GFS-CNN: A GPU-friendly Secure Computation Platform for Convolutional Neural Networks

Chao Guo[1], Ke Cheng[2], Jiaxuan Fu[2], Ruolu Fan[3], Zhao Chang[2], Zhiwei Zhang[2], and Anxiao Song[2]
[1]Hangzhou Institute of Technology, Xidian University, Hangzhou, 311231, China
[2]School of Computer Science and Technology, Xidian University, Xi'an, 710071, China
[3]China Telecom Corporation Limited, Shaanxi Branch, Xi'an, 710075, China

Outsourcing convolutional neural network (CNN) inference services to the cloud is extremely beneficial, yet raises critical privacy concerns on the proprietary model parameters of the model provider and the private input data of the user. Previous studies have indicated that some cryptographic tools such as secure multi-party computation (MPC) can be used to achieve secure outsourced inferences. However, MPC-based approaches often require a large number of communication rounds across two or more non-colluding servers, which make them hard to exploit GPU acceleration. In this paper, we propose GFS-CNN, a GPU-friendly secure computation platform for convolutional neural networks. The following two specific efforts of GFS-CNN have been made by combining machine learning and cryptography techniques. Firstly, We use quadratic activation functions to replace most of the ReLU functions without losing much accuracy, so as to create a mixed linear layer for better efficiency by integrating convolution, batch normalization, and quadratic activation. Secondly, for the rest ReLU functions, we implement the secure ReLU protocol using function secret sharing, enabling GFS-CNN to evaluate the secure comparison function via a single interaction during the online phase. Extensive experiments demonstrate that GFS-CNN is accuracy-preserving and reduces online inference time by 16.4% on VGG-16 models compared to Delphi (USENIX Security'20).

***Index Terms*—Secure inference, convolutional neural network, function secret share, secure multi-party computation, GPU acceleration.**

I. Introduction

The convolutional neural network (CNN) inferences have enabled numerous applications in diverse fields, such as image classification, voice assistant, and security systems. But CNN inferences[1], [2], [3], [4] are largely data-driven and rely on aggregating and analyzing users' massive data, which requires strong computing power. Towards these needs, outsourcing the CNN inference to cloud servers has rapidly become an appealing offering.

However, uploading the users' personal data to the cloud will raise critical privacy concerns [5], [6], [7], [8]. To address the privacy concerns, a number of works [3], [9], [10] in the last few years have introduced cryptographic frameworks based on the secure multiparty computation (MPC) to enable secure CNN inference. From a broad perspective, MPC protocols enable a group of mutually untrusting parties to calculate an arbitrary function by inputting their private data collaboratively. In the calculation process, each party only learns about the secret shares of output and nothing more. However, the existing MPC-based CNN inference protocols remain the problems in expensive calculations and high communication rounds. These problems occurs for the following two reasons.

Firstly, some CNN inference protocols [1], [2], [11] based on MPC techniques require high communication rounds, which makes them not scale well to the hardware acceleration with graphics processing units (GPUs). Specifically, in the GPU-accelerated scenario, data is typically stored in GPU memory, and the communication between GPU memories of different parties needs to go through CPU bridging. Therefore, high communication rounds increase the communication time between parties and the data exchange time between the GPU and the CPU.

Secondly, some approaches [6], [7] replace some ReLU activations with quadratic activation functions to minimize time-consuming nonlinear computation while maintaining accuracy. However, replacing all ReLU layers in the CNN with quadratic activations is impracticable as it can result in reduced inference accuracy and even makes the inference unavailable.

To address the above problems, we introduce GFS-CNN, a GPU-friendly privacy-preserving computation framework convolutional neural network inference, in which all of the secure protocols for both linear and nonlinear are implemented on the GPU. Our main contributions are as follows:

- We use quadratic activation functions to replace most of the ReLU functions without losing much accuracy, so as to create a mixed linear layer for better efficiency by integrating convolution, batch normalization, and quadratic activation. To implement the calculation, we propose a basic protocol, secure continuous multiplication protocol, which enables to compute the multiple secret-shared matrices via two communication rounds.
- We propose the secure ReLU protocol, which combines the function secret sharing and additive secret sharing to provide a privacy-preserving implementation of the ReLU layer. This enables us to evaluate secure ReLU on secret-shared data via only two communication rounds, in which the secure comparison function only needs a single communication round.

We evaluate the performance of the GFS-CNN basic protocols and the overall performance of GFS-CNN on VGG-16.

Manuscript received May 27, 2023; revised June 26, 2023. Corresponding author: Ke Cheng (email: chengke@xidian.edu.cn).

Typically, the online calculation time of the mixed linear layer and secure ReLU function is 27.8% and 21.1% faster than the previous works [5], [7]. For evaluating VGG-16 on CIFAR-10, GFS-CNN reduces online inference time by 16.4% compared to the current state-of-the-art Delphi [7].

The rest of the paper is organized as follows. Section II presents the related work. Section III and Section IV introduce preliminaries and system overview. The specific schemes of our system implementation are in Section V. The detailed experimental evaluations are shown in Section VI. Finally, Section VII concludes the paper.

II. Relateed Works

In privacy-preserving neural network inference outsourcing, the provider's model is stored in ciphertext on the cloud server. The users can obtain the inference output without revealing their input data. Many prior works have explored the privacy-preserving CNN inference methods [1], [6], [7], [9], [12], [13].

SecureML [9] is a classic secure machine learning framework that uses a two-server secure computational model to enable the ciphertext model for both inference and training on ciphertext input. XONN [1] is a privacy-preserving inference framework based on garble circuit. It replaces the costly matrix multiplication operations of deep learning models with the nearly free XNOR operator in GC, resulting in a $37\times$ increase in operational efficiency improvement compared to SecureML.

The state-of-the-art works CryptGPU [13] and Piranha [14] argue that hardware acceleration with GPUs is essential for accelerating privacy-preserving CNN inference. CryptGPU introduces a new interface that can process secret-shared values by highly-optimized CUDA kernels for linear algebra. Based on the interface, CryptGPU proposes a sequence of cryptographic protocols to enable privacy-preserving evaluation of both linear and nonlinear operations on the GPU. Piranha is a modular structure platform for GPU-accelerated MPC protocol development, which can accelerate secret-sharing protocols by providing integer-based kernels in current general-purpose GPU libraries. Piranha's modular structure provides wide applicability for other projects to use GPU acceleration without requiring expert knowledge.

For the ReLU approximation, Manto [6] and Delphi [7] propose good solutions. Delphi [7] replaces the ReLU activation function with a quadratic activation function and designs a planner that automatically determines the number of quadratic approximation polynomials and their positions in the network for the better trade-off of protocol efficiency and classification accuracy. However, for deep neural networks, Delphi still needs to keep many ReLU functions and implement these quadratic activation functions in practice through time-consuming confusion circuits implement these nonlinear operations in practice. Manto [6] presents a quadratic-fitting function search algorithm and a fine-tuning approach to produce accurate CNNs with fewer ReLU activations. Additionally, it proposes a collection of secure protocols employing lightweight cryptographic primitives for both quadratic and ReLU6 activations.

TABLE I: Notations and Definitions

Notations	Definitions
a	a scalar a
$\mathbf{X}$	a matrix $\mathbf{X}$
$\mathbf{E}$	the unit matrix with all elements of 1
$a\mathbf{X}$	the matrix $\mathbf{X}$ multiplied by the number a
$\mathbf{X} \times \mathbf{Y}$	the Hadamard product of matrices $\mathbf{X}$ and $\mathbf{Y}$
$\mathbf{X} \cdot \mathbf{Y}$	the Cartesian product of matrices $\mathbf{X}$ and $\mathbf{Y}$
$\mathbf{X} \otimes \mathbf{Y}$	the 2D-convolutional calculation of $\mathbf{X}$ and $\mathbf{Y}$

III. Preliminaries

This section introduces some basic protocols used in our system, including additive secret sharing, function secret sharing, linear layer computation, and ReLU approximation. Table I lists the main notations and definitions used in GFS-CNN.

A. Additive Secret Sharing

A 2-out-of-2 additive secret sharing scheme (ASS) [4] is a cryptographic primitive that protects a private value x by splitting it into two secret shares, where each share alone reveal nothing about x. It consists of two algorithm: Share and Restore, which are defined as follows:

- Share(x): Server S_i generates a random number r (for $i \in \{0,1\}$), sets $\langle x \rangle_i = (x - r)$, and sends r to S_{1-i}, who sets $\langle x \rangle_{1-i} = r$. We denote $\langle x \rangle = (\langle x \rangle_0, \langle x \rangle_1)$ as their shorthand.
- Restore($\langle x \rangle$): Server S_i send $\langle x \rangle_i$ to server S_{1-i}, who compute $x = (\langle x \rangle_i + \langle x \rangle_{1-i})$

Next, we introduce two secure protocols for fundamental operations over the above secret-shared data. SecAdd takes two shares $\langle x \rangle$,$\langle y \rangle$ as input, and outputs $\langle x+y \rangle$ without revealing x, y. Similarly SecMul takes $\langle x \rangle$,$\langle y \rangle$ as input, and outputs $\langle xy \rangle$ without revealing any values of x, y. The above operations are defined as follows:

- SecAdd($\langle x \rangle$,$\langle y \rangle$): S_i locally computes $\langle x+y \rangle_i \leftarrow \langle x \rangle_i + \langle y \rangle_i$(for $i \in 0, 1$)
- SecMul($\langle x \rangle$,$\langle y \rangle$): We use Beaver's triples [15] to implement SecMul. First, randomly generate triples $\{a, b, c\}$, where $c = ab$. Then share them among the two servers of the form $\{\langle a \rangle, \langle b \rangle, \langle c \rangle\}$. For each server, S_i locally computes $\langle e \rangle_i \leftarrow \langle x \rangle_i - \langle a \rangle_i$ and $\langle f \rangle_i \leftarrow \langle y \rangle_i - \langle b \rangle_i$. Both servers perform Restore($\langle e \rangle$) and Restore($\langle f \rangle$), then set output $\langle xy \rangle_i \leftarrow i \cdot ef + e \cdot \langle b \rangle_i + f \cdot \langle a \rangle_i + \langle c \rangle_i$.

B. Function Secret Sharing

Function secret sharing (FSS) [16], [17], [18], [19], a primitive introduced by Boyle [16], [17], [18], provides a way to allow secure two-server evaluation of non-linear functions (e.g., comparison) with low interactions in the secret sharing domain. Thus, in a two-server FSS scheme, a target function f is split into two succinct keys f_1, f_2, where $f(x) = f_1(x) + f_2(x)$. Each key alone does not reveal private information about the target function f. A 2-server function secret sharing (FSS) scheme is a pair of algorithms (KeyGen, Eval) with the following syntax:

- KeyGen$(1^\lambda, f_{\alpha,\beta})$ is a probabilistic polynomial-time key generation algorithm, which on input 1^λ (security parameter) and $f_{\alpha.\beta} \in \{0,1\}^*$, output a pair of succinct FSS keys (k_0, k_1).
- Eval (i, k_i, x) is a polynomial-time evaluation algorithm, in which input is server index $i \in \{0,1\}$, k_i(the i-th FSS key) and x, and the outputs is $\langle f_{\alpha,\beta}(x)\rangle_i$.

FSS provides security guarantees such that if an adversary has access to only one of the keys (k_0, k_1), the adversary cannot obtain any confidential information about the target function for its output $f(x)$.

C. Linear Layer Computation

The secure computation of linear layers has been widely used in many previous works [5], [6], [8]. We employ a state-of-the-art convolutional protocol [6], which transforms the convolution to the matrix multiplication, denoted as $\mathbf{Y} = \mathbf{X} \otimes \mathbf{W}_1 + \mathbf{B}_1$ (The notation "$\otimes$" denotes the 2D-convolutional calculation of two matrices). Similarly, we can denote the equation for the batch normalization (BN) as $\mathbf{Y} = \mathbf{A}_1 \times \mathbf{X} + \mathbf{B}_2$.

D. ReLU Approximation

Various prior works [6], [7], [20] have attempted to provide effective quadratic activation functions to replace the ReLU function. The output of the quadratic activation function in a certain interval is very close to the output of the ReLU activation function. The secure quadratic activation (SQA) protocol assumes that S_0 holds $\langle \mathbf{X}\rangle_0, \langle k_2\rangle_0, \langle k_1\rangle_0, \langle k_0\rangle_0$ and S_1 holds $\langle \mathbf{X}\rangle_1, \langle k_2\rangle_1, \langle k_1\rangle_1, \langle k_0\rangle_1$. The SQA protocol aims to securely compute $f(x) = k_2\mathbf{X}^2 + k_1\mathbf{X} + k_0$, without revealing any private information about $\mathbf{X}, k_2, k_1, k_0$.

IV. System overview

A. System model

Fig. 1 shows our system architecture. Our system involves a user, a model provider, two non-colluding cloud servers, and a trusted dealer (denoted as *TD*). The user holds the privacy input data. The model provider holds the privacy model parameters. The trusted dealer generates auxiliary parameters for subsequent secure calculations. The cloud servers offer efficient and secure neural network inference services. A pre-trained model can be deployed in a secret-shared form on the cloud through the service, which can then provide inference services without exposing the model in plaintext. The user encrypts the data locally and uploads it to two servers. The cloud server performs a series of secure computing protocols and then sends the encrypted result to the designated user.

B. Threat Model

In this paper, we present a set of secure computing protocols designed under the semi-honest security model. Specifically, the protocols are designed for two cloud servers assumed to be semi-honest, meaning they will faithfully perform operations as specified and will not actively disrupt the execution of the protocol, but may attempt to snoop or infer private information related to the original data. Additionally, we assume no collusion between the two cloud servers, as large service providers typically value their reputations in practice. This assumption is widely used in previous works [2], [5], [21].

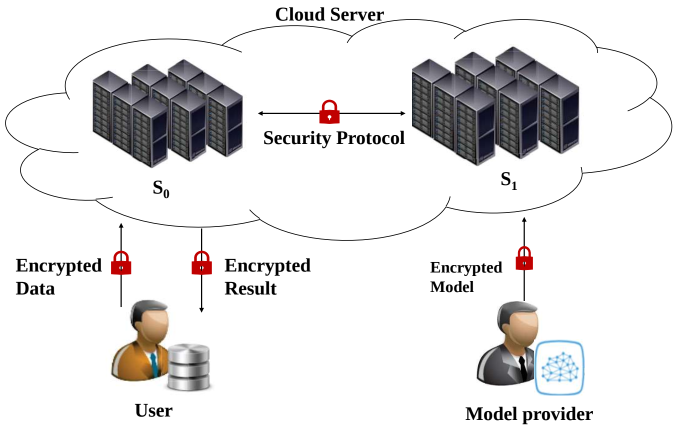


Fig. 1: System architecture.

Algorithm 1 Secure Continuous Multiplication (SecCMul) Protocol

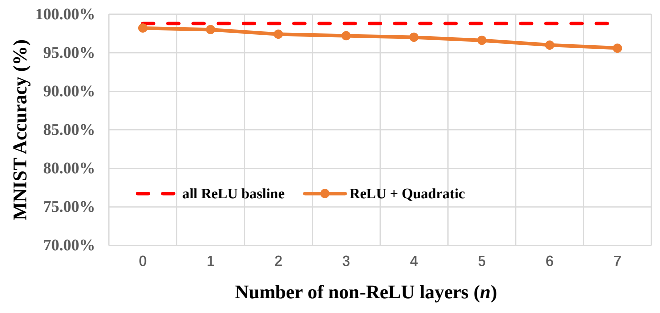
```
Input: $S_0$ holds $\langle \mathbf{X}_0\rangle_0, \langle \mathbf{X}_1\rangle_0, \cdots, \langle \mathbf{X}_N\rangle_0$,
       $S_1$ holds $\langle \mathbf{X}_0\rangle_1, \langle \mathbf{X}_1\rangle_1, \cdots, \langle \mathbf{X}_N\rangle_1$
Output: $S_0$ outputs $\langle \mathbf{Y}\rangle_0 = \langle \mathbf{X}_0 \times \mathbf{X}_1 \times \cdots \times \mathbf{X}_N\rangle_0$,
        $S_1$ outputs $\langle \mathbf{Y}\rangle_1 = \langle \mathbf{X}_0 \times \mathbf{X}_1 \times \cdots \times \mathbf{X}_N\rangle_1$
1: On the offline phase
2: TD randomly choose $a_1, a_2, \cdots, a_N$, $t_0, t_1, \cdots, t_N$
3: TD computes $t_0^{-1}, t_1^{-1}, \cdots, t_N^{-1}$, $p_j = t_{j-1}t_j^{-1}$, $a_jp_j = a_it_{j-1}t_j^{-1}, (j \in [1,N])$, and computes $q = t_0^{-1}t_N$
4: TD send $\{(\langle a_j\rangle_0, \langle p_j\rangle_0, \langle a_jp_j\rangle_0)_{j\in[1,N]}, \langle q\rangle_0\}$ to $S_0$ and send $\{(\langle a_j\rangle_1, \langle p_j\rangle_1, \langle a_jp_j\rangle_1)_{j\in[1,N]}, \langle q\rangle_1\}$ to $S_1$
5: On the online phase
6: For $(j = 1, 2, ..., N)$ do in parallel:
7:     For $i \in \{0,1\}$, $S_i$ computes $\langle \mathbf{X}_j\rangle_i - \langle a_j\mathbf{E}\rangle_i$
8:     For $i \in \{0,1\}$, $S_i$ sends $\langle \mathbf{X}_j\rangle_i - \langle a_j\mathbf{E}\rangle_i$ to $S_{1-i}$ and obtains $(\mathbf{X} - a_j\mathbf{E})$ = Restore($\langle \mathbf{X}_j\rangle_i - \langle a_j\mathbf{E}\rangle_i$)
9:     For $i \in \{0,1\}$, $S_i$ computes $\langle \mathbf{D}_j\rangle_i = ((\mathbf{X}_j - a_j\mathbf{E}) \times \langle p_j\mathbf{E}\rangle_i + \langle (a_jp_j)\mathbf{E}\rangle_i)$ and obtain $\mathbf{D}_j$ = Restore($\langle \mathbf{D}_j\rangle$) $\leftarrow (\mathbf{X}_j \times p_j\mathbf{E})$
10: End for
11: $S_0$ computes $\langle \mathbf{Y}\rangle_0 = (\mathbf{D}_1 \times \mathbf{D}_2 \times \cdots \times \mathbf{D}_N \times \langle q\mathbf{E}\rangle_0)$
12: $S_1$ computes $\langle \mathbf{Y}\rangle_1 = (\mathbf{D}_1 \times \mathbf{D}_2 \times \cdots \times \mathbf{D}_N \times \langle q\mathbf{E}\rangle_1)$
```

V. GPU-friendly privacy-preserving computing

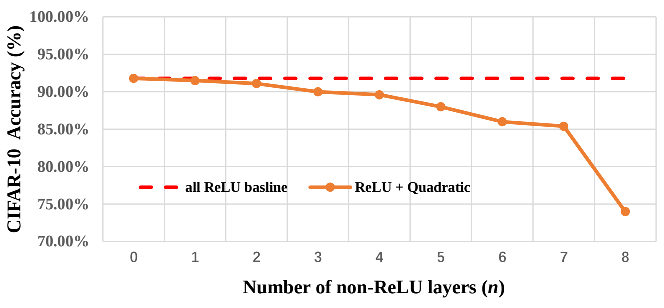
In this section, we propose a series of GPU-friendly secure protocols to compute linear and nonlinear layers in the CNN model efficiently.

A. Secure Mixed Linear Layer Protocol

Previous works [6], [7] have shown that the online latency in CNN inference is most in ReLU functions as it involves many nonlinear operations, so we use quadratic activation functions [7] to replace most of the ReLU functions without losing



(a) MNIST accuracy of AlexNet



(b) CIFAR-10 accuracy of VGG-16

Fig. 2: Comparisons of the inference accuracy between two approaches

Algorithm 2 Secure Mixed Linear Layer (SMLL) protocol

```
Input: $S_0$ holds $\langle \mathbf{X}_0\rangle_0$, $\langle \mathbf{W}_1\rangle_0$, $\langle \mathbf{B}_1\rangle_0$, $\langle \mathbf{A}_1\rangle_0$, $\langle \mathbf{B}_2\rangle_0$, $\langle k_2\rangle_0$, $\langle k_1\rangle_0$, $\langle k_0\rangle_0$, $S_1$ holds $\langle \mathbf{X}_0\rangle_1$, $\langle \mathbf{W}_1\rangle_1$, $\langle \mathbf{B}_1\rangle_1$, $\langle \mathbf{A}_1\rangle_1$, $\langle \mathbf{B}_2\rangle_1$, $\langle k_2\rangle_1$, $\langle k_1\rangle_1$, $\langle k_0\rangle_1$.
Output: $S_0$ outputs $\langle \mathbf{Y}\rangle_0 = \langle k_2[\mathbf{A}_1 \times (\mathbf{X} \otimes \mathbf{W}_1 + \mathbf{B}_1) + \mathbf{B}_2]^2\rangle_0 + \langle k_1[\mathbf{A}_1 \times (\mathbf{X}\otimes\mathbf{W}_1+\mathbf{B}_1)+\mathbf{B}_2]\rangle_0 + \langle k_0\rangle_0$, $S_1$ outputs $\langle \mathbf{Y}\rangle_1 = \langle k_2[\mathbf{A}_1 \times (\mathbf{X} \otimes \mathbf{W}_1 + \mathbf{B}_1) + \mathbf{B}_2]^2\rangle_1 + \langle k_1[\mathbf{A}_1 \times (\mathbf{X} \otimes \mathbf{W}_1 + \mathbf{B}_1) + \mathbf{B}_2]\rangle_1 + \langle k_0\rangle_1$
1: For $i \in \{0,1\}$, $S_i$ computes $\langle \mathbf{X}_{out}\rangle_i = \langle \mathbf{X}_0\rangle_i \otimes \langle \mathbf{W}_1\rangle_i$
2: For $i \in \{0,1\}$, $S_i$ computes $\langle \mathbf{K}_2\rangle_i = \langle k_2\rangle_i\mathbf{E}$, $\langle \mathbf{K}_1\rangle_i = \langle k_1\rangle_i\mathbf{E}$, $\langle \mathbf{K}_0\rangle_i = \langle k_0\rangle_i\mathbf{E}$
3: $S_0$ and $S_1$ (do in parallel):
4:     $\langle \mathbf{Z}_0\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_2\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{X}_{out}\rangle, \langle \mathbf{X}_{out}\rangle)$
5:     $\langle \mathbf{Z}_1\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_2\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{B}_1\rangle, \langle \mathbf{X}_{out}\rangle) \cdot 2$
6:     $\langle \mathbf{Z}_2\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_2\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{B}_2\rangle, \langle \mathbf{X}_{out}\rangle) \cdot 2$
7:     $\langle \mathbf{Z}_3\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_1\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{X}_{out}\rangle)$
8:     $\langle \mathbf{Z}_4\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_2\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{B}_1\rangle, \langle \mathbf{B}_1\rangle)$
9:     $\langle \mathbf{Z}_5\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_2\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{B}_2\rangle, \langle \mathbf{B}_1\rangle) \cdot 2$
10:    $\langle \mathbf{Z}_6\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_2\rangle, \langle \mathbf{B}_2\rangle, \langle \mathbf{B}_2\rangle)$
11:    $\langle \mathbf{Z}_7\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_1\rangle, \langle \mathbf{A}_1\rangle, \langle \mathbf{B}_1\rangle)$
12:    $\langle \mathbf{Z}_8\rangle \leftarrow \mathsf{SecCMul}(\langle \mathbf{K}_1\rangle, \langle \mathbf{B}_2\rangle)$
13: For $i \in \{0,1\}$, $S_i$ computes $\langle \mathbf{Y}\rangle_i \leftarrow \langle \mathbf{Z}_0\rangle_i + \langle \mathbf{Z}_1\rangle_i + \cdots + \langle \mathbf{Z}_8\rangle_i + \langle \mathbf{K}_0\rangle_i$
```

much accuracy. Fig.2 plots the accuracy against the varying number (n) of non-ReLU layers for two CNN architectures, respectively. We observed that we can replace part of the ReLU layer on the basis of ensuring accuracy.

We first propose **Secure Mixed Linear Layer Computation** protocol (SMLL), which combines one convolutional layer, one batch normalization layer, and one quadratic activation layer for calculation. It reduces the communication rounds and calculation times. Recall that the convolutional layer is $\mathbf{Y} = \mathbf{X} \otimes \mathbf{W}_1 + \mathbf{B}_1$, the BN layer is $\mathbf{Y} = \mathbf{A}_1 \times \mathbf{X} + \mathbf{B}_2$, and the quadratic activation layer is $\mathbf{Y} = k_2\mathbf{X}^2 + k_1\mathbf{X} + k_0$. Such that the combined equation is

$$\begin{aligned}\mathbf{Y} =& k_2[\mathbf{A}_1 \times (\mathbf{X} \otimes \mathbf{W}_1 + \mathbf{B}_1) + \mathbf{B}_2]^2 \\ &+ k_1[\mathbf{A}_1 \times (\mathbf{X} \otimes \mathbf{W}_1 + \mathbf{B}_1) + \mathbf{B}_2] + k_0\end{aligned}$$

Algorithm 3 Secure Comparison (SecCmp) protocol

```
Input: $S_0$ holds $\langle x\rangle_0$, $S_1$ holds $\langle x\rangle_1$
Output: $S_0$ outputs $\langle z\rangle_0 = \langle x \le 0\rangle_0$,
        $S_1$ outputs $\langle z\rangle_1 = \langle x \le 0\rangle_1$
1: On the offline phase
2: $TD$ generates $\langle\gamma\rangle_0$ and $\langle\gamma\rangle_1$, where $\gamma = \langle\gamma\rangle_0 + \langle\gamma\rangle_1$
3: $TD$ obtains $k_0, k_1 \leftarrow \mathsf{KeyGen}(1^\lambda, f_\gamma)$
4: $TD$ sends $k_0$, $\langle\gamma\rangle_0$ to $S_0$ and sends $k_1$, $\langle\gamma\rangle_1$ to $S_1$.
5: On the online phase
6: For $i \in \{0,1\}$, $S_i$ computes $\langle x\rangle_i + \langle\gamma\rangle_i$
7: For $i \in \{0,1\}$, $S_i$ sends $\langle x\rangle_i + \langle\gamma\rangle_i$ to $S_{1-i}$ and obtain $x + \gamma$
8: For $i \in \{0,1\}$, $S_i$ obtains $\langle z\rangle_i \leftarrow \mathsf{Eval}(i, k_i, x + \gamma)$
```

We assume that the size of $\mathbf{X}$ is $m \times m$, $\mathbf{W}_1$ is $f \times f$, such that the size of $(\mathbf{B}_1, \mathbf{A}_1, \mathbf{B}_2, (\mathbf{X}\otimes\mathbf{W}_1))$ is $q \times q$ ($q = m - f + 1$). And the combined equation can be deformed to

$$\begin{aligned}\mathbf{Y} =& k_2\mathbf{A}_1^2 \times \underline{(\mathbf{X} \otimes \mathbf{W}_1)^2} \\ &+ (2k_2\mathbf{A}_1^2 \times \mathbf{B}_1 + 2k_2\mathbf{A}_1 \times \mathbf{B}_2 + k_1\mathbf{A}_1) \times \underline{(\mathbf{X} \otimes \mathbf{W}_1)} \\ &+ k_2\mathbf{A}_1^2 \times \mathbf{B}_1^2 + 2k_2\mathbf{A}_1 \times \mathbf{B}_2 \times \mathbf{B}_1 \\ &+ k_2\mathbf{B}_2^2 + k_1\mathbf{A}_1 \times \mathbf{B}_1 + k_1\mathbf{B}_2 + k_0\end{aligned}$$

The underline part of the equation can be calculated by SecConv [6], and the rest can be calculated by the secure continuous multiplication protocol (SecCMul). SecCMul aims to compute $\langle \mathbf{Y}\rangle \leftarrow f(\langle \mathbf{X}_1\rangle, \langle \mathbf{X}_2\rangle, \cdots, \langle \mathbf{X}_n\rangle)$, where $\mathbf{Y} = \mathbf{X}_1 \times \mathbf{X}_2 \times \cdots \times \mathbf{X}_n$.

Assume that the S_0 holds $\langle \mathbf{X}_0\rangle_0, \langle \mathbf{X}_1\rangle_0, \cdots, \langle \mathbf{X}_N\rangle_0$, S_1 holds $\langle \mathbf{X}_0\rangle_1, \langle \mathbf{X}_1\rangle_1, \cdots, \langle \mathbf{X}_N\rangle_1$. **On the offline phase**, The TD generates auxiliary random numbers, and send $\{(\langle a_j\rangle_i, \langle p_j\rangle_i, \langle a_jp_j\rangle_i)_{j\in[1,N]}, \langle q\rangle_i\}$ to S_i respectively, where $i \in \{0,1\}$. **On the online phase**, S_i computes $\langle \mathbf{X}_j\rangle_i - \langle a_j\mathbf{E}\rangle_i$ and obtain $(\mathbf{X} - a_j\mathbf{E})$. Then S_i computes $\langle \mathbf{D}_j\rangle_i = (\langle \mathbf{X}_j - a_j\mathbf{E}\rangle_i \times \langle p_j\mathbf{E}\rangle_i + \langle (a_jp_j)\mathbf{E}\rangle_i)$ and obtain $\mathbf{D}_j \leftarrow \mathsf{ReStore}(\langle \mathbf{D}_j\rangle)$. Finally, S_i obtains $\langle \mathbf{Y}\rangle_i = (\mathbf{D}_1 \times \mathbf{D}_2 \times \cdots \times \mathbf{D}_n \times \langle q\mathbf{E}\rangle_i)$. The specific calculating process of the SecCMul protocol is shown in Algorithm 1.

TABLE II: Running time and communication cost of SMLL

Parameters			System	Time(ms)	Comm.	
Input $C \times H \times W$	Kernel $N \times K \times K$	Stride& Padding			Rounds	Volume(KB)
$16 \times 32 \times 32$	$16 \times 3 \times 3$	$(1,1)$	GFS-CNN	**15.02**	3	128
			Delphi [7]	$19.2(14.05 + 2.51 + 2.64)$	5	104
$32 \times 16 \times 16$	$32 \times 3 \times 3$	$(1,1)$	GFS-CNN	**12.24**	3	64
			Delphi [7]	$14.92(11.85 + 1.34 + 1.73)$	5	52
$64 \times 8 \times 8$	$64 \times 3 \times 3$	$(1,1)$	GFS-CNN	**9.16**	3	32
			Delphi [7]	$10.35(8.7 + 0.76 + 0.89)$	5	26

TABLE III: Running time and communication cost of SReLU and MSB-ReLU

ReLU	Time(ms)	Comm. Rounds	Comm. Volume(KB)
SReLU	**10.9**	**2**	**4**
MSB-ReLU [5]	13.2	7	364

TABLE IV: Performance Evaluation On CNNs

Network	Test Set	System	Acc	Time(s)	Comm. Rounds	Comm. Volume(KB)
AlexNet	MNIST	Delphi(GPU) [7]	96.0%	1.59	34	190.59
		GFS-CNN(CPU)	95.6%	2.24	**20**	201.51
		GFS-CNN(GPU)	95.6%	**1.44**	**20**	201.51
VGG-16	CIFAR-10	Delphi(GPU) [7]	85.6%	3.86	119	8092.5
		GFS-CNN(CPU)	85.4%	5.26	**53**	**1537**
		GFS-CNN(GPU)	85.4%	**3.22**	**53**	**1537**

Algorithm 4 Secure ReLU (SReLU) Protocol

```
Input: $S_0$ holds $\langle x\rangle_0$, $S_1$ holds $\langle x\rangle_1$
Output: $S_0$ outputs $\langle y\rangle_0 = \langle x > 0\rangle_0 \cdot \langle x\rangle_0$, $S_1$ outputs
        $\langle y\rangle_1 = \langle x > 0\rangle_1 \cdot \langle x\rangle_1$
1: $S_0$ and $S_1$ computes $\langle z\rangle = 1 - \mathsf{SecCmp}(\langle x\rangle)$
2: $S_0$ and $S_1$ computes $\langle y\rangle = \langle zx\rangle \leftarrow \mathsf{SecMul}(\langle z\rangle, \langle x\rangle)$
```

B. Secure ReLU Protocol

Previous work [6], [7] and our experiments show that replacing the ReLU layer with a quadratic activation function completely will decrease inference accuracy and make the inference results unavailable. This is a composite loss of accuracy due to the increase in the amount of calculation of the square term. Therefore, we propose the **Secure ReLU Protocol** (SReLU), which replaces the ReLU function using function secret sharing in CNNs.

For the comparison operations in ReLU functions, we propose the **Secure Comparison** (SecCmp) protocol based on FSS, which aims to compute $\langle z\rangle \leftarrow \langle x\rangle \leq \langle y\rangle$. We choose the latest FSS-based comparison construction [8], [17], [19], named distributed comparison function (DCF), allowing secure comparison with low communication costs. Specifically, the DCF can perform secure computation of the function $f_{\alpha,\beta}(x)$, in which the output is β if x is less than or equal to α, and 0 if x is greater than α. In the two-server scenario, the *TD* generates two DCF keys (k_0, k_1) from KeyGen function and sends them to S_0 and S_1 respectively. S_0 and S_1 then obtain the secret-shared output by evaluate the $\mathsf{EVAL}(i, k_i, x)$ function using the DCF key and the input x.

However, as the DCF evaluation mandates both parties to provide specific inputs, evaluating DCF on a private (additive-secret-shared) value is not feasible through the above procedure. To address this challenge, we maintain the secrecy of the private value x by revealing its masked version. To be more specific, for a given target function $f_{0,1}$ (in which the output is 1 if $x \leq 0$ and 0 if $x > 0$), we generate a set of DCF keys for the offset function $f_{\gamma,1}$, in which $f_{0,1}(x) = f_{\gamma,1}(x+\gamma)$. Here, γ denotes a random value and is divided into two additive secret shares, $\gamma = \langle\gamma\rangle_0 + \langle\gamma\rangle_1$. Next, S_i sends the masked share $\langle x\rangle_i + \langle\gamma\rangle_i$ to S_{1-i}, and the two servers cooperatively obtain $x + \gamma$ without leaking x. The S_0 and S_1 securely perform the offset FSS evaluation functions on the input $x+\gamma$ and obtain the secret-shared output of $f_{\gamma,1}(x+\gamma)$, where $f_{\gamma,1}(x+\gamma) = f_{0,1}(x)$. Based on the offset function, we introduce the calculation process of SecCmp protocol in Algorithm 3.

Given the above SecCmp protocol and the secret-shared input, the $\mathsf{SReLU}(\langle x\rangle)$ protocol is shown in Algorithm 4. Assume S_i holds $\langle x\rangle_i$ respectively, and $\alpha = 0, \beta = 1$. Then S_i can obtain $\langle x\rangle_i$ if $x > 0$ and obtain $\langle 0\rangle_i$ if $x \leq 0$.

VI. EXPERIMENT

A. Implementation and Experimental Setup

Our system is implemented in Python3.8 with the Pytorch library for matrix operations. All our secure MPC protocols are multi-threaded. We use two separate servers to act as the server S_0 and S_1, which run Ubuntu 22.04 and are equipped with a 2.5GHz Intel Xeon processor with four cores, 128GB of RAM, and an Nvidia 2080ti GPU. We put the two servers in the practical WAN, which communication bandwidth is 1000 Mbps.

B. *Microbenchmarks*

We provide microbenchmarks of GFS-CNN performance on SMLL protocol and SReLU protocol, comparing both with previous works.

SMLL performance. The complexity of the SMLL is determined by the input dimensions, the size and number of convolution kernels, and the padding and stride. In Table II, we evaluate the costs of SMLL and compare it with Delphi. The experimental data show that our online time is over $1.2\times$ faster than Delphi's, and our online communication rounds are only three times. However, our communication volume is higher than Delphi due to the high parallelism of our SMLL protocol.

SReLU performance. We evaluated the SReLU protocol on the $(16, 32, 32)$ feature images and compared it with the MSB-based Relu protocol. As shown in Table III, the SReLU protocol is $1.2\times$ faster than the MSB-ReLU protocol [5]. And SReLU protocol requires only one round of communication and 4KB of communication volumes, which is 83.3% and 98% lower than MSB-ReLU, respectively.

C. *Performance Evaluation on CNNs*

By demonstrating the SMLL and SMReLU protocols of GFS-CNN and comparing them with previous work, we significantly reduce the communication rounds and prove the low latency of GFS-CNN. Next, we will provide a detailed evaluation of AlexNet and VGG-16 using the test sets MNIST and CIFAR-10, respectively. Table IV shows the performance of GFS-CNN and the comparisons with Delphi. We find that, under the same level of inference accuracy, our work reduces the online inference time for evaluating AlexNet on MNIST and VGG-16 on CIFAR-10 by 10.2% and 16.4%, respectively. And on this basis, our communication rounds are reduced by 41.1% and 55.4% on AlexNet and VGG-16, respectively. However, our communication volume is higher on Alexnet but much smaller on VGG-16 than Delphi. This is because we replace all the ReLU layers on Alexnet with the quadric activation function, while only part of the ReLU function is replaced on VGG-16, and the rest uses the DCF function as the activation layer.

VII. CONCLUSION

In this paper, we propose GFS-CNN, a GPU-friendly privacy-preserving computing framework for CNN inference, which provides privacy protection for users' private data and model providers' privacy models. Technically, we propose a parallelized protocol SMLL, reducing communication rounds to only three times in the mixed linear layers. In addition,the FSS-based secure ReLU protocol only involves two communication rounds, thereby better exploiting GPU acceleration. Compared with the state-of-the-art solution Delphi, GFS-CNN reduces the online inference time by $10.2\% - 16.4\%$, and the number of communication rounds by $41.1\% - 55.4\%$.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3101100; in part by the National Natural Science Foundation of China under Grant 62220106004; in part by the Major Research Plan of the National Natural Science Foundation of China under Grant 92267204; in part by the Key Research and Development Program of Shaanxi under Grant 2022KXJ-093 and Grant 2021ZDLGY07-05; in part by the Natural Science Basis Research Plan in Shaanxi Province of China under Grant 2022JM-338; in part by the Shandong Provincial Natural Science Foundation under Grant ZR2021LZH006; and in part by the Fundamental Research Funds for the Central Universities under Grant XJSJ23040.

REFERENCES

[1] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. E. Lauter, and F. Koushanfar, "Xonn: Xnor-based oblivious deep neural network inference." in *USENIX Security Symposium*, 2019, pp. 1501–1518.

[2] S. Wagh, D. Gupta, and N. Chandran, "Securenn: 3-party secure computation for neural network training." *Proc. Priv. Enhancing Technol.*, vol. 2019, no. 3, pp. 26–49, 2019.

[3] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 35–52.

[4] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation." in *NDSS*, 2015.

[5] X. Liu, Y. Zheng, X. Yuan, and X. Yi, "Securely outsourcing neural network inference to the cloud with lightweight techniques," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[6] K. Cheng, J. Fu, Y. Shen, H. Gao, N. Xi, Z. Zhang, and X. Zhu, "Manto: A practical and secure inference service of convolutional neural networks for iot," *IEEE Internet of Things Journal*, 2023.

[7] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: a cryptographic inference system for neural networks," in *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, 2020, pp. 27–30.

[8] M. Zhou, Y. Zheng, S. Wang, Z. Hua, H. Huang, Y. Gao, and X. Jia, "Ppta: A location privacy-preserving and flexible task assignment service for spatial crowdsourcing," *Computer Networks*, vol. 224, p. 109600, 2023.

[9] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.

[10] Z. Xia, Q. Gu, W. Zhou, L. Xiong, J. Weng, and N. Xiong, "Str: Secure computation on additive shares using the share-transform-reveal strategy," *IEEE Transactions on Computers*, 2021.

[11] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "F: Honest-majority maliciously secure framework for private deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 1, pp. 188–208, 2021.

[12] X. Mu, Y. Shen, K. Cheng, X. Geng, J. Fu, T. Zhang, and Z. Zhang, "Fedproc: Prototypical contrastive federated learning on non-iid data," *Future Generation Computer Systems*, vol. 143, pp. 93–104, 2023.

[13] S. Tan, B. Knott, Y. Tian, and D. J. Wu, "Cryptgpu: Fast privacy-preserving machine learning on the gpu," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1021–1038.

[14] J.-L. Watson, S. Wagh, and R. A. Popa, "Piranha: A {GPU} platform for secure computation," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 827–844.

[15] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology—CRYPTO'91: Proceedings 11*. Springer, 1992, pp. 420–432.

[16] E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing," in *Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*. Springer, 2015, pp. 337–367.

[17] Boyle, Elette and Gilboa, Niv and Ishai, Yuval, "Secure computation with preprocessing via function secret sharing," in *Theory of Cryptography: 17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part I 17*. Springer, 2019, pp. 341–371.

[18] E. Boyle, N. Chandran, N. Gilboa, D. Gupta, Y. Ishai, N. Kumar, and M. Rathee, "Function secret sharing for mixed-mode and fixed-point secure computation," in *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part II*. Springer, 2021, pp. 871–900.

[19] S. Wagh, "Pika: Secure computation using function secret sharing over rings," *Cryptology ePrint Archive*, 2022.

[20] J. Park, M. J. Kim, W. Jung, and J. H. Ahn, "Aespa: Accuracy preserving low-degree polynomial activation for fast private inference," *arXiv preprint arXiv:2201.06699*, 2022.

[21] T. Zhang, A. Song, X. Dong, Y. Shen, and J. Ma, "Privacy-preserving asynchronous grouped federated learning for iot," *IEEE Internet of Things Journal*, vol. 9, no. 7, pp. 5511–5523, 2021.

Chao Guo received his B.E. degree from Shandong University of Science and Technology, Qingdao, China, in 2021. He is currently working toward the M.S. degree in computer science at the School of Computer Science and Technology, Xidian University, Xi'an, China. His research interests include cloud computing security and privacy protection.

Ke Cheng is a lecturer in the School of Computer Science and Technology at Xidian University. He received his B.S. and M.S. degrees from Anhui University, Hefei, China, in 2015 and 2018, respectively. He received his Ph.D. degree in computer science and technology from Xidian University in 2022. His research interests include cloud computing security, data security, and privacy protection.

Jiaxuan Fu received his B.S. degree from Xidian University, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xidian University, China. His research interests include IoT data security and machine learning security.

Ruolu Fan received her B.S. degree from Xi'an University of Posts & Telecommunications, China, in 2013. She is an engineer at China Telecom Corporation Limited, Shaanxi Branch, China. Her research interests include data management and data analysis.

Zhao Chang received a BS degree in computer science and technology from Peking University in 2013, and a Ph.D. degree in computing from University of Utah in 2021. He currently works as an associate professor in the School of Computer Science and Technology at Xidian University. His research interests focus on security and privacy issues in large scale data management.

Zhiwei Zhang received the Ph.D. degree in cryptography from Xidian University, Xi'an, Shaanxi, China, in 2019. He is currently an Associate Professor at the School of Computer Science and Technology, Xidian University. His research interest includes data security management, data storage security, and data location verification in cloud computing.

Anxiao Song received the B.E. degree in software engineering from the Henan University of Technology, Zhengzhou, China, in 2019. He is currently pursuing the M.E. degree with the School of Computer Science and Technology, Xidian University, Xi'an, China. His research interests include network security and privacy protection.