FPPNet: Fast Privacy-Preserving Neural Network via Three-Party Arithmetic Secret Sharing

Renwan Bi[1], Jinbo Xiong[1(✉)], Qi Li[2,3], Ximeng Liu[4], and Youliang Tian[5]

[1] Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China
jbxiong@fjnu.edu.cn

[2] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
liqics@njupt.edu.cn

[3] Key Laboratory of Cryptography of Zhejiang Province, Hangzhou Normal University, Hangzhou 311121, China

[4] Key Laboratory of Information Security of Network Systems, College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China

[5] State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China

Abstract. Jointing multi-source data for model training can improve the accuracy of neural network. To solve the raising privacy concerns caused by data sharing, data are generally encrypted and outsourced to a group of cloud servers for computing and processing. In this client-cloud architecture, we propose FPPNet, a fast and privacy-preserving neural network for secure inference on sensitive data. FPPNet is deployed in three cloud servers, who collaboratively execute privacy computing via three-party arithmetic secret sharing. We develop the secure conversion method between additive shares and multiplicative shares, and propose three secure protocols to calculate non-linear functions, such as comparison, exponent and division that are superior to prior three-party works. Some secure modules for running convolutional, ReLU, max-pooling and Sigmoid layers are designed to implement FPPNet. We theoretically analyze the security and complexity of the proposed protocols. With MNIST dataset and two types of neural networks, experimental results validate that our FPPNet is faster than the related works, and the accuracy is the same as that of plaintext neural network.

Keywords: Privacy-preserving · Neural network · Secure computing · Arithmetic secret sharing · Privacy computing

1 Introduction

In recent years, machine learning (ML) research has made continuous breakthroughs, such as neural network (NN) model, and has been widely used in

Y. Chenggang et al. (Eds.): MobiMedia 2022, LNICST 451, pp. 165–178, 2022.
https://doi.org/10.1007/978-3-031-23902-1_13

intelligent Internet of Things, medical care, connected autonomous vehicles [1] and other scenarios, profoundly changing the life style of people. For example, the model trained from a large amount of medical data can provide valuable predictive services to assist physicians in making more accurate diagnosis. Although advances in technologies such as cloud computing have made big data processing and model training more efficient, jointing multi-source data remains a major challenge. The data with characteristic of sensitivity and confidentiality are important assets of clients or organizations. Data sharing and union in the plaintext environment raise privacy concerns among data owners [2].

Privacy-preserving neural network based on secure multi-party computing (MPC) offers a promising solution [3]. Each independent client encrypted and outsourced their data to a group of cloud servers, who collaboratively execute neural network computing and process on joint data view, without disclosing any information beyond the results [4]. Outsourced cloud computing can effectively alleviate the clients' resource bottleneck. However, how to realize the correct and efficient computing over encrypted data is faced with great challenges [5]. At present, researchers combine cloud computing and MPC primitives, such as garbled circuit (GC), oblivious transfer (OT), secret sharing (SS), to construct some privacy-preserving schemes, mainly consisting of two types of two-party and multi-party settings. These schemes have achieved pretty great effect in the aspects of security, accuracy and overhead. It is worth mentioning that recent FALCON [6] is several orders of magnitude faster than the SecureML [3]. This is a favorable trend, whereas, the existing MPC protocol is still difficult to be deployed in large-scale applications with the reason of high-level computational and communication complexity. Therefore, we further devote to improving the efficiency of privacy computing without compromising security.

The motivations mainly include the following three parts. (i) On the two-party setting, the secure multiplication protocol using beaver triple [7] requires vast memory space, and the computational cost linearly increases with input data size. In contrast, the multiplication calculation no longer requires pre-computing triples and interaction when extended to the three-party setting [8]. (ii) The conversions among arithmetic, boolean and Yao shares require extra overhead in ABY3 [8], and inspired by FALCON [6], adopting only arithmetic secret sharing (ASS) can realize secure and efficient computing protocols. (iii) The secure design of non-linear functions, such as division and exponent, is frequently approximated by low-order polynomials and lacks a general non-linear calculation method [6,9].

On the basis of the above observations, the contributions of this paper are summarized as follows.

- We propose FPPNet, that clients randomly split their data into three shares and outsource them to a group of severs, and severs collaboratively execute neural network inference without compromising the privacy of uploaded data and intermediate results.
- We develop a general non-linear computing method by switching between additive and multiplicative shares, and design secure comparison, secure expo-

nent and secure division protocols, as well as securely realize ReLU, max-pooling and Sigmoid layers embedded in neural network.
– We analyze the security of the proposed secure computing protocols, and the computational and communication complexity are superior to the prior three-party works. Experimental results with MNIST dataset, validate that the runtime of our FPPNet is faster than the related works, and the accuracy is the same as that of the plaintext neural network.

The rest of this paper is organized as follows. In Sect. 2, we introduce the related works. The server-aided model and security model are presented in Sect. 4. Next, we discuss the design of basic secure computing protocols and the implementation of FPPNet in Sects. 3 and 5. Theoretical analysis and experimental results are described in Sects. 6 and 7 respectively. Lastly, we conclude the main work in Sect. 8.

2 Related Work

Privacy-preserving neural network based on MPC has been the hot research issues, including two-party and multi-party schemes.

Two-Party Schemes. Mohasse *et al.* [3] first proposed a privacy-preserving neural network SecureML scheme based on 2PC protocol [10], using beaver triple [7] and OT to perform multiplication and comparison operations, respectively. DeepSecure [11] adopted GC to compute and inference over encrypted data. MiniONN [12] removed some overhead to the offline phase, effectively improving the performance of DeepSecure. Additionally, some mixed-protocol frameworks executed in two non-collusive servers setting have been proposed, such as Chameleon [13], GAZELLE [14], DELPHI [15]. However, they are not easy to expand, because of the existence of computation-intensive cryptographic primitives. Huang *et al.* [16] proposed a lightweight privacy-preserving CNN feature extraction framework leveraging the additive SS, and obtained the good effect in terms of efficiency.

Multi-party Schemes. In order to overcome the offline storage problem of beaver triple, ABY3 [8] adopted $(2, 3)$-sharing, that each party possesses two shares of three shares, to design a non-interactive multiplication protocol. Also, ABY3 realized the secure conversion between Cleartexts and three types of arithmetic, boolean, and Yao sharing, so as to compute using the suitable sharing type. SecureNN [9] designed an effective and secure comparison protocol, which transformed the most significant bit (MSB) in even ring into the least significant bit (LSB) in odd ring. ASTRA [17] is a 3PC scheme with semi-honest security, which abandons some expensive GC protocol and prefix adder in works [3,8]. On the basis, BLAZE [18] has the stronger guarantee of fairness, and tolerates one of the three parties being malicious. Wagh *et al.* [6] proposed FALCON, an honest-majority secure 3PC framework supporting privacy-preserving deep neural network training, which realize the non-linear operation just using ASS instead of the sharing conversion. Furthermore, FLASH [19] and Trident [20]

introduce the fourth party to evaluate the behavior of each participant in the protocols, aiming to resist the potential attacks.

Inspired by ABY3 [8] and FALCON [6], we design a fast privacy-preserving neural network (FPPNet) leveraging only the three-party ASS, and devolop a general non-linear computing method by switching between additive shares and multiplicative shares.

3 Problem Statement

We consider a server-aided setting where the clients *Us* outsource the computation to three honest but curious (HbC) servers S_1, S_2 and S_3, so as to realize privacy-preserving neural network on outsourced data. It contains the following steps.

Setting. *Us* secretly share their data based on ASS and upload them to S_1, S_2 and S_3. No assumptions are made about the distribution of data, and we only split each value (e.g., the pixels of image data) into three shares in an addition manner.

Computing. Using the trained model, S_1, S_2 and S_3 collaboratively execute secure protocols and model inference over the received data shares. The real inference results can be recovered through simply adding three shares.

Also, we consider that dividing the power of cloud into three parts and allocating to three non-collusive servers. Similar to the semi-trusted model in FALCON [6], we assume an adversary $\mathcal{A}$ who can corrupt at most one of S_1, S_2 and S_3 and obtain the corresponding information. The raw data and intermediate results are private to *Us* and cannot be known by S_1, S_2 or S_3; while the trained model is private to servers [21]. Our goal is to realize correct privacy-preserving neural network inference while satisfying the privacy constraints of each entity.

4 Basic Protocols

4.1 Concepts and Descriptions

ASS is the addition and multiplication calculation in the form of shares. Due to the advantage of 3PC multiplication in ABY3 [8], we adopt 2-out-of-3 ASS (i.e., $(2,3)$-share) to split data. A secret u is randomly split into three shares $[u]_1$, $[u]_2$ and $[u]_3$ such that $u = [u]_1 + [u]_2 + [u]_3$, each server S_i ($i \in \{1,2,3\}$) holds 2-out-of-3 shares $([u]_i, [u]_{i+1})$. Note that, if $i = 3$, then $i + 1 = (i + 1) \mod 3 = 1$; and if $i = 1$, then $i - 1 = (i - 1) \mod 3 = 3$. In other word, any two servers (e.g., S_1 and S_2) can recover the real secret u. Except of additive shares $[u]_i$, we define the multiplicative shares $\langle v \rangle_i$, such that $v = \langle v \rangle_1 \cdot \langle v \rangle_2 \cdot \langle v \rangle_3$. Next, we introduce some preliminary protocols [6], as follows.

$\Pi_{Zeroshare}$: Random 3-out-of-3 additive shares $[\alpha]_1$, $[\alpha]_2$, and $[\alpha]_3$ such that $[\alpha]_1 + [\alpha]_2 + [\alpha]_3 = 0$, S_i holds $[\alpha]_i$. The random shares are generated by psuedo-random function (PRF).

$\Pi_{Oneshare}$: Random 3-out-of-3 multiplicative shares $\langle\beta\rangle_1$, $\langle\beta\rangle_2$, and $\langle\beta\rangle_3$ such that $\langle\beta\rangle_1 \cdot \langle\beta\rangle_2 \cdot \langle\beta\rangle_3 = 1$, S_i holds $\langle\beta\rangle_i$.

Π_{Mul}: To calculate $f = u \cdot v$, such that $u = [u]_1 + [u]_2 + [u]_3$, $v = [v]_1 + [v]_2 + [v]_3$ and $f = [f]_1 + [f]_2 + [f]_3$, S_i locally calculates 3-out-of-3 additive share $[f]_i \leftarrow [u]_i \cdot [v]_i + [u]_{i+1} \cdot [v]_i + [u]_i \cdot [v]_{i+1}$. To strengthen the randomness of output, invoking one time of $\Pi_{Zeroshare}$, S_i locally calculates $[f]_i \leftarrow [f]_i + [\alpha]_i$. Then, S_i obtains 2-out-of-3 shares $([f]_i, [f]_{i+1})$ by receiving $[f]_{i+1}$ from party S_{i+1}.

Additionally, all data is represented as double-precision float-point format deployed in IEEE 754–2008 [22], which consists of the sign, mantissa and exponent bits. The fixed mantissa bits can be encoded into integer through the enlargement method, i.e., $\lfloor u \cdot 10^q \rfloor$ from $\mathbb{Z}_n$, where q represents the decimal places.

4.2 Secure Conversion Protocols

Through investigation, the switch between additive and multiplicative shares can help S_i split and share secret, then saving communication rounds. Hence, we design a group of secure protocols Π_{M2A} and Π_{A2M} using $\Pi_{Zeroshare}$, $\Pi_{Oneshare}$ and Π_{Mul} protocols, as illustrated in Protocol 1 and 2, respectively. In Π_{M2A}, given the multiplicative shares $\langle u\rangle_i$, S_1, S_2 and S_3 jointly calculate $[d]_1 + [d]_2 + [d]_3 = \langle u\rangle_1 \cdot \langle u\rangle_2$ and $[f]_1 + [f]_2 + [f]_3 = \langle u\rangle_1 \cdot \langle u\rangle_2 \cdot \langle u\rangle_3$. Similarly given the additive shares $[u]_i$, Π_{A2M} obtains $\langle f\rangle_1 \cdot \langle f\rangle_2 \cdot \langle f\rangle_3 = [u]_1 + [u]_2 + [u]_3$. S_i sets the relationship of $[\alpha]_1 + [\alpha]_2 + [\alpha]_3 + \Delta = \langle\beta\rangle_1 \cdot \langle\beta\rangle_2 \cdot \langle\beta\rangle_3 \cdot \Delta$ (i.e., equals Δ). Since $g = [g]_1 + [g]_2 + [g]_3 = \Delta \cdot ([u]_1 + [u]_2 + [u]_3)$, S_i can obtain multiplicative share $\langle h\rangle_i$ through re-constructing g executed by S_1 (or S_2). Note that g cannot be known by S_3, otherwise S_3 can infer the real value of $[u]_1 + [u]_2 + [u]_3$. The new $\langle\beta\rangle_i^{\dagger}$ is used to randomize $\langle h\rangle_i$ and re-share $\langle f\rangle_i$.

Protocol 1: Additive to Multiplicative Shares Protocol (Π_{M2A})

Input: S_i ($i \in \{1, 2, 3\}$) holds $\langle u\rangle_i$.
Output: S_i outputs $[f]_i$.
1 S_1, S_2 and S_3 calculate $[d]_1, [d]_2, [d]_3 \leftarrow \Pi_{Mul}(\langle u\rangle_1, 0, 0; 0, \langle u\rangle_2, 0)$;
2 S_1, S_2 and S_3 calculate $[f]_1, [f]_2, [f]_3 \leftarrow \Pi_{Mul}([d]_1, [d]_2, [d]_3; 0, 0, \langle u\rangle_3)$;
3 S_i returns $([f]_i, [f]_{i+1})$.

5 The Construction of FPPNet

In this section, we discuss the implementation details of FPPNet, consisting of secure linear, ReLU, max-pooling and sigmoid layers.

5.1 Secure Linear Layer

In a neural network, the linear full-connected layers connect the relationship between sample feature and category using the trained weight and bias

Protocol 2: Multiplicative to Additive Shares Protocol (Π_{A2M})

Input: S_i ($i \in \{1,2,3\}$) holds $[u]_i$.
Output: S_i outputs $\langle f \rangle_i$.

1 S_i gets $[\alpha]_i$ and $\langle \beta \rangle_i$ through $\Pi_{Zeroshare}$ and $\Pi_{Oneshare}$;
2 S_3 randomly generates Δ, calculates $[\alpha]_3 \leftarrow [\alpha]_3 + \Delta$ and $\langle \beta \rangle_3 \leftarrow \langle \beta \rangle_3 \cdot \Delta$;
3 S_i calculates $[g]_i \leftarrow [\alpha]_i \cdot [u]_i + [\alpha]_i \cdot [u]_{i+1} + [\alpha]_{i+1} \cdot [u]_i$;
4 S_2 sends $[g]_2$ to S_1, S_3 sends $[g]_3$ to S_1;
5 S_1 calculates $\langle h \rangle_1 \leftarrow ([g]_1 + [g]_2 + [g]_3)/\langle \beta \rangle_1$, S_2 calculates $\langle h \rangle_2 \leftarrow 1/\langle \beta \rangle_2$, S_3 calculates $\langle h \rangle_3 \leftarrow 1/\langle \beta \rangle_3$;
6 S_i gets new $\langle \beta \rangle_i^{\dagger}$ through $\Pi_{Oneshare}$;
7 S_i calculates $\langle f \rangle_i \leftarrow \langle h \rangle_i \cdot \langle \beta \rangle_i^{\dagger}$, and sends $\langle f \rangle_i$ to S_{i-1};
8 S_i returns $(\langle f \rangle_i, \langle f \rangle_{i+1})$.

parameters (w, b). The expression is $y = w \cdot x + b$, where x and y are the input and output of this layer. For the data privacy, each data is randomly split into the three-party shares $[x]_1$, $[x]_2$ and $[x]_3$, and single S_i ($i \in \{1,2,3\}$) can locally calculates $[y]_i = w \cdot [x]_i + b/3$. Obviously, the sum of all shares $[y]_i$ equals $[y]$. Likewise, S_1, S_2 and S_3 can realize the convolutional layer through locally calculating the linear combination between input share and convolutional kernel parameters.

5.2 Secure ReLU and Max-Pooling Layers

The ReLU layer performs the operation $\mathrm{ReLU}(x) = \max(x, 0)$ over each neuron with the corresponding input x. Each neuron is activated or suppressed by comparing the relationship between x and 0. Instead of using GC in previous work [3,8], we design an efficient and secure comparison protocol Π_{Comp} adopting only ASS. As illustrated in Protocol 3, given $[u]_i$ and $[v]_i$, Π_{Comp} aims to obtain the comparison shares $[f]_i$ between u and v, such that $u = [u]_1 + [u]_2 + [u]_3$ and $v = [v]_1 + [v]_2 + [v]_3$. If $[f]_1 + [f]_2 + [f]_3 = 0$, then $u \geq v$; otherwise $([f]_1 + [f]_2 + [f]_3 = 1)$, $u < v$. In line 2, the additive share $[p]_i$ of the difference of $u - v$ are converted into multiplicative share $\langle r \rangle_i$, The exclusive-or (XOR) result of MSB of $\langle r \rangle_i$ determines whether $u \geq v$ or $u < v$. In line 7–8, the XOR operation of single bit can be converted into arithmetic operation, i.e., $\langle s \rangle_1 \oplus \langle s \rangle_2 \oplus \langle s \rangle_3 = \langle s \rangle_1 + \langle s \rangle_2 + \langle s \rangle_3 - 2 \cdot \langle s \rangle_1 \cdot \langle s \rangle_2 - 2 \cdot \langle s \rangle_2 \cdot \langle s \rangle_3 - 2 \cdot \langle s \rangle_1 \cdot \langle s \rangle_3 + 4 \cdot \langle s \rangle_1 \cdot \langle s \rangle_2 \cdot \langle s \rangle_3 = [f]_1 + [f]_2 + [f]_3$.

On the basis, the secure realization of ReLU layer is executed by Π_{ReLU} protocol. Given the input feature shares $[x]_1$, $[x]_2$, and $[x]_3$, Π_{ReLU} obtains $y = x$ if $x \geq 0$ or $y = 0$ if $x < 0$. In fact, it satisfies $y = x \cdot (1 - \eta)$, where η (*i.e.*, $[\eta]_1 + [\eta]_2 + [\eta]_3$) is the MSB of x.

Π_{ReLU}: To calculate $y = \max(x, 0)$, such that $x = [x]_1 + [x]_2 + [x]_3$ and $y = [y]_1 + [y]_2 + [y]_3$, S_1, S_2 and S_3 jointly calculate $[\eta]_1, [\eta]_2, [\eta]_3 \leftarrow \Pi_{Comp}([x]_1, [x]_2, [x]_3;\ 0, 0, 0)$ and $[y]_1, [y]_2, [y]_3 \leftarrow \Pi_{Mul}(1 - [\eta]_1, -[\eta]_2, -[\eta]_3; [x]_1, [x]_2, [x]_3)$. Then, S_i obtains 2-out-of-3 shares $([y]_i, [y]_{i+1})$.

Protocol 3: Secure Comparison Protocol (Π_{Comp})

Input: S_i ($i \in \{1,2,3\}$) holds $[u]_i$ and $[v]_i$.
Output: S_i outputs $[f]_i$.

S_i calculates $[p]_i \leftarrow [u]_i - [v]_i$;
S_1, S_2 and S_3 calculate $\langle r\rangle_1, \langle r\rangle_2, \langle r\rangle_3 \leftarrow \Pi_{A2M}([p]_1, [p]_2, [p]_3)$;
if $\langle r\rangle_i \geq 0$ **then**
  S_i assigns $\langle s\rangle_i \leftarrow 0$;
else
  S_i assigns $\langle s\rangle_i \leftarrow 1$;
S_1, S_2 and S_3 calculate $[t]_1, [t]_2, [t]_3 \leftarrow \Pi_{M2A}(\langle s\rangle_1, \langle s\rangle_2, \langle s\rangle_3)$;
S_i calculates $[f]_i \leftarrow \langle s\rangle_i - 2 \cdot \langle s\rangle_i \cdot \langle s\rangle_{i+1} + 4 \cdot [t]_i$;
S_i sends $[f]_i$ to S_{i-1}, and returns $([f]_i, [f]_{i+1})$.

The max-pooling layer divides the feature map into some pooling regions with the same size (e.g., 2×2) evenly, and selects the maximum feature element from each pooling region. Here, we design a secure max-pooling protocol $\Pi_{Maxpool}$ adopting the designed Π_{Comp} protocol. For each pooling region, given the input feature shares $[\boldsymbol{x}]_1$, $[\boldsymbol{x}]_2$, and $[\boldsymbol{x}]_3$, $\Pi_{Maxpool}$ obtains the maximum feature $y = \max(\boldsymbol{x})$ by executing Π_{Comp} and Π_{Mul} for $\mu^2 - 1$ times. Draw the idea of work [21], we can load the feature with the same position in each pooling region into a block, that is, the entire feature map are turned into μ^2 blocks. Then, we perform $\Pi_{Maxpool}$ over these block in parallel, and obtain the max-pooling output eventually.

$\Pi_{Maxpool}$: For the shares $[\boldsymbol{x}_{j,k}]_i, j = (1, 2, \cdots, \mu), k = (1, 2, \cdots, \mu)$ of each pooling region with size $\mu \times \mu$ in entire feature map, it calculates the maximum feature $y = \max(\boldsymbol{x})$, such that $\boldsymbol{x}_{j,k} = [\boldsymbol{x}_{j,k}]_1 + [\boldsymbol{x}_{j,k}]_2 + [\boldsymbol{x}_{j,k}]_3$ and $y = [y]_1 + [y]_2 + [y]_3$. S_1, S_2 and S_3 jointly calculate $[\delta]_1, [\delta]_2, [\delta]_3 \leftarrow \Pi_{Comp}([\boldsymbol{x}_{1,1}]_1, [\boldsymbol{x}_{1,1}]_2, [\boldsymbol{x}_{1,1}]_3; [\boldsymbol{x}_{1,2}]_1, [\boldsymbol{x}_{1,2}]_2, [\boldsymbol{x}_{1,2}]_3)$ and $[y]'_1, [y]'_2, [y]'_3 \leftarrow \Pi_{Mul}([\delta]_1, [\delta]_2, [\delta]_3; [\boldsymbol{x}_{1,2}]_1 - [\boldsymbol{x}_{1,1}]_1, [\boldsymbol{x}_{1,2}]_2 - [\boldsymbol{x}_{1,1}]_2, [\boldsymbol{x}_{1,2}]_3 - [\boldsymbol{x}_{1,1}]_3)$. $\mathcal{S}_i$ obtains the larger feature share $[y]_i \leftarrow [y]'_i + [\boldsymbol{x}_{1,1}]_i$ between $\boldsymbol{x}_{1,1}$ and $\boldsymbol{x}_{1,2}$. Then, S_1, S_2 and S_3 compare y with the remaining $\mu^2 - 2$ features, and obtain the maximum feature share $[y]_i$. Finally, S_i obtains 2-out-of-3 shares $([y]_i, [y]_{i+1})$.

5.3 Secure Sigmoid Layer

Sigmoid function is used to map the classification score to the interval of $(0, 1)$, so as to obtain the predicted probability of each category. The expression for this function is $\text{sigmoid}(x) = 1/(1 + \mathrm{e}^{-x})$. To protect the privacy of Sigmoid operation, we design secure exponent and division protocols (i.e., Π_{Exp} and Π_{Div}). Unlike the previous FALCON [6], we only adopt the conversion between additive and multiplicative shares, instead of extra polynomial approximation method. With the help of Π_{Exp} and Π_{Div}, the secure realization $\Pi_{Sigmoid}$ of Sigmoid operation can be finished.

Π_{Exp}: To calculate $f = e^u$, such that $u = [u]_1 + [u]_2 + [u]_3$ and $f = [f]_1 + [f]_2 + [f]_3$, S_1, S_2 and S_3 jointly calculate $[f]_1, [f]_2, [f]_3 \leftarrow \Pi_{M2A}(e^{[u]_1}, e^{[u]_2}, e^{[u]_3})$. Then, S_i obtains 2-out-of-3 shares $([f]_i, [f]_{i+1})$.

Π_{Div}: To calculate $f = u/v$, such that $u = [u]_1 + [u]_2 + [u]_3$, $v = [v]_1 + [v]_2 + [v]_3$ and $f = [f]_1 + [f]_2 + [f]_3$, S_1, S_2 and S_3 jointly calculate $\langle\lambda\rangle_1, \langle\lambda\rangle_2, \langle\lambda\rangle_3 \leftarrow \Pi_{A2M}([u]_1, [u]_2, [u]_3)$, $\langle\gamma\rangle_1, \langle\gamma\rangle_2, \langle\gamma\rangle_3 \leftarrow \Pi_{A2M}([v]_1, [v]_2, [v]_3)$ and $[f]_1, [f]_2, [f]_3 \leftarrow \Pi_{M2A}(\langle\lambda\rangle_1/\langle\gamma\rangle_1, \langle\lambda\rangle_2/\langle\gamma\rangle_2, \langle\lambda\rangle_3/\langle\gamma\rangle_3)$. Then, S_i obtains 2-out-of-3 shares $([f]_i, [f]_{i+1})$.

$\Pi_{Sigmoid}$: To calculate $\text{sigmoid}(x) = 1/(1 + e^{-x})$, such that $x = [x]_1 + [x]_2 + [x]_3$ and $y = [y]_1 + [y]_2 + [y]_3$, S_1, S_2 and S_3 jointly calculate $[\phi]_1, [\phi]_2, [\phi]_3 \leftarrow \Pi_{Exp}(e^{-[x]_1}, e^{-[x]_2}, e^{-[x]_3})$ and $[y]_1, [y]_2, [y]_3 \leftarrow \Pi_{Div}(1, 0, 0; 1 + [\phi]_1, [\phi]_2, [\phi]_3)$. Then, S_i obtains 2-out-of-3 shares $([y]_i, [y]_{i+1})$.

6 Theoretical Analysis

6.1 Security Analysis

The non-collusion assumption can guarantee that each S_i ($i \in \{1, 2, 3\}$) cannot recover the real information from three-party shares. The security of FPPNet depends on the proposed protocols, we provide the security proof using universal composition (UC) framework [23]. We assume that there is a probabilistic polynomial time (PPT) simulator $\mathcal{M}$ which can generate a set of simulatable view $View_{sim}$ for adversary $\mathcal{A}$. In a certain protocol Π, if $View_{sim}$ cannot be computationally distinguished from the real view $View_{real}$ of a physical entity (e.g., S_i), then Π is considered secure. The security proof relies on the following lemmas [24] and theorems.

Lemma 1. *A protocol is perfectly simulatable if all its sub-protocols are perfectly simulatable.*

Lemma 2. *If a random element a is uniformly distributed on $\mathbb{Z}_n$ and independent from any variable $b \in \mathbb{Z}_n$, then $a \pm b$ is also uniformly random and independent from b.*

Theorem 1. *The proposed protocols are secure in the HbC model.*

Proof. For Π_{M2A} protocol, the $View_{real}$ of S_i ($i \in \{1, 2, 3\}$) is $\{\langle u\rangle_i, [d]_i, [f]_i\}$, where $\langle u\rangle_i$ is the input. According to Lemma 2, $[d]_i$ and $[f]_i$ are random. Since Π_{Mul} has been proved to be secure and simulatable in FALCON [6], Π_{M2A} is also simulatable on the grounds of Lemma 1. $\mathcal{M}$ can generate the $View_{sim}$ of S_i, and $\mathcal{A}$ cannot distinguish between $View_{real}$ and $View_{sim}$ computationally. For Π_{A2M} protocol, the $View_{real}$ of S_1 is $\{[u]_1, [\alpha]_1, \langle\beta\rangle_1, [g]_1, g, \langle h\rangle_1, \langle f\rangle_1\}$. S_1 can re-construct $g = u \cdot \Delta$, but not predict real u without knowing Δ. Inversely, although S_3 knows Δ, but not g. Both g and Δ are unknown to S_2. In this way, the data possessed by S_i ($i \in \{1, 2, 3\}$) are random according to Lemma 2. $\mathcal{A}$ cannot computationally distinguish between $View_{real}$ and $View_{sim}$ of S_i. Thus, Π_{M2A} and Π_{A2M} protocols are considered secure in the HbC model. Since

Table 1. Computational complexity of proposed protocols.

Protocols	Input sizes	Plaintext	FPPNet (ours)
Π_{A2M}	n	–	$O(n)$
Π_{M2A}	n	–	$O(n)$
Π_{Comp}	n	$O(n)$	$O(n)$
Π_{Exp}	n	$O(n)$	$O(n)$
Π_{Div}	n	$O(n)$	$O(n)$
Π_{ReLU}	n	$O(n)$	$O(n)$
$\Pi_{Maxpool}$	$n, (\mu, \mu)$	$O(n(\mu^2-1)/\mu^2)$	$O(n(\mu^2-1)/\mu^2)$
$\Pi_{Sigmoid}$	n	$O(n)$	$O(n)$

Table 2. Communication complexity of proposed protocols.

Protocols	SecureNN [9]		FALCON [6]		FPPNet (ours)	
	Rounds	Message sizes	Rounds	Message sizes	Rounds	Message sizes
Π_{M2A}	–	–	–	–	2	$6nl$
Π_{A2M}	–	–	–	–	2	$5nl$
Π_{Comp}	5	$n(4l\log p + 13l)$	$\log p + 2$	$2nl$	4	$11nl$
Π_{Exp}	–	–	–	–	2	$6nl$
Π_{Div}	$10l_D$	$n(8l\log p + 24l)l_D$	$p\log p + 5p + 7$	$nl(4p+7)$	4	$16nl$
Π_{ReLU}	10	$n(8l\log p + 24l)$	$\log p + 5$	$4nl$	5	$14nl$
$\Pi_{Maxpool}$	$9(n-1)$	$(n-1)(8l\log p + 29l)$	$(\mu^2-1)(\log p + 7)$	$n(\mu^2+5l)/\mu^2$	$5(\mu^2-1)$	$14n(\mu^2-1)l/\mu^2$
$\Pi_{Sigmoid}$	–	–	–	–	6	$22nl$

the remaining Π_{Comp}, Π_{Exp}, Π_{Div}, Π_{ReLU}, $\Pi_{Maxpool}$ and $\Pi_{Sigmoid}$ protocols is designed on the basis of Π_{Mul}, Π_{M2A} and Π_{A2M} protocols, likewise, these protocols can be proved to be secure in the HbC model.

Theorem 2. *FPPNet is secure in the HbC model.*

Proof. Since Π_{ReLU}, $\Pi_{Maxpool}$ and $\Pi_{Sigmoid}$ protocols have been proved to be secure in Theorem 1, the outputs of these protocols are random. According to Lemma 1, FPPNet is also simulatable. The $View_{real}$ of FPPNet and $View_{sim}$ generated by $\mathcal{M}$ are computationally indistinguishable for $\mathcal{A}$. Therefore, FPPNet is secure in the HbC model.

6.2 Complexity Analysis

In this section, we analyze the computational and communication complexity of the proposed protocols. As illustrated in Table 1, we assume that n denotes the size of the vector in vectorized implementation, (μ, μ) is the size of pooling region. The secure conversion Protocols Π_{M2A} and Π_{A2M} are designed by only three-party ASS, that cost is $O(n)$. The remaining protocols are on the basis of the Π_{M2A} and Π_{A2M}. In the $\Pi_{Maxpool}$ protocol, it performs Π_{Comp} for $\mu^2 - 1$

times. Clearly, the computational complexity of our protocols is linear, and is the same order as that in plaintext environment.

Compared with the previous works [6,9], Table 2 gives the communication complexity results. l denotes the bit-width, l_D denotes the precision of bits, and p denotes the logarithm of the ring size. For Π_{Comp} protocol, SecureNN [9] transforms MSB in even ring into LSB in odd ring, removing the dependence of communication rounds on l. FALCON [6] adopts (2, 3)-sharing instead of (2, 2)-sharing using in SecureNN, and further reduces the size of ring structure (i.e., $\mathbb{Z}_{2^p}$). By contrast, our Π_{Comp} only uses arithmetic addition and multiplication. Although the communication overhead is higher than FALCON, it greatly compresses the communication rounds and is independent of l and p. For Π_{Div} protocol, we do not require any extra polynomial iteration compared with SecureNN and FALCON. Additionally, we design Π_{Exp} and $\Pi_{Sigmoid}$ protocols, which does not requires to be approximated using ReLU. Overall, our protocol can realize constant rounds for communication, regardless of the size of the input data.

7 Experimental Results

Experiments are performed on a 64-bit personal computer with 1.80 GHz CPU and 20 GB RAM. We implement FPPNet using PyTorch 1.6 deep learning framework and Python 3 language. The tensor package is used as a multidimensional container of numbers to execute the calculation in parallel. We use MNIST dataset to conduct experiments, which consists of $60,000$ images for training and $10,000$ images for inference. Each image is a 28×28 pixel image of a hand-written digit along with a label between 0 and 9. Similar to the previous works [6,8,9], we evaluate the performance of FPPNet by using MNIST dataset and adopting two different networks (i.e., NN-3 and LeNet) in terms of efficiency and accuracy. NN-3 is a three-layer full-connected network with ReLU activation after each layer, which has around 118K parameters. LeNet is a standard convolutional neural network, which contains two convolutional layers, two max-pooling layers, three full-connected layers and five ReLU layers with 431K parameters.

The efficiency of FPPNet depends on the designed protocols. We firstly analyze the runtime of various layers in a certain network, as shown in Table 3. Since we adopt the single computer to simulate the running of three servers, the runtime of secure linear convolutional and full-connected layers are around $3\times$ times than plaintext LeNet. Adopting Π_{Comp} and Π_{Mul} protocols, the secure realization cost of ReLU layer is around $55\times$ to $72\times$ times than the original activation function. Max-pooling and Sigmoid layers have the similar trend, and the runtime linearly increases with the input size. Over all, the runtime of each layer can be maintained in microseconds.

Next, we discuss the performance of FPPNet from the aspect of efficiency and accuracy. Table 4 gives the inference efficiency comparison result between our FPPNet and the existing three-party schemes [6,8,9]. The single number is

Table 3. Comparison of runtime of various layers for FPPNet and plaintext LeNet. All runtimes are reported in microsecond (μs).

Layers	Input sizes	LeNet	FPPNet	Multiple
Convolution	$(1, 28, 28)$	26.7	80.7	3.02×
	$(6, 12, 12)$	27.2	61.2	2.25×
ReLU	$(6, 24, 24)$	29.0	1601.1	55.21×
	$(16, 8, 8)$	6.9	498.0	72.17×
Max-pooling	$(6, 24, 24)$	45.7	1615.0	35.34×
	$(16, 8, 8)$	12.9	460.7	35.71×
Full-connected	$(1, 256)$	1.0	2.2	2.20×
	$(1, 120)$	0.4	1.2	3.0×
	$(1, 84)$	0.1	0.3	3.0×
Sigmoid	$(1, 10)$	0.2	5.4	27.0×

Table 4. Comparison of inference efficiency of various schemes for single image from MNIST dataset. All runtimes are reported in second (s) and communication in MB.

Schemes	NN-3		LeNet	
	Runtime	Communication	Runtime	Communication
ABY3 [8]	0.008	0.5	–	–
SecureNN [9]	0.043	2.10	0.23	18.94
FALCON [6]	0.011	0.012	0.047	0.74
FPPNet (ours)	0.007	0.030	0.031	0.862
Plaintext	0.001	–	0.004	–

stored as fixed 64-bit data type. FALCON [6] combines the advantages of ABY3 [8] and SecureNN [9], and adopt $(2, 3)$-sharing method to perform the end-to-end privacy-preserving inference in a small ring. To pursue the higher performance gains, different from FALCON, our FPPNet transforms the calculation over boolean ring into pure arithmetic calculation, and realizes the non-linear functions by adopting the designed secure conversion protocols. Inevitably, the communication overhead of FPPNet is slightly higher than that of FALCON, but it is acceptable. More importantly, FPPNet is able to achieve faster inference efficiency due to the fewer rounds of communication between servers. In other word, FPPNet further narrows the generation gap between ciphertext and plaintext inference.

Additionally, the inference accuracy is an important factor to evaluate the effectiveness of FPPNet. We set the learning rate as 0.1 and the batch size as 128 in training. After 40 iterations, as shown in Table 5, the inference accuracy of NN-3 and LeNet can achieve 98.09% and 99.03% in plaintext environment, respectively. Clearly, the accuracy of FALCON has low loss. This is because

Table 5. Comparison of inference accuracy for batch size 128 with MNIST dataset.

Networks	Plaintext	FALCON [6]	FPPNet (ours)
NN-3	98.09%	97.42%	98.09%
LeNet	99.03%	96.85%	99.03%

the fixed-point 32-bit algorithm in FALCON is still slightly different from high-precision 64-bit computing. By contrast, we adopt the fixed-point 64-bit data type to ensure that our FPPNet can achieve the consistent accuracy as that of plaintext network as long as multiplicative results do not overflow.

8 Conclusion

In this paper, we proposed a fast and privacy-preserving neural network, referred to FPPNet, leveraging only three-party arithmetic secret sharing. Specifically, we design a group of secure conversion protocols for switching additive and multiplicative shares, and develop a series of secure computing protocols to realize each layer of FPPNet. Three non-collusive servers can collaboratively perform the secure inference while not disclosing the intermediate calculation results. Experimental results indicated that with MNIST dataset, our FPPNet is faster than FALCON, and can achieve the same accuracy as that of plaintext neural network, having the stronger practicality. In the future work, we further try to applied FPPNet to more large-scale deep neural network, and explore the end-to-end privacy-preserving training using the designed secure conversion protocols.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China under Grant 61872088, Grant U1905211, Grant 62072109, and Grant U1804263; in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS202105; in part by the Science and Technology Major Support Program of Guizhou Province under Grant 20183001; in part by the Science and Technology Program of Guizhou Province under Grant 20191098; in part by the Project of High-level Innovative Talents of Guizhou Province under Grant 20206008; and in part by the Open Research Fund of Key Laboratory of Cryptography of Zhejiang Province under Grant ZCL21015.

References

1. Xiong, J., Bi, R., Chen, Q., et al.: Towards edge-collaborative, lightweight and secure region proposal network. J. Commun. **41**(10), 188–201 (2020)
2. Xiong, J., Bi, R., Zhao, M., et al.: Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles. IEEE Wirel. Commun. **27**(3), 24–30 (2020)
3. Mohassel, P., Zhang, Y.: SecureML: a system for scalable privacy-preserving machine learning. In: IEEE Symposium on Security and Privacy (SP), USA, pp. 19–38 (2017)

4. Bi, R., Chen, Q., Xiong, J., et al.: Design method of secure computing protocol for deep neural network. Chin. J. Netw. Inf. Secur. **6**(4), 130–139 (2020)
5. Xiong, J., Zhou, Y., Bi, R., et al.: Towards edge-collaborative, lightweight and privacy-preserving classification framework. J. Commun. **43**(1), 127–137 (2022)
6. Wagh, S., Tople, S., Benhamouda, F., et al.: FALCON: honest-majority maliciously secure framework for private deep learning. Proc. Priv. Enhancing Technol. **1**, 188–208 (2021)
7. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_34
8. Mohassel, P., Rindal, P.: ABY3: a mixed protocol framework for machine learning, In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), Los Angeles, USA, pp. 35–52 (2018)
9. Wagh, S., Gupta, D., Chandran, N.: SecureNN: 3-party secure computation for neural network training. Proc. Priv. Enhancing Technol. **2019**(3), 26–49 (2019)
10. Demmler, D., Schneider, T., Zohner, M.: ABY-A framework for efficient mixed-protocol secure two-party computation. In: Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, USA, pp. 1–15 (2015)
11. Rouhani, B.D., Riazi, M.S., Koushanfar, F.: DeepSecure: scalable provably-secure deep learning. In: Proceedings of the 55th Annual Design Automation Conference (DAC), San Francisco, USA, pp. 1–6 (2018)
12. Liu, J., Juuti, M., Lu, Y., et al.: Oblivious neural network predictions via MiniONN transformations. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), Los Angeles, USA, pp. 619–631 (2017)
13. Riazi, M.S., Weinert, C., Tkachenko, O., et al.: Chameleon: a hybrid secure computation framework for machine learning applications. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS), New York, USA, pp. 707–721 (2018)
14. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: GAZELLE: a low latency framework for secure neural network inference. In: 27th USENIX Security Symposium (USENIX Security), Berkeley, USA, pp. 1651–1669 (2018)
15. Mishra, P., Lehmkuhl, R., Srinivasan, A., et al.: DELPHI: a cryptographic inference service for neural networks. In: 29th USENIX Security Symposium (USENIX Security), Boston, USA, pp. 2505–2522 (2020)
16. Huang, K., Liu, X., Fu, S., et al.: A lightweight privacy-preserving CNN feature extraction framework for mobile sensing. IEEE Trans. Dependable Secure Comput. **18**(3), 1441–1455 (2021)
17. Chaudhari, H., Choudhury, A., Patra, A., et al.: ASTRA: high throughput 3PC over rings with application to secure prediction. In: Proceedings of the ACM SIGSAC Conference on Cloud Computing Security Workshop (CCSW), Los Angeles, USA, pp. 81–92 (2019)
18. Patra, A., Suresh, A.: BLAZE: blazing fast privacy-preserving machine learning. In: 27th Annual Network and Distributed System Security Symposium (NDSS), San Diego, USA, pp. 1–18 (2020)
19. Byali, M., Chaudhari, H., Patra, A., et al.: FLASH: fast and robust framework for privacy-preserving machine learning. Proc. Priv. Enhancing Technol. **2**, 459–480 (2020)
20. Chaudhari, H., Rachuri, R., Suresh, A.: Trident: efficient 4PC framework for privacy preserving machine learning. In: Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, USA, pp. 1–18 (2020)

21. Xiong, J., Bi, R., Tian, Y., et al.: Towards lightweight, privacy-preserving cooperative object classification for connected autonomous vehicles. IEEE Internet Things J. **9**(4), 2787–2801 (2022)
22. Markstein, P.: The new IEEE-754 standard for floating point arithmetic. In: Dagstuhl Seminar Proceedings, pp. 1–3 (2008)
23. Damgård, I., Fitzi, M., Kiltz, E., et al.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Theory of Cryptography Conference (TCC), New York, USA, pp. 285–304 (2006)
24. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: a framework for fast privacy-preserving computations. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 192–206. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88313-5_13