layer	Number of neurons	Activation
Conv2D(10,(3, 3),batch_input_shape=(1, 28, 28, 1))	$10 \times 3 \times 3 = 90$	-
AveragePooling2D((2, 2))	-	relu
Conv2D(32, (3, 3))	$32 \times 3 \times 3 = 288$	-
AveragePooling2D((2, 2))	-	relu
Conv2D(64, (3, 3))	$64 \times 3 \times 3 = 576$	-
AveragePooling2D((2, 2))	-	relu
Flatten	-	-
Dense	10	logit

6.2.2. Experimental Results

Here we display the results of our experiments. We do not count the pre-training time because we consider the scenario that the server S first trains a deep learning model with its collected data and then provides DLIS. When counting the running times, we separate each scheme into two phases: model sharing and secure prediction (including input data sharing, inference and output recovery). Specifically, for easy presentation, the two phases in *Scheme 1* are denoted as Scheme 1-1 and Scheme 1-2 respectively, which is similar in *Scheme 2*. And the results are the average time costs of providing prediction service for 100 input data items. Besides, the unit of time in our paper is second (s).

Influence of pre-trained model accuracy *ma***:** First, we show the time costs of different groups which are grouped by model and scheme. In order to explore the influence of the accuracy of pre-trained model, we pre-train two deep CNN models with different model accuracy, denoted as *model a* ($ma = 98\%$) and *model b* ($ma = 99\%$) respectively. But the network architectures (such as activation function, layer number and node number, etc.) of these two models are the same, which are shown in Table 4. We just change the training epochs to get different model accuracy. We take 100 test cases and set key length of RSA signature as 2048 here. The average time costs for providing 100 input data items inference of different models are listed in Table 5.

Table 5. Average time costs of different models.

Time (s)	Scheme 1-1	Scheme 1-2	Scheme 2-1	Scheme 2-2	Accuracy
model a	2.274	2.851	2.229	1.885	98%
model b	2.508	3.425	2.151	1.808	99%

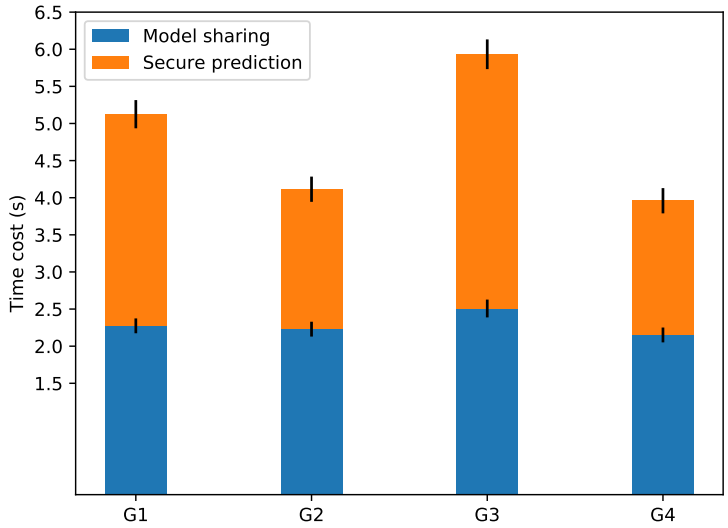

Figure 3. Average time costs of groups grouped by model and scheme.

As depicted in Figure 3, there are four groups ($G1 \sim G4$) representing four different combinations of model and scheme. *G1* represents the average time cost of providing *model a* prediction service for 100 input data items in *Scheme 1*. That is to say, in *Scheme 1*, sharing the *model a* with 98% accuracy among three workers costs about 2.274 s (the blue bar in *G1*) and performing secure prediction costs average 2.85 s per 100 input images (the orange bar in *G1*). And the black vertical line shows the error range. Furthermore, the whole bar shows the total time cost of the complete privacy-preserving and fair prediction service. Similarly, $G2 \sim G4$ display time costs of providing *model a* prediction service in *Scheme 2*, providing *model b* prediction service in *Scheme 1* and providing *model b* prediction service in *Scheme 2* respectively.

When we compare *Scheme 1* with *Scheme 2* in serving *model a* with 98% accuracy, the extra computational cost is approximate 0.045 s in model sharing procedure and 0.966 s in secure prediction process. And when it comes to *model b* with 99% accuracy, the extra computational cost becomes 0.356 s in model sharing procedure and 1.617 s in secure prediction process. The above experimental results demonstrate that the extra computational cost introduced by our scheme is acceptable. Moreover, the higher model accuracy is, the higher extra computational cost is.

Influence of ratio of test cases to the data items in the input dataset $\frac{k}{m}$: Then we show the computational costs of difference groups which are grouped by scheme and number of test cases. In order to explore the influence of the number of test cases, we respectively take 100 ($\frac{k}{m} = 1\%$) and 200 ($\frac{k}{m} = 2\%$) data items from the 10,000 data items in MNIST test dataset randomly and evenly as test cases. And we use the pre-trained *model a* with 98% accuracy and set key length of RSA signature as 2048 here. The average time costs of different number of test cases are listed in Table 6.

Table 6. Average time cost of different number of test cases.

Time (s)	Scheme 1-1	Scheme 1-2	Scheme 2-1	Scheme 2-2	Accuracy
k = 100	2.274	2.851	2.229	1.885	98%
k = 200	2.992	4.222	2.180	3.301	98.5%

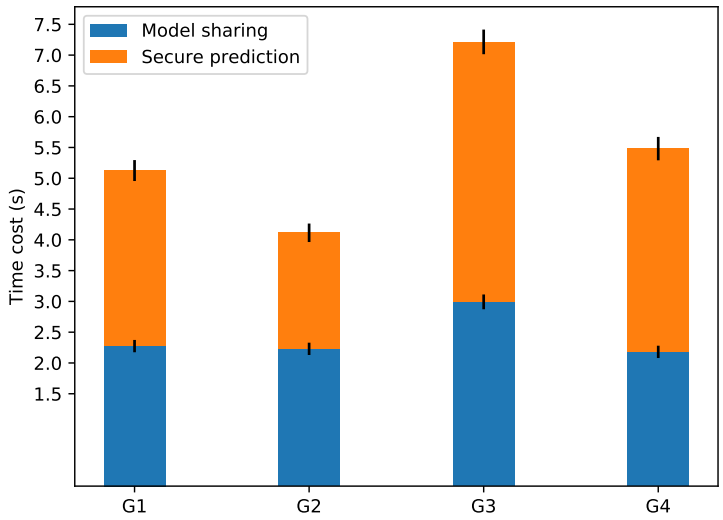

Figure 4. Average time costs of groups grouped by scheme and number of test cases.

As depicted in Figure 4, there are four groups ($G1 \sim G4$) representing four different combinations of scheme and number of test cases. *G1* represents the time cost of providing 100 ($\frac{k}{m} = 1\%$) test cases prediction service in *Scheme 1*. That is to say, in *Scheme 1*, sharing the *model a* among three workers costs about 2.274 s (the blue bar in *G1*) and performing secure prediction costs average 2.85 s per 100 input images (the orange bar in *G1*). And the black vertical line shows the error range. Furthermore, the whole bar shows the total time cost of the complete privacy-preserving and fair prediction service. Similarly, $G2 \sim G4$ display time costs of providing 100 ($\frac{k}{m} = 1\%$) test cases prediction service in

Scheme 2, providing 200 ($\frac{k}{m} = 2\%$) test cases prediction service in *Scheme 1* and providing 200 ($\frac{k}{m} = 2\%$) test cases prediction service in *Scheme 2* respectively.

When we compare *Scheme 1* with *Scheme 2* in providing 100 ($\frac{k}{m} = 1\%$) test cases prediction service, the extra computation cost is approximate 0.045 s in model sharing procedure and 0.966 s in secure prediction process. And when it comes to providing 200 ($\frac{k}{m} = 2\%$) test cases prediction service, the extra computation cost becomes 0.812 s in model sharing procedure and 0.921 s in secure prediction process. The above experimental results demonstrate that the extra computational cost introduced by our scheme is acceptable. Moreover, the higher ratio of test cases is, the higher extra computational cost is.

Influence of key length *len*: Finally, we show the computational costs of difference groups which are grouped by scheme and key length of RSA signature. In order to explore the influence of the length of key used in our scheme for RSA signature, we respectively set key length $len = 1024$, $len = 2048$ and $len = 4086$ (the least key length in RSA is 1024). And we use the pre-trained *model a* with 98% accuracy and set $\frac{k}{m} = 1\%$ here. The average time costs for providing 100 input data items inference of different key length of RSA signature are listed in Table 7.

Table 7. Average time costs of different key length of RSA signature.

Time (s)	Scheme 1-1	Scheme 1-2	Scheme 2-1	Scheme 2-2	Accuracy
len = 1024	2.147	2.133	1.970	1.773	98%
len = 2048	2.274	2.851	2.200	1.790	98%
len = 4096	3.529	11.171	2.388	1.792	98%

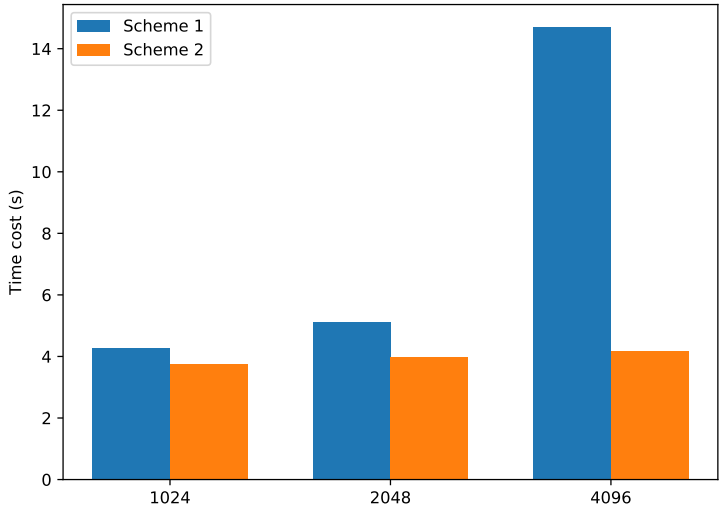

Figure 5. Average time costs of groups grouped by scheme and key length.

As depicted in Figure 5, there are three groups representing three different combinations of scheme and key length of RSA signature ($len = 1024$, $len = 2048$ and $len = 4086$). And each bar shows the total time cost of the complete privacy-preserving and fair prediction service. The extra computational cost of *Scheme 1* does not increase much when key length *len* increases from 1024 to 2048 but skyrockets to 14.7 s when $len = 4086$. We use the RSA cryptography to generates key pairs for signature in our scheme *Scheme 1*. And security of RSA with key length $len = 2048$ is satisfactory now. The above experimental results demonstrate that the extra computational cost introduced by our scheme is acceptable while keeping our scheme secure. Moreover, the higher the security is, the higher extra computational cost is.

6.3. Summary

The above performance analysis and experimental results demonstrate that the extra computational cost of our scheme is acceptable through theoretical and empirical evaluations. Additionally, experimental results show that prediction accuracy does not decrease after model sharing in our scheme.

7. Conclusion and Future Work

7.1. Conclusion

In this paper, we make the first attempt to design a privacy-preserving and fair scheme for deep learning inference service under publicly verifiable covert security setting. Focusing on protecting the privacy of sensitive input data of the client and the pre-trained model parameters of the server, we carefully analyze the merits and deficiencies of related works to define the problem of deep learning inference service scheme. Then, we propose a novel approach based on secure three-party computation and making commitments to solve the problem. The security analysis demonstrates that our scheme has the following desirable security properties: input data privacy, model privacy and defamation freeness. And the experimental results verify that our scheme is able to achieve the same prediction accuracy (up to 99%) as the pre-trained model while providing privacy and fairness guarantees. We also explore the effects of pre-trained model accuracy, the ratio of test cases and the key length of RSA signature respectively. Experimental results show that the higher pre-trained model accuracy is, the higher computational cost is: when the accuracy of pre-trained model accuracy grows from 98% to 99%, the total computational time is 0.808 s higher for 100 input images in our scheme. And the higher ratio of test cases is, the higher computational cost is: when the ratio of test cases rises from 1% to 2%, the total computational time is 2.089 s higher. Furthermore, the larger key length of RSA signature is, the higher computational cost is: when the key length of RSA signature increases from 1024 to 4086, the total computational time is 10.42 s higher. So we suggest that $len = 2048$ is proper considering the balance of security and computational cost. The results are quite reasonable intuitively and the extra computational costs of our scheme is acceptable.

7.2. Limits and Suggestions for Future Works

The deep neural network used for verifying our scheme in implementation is a medium scale CNN. The feasibility study of our scheme on large scale and various kinds of deep neural networks are the future works. We believe that the design rationale and the solution developed in this paper will motivate more research on privacy-preserving and fair deep learning inference service.

Author Contributions: F.T. contributed to writing—original draft preparation, methodology, software and validation of the proposed scheme; J.H. contributed to conceptualization; J.L. contributed to writing—review and editing and funding acquisition; H.W. contributed to supervision; and M.X. contributed to project administration.

Funding: This research was funded by the National Natural Science Foundation of China under Grant No. 61801489.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

Table A1. Notations.

Symbol	Meaning
C	the client (data provider and service consumer)
S	the server (model provider)
W_1, W_2, W_3	three workers (computation implementers)
$\langle X \rangle$	the input data set that client C holds
k	the number of test cases
$\langle x \rangle$	test cases data set
$\langle y \rangle$	the labels of $\langle x \rangle$
m	the number of input data items that C shares
I_j	the index of j_{th} test case
L_j	the corresponding label of j_{th} test case
$share\_data\_com$	the commitment of the shared data
$Sign_C(*)$	the signature of input of C
$hash(*)$	the hash value of input
$\|\|$	the operation of concatenate strings
λ	the model accuracy threshold
$share\_model\_com$	the commitment of the shared model
$model\_params$	the parameter values of the shared model
$\langle Y \rangle_i$	the inference result W_i obtains
$\langle X \rangle_i$	the shared input data worker W_i gets
y_{in}	intermediate inference result
n	the number of test cases judged correctly
$\langle Y \rangle$	the recovered output of inference
Ad	a malicious adversary
Sim	a simulator set to simulate Ad in the ideal world
Scheme 1	our scheme
Scheme 2	scheme proposed in Reference [31]
ma	pre-trained model accuracy
len	the key length of RSA signature

References

1. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90.
2. Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* **2018**, *37*, 421–436.
3. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **2018**, *13*, 55–75.
4. Chen, H.; Engkvist, O.; Wang, Y.; Olivecrona, M.; Blaschke, T. The rise of deep learning in drug discovery. *Drug Discov. Today* **2018**, *23*, 1241–1250.
5. Kermany, D.S.; Goldbaum, M.; Cai, W.; Valentim, C.C.; Liang, H.; Baxter, S.L.; McKeown, A.; Yang, G.; Wu, X.; Yan, F.; et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell* **2018**, *172*, 1122–1131.
6. Rajkomar, A.; Oren, E.; Chen, K.; Dai, A.M.; Hajaj, N.; Hardt, M.; Liu, P.J.; Liu, X.; Marcus, J.; Sun, M.; et al. Scalable and accurate deep learning with electronic health records. *NPJ Digit. Med.* **2018**, *1*, 18.
7. Ching, T.; Himmelstein, D.S.; Beaulieu-Jones, B.K.; Kalinin, A.A.; Do, B.T.; Way, G.P.; Ferrero, E.; Agapow, P.M.; Zietz, M.; Hoffman, M.M.; et al. Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface* **2018**, *15*, 20170387.
8. Zhong, P.; Gong, Z.; Li, S.; Schönlieb, C.B. Learning to diversify deep belief networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 3516–3530.
9. Ravanelli, M.; Brakel, P.; Omologo, M.; Bengio, Y. A network of deep neural networks for distant speech recognition. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 4880–4884.

10. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554.
11. Ranzato, M.A.; Poultney, C.; Chopra, S.; Cun, Y.L. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2007; pp. 1137–1144.
12. Marcus, G. Deep learning: A critical appraisal. *arXiv* **2018**, arXiv:1801.00631.
13. Bhattacharjee, B.; Boag, S.; Doshi, C.; Dube, P.; Herta, B.; Ishakian, V.; Jayaram, K.; Khalaf, R.; Krishna, A.; Li, Y.B.; et al. IBM deep learning service. *IBM J. Res. Dev.* **2017**, *61*, 10–1.
14. Buccafurri, F.; Fotia, L.; Lax, G.; Saraswat, V. Analysis-preserving protection of user privacy against information leakage of social-network Likes. *Inf. Sci.* **2016**, *328*, 340–358.
15. Phong, L.; Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1333–1345.
16. Tang, F.; Wu, W.; Liu, J.; Wang, H.; Xian, M. Privacy-Preserving Distributed Deep Learning via Homomorphic Re-Encryption. *Electronics* **2019**, *8*, 411.
17. Hesamifard, E.; Takabi, H.; Ghasemi, M. Cryptodl: Deep neural networks over encrypted data. *arXiv* **2017**, arXiv:1711.05189.
18. Mohassel, P.; Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 19–38.
19. Rouhani, B.D.; Riazi, M.S.; Koushanfar, F. Deepsecure: Scalable provably-secure deep learning. In Proceedings of the 55th Annual Design Automation Conference, San Francisco, CA, USA, 24–29 June 2018; p. 2.
20. Mohassel, P.; Rindal, P. ABY 3: A mixed protocol framework for machine learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 35–52.
21. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 201–210.
22. Kwabena, O.; Qin, Z.; Zhuang, T.; Qin, Z. MSCryptoNet: Multi-Scheme Privacy-Preserving Deep Learning in Cloud Computing. *IEEE Access* **2019**, *7*, 29344–29354. doi:10.1109/ACCESS.2019.2901219.
23. Boemer, F.; Lao, Y.; Wierzynski, C. nGraph-HE: A Graph Compiler for Deep Learning on Homomorphically Encrypted Data. *arXiv* **2018**, arXiv:1810.10121.
24. Liu, J.; Juuti, M.; Lu, Y.; Asokan, N. Oblivious neural network predictions via minionn transformations. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 619–631.
25. Juvekar, C.; Vaikuntanathan, V.; Chandrakasan, A. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1651–1669.
26. Hong, C.; Katz, J.; Kolesnikov, V.; Lu, W.-J.; Wang, X. Covert Security with Public Verifiability: Faster, Leaner, and Simpler. In Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, 19–23 May 2019, pp. 97–121.
27. Gutub, A.; Al-Juaid, N.; Khan, E. Counting-based secret sharing technique for multimedia applications. *Multimed. Tools Appl.* **2019**, *78*, 5591–5619.
28. Deshmukh, M.; Nain, N.; Ahmed, M. Efficient and secure multi secret sharing schemes based on boolean XOR and arithmetic modulo. *Multimed. Tools Appl.* **2018**, *77*, 89–107.
29. Araki, T.; Furukawa, J.; Lindell, Y.; Nof, A.; Ohara, K. High-throughput semi-honest secure three-party computation with an honest majority. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 805–817.
30. Mohassel, P.; Rosulek, M.; Zhang, Y. Fast and secure three-party computation: The garbled circuit approach. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 591–602.
31. Ryffel, T.; Trask, A.; Dahl, M.; Wagner, B.; Mancuso, J.; Rueckert, D.; Passerat-Palmbach, J. A generic framework for privacy preserving deep learning. *arXiv* **2018**, arXiv:1811.04017.

32. Araki, T.; Barak, A.; Furukawa, J.; Keller, M.; Lindell, Y.; Ohara, K.; Tsuchida, H. Generalizing the SPDZ compiler for other protocols. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 880–895.
33. Soifer, J.; Li, J.; Li, M.; Zhu, J.; Li, Y.; He, Y.; Zheng, E.; Oltean, A.; Mosyak, M.; Barnes, C.; et al. Deep Learning Inference Service at Microsoft. In Proceedings of the 2019 USENIX Conference on Operational Machine Learning (OpML 19), Santa Clara, CA, USA, 20 May 2019; USENIX Association: Santa Clara, CA, USA, 2019; pp. 15–17.
34. Catanzaro, B.; Chen, J.; Chrzanowski, M.; Elsen, E.; Engel, J.; Fougner, C.; Han, X.; Hannun, A.; Prenger, R.; Satheesh, S.; et al. Deployed End-to-End Speech Recognition. U.S. Patent App. 15/358,083, 25 May 2017.
35. Buccafurri, F.; Fotia, L.; Lax, G. *Social Signature: Signing by Tweeting*; Springer: Cham, Switzerland, 2014; pp. 1–14.