5-2020

A lightweight privacy-preserving CNN feature extraction framework for mobile sensing

Kai HUANG
National University of Defense Technology

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Shaojing FU
National University of Defense Technology

Deke GUO
National University of Defense Technology

Ming XU
National University of Defense Technology

Published in IEEE Transactions on Dependable and Secure Computing, 2020, 18 (3), 1441-1455. DOI: 10.1109/TDSC.2019.2913362

A Lightweight Privacy-Preserving CNN Feature Extraction Framework for Mobile Sensing

Kai Huang, Ximeng Liu, *Member, IEEE*, Shaojing Fu, *Member, IEEE*, Deke Guo, *Senior Member, IEEE*, and Ming Xu, *Member, IEEE*

Abstract—The proliferation of various mobile devices equipped with cameras results in an exponential growth of the amount of images. Recent advances in the deep learning with convolutional neural networks (CNN) have made CNN feature extraction become an effective way to process these images. However, it is still a challenging task to deploy the CNN model on the mobile sensors, which are typically resource-constrained in terms of the storage space, the computing capacity, and the battery life. Although cloud computing has become a popular solution, data security and response latency are always the key issues. Therefore, in this paper, we propose a novel lightweight framework for privacy-preserving CNN feature extraction for mobile sensing based on edge computing. To get the most out of the benefits of CNN with limited physical resources on the mobile sensors, we design a series of secure interaction protocols and utilize two edge servers to collaboratively perform the CNN feature extraction. The proposed scheme allows us to significantly reduce the latency and the overhead of the end devices while preserving privacy. Through theoretical analysis and empirical experiments, we demonstrate the security, effectiveness, and efficiency of our scheme.

Index Terms—Privacy-preserving, CNN, feature extraction, mobile sensing

1 Introduction

The mobile sensing has received a lot of attention in recent years and transformed our lives as an efficient sensing paradigm. This is greatly attributed to the popularity of smartphones and other mobile devices equipped with a variety of sensors, such as accelerometers, gyroscopes, microphones, and cameras. These sensors can be used to collect data from our surrounding environment and provide valuable information for various applications. Particularly, the camera is the most pervasive one among them. Massive images acquired by the cameras have been utilized to support a large number of visual applications, such as object recognition, variety identification, scene understanding, and environment modeling. Specifically, with the recent advances in the deep learning with convolutional neural networks (CNN), the accuracy of these applications can be vastly improved. Literatures [1], [2], [3] have shown that the features extracted by CNN significantly outperform traditional hand-crafted features and other highly tuned state-of-the-art methods. It is suggested that features obtained from CNN should be the primary candidate in essentially any visual tasks [1].

This motivates many researchers to work on putting CNN feature extraction on mobile devices. However, it still has a long way to go. The explosive growth in the volume of images and the complicated CNNs impose significant challenges on the storage and processing capacity of resource-constrained mobile sensors. With the cloud computing being more widely utilized, more and more users choose to outsource their huge amount of image data and computation-intensive visual tasks to the cloud data centers. Unfortunately, the cloud data centers are usually far away from the mobile users, which usually connect to the internet via the wireless network. For the massive image data, communication between the mobile users and the cloud demands substantial bandwidth and incurs unpredictable delays, which will result in a degraded quality of experience [4].

Edge computing has been proposed as a method of optimizing cloud computing architecture by performing data processing at the edge of the network and close to the source of the data [5]. This can significantly reduce the bandwidth consumption of delivering large amounts of data to the data center and reduce the latency between the mobile device and the data center. Nevertheless, security remains a challenge in this architecture, since the images usually contain confidential or sensitive information. The most intuitive way is to use a homomorphic encryption scheme, which allows computation on encrypted data, to protect the privacy of the images before outsourcing to the edge. CryptoNets [6] proposed by Microsoft was the first work to show how to apply CNN to encrypted data based on homomorphic encryption. Following that, literatures [7] and [8] made some

- *K. Huang and S. Fu are with the College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China, and also with the State Key Laboratory of Cryptology, Beijing 1816670, China. E-mail: kai.huang@nudt.edu.cn, shaojing1984@163.com.*
- *M. Xu is with the College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China. E-mail: xuming@nudt.edu.cn.*
- *X. Liu is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian 350108, China, and also with the Key Lab of Information Security of Network Systems, Fuzhou University, Fuzhou, Fujian Province 350108, China. E-mail: snbnix@gmail.com.*
- *D. Guo is with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, Hunan 410073, China. E-mail: dekeguo@nudt.edu.cn.*

improvements and obtained better accuracy. To support homomorphic operations, the non-linear functions in their networks need to be approximated by low degree polynomials, which essentially reduces the accuracy of their schemes. Moreover, current homomorphic encryption techniques are too computation-intensive. Recently, Liu et al. [9] proposed MiniONN for transforming an existing neural network to an oblivious neural network supporting privacy-preserving predictions. Although they put the homomorphic encryption in the offline phase, the garbled circuits are needed in the online phase. Generating and storing such garbled circuits would be a challenging task. Riazi et al. [10] proposed a mixed protocol framework. They claimed that they can choose the most efficient protocol for different operations. However, they suffer from the complicated data structures and conversion of protocols. Therefore, while providing security, existing solutions still require significant computation and storage on the end devices. In addition, they incur high communication cost between the end devices and the data center. As a consequence, they are impractical to be used in the setting of mobile sensing.

Accordingly, it is important to design a novel lightweight framework to address the issue of privacy-preserving CNN feature extraction for mobile sensing. Our aim is to greatly reduce the latency and the overhead of the end devices while preserving the accuracy of the neural networks and the privacy of the data. To this end, we perform the task of CNN feature extraction at the edge close to the mobile sensors. To protect the privacy of the data, we utilize the additive secret-sharing techniques and design a series of secure interaction protocols corresponding to the various CNN layers. In particular, two independent edge servers and a trusted third party are utilized in our novel architecture. The trusted third party is responsible for generating random values in the offline phase. In the online phase, the two edge servers perform the CNN feature extraction over the received encrypted images via the series of secure interaction protocols. The user can obtain the CNN features from the encrypted results generated by the two edge servers.

Our main contributions can be summarized as follows:

- We propose a novel lightweight framework for privacy-preserving CNN feature extraction in the setting of mobile sensing. By utilizing the secret sharing-based encryption technique and moving the computation-intensive work to the edge servers, we can greatly reduce the overhead on the mobile sensors and the end users. Meanwhile, it requires no interaction between the end devices and the servers.
- We develop a series of building blocks that don't rely on the computation-intensive cryptographic primitives like homomorphic encryption and garbled circuits. Thus, our solution can greatly improve the overall performance with a minimal amount of computation and communication overheads.
- We design a privacy-preserving CNN feature extraction scheme that does not need to make any approximation for recommended common CNN layers as previous schemes based on homomorphic encryption. Thus, we can maintain the accuracy of the CNN model while preserving privacy.
- We conduct comprehensive theoretical analysis and empirical experiments to measure the performance of our scheme. The results indicate that our scheme outperforms previous work in terms of the runtime and the communication overhead while preserving the accuracy of the neural networks and the privacy of the data.

The remainder of this paper is organized as follows. We introduce the related work in Section 2 and the necessary preliminaries in Section 3. Then we formulate the problem and give the system model and security model in Section 4. The building blocks that support efficient secure computation based on secret sharing techniques are given in Section 5. On the basis of that, we propose the details of our scheme in Section 6. We give the security analysis and experimental results in Sections 7 and 8. Finally, we conclude our work in Section 9.

2 Related Work

Feature extraction is a fundamental problem in various visual tasks such as image classification, object recognition, and image retrieval. Recently, to preserve the privacy of the images, numerous schemes have been proposed to extract traditional features over the encrypted images. These include privacy-preserving global feature detection [11], shape-based feature extraction [12], SIFT [13], [14], [15], SURF [16], [17], and HOG [18]. In [19], color and texture information were used to support privacy-preserving content-based image retrieval in the cloud. Lu et al. [20] used the global color histogram in the HSV space as image features and studied the problem of confidentiality-preserving image retrieval. Xia et al. [21] proposed a privacy-preserving and copy-deterrence content-based image retrieval scheme by using the local features. Yuan et al. [22] adopted the Bag-of-Words model to aggregate the local features and proposed a privacy-preserving social discovery architecture based on encrypted images. In addition, there are some other schemes using the finger code or other biometric data to design privacy-preserving biometric authentication and identification [23], [24]. However, these traditional hand-crafted features have their inherent defects, since they often rely on expert knowledge and require expensive human labor.

To the best of our knowledge, the first related work for applying the neural network to the encrypted data to make encrypted predictions is CryptoNets [6], [7], [8]. The recent work CryptoDL [25] considers both training and inference phases. However, the main ingredient of both CryptoNets and CryptoDL are homomorphic encryption. The accuracy is essentially reduced and the computation overhead is too high to be practical. Therefore, they are not appropriate in the setting of mobile sensing. SecureML [26] seems to be the first work based on SMC techniques for privacy-preserving training and prediction. However, they are inefficient and can only support very simple networks and very few hidden layers. Recently, various mixed-protocol frameworks have been proposed, like MiniONN [9], DeepSecure [27], Chameleon [10], GAZELLE [28], and ABY^3 [29]. However, they are not easy to expand since they suffer from computation-intensive cryptographic primitives and cannot fully utilize the efficient parallel data structure of CNN. An

orthogonal line of work focused on privacy-preserving learning. [30] and [31] are both based on homomorphic encryption. However, they only supported low-depth network and had high computation and communication complexity. In [32], [33], [34], the authors developed new algorithmic techniques for learning within the framework of differential privacy. They tried to make a trade-off between accuracy and privacy, however, the server has full access to the data in plaintext and the privacy of the original image content was not protected.

3 Preliminaries

In this section, we briefly review the CNN features and the cryptographic tools we use in our scheme.

3.1 CNN Features

In this work, we focus on the features extracted from the publicly available pre-trained CNN.

CNN is a sequence of layers that transform an input layer to an output layer. Each layer is made up of a set of neurons. Each neuron of one layer (except the input layer) is the output of a function applied on the neurons of the previous layer, i.e., $y = f(x)$. Several commonly used layers are the fully connected layer, the convolutional layer, the activation layer, and the pooling layer.

- *Fully connected layer.* Each neuron of this layer is connected to each neuron of the previous layer. Let w_{jk} denote the weight on the connection between the kth neuron of the previous layer and the jth neuron of the current layer. Let b_j denote the bias of the jth neuron of the current layer. Then, the output of the jth neuron of this layer will be $y_j = \sum_k w_{jk} x_k + b_j$.
- *Convolutional layer.* The neurons in this layer share the same weights and bias, which are often said to define a kernel or filter. Let the size of the filter be $n \times n$, then each neuron in this layer will be connected to a $n \times n$ region of the neurons in the previous layer. Correspondingly, for the (j,k)th neuron, the output will be $y_{j,k} = \sum_{l=0}^{n-1} \sum_{m=0}^{n-1} w_{l,m} x_{j+l,k+m} + b$.
- *Activation layer.* Activation layers apply elementwise non-linearity and are usually used immediately after convolutional or fully connected layers. Common activation functions include sigmoid, tanh, and the rectified linear unit (ReLU), in which ReLU has become the default recommendation in modern neural networks. ReLU is defined by the activation function $f(x) = max(x, 0)$.
- *Pooling layer.* The pooling layer partitions the neurons of the previous layer into a set of non-overlapping rectangles and performs a downsampling operation on each sub-area to obtain the value of one neuron in the current layer. The most common pooling functions are the max-pooling that outputs the maximum value within the sub-area and the average-pooling that outputs the average of the values of the sub-area.

A convolutional neural network usually stacks a sequence of convolutional (Conv)-ReLU layers, following with the pooling layers (Pool), and repeats this pattern until the image has been merged spatially to a small size. At some point, it is common to transit to fully-connected layers (FC). For clarity, the most common CNN architecture follows the pattern: Input $\rightarrow$ [[Conv $\rightarrow$ ReLU] $\times n \rightarrow$ Pool?] $\times m \rightarrow$ [FC $\rightarrow$ ReLU] $\times l \rightarrow$ FC, where the $\times$ indicates repetition, and the question mark ? indicates an optional layer. In addition, $n \geq 0$ (and usually $n \leq 3$), $m \geq 0$, $l \geq 0$ (and usually $l < 3$). We can take the responses from one of the network's layers as our CNN feature vector, which can be used for different visual tasks in combination with some other techniques.

3.2 Secure Multiparty Computation

Secure Multiparty Computation (SMC) is an interactive computation paradigm that enables multiple parties to collaboratively evaluate a function over their respective inputs while keeping those inputs private.

SMC was originally introduced for the so-called Millionaire Problem in the two-party setting (2PC) by Yao [35]. Since it turns out that information theoretic security would not be possible for 2PC [36], most solutions are based on cryptographic tools like homomorphic encryption (HE) or garbled circuits (GC). However, HE schemes are computationally expensive since they require relatively expensive public-key operations in the online phase. Although GC allows to pre-compute the expensive operations, it requires to generate a garbled circuit for evaluating a function. Generating and storing such garbled circuits would be challenging for large-scale problems [37].

Following that, SMC was generalized to the multi-party setting (MPC), which works quite differently from the secure two-party computation. Many schemes for secure MPC have been proposed, like VIFF [36], FairplayMP [38], Sharemind [39], SPDZ [40], and so on. They make use of secret sharing techniques that were introduced by Shamir [41] and Blakley [42]. Particularly, MPC with an honest majority could be implemented with secret sharing alone. Two commonly used secret sharing schemes are Shamir secret sharing and additive secret sharing. Compared with the secure two-party computations, they are much more efficient since they don't rely upon any computationally expensive cryptographic primitives.

4 Problem Formulation

4.1 System Architecture

In this paper, we aim to address the issue of privacy-preserving CNN feature extraction for mobile sensing. Several schemes have been proposed to outsource the tasks of CNN inference to the cloud servers. However, a deep CNN model typically has a substantially sophisticated structure that consists of many layers of non-linear feature extractors. The extra latency brought by the user-cloud and the inter-cloud interaction will become unacceptable since the users and the cloud servers are usually far away from each other. Moreover, to protect the privacy of the data, they usually utilize the computation-intensive cryptographic primitives like homomorphic encryption or garbled circuits. As described previously, this will incur lots of computation and storage overheads on the end devices and high communication cost between the end devices and the data center. Therefore, we propose a novel lightweight framework based on the edge computing and most of the data processing is moved away from the centralized cloud to the edge of the network (e.g., the edge gateways or the edge servers).

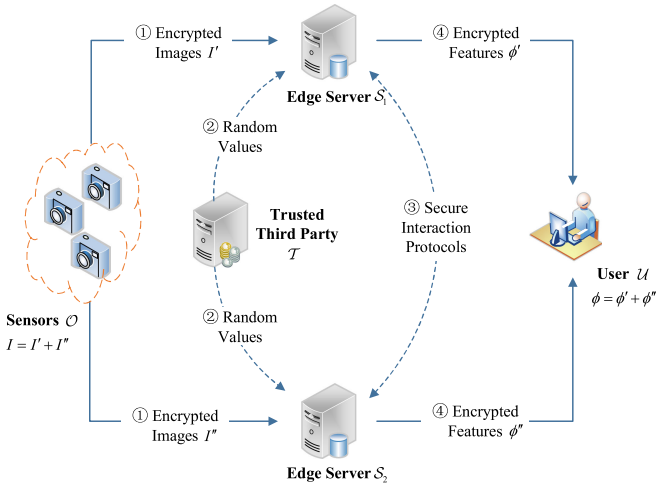


Fig. 1. System architecture.

To protect the privacy of the data and enable the CNN feature extraction over the encrypted data, we design a series of secure computation protocols based on the additive secret sharing techniques. However, different from previous secret sharing schemes [37], [39], [43], [44], [45] that are designed for at least three parties and the multi-parties play almost the same role, we put the computation-intensive work on just two edge servers. This is due to the fact that the required storage and bandwidth resources for the storage and transmission of the shares have to be at least of the size of the data times the number of shares. Correspondingly, every additional party will increase the communication traffic and the risk of being attacked [39].

As illustrated in Fig. 1, our proposed framework consists of four main entities: the mobile sensor $\mathcal{O}$, two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$, the trusted third party $\mathcal{T}$, and the user $\mathcal{U}$. The trained CNN model is publicly available and can be obtained by both $\mathcal{S}_1$ and $\mathcal{S}_2$. The user collected a large number of images with the mobile sensors and would like to process the images on the edge servers by leveraging their storage and computation resources. Let I denote the original image and ϕ denote the extracted CNN feature. To protect the privacy of the images, $\mathcal{O}$ will first encrypt the image data I by randomly splitting it into two shares I' and I'' in an element-wise manner such that I can be recovered by adding the two shares together. Then the ciphertexts I' and I'' will be distributed to the two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$. To reduce the interactions between the parties, $\mathcal{T}$ is just responsible for the generation of random values, which is simple and can be done at a light server or a client. The computation-intensive work is performed by $\mathcal{S}_1$ and $\mathcal{S}_2$. After running the CNN feature extraction in the ciphertext domain via a sequence of secure interaction protocols, $\mathcal{S}_1$ and $\mathcal{S}_2$ will return the encrypted CNN features ϕ' and ϕ'' to $\mathcal{U}$. Then, $\mathcal{U}$ can eventually recover the CNN features ϕ from their encrypted versions to perform various visual tasks such as object recognition, attribute detection, image classification, and image retrieval.

4.2 Security Model

Similar to many secure computation protocols found in the literature [14], [37], [39], [43], [46], [47], [48], we use the semi-honest (also known as passive or honest-but-curious) security model. That is, each edge server will perform the protocols exactly as specified, yet may attempt to learn as much as possible about private information of interest based on the data stored and processed on it.

In addition, the two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$ are assumed to be independent and non-colluding. It means that one edge server will not reveal more information than protocol messages to the other. This is a practical assumption since the two edge servers can be deployed on and managed by two different or even competitive service providers (e.g., AWS's Lambda@Edge and Microsoft Azure IoT Edge).

As described above, the third party $\mathcal{T}$ is just responsible for the random value generation. The work can simply be performed by a light server or a client that is controlled by the valid users. Therefore, it is reasonable to assume that the third party is always honest and can be trusted. Besides, we also assume that the sensors and valid users are always honest and secure communication channels are available between the entities.

4.3 Design Goals

Our goal is to design a secure and lightweight CNN feature extraction scheme for mobile sensing. Specifically, we have the following goals:

- *Correct*. Our primary goal is to support the CNN feature extraction over the encrypted images. The edge servers included in our architecture should be able to correctly extract the CNN features over the encrypted images. The encrypted CNN features can later be decrypted to perform other visual tasks by the user.
- *Privacy-preserving*. The main concern in our design is to protect the privacy of the images. During the process of the CNN feature extraction, the edge servers or the attackers should be prevented from learning any content of the images or the extracted features.
- *Lightweight*. Since the mobile sensors are typically resource-constrained, our design should take into full account the computation overhead on the mobile devices. Meanwhile, we should greatly reduce the communication overhead between the mobile devices and the edge servers and therefore reduce the response latency.

5 Building Blocks: Efficient Secret Sharing-based Secure Computation

To enable the secure interactions between the two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$, we design a series of efficient secure computation sub-protocols based on the additive secret sharing techniques.

Given two numbers u and v, they will be randomly split into two parts, respectively. Let $u = u_1 + u_2$ and $v = v_1 + v_2$, where the u_i's and v_i's are called the shares of u and v and will be stored in $\mathcal{S}_i$ $(i = 1, 2)$. Then $\mathcal{S}_1$ and $\mathcal{S}_2$ will work together to compute $f(u, v)$. $\mathcal{S}_1$ will output f_1 and $\mathcal{S}_2$ will output f_2, where $f_1 + f_2 = f$. Note that, to preserve the privacy of the inputs of the two edge servers, both parties won't reveal the outputs of the functions to each other in the secure computation except the secure comparison.

Input: $\mathcal{S}_1$ has $u_1, v_1 \in \mathbb{F}$; $\mathcal{S}_2$ has $u_2, v_2 \in \mathbb{F}$.
Output: $\mathcal{S}_1$ outputs f_1; $\mathcal{S}_2$ outputs f_2.

Offline Phase :

1: $\mathcal{T}$ generates random numbers $a, b \in \mathbb{F}$ and computes $c \leftarrow a \cdot b$.
2: $\mathcal{T}$ splits a, b, and c into random shares: $a = a_1 + a_2$, $b = b_1 + b_2$, and $c = c_1 + c_2$.
3: $\mathcal{T}$ sends a_i, b_i, and c_i to $\mathcal{S}_i$ $(i = 1, 2)$.

Online Phase :

4: $\mathcal{S}_1$ computes $\alpha_1 \leftarrow u_1 - a_1$, $\beta_1 \leftarrow v_1 - b_1$, and sends α_1, β_1 to $\mathcal{S}_2$.
5: $\mathcal{S}_2$ computes $\alpha_2 \leftarrow u_2 - a_2$, $\beta_2 \leftarrow v_2 - b_2$, and sends α_2, β_2 to $\mathcal{S}_1$.
6: $\mathcal{S}_1$ computes $\alpha \leftarrow \alpha_1 + \alpha_2$, $\beta \leftarrow \beta_1 + \beta_2$, and computes $f_1 \leftarrow c_1 + b_1 \cdot \alpha + a_1 \cdot \beta$.
7: $\mathcal{S}_2$ computes $\alpha \leftarrow \alpha_1 + \alpha_2$, $\beta \leftarrow \beta_1 + \beta_2$, and computes $f_2 \leftarrow c_2 + b_2 \cdot \alpha + a_2 \cdot \beta + \alpha \cdot \beta$.

Fig. 2. Secure multiplication protocol `SecMul`.

5.1 Secure Addition and Subtraction

In this protocol, we want to compute $f(u, v) = u \pm v$. Since $u \pm v = (u_1 + u_2) \pm (v_1 + v_2) = (u_1 \pm v_1) + (u_2 \pm v_2)$, it is easy to find out that $\mathcal{S}_1$ and $\mathcal{S}_2$ can perform the secure addition and subtraction locally without interaction with each other. After the computation, $\mathcal{S}_i$ $(i = 1, 2)$ will output $f_i = u_i \pm v_i$. Obviously, we have $f_1 + f_2 = u \pm v$.

Note that, when it comes to the field of $\mathbb{Z}_2$, we actually implement the secure XOR ($\oplus$) computation.

Besides, we can also execute the secure multiplication by a publicly known constant c in the same way. To compute $f(u) = c \cdot u$, $\mathcal{S}_i$ $(i = 1, 2)$ will output $f_i = c \cdot u_i$. This can also be done by $\mathcal{S}_i$ independently without interaction.

5.2 Secure Multiplication

To greatly reduce the computation and communication overheads, we design our secure multiplication protocol by incorporating the main idea of *Beaver's triplet* [49]. A secret multiplication triplet would be of the form (a, b, c), where a and b are random and private, and $c = a \cdot b$. The triplet would be shared between the two edge servers in the offline phase. Let $\mathcal{S}_i$ $(i = 1, 2)$ have the shares a_i, b_i, and c_i. In the online phase, the two edge servers would compute the product $f(u, v) = u \cdot v$ over a finite field $\mathbb{F}$. The basic idea of Beaver's triplet is to mask u and v with a and b. And then $u \cdot v$ can be expressed as the linear combination of a, b, and c, whose components will be publicly known and will not disclose any information about u and v.

As illustrated in Fig. 2, we divide our secure multiplication protocol `SecMul` into an offline phase and an online phase. Note that the offline phase is independent of the private input numbers. It can be pre-computed and stored by the trusted third party $\mathcal{T}$ before performing the actual computation. This will lead to a very efficient online phase that just takes place between the two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$.

Offline Phase. The trusted third party $\mathcal{T}$ generates the Beaver's triplet (a, b, c), where a, $b \in \mathbb{F}$ and $c = a \cdot b$, Then a, b, and c are split into random shares: $a = a_1 + a_2$, $b = b_1 + b_2$, and $c = c_1 + c_2$. The shares a_i, b_i, and c_i will be distributed to the two edge servers $\mathcal{S}_i$.

Online Phase. The two edge servers $\mathcal{S}_i$ $(i = 1, 2)$ first mask their inputs by computing $\alpha_i = u_i - a_i$ and $\beta_i = v_i - b_i$. Then they send α_i and β_i to each other and reconstruct α and β. Finally, $\mathcal{S}_1$ and $\mathcal{S}_2$ will compute and output $f_1 = c_1 + b_1 \cdot \alpha + a_1 \cdot \beta$ and $f_2 = c_2 + b_2 \cdot \alpha + a_2 \cdot \beta + \alpha \cdot \beta$, respectively.

Similarly, when it comes to the field of $\mathbb{Z}_2$, we actually implement the secure AND ($\wedge$) computation.

5.3 Secure Comparison

The secure comparison is one of the most fundamental building blocks in our scheme. Given two numbers $u, v \in \mathbb{R}$, we want to compute $f(u, v) = (u < v)$ and $f \in \{0, 1\}$, where $f = 1$ if and only if $u < v$.

We first design a protocol for secure comparison of a number and zero, which is actually to determine the sign of the number. It could then naturally expand to the secure greater than or equal to (GTE) and less-than (LT) comparison of two arbitrary numbers. Our secure comparison protocol is based on the fact that the most significant bit (MSB) of the two's complement representation of a signed integer indicates the sign of the number. Thus the evaluation of the GTE and LT predicates could be reduced to the bit extraction operations.

To extract the MSB and preserve the privacy of the data, we adapt the bit-decomposition method in [50] to implement bit-level operations in our scheme. The online operations are performed between just two parties, which can greatly reduce the communication overhead between the parties. Given a shared l-bit number $u = u_1 + u_2$, we need to securely compute the shares of the MSB $u^{(l-1)} = u_1^{(l-1)} \oplus u_2^{(l-1)}$, where u_i and $u_i^{(l-1)}$ are the input and output of $\mathcal{S}_i$ $(i = 1, 2)$. This is achieved by masking the inputs with random values that will be converted into sharing of bits. And the bits of original inputs can be revealed by performing the bit-wise operations. To ensure computation and communication efficiency, we also put the random value generation on the trusted third party $\mathcal{T}$, which would be done in the offline phase.

Specifically, the following sub-protocols are included in our secure comparison protocol.

Data Representation. The bit-decomposition approach we use is over the signed integers, while the numbers we compare may be decimals. Since we just require to determine the sign of a number, we could convert the numbers into integers by multiplying the numbers by 10^p and remove the remaining decimal places, where p is the number of decimal places we want. Then, for the shared number $u = u_1 + u_2$, the two edge servers $\mathcal{S}_i$ $(i = 1, 2)$ can compute $\bar{u_i} = \lfloor u_i \cdot 10^p \rfloor$, respectively, where $\lfloor \cdot \rfloor$ denotes the round-down operation. For the sake of simplicity in notation, we will omit the overbar in the following if there is no confusion.

To perform the bit-wise operations, we make use of the two's complement binary representation of the number. The weight of each bit is a power of two, except for the MSB, whose weight is the negative of the corresponding power of two. Specifically, for the l-bit signed integer u, it can be converted into the form $u^{(l-1)}u^{(l-2)} \cdots u^{(0)}$ with $u^{(l-1)}$ being the MSB and

$$u = -u^{(l-1)} \cdot 2^{l-1} + \sum_{j=0}^{l-2} u^{(j)} \cdot 2^j.$$

Input: $\mathcal{S}_1$ has $v_1^{(l-1)} \cdots v_1^{(0)}$ and $r_1^{(l-1)} \cdots r_1^{(0)}$; $\mathcal{S}_2$ has $v_2^{(l-1)} \cdots v_2^{(0)}$ and $r_2^{(l-1)} \cdots r_2^{(0)}$.
Output: $\mathcal{S}_1$ outputs $u_1^{(l-1)} \cdots u_1^0$; $\mathcal{S}_2$ outputs $u_2^{(l-1)} \cdots u_2^0$.

1: **for** $j = 0$ to $l-1$ **do**
2: $\mathcal{S}_i$ compute $\alpha_i^{(j)} \leftarrow v_i^{(j)} \oplus r_i^{(j)}$.
3: $\mathcal{S}_1$ and $\mathcal{S}_2$ compute $(\beta_1^{(j)}, \beta_2^{(j)}) \leftarrow \texttt{SecMul}(v_1^{(j)}, v_2^{(j)}, r_1^{(j)}, r_2^{(j)})$.
4: **end for**
5: $\mathcal{S}_i$ set $c_i^{(0)} \leftarrow 0$.
6: $\mathcal{S}_i$ compute $u_i^{(0)} \leftarrow v_i^{(0)} \oplus r_i^{(0)}$.
7: **for** $j = 1$ to $l-1$ **do**
8: $\mathcal{S}_1$ and $\mathcal{S}_2$ compute $(\alpha_1^{(j-1)}, \alpha_2^{(j-1)}) \leftarrow \texttt{SecMul}(\alpha_1^{(j-1)}, \alpha_2^{(j-1)}, c_1^{(j-1)}, c_2^{(j-1)})$.
9: $\mathcal{S}_i$ compute $c_i^{(j)} \leftarrow \alpha_i^{(j-1)} \oplus \beta_i^{(j-1)}$.
10: $\mathcal{S}_i$ compute $u_i^{(j)} \leftarrow v_i^{(j)} \oplus r_i^{(j)} \oplus c_i^{(j)}$.
11: **end for**
12: $\mathcal{S}_i$ return $u_i^{(l-1)} \cdots u_i^0$.

Fig. 3. Secure bit-wise addition protocol `BitAdd`.

Random Bits Generation. It is fairly easy for the third party $\mathcal{T}$ to generate a random shared l-bit number r along with its sharing of bits in $\mathbb{Z}_2$.

Specifically, $\mathcal{T}$ first generates two shares of random bits $r_1^{(l-1)} \cdots r_1^{(0)}$ and $r_2^{(l-1)} \cdots r_2^{(0)}$. Then he performs bit-wise XOR and generates $r^{(l-1)} \cdots r^{(0)}$, where $r^{(j)} = r_1^{(j)} \oplus r_2^{(j)}, j = 0, 1, \ldots, l-1$. According to the two's complement binary representation, r can be reconstructed by computing $r = -r^{(l-1)} \cdot 2^{l-1} + \sum_{j=0}^{l-2} r^{(j)} \cdot 2^j$. Then, $\mathcal{T}$ splits r into two shares s_1 and s_2 such that $r = s_1 + s_2$. Note that $s_i \neq -r_i^{(l-1)} \cdot 2^{l-1} + \sum_{j=0}^{l-2} r_i^{(j)} \cdot 2^j, i = 1, 2$. Finally, $\mathcal{T}$ sends s_i along with $(r_i^{(l-1)}, \ldots, r_i^{(0)})$ to $\mathcal{S}_i$, respectively.

Secure Bit-Wise Addition. Given two bit-wise shares $v_i^{(l-1)} \cdots v_i^{(0)}$ and $r_i^{(l-1)} \cdots r_i^{(0)}$, $\mathcal{S}_1$ and $\mathcal{S}_2$ need to work together to reveal the bit-wise shares $u_i^{(l-1)} \ldots u_i^{(0)}$ of their sum $u = v + r$.

Although the *carry lookahead adder* (CLA) can solve the carry delay problem of the *ripple-carry adder* (RCA) by calculating the carry bits in advance, it requires much more rounds of communication between the two edge servers. Therefore, in our scheme, we design the secure bit-wise addition protocol based on the RCA. Specifically, we compute the carry bit by iterating from the least significant bit to the most significant one.

For the two input bit-wise numbers $v^{(l-1)} \cdots v^{(0)}$ and $r^{(l-1)} \cdots r^{(0)}$, we let $u^{(j)}$ be the sum of the bits at position j, and $c^{(j)}$ be the carry bit at position j that is propagated through from a less significant bit position. With $c^{(0)} = 0$, we have

$$u^{(j)} = v^{(j)} \oplus r^{(j)} \oplus c^{(j)},$$
$$c^{(j+1)} = (v^{(j)} \wedge r^{(j)}) \oplus ((v^{(j)} \oplus r^{(j)}) \wedge c^{(j)}).$$

We can easily find out that only XOR ($\oplus$) and AND ($\wedge$) operations are included. As illustrated in Fig. 3, when the inputs are shared between the two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$, the massive XOR operations can be performed by them locally without any interaction with each other. For the AND operations, they can also be implemented by invoking the secure multiplication protocol `SecMul` over $\mathbb{Z}_2$.

Input: $\mathcal{S}_1$ has $u_1 \in \mathbb{Z}_{2^l}$; $\mathcal{S}_2$ has $u_2 \in \mathbb{Z}_{2^l}$.
Output: $\mathcal{S}_1$ outputs $u_1^{(l-1)}$; $\mathcal{S}_2$ outputs $u_2^{(l-1)}$.

Offline Phase :
1: For $j = 0, \cdots, l-1$, $\mathcal{T}$ generates random $r_1^{(j)}, r_2^{(j)} \in \mathbb{Z}_2$.
2: For $j = 0, \cdots, l-1$, $\mathcal{T}$ computes $r^{(j)} \leftarrow r_1^{(j)} \oplus r_2^{(j)}$.
3: $\mathcal{T}$ computes $r \leftarrow -r^{(l-1)} \cdot 2^{l-1} + \sum_{j=0}^{l-2} r^{(j)} \cdot 2^j$.
4: $\mathcal{T}$ generates random $s_1 \in \mathbb{Z}_{2^l}$ and computes $s_2 \leftarrow r - s_1$.
5: $\mathcal{T}$ sends s_i and $r_i^{(l-1)} \cdots r_i^{(0)}$ to $\mathcal{S}_i$ $(i = 1, 2)$.

Online Phase :
6: $\mathcal{S}_1$ computes $t_1 \leftarrow u_1 - s_1$.
7: $\mathcal{S}_2$ computes $t_2 \leftarrow u_2 - s_2$ and sends it to $\mathcal{S}_1$.
8: $\mathcal{S}_1$ computes $v \leftarrow t_1 + t_2$ and generates its two's complement binary representation $v^{(l-1)} \cdots v^{(0)}$.
9: For $j = 0, \cdots, l-1$, $\mathcal{S}_1$ generates random $v_1^{(j)} \in \mathbb{Z}_2$ and compute $v_2^{(j)} \leftarrow v^{(j)} \oplus v_1^{(j)}$.
10: $\mathcal{S}_1$ sends $v_2^{(l-1)} \cdots v_2^{(0)}$ to $\mathcal{S}_2$.
11: $\mathcal{S}_1$ and $\mathcal{S}_2$ compute $(u_1^{(l-1)} \cdots u_1^{(0)}, u_2^{(l-1)} \cdots u_2^{(0)}) \leftarrow \texttt{BitAdd}(v_1^{(l-1)} \cdots v_1^{(0)}, v_2^{(l-1)} \cdots v_2^{(0)}, r_1^{(l-1)} \cdots r_1^{(0)}, r_2^{(l-1)} \cdots r_2^{(0)})$.
12: $\mathcal{S}_i$ return $u_i^{(l-1)}$.

Fig. 4. Secure bit-extraction protocol `BitExtra`.

Secure Bit Extraction. To extract the bits of the shared number u, $\mathcal{S}_1$ and $\mathcal{S}_2$ first mask their input shares u_1 and u_2 with the received random number shares s_1 and s_2 by computing $t_1 = u_1 - s_1$ and $t_2 = u_2 - s_2$, respectively. Then they send t_1 and t_2 to each other and compute $v = t_1 + t_2$. We can easily find out that $u = u_1 + u_2 = v + r$. Since $\mathcal{S}_1$ and $\mathcal{S}_2$ have known the value v and the bit-wise shares of r, they can work together to compute the bit-wise shares of u by invoking the above bit-wise addition protocol `BitAdd`. The secure bit extraction protocol `BitExtra` is as illustrated in Fig. 4.

So far, we have extracted the shared bits of the number u. Then the secure `MSB` protocol would just involve computing $u^{(l-1)} = u_1^{(l-1)} \oplus u_2^{(l-1)}$, which can determine the sign of the shared number u, i.e., if $u^{(l-1)} = 0$, $u \geq 0$; otherwise, $u < 0$.

Finally, if we want to compare two shared numbers u and v, we can transform the problem into determining the sign of $u - v$. Specifically, given $u = u_1 + u_2$ and $v = v_1 + v_2$, $\mathcal{S}_1$ and $\mathcal{S}_2$ can compute $w_1 = u_1 - v_1$ and $w_2 = u_2 - v_2$, respectively. By invoking the secure `MSB` protocol, we can get the most significant bit of $w = w_1 + w_2$. Correspondingly, we can obtain the comparison result of u and v, i.e., if $w^{(l-1)} = 0$, we have $u \geq v$; otherwise, $u < v$.

5.4 Vectorization

By using the efficient matrix-vector operations, we can make the most of the nature of CNN, that is the data and parameters can be organized into vectors and matrices. Moreover, multiple instances can also be processed in a batch simultaneously. Correspondingly, we can speed up calculations in the network significantly. Besides, in order to take better advantage of the parallelism and efficient matrix operations, we could have our secret-sharing based secure computation be performed simultaneously. This can be easily

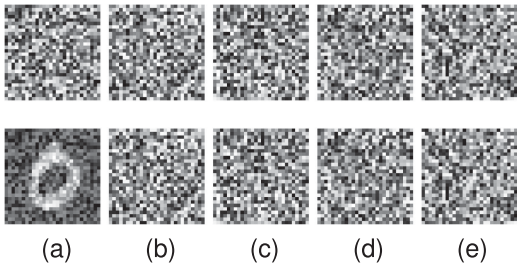


Fig. 5. An example of splitting an image in the MNIST dataset into two shares with regard to different choices of n. One share is in the top row and the other is in the bottom row. (a) n = 8; (b) n = 16; (c) n = 24; (d) n = 32; (e) n = 40.

implemented since the data is always in the form of shares in our design and we don't need to make any changes to the data structures.

6 Lightweight Privacy-Preserving CNN Feature Extraction for Mobile Sensing

In this section, we present the details of our design and the secure interaction protocols. Note that for the sake of clarity in notation, we use the marks′ and ″ to denote the shares distributed to the two edge servers in the following.

6.1 Image Encryption

To protect the privacy of the original images, the sensor $\mathcal{O}$ encrypts them based on the additive secret sharing technique. Specifically, for each image I of size $w \times h \times d$ (width, height, and depth, respectively), $\mathcal{O}$ first generates a random volume V of the same size as I. To preserve the privacy of the original elements of I in the range $[0, 2^8 - 1]$, the elements of V are drawn uniformly at random from a much larger interval $[-2^{n-1}, 2^{n-1} - 1]$, where $n > 8$ acts as security parameter to define the message space. After that, $\mathcal{O}$ encrypts I by splitting it into two shares in an element-wise manner: $I' = V$, and $I'' = I - I'$. As illustrated in Fig. 5, we give an example of splitting a handwritten image into two shares with regard to different choices of n. The two shares I' and I'' are shown in the top row and the bottom row, respectively. Then, I' and I'' will be sent to two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively.

6.2 Secure CNN Feature Extraction

As described above, a common convolutional neural network architecture usually consists of the following four types of layers: Convolutional Layer, ReLU layer, Pooling Layer, and Fully-Connected Layer. To enable privacy-preserving CNN feature extraction, we design a series of secure interactive protocols between the two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$, as shown in Fig. 6. These protocols correspond to the various layers and will be the building blocks of our scheme. Note that we aim to utilize the publicly available model of CNN and focus on the inference phase. Thus, the parameters such as weights and biases are considered to be public to all participants.

6.2.1 Convolutional Layer

The convolution operation essentially performs dot products between the filters and local regions of the input, which is associative and distributive over addition. Therefore, we can take advantage of this fact and let $\mathcal{S}_1$ and $\mathcal{S}_2$ perform the forward pass of the convolutional layer locally with the public weights w and biases b based on our secure addition protocol.

Specifically, for the $(j,k)^{\text{th}}$ hidden neuron in this layer, $\mathcal{S}_1$ will compute the output:

$$y'_{j,k} = \sum_{l=0}^{n-1}\sum_{m=0}^{n-1} w_{l,m} x'_{j+l,k+m} + b. \tag{1}$$

And $\mathcal{S}_2$ will compute the output:

$$y''_{j,k} = \sum_{l=0}^{n-1}\sum_{m=0}^{n-1} w_{l,m} x''_{j+l,k+m}. \tag{2}$$

Here, we use $x_{j,k}$ to denote the input activation at position j, k. And the filters have size $n \times n$ with shared weights $w_{l,m}(l, m = 0, 1, \ldots, n-1)$ and bias b. It is important to note that we have set all the biases in $\mathcal{S}_2$ to be 0.

6.2.2 Fully-Connected Layer

Neurons in a fully connected layer have full connections to all activations in the previous layer. Their activations can then be computed with a matrix multiplication followed by a bias offset, which also satisfies the associativity and distributivity. Therefore, the forward pass of the full-connected layer can also be performed locally in $\mathcal{S}_1$ and $\mathcal{S}_2$ based on our secure addition protocol. Specifically, for the jth hidden neuron in this layer, $\mathcal{S}_1$ will compute one component of the activation:

$$y'_j = \sum_k w_{jk} x'_k + b_j. \tag{3}$$

And $\mathcal{S}_2$ will compute the other component of the activation

$$y''_j = \sum_k w_{jk} x''_k. \tag{4}$$

We use x_k for the activation of the kth neuron in the previous layer. In addition, we use w_{jk} to denote the weights for

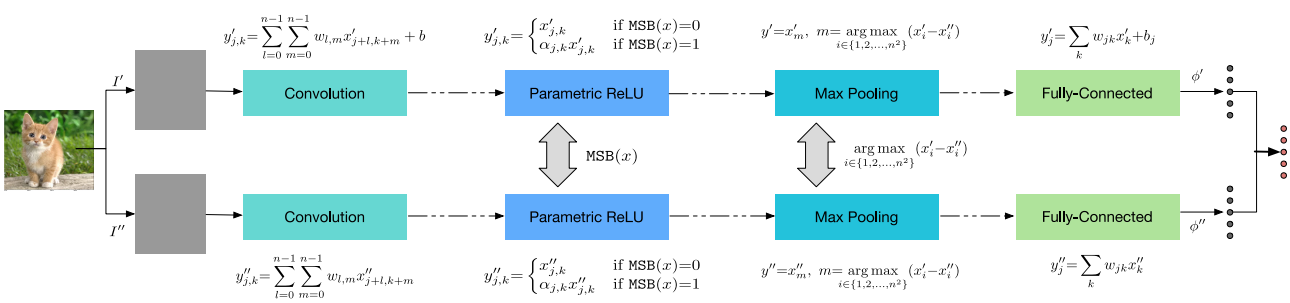


Fig. 6. Privacy-preserving CNN feature extraction protocols.

the connection from the kth neuron in the previous layer to the jth neuron in the current layer, and b_j to denote the bias of the jth neuron in the current layer. Note that the sum is over all neurons k in the previous layer and we have also set the biases in $\mathcal{S}_2$ to be 0.

6.2.3 Activation Layer

Unlike the former two linear layers, the purpose of the activation layer is to introduce non-linearity into the network. As described above, the default recommendation in modern neural networks is to use ReLU. Therefore, we first describe the privacy preserving ReLU layer.

ReLU layer applies the function $max(x, 0)$ thresholding at zero. Note that each element x is shared by $\mathcal{S}_1$ and $\mathcal{S}_2$, i.e., $x = x' + x''$. By applying the above secure $\texttt{MSB}$ protocol, $\mathcal{S}_1$ and $\mathcal{S}_2$ will work together to determine the MSB of x. If the MSB is 0, we have $max(x, 0) = x$; and otherwise, $max(x, 0) = 0$. Correspondingly, for the $(j, k)^{\text{th}}$ hidden neuron, we let $\mathcal{S}_1$ have:

$$y'_{j,k} = \begin{cases} x'_{j,k} & \text{if } \texttt{MSB}(x) = 0 \\ 0 & \text{if } \texttt{MSB}(x) = 1. \end{cases} \tag{5}$$

Meanwhile, we let $\mathcal{S}_2$ have

$$y''_{j,k} = \begin{cases} x''_{j,k} & \text{if } \texttt{MSB}(x) = 0 \\ 0 & \text{if } \texttt{MSB}(x) = 1. \end{cases} \tag{6}$$

The major drawback of ReLU is that it cannot learn via gradient-based methods when $x < 0$ [51]. Several generalizations of ReLU are proposed to address this issue, including absolute value rectification, leaky ReLU, and parametric ReLU. They are based on using a non-zero slope α when $x < 0$: $max(x, 0) + \alpha min(x, 0)$. The main difference between them lies in the value of α, which can be fixed to -1, 0.01, or even be treated as a learnable parameter, respectively.

For the generalizations of ReLU, we need to make corresponding modifications. Particularly, for the $(j, k)^{\text{th}}$ hidden neuron, we let $\mathcal{S}_1$ have:

$$y'_{j,k} = \begin{cases} x'_{j,k} & \text{if } \texttt{MSB}(x) = 0 \\ \alpha_{j,k} x'_{j,k} & \text{if } \texttt{MSB}(x) = 1. \end{cases} \tag{7}$$

Meanwhile, we let $\mathcal{S}_2$ have

$$y''_{j,k} = \begin{cases} x''_{j,k} & \text{if } \texttt{MSB}(x) = 0 \\ \alpha_{j,k} x''_{j,k} & \text{if } \texttt{MSB}(x) = 1. \end{cases} \tag{8}$$

Other Non-Linear Activation Layers. We note that, prior to the introduction of ReLU, most neural networks used the sigmoid activation function $\sigma(x) = 1/(1 + e^{-x})$ or the tanh activation function $tanh(x)$. However, it is accepted that they suffer from the problem of vanishing gradient and therefore make gradient-based learning difficult. Thus, their use as activation functions in feedforward networks is now discouraged. In addition, many other types of activation functions are possible but are used less frequently.

Even so, for the sake of completeness, we still give a generalized approach for securely processing the non-linear activation layers. As previous work [9], we also adopt the *piecewise polynomial spline* because of its ease and accuracy of construction and evaluation, and its capacity to approximate complex shapes [52]. Then, the non-linear functions can be approximated with piecewise polynomials

$$f(x) = \begin{cases} P_0(x), & x_0 \le x < x_1 \\ P_1(x), & x_1 \le x < x_2 \\ & \cdots \\ P_{k-1}(x), & x_{k-1} \le x < x_k, \end{cases}$$

where $P(x)$ is an n-degree polynomial, i.e., $P(x) = a_0 + a_1 x + \cdots + a_n x^n$. There is no doubt that higher degree polynomials give better approximations.

To further protect the privacy of the elements x, we propose to split the piecewise polynomials as well. We let the coefficients $a_i = a'_i + a''_i$. Correspondingly, $P(x) = P'(x) + P''(x)$, where $P'(x) = a'_0 + a'_1 x + \cdots + a'_n x^n$ and $P''(x) = a''_0 + a''_1 x + \cdots + a''_n x^n$ are stored in $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Meanwhile, the endpoints of the intervals are also split into shares distributed to $\mathcal{S}_1$ and $\mathcal{S}_2$, i.e., $x_j = x'_j + x''_j$. It's not hard to find out that the polynomials can be solved securely by applying the above secure computation protocols. Specifically, the first step is to decide which interval x lies in, which could be easily implemented with the secure comparison protocol. Since the polynomials only include addition and multiplication operations, we can then securely solve them by invoking the secure addition and multiplication protocols.

We must point out that, compared with previous work based on computation-intensive cryptographic primitives like homomorphic encryption and garbled circuits, we can ensure much better accuracy by representing these non-linear layers with higher degree polynomials. This is due to the fact that we can achieve high efficiency with our secret-sharing based secure computation protocols, and meanwhile, we don't need to deal with the noise growth in the ciphertext.

6.2.4 Pooling Layer

Pooling layer performs a downsampling operation along the spatial dimensions (width, height). As described above, popular pooling functions include the average pooling and the max pooling. For the average pooling that outputs the average of the values within a rectangular neighborhood, it is trivial to check that computing average is also associative and distributive over addition. If we use the average pooling in our network, the two edge servers can still perform the pooling operations locally.

However, the max pooling is a bit more complicated. It reports the maximum output within a rectangular neighborhood. Let the stride size be n, then each max is taken over $n \times n$ numbers. In our scenario, each number $x_i (i \in \{1, 2, \ldots, n^2\})$ is split into two components x'_i and x''_i. Here, x'_i is owned by $\mathcal{S}_1$, and x''_i is owned by $\mathcal{S}_2$. To perform the max operation, $\mathcal{S}_1$ and $\mathcal{S}_2$ need to work collaboratively based on our secure comparison protocol to obtain the index m of the maximum number without revealing the actual numbers.

Specifically, for two numbers x_i and x_j, we have

$$\begin{aligned} (x_i < x_j) &= \texttt{MSB}(x_i - x_j) \\ &= \texttt{MSB}((x'_i + x''_i) - (x'_j + x''_j)) \\ &= \texttt{MSB}((x'_i - x'_j) + (x''_i - x''_j)). \end{aligned}$$

Correspondingly, $\mathcal{S}_1$ and $\mathcal{S}_2$ will compute $\Delta x' = x'_i - x'_j$ and $\Delta x'' = x''_i - x''_j$, respectively. Then, they will work together to compute $\texttt{MSB}(\Delta x', \Delta x'')$ by using the above bit-extraction protocol. If the output of MSB is 0, we will claim that $x_i \geq x_j$; otherwise, $x_i < x_j$.

In this way, after iterating through all the n^2 values and performing $n^2 - 1$ encrypted comparisons, $\mathcal{S}_1$ and $\mathcal{S}_2$ can determine the index m of the largest value. Correspondingly, $\mathcal{S}_1$ will output

$$y' = x'_m, \quad m = \underset{i \in \{1,2,\ldots,n^2\}}{\arg\max} (x'_i - x''_i). \tag{9}$$

And $\mathcal{S}_2$ will output

$$y'' = x''_m, \quad m = \underset{i \in \{1,2,\ldots,n^2\}}{\arg\max} (x'_i - x''_i). \tag{10}$$

To reduce the interactions between the edge servers, $\mathcal{S}_1$ and $\mathcal{S}_2$ can compute $\Delta x'$ and $\Delta x''$ between any two numbers within the $n \times n$ region simultaneously. Then, they can perform the secure comparison protocols for all Δx's at once.

6.2.5 Remarks

Since we focus on the inference phase, many layers that work during the learning phase would not make any difference to our scheme. In addition, some layers have since fallen out of favor because their impact has been shown to be limited in practice. For instance, it is shown that the Local Response Normalisation (LRN) does not improve the performance, but leads to increased memory consumption and computation time [53]. Moreover, it is impractical to describe all the layers here. Therefore, we just highlight the building blocks that compose the current best CNN architectures such as ResNets [54].

In particular, the residual layers and the skip connections in ResNets make the training of very deep networks possible. They have become indispensable components in a variety of neural network architectures [55] and have a very positive effect in the practical application. Specifically, the skip connections are added to perform identity mapping. A residual block that consists of a few stacked non-linear layers are learned to approximate the residual functions, i.e., $\mathcal{F}(x) = y - x$, where x and y are the input and output of the residual block. Then, the output of the residual block will be $y = \mathcal{F}(x) + x$. When the input x is split into two shares x' and x'' in our scheme, we let $\mathcal{S}_1$ compute

$$y' = \mathcal{F}(x') + x'. \tag{11}$$

Meanwhile, we let $\mathcal{S}_2$ compute

$$y'' = \mathcal{F}(x'') + x''. \tag{12}$$

Since the skip connections add neither extra parameter nor computational complexity [54], the residual layers can be performed by the two edge servers locally and simultaneously.

6.3 CNN Feature Decryption

After performing the series of interactive protocols corresponding to the pre-trained CNN architecture, $\mathcal{S}_1$ and $\mathcal{S}_2$ will take the outputs of some layer in the network as the components of the CNN feature ϕ. Thus, $\mathcal{S}_1$ will output one component ϕ'; and $\mathcal{S}_2$ will output the other component ϕ''. Both components ϕ' and ϕ'' will then be sent to the image user $\mathcal{U}$. According to the nature of the additive secret sharing, $\mathcal{U}$ can decrypt the CNN feature by simply computing $\phi = \phi' + \phi''$.

Note that we consider feature extraction as a fundamental problem in the many visual tasks. Combined with some classifiers (e.g., the linear SVM classifier), or some distance metrics (e.g., the L2 distance), the decrypted ϕ can be utilized to tackle the various visual tasks of image classification, image retrieval, and so on. Certainly, we can also combine these classifier into the network and trained an end-to-end deep learning model. Since we have constructed a series of building blocks and implemented the various secure computation protocols, the privacy-preserving inference can also be fully supported.

7 Theoretical Analysis

7.1 Correctness

In the process of CNN feature extraction, the network transforms the original image I layer by layer from the raw pixels to a single vector ϕ. In our scheme, I is divided into two components by $I = I' + I''$ based on the additive secret sharing. Intuitively, we don't necessarily have the output $\phi = \phi' + \phi''$ after a sequence of linear and non-linear transformations. However, through our collaborative design, we can ensure that the system will return the exact feature vector.

First, the convolutional layers, the fully-connected layers, and even the average pooling layers in the network essentially perform linear dot products. For the input $x = x' + x''$, we naturally have the output $y' + y'' = y$. Second, for the ReLU layers, whether the input is larger than 0 or not is determined collaboratively by the two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$. For the input that is larger than 0, both $\mathcal{S}_1$ and $\mathcal{S}_2$ will retain the original x' and x''. Then $y' + y'' = x' + x'' = y$. When the input is less than 0, for the basic ReLU, the output of $\mathcal{S}_1$ and $\mathcal{S}_2$ will both be 0; for the other generalizations of ReLU, we still have $y' + y'' = \alpha x' + \alpha x'' = \alpha x = y$. Third, for the max-pooling layers, since the index m of the largest value is also determined collaboratively by the two edge servers, we have $x'_m + x''_m = y' + y'' = y$. Finally, for the residual layers in ResNets, since the skip connections are identity mapping, we have $y' + y'' = [\mathcal{F}(x') + x'] + [\mathcal{F}(x'') + x''] = \mathcal{F}(x' + x'') + (x' + x'') = y$. The second equality is from the above analysis that the output of the non-linear layers can still preserve the additive property. Therefore, the skip connections have no effect on the results.

Note that we convert the decimal numbers into integers when performing secure comparison. Normally, this will not impact the comparison result except that the two numbers are very close to each other. However, to keep the desired precision, e.g., p decimal places, we can scale up the values by multiplying the original numbers by 10^p. Meanwhile, to prevent overflow when representing the converted integers in binary form, we should let $0 < p < (l - n) \cdot \lg 2$, where l and n are the bit-widths of the converted integers and the generated random numbers, respectively.

Overall, although the inputs are divided into two shares, the output of each layer and even the whole network still

preserves the additive property. This ensures that the data user will finally be able to extract the CNN features correctly. Moreover, we don't need to make any approximations for recommended common CNN layers during the whole process of our scheme like other schemes based on homomorphic encryption. Thus, our scheme can be applied to any such CNN architectures without loss in accuracy.

7.2 Security

We prove the security of our protocols in the universal composability framework [56], [57]. In our honest-but-curious model, the adversary is allowed to corrupt at most one of the two edge servers $\mathcal{S}_1$ and $\mathcal{S}_2$. To prove that a protocol is secure, it suffices to show that the view of the corrupted party is simulatable given its input and output [39]. Specifically, we use the following definition.

Definition 1. *We say that a protocol is secure if there exists a probabilistic polynomial-time simulator $\mathcal{S}$ that can generate a view for the adversary $\mathcal{A}$ in the real world and the view is computationally indistinguishable from its real view.*

To prove the security of our protocols, the following lemmas will be used.

Lemma 1. *[39]. A protocol is perfectly simulatable if all its subprotocols are perfectly simulatable.*

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

We refer the reader to [39] and [37] for the proofs of Lemmas 1 and 2. Since most of the subprotocols in our scheme are performed locally, they can be perfectly simulated. Therefore, we mainly prove the security for the ones that need interactions between $\mathcal{S}_1$ and $\mathcal{S}_2$ in the following.

Theorem 1. *The protocol `SecMul` is secure in the honest-but-curious model.*

Proof. For $\mathcal{S}_1$, the view in the protocol execution will be $\mathsf{view}_1 = (u_1, v_1, a_1, b_1, c_1, \alpha_2, \beta_2)$, where $\alpha_2 = u_2 - a_2$ and $\beta_2 = v_2 - b_2$. According to Lemma 2, it is trivial to see that all these values are uniformly random. Besides, the output of $\mathcal{S}_1$ will be $\mathsf{output}_1 = (f_1 = c_1 + b_1 \cdot \alpha + a_1 \cdot \beta)$, where f_1 is also uniformly random. Therefore, both view_1 and output_1 are simulatable by the simulator $\mathcal{S}$ and the views of $\mathcal{S}$ and $\mathcal{A}$ will be computationally indistinguishable. Likewise, for $\mathcal{S}_2$, it is easy for the simulator $\mathcal{S}$ to generate a view that is computationally indistinguishable from its real view. □

Theorem 2. *The protocol `BitAdd` is secure in the honest-but-curious model.*

Proof. For $\mathcal{S}_i$, the view during an execution of `BitAdd` will be $\mathsf{view}_i = (v_i^{(j)}, r_i^{(j)}, \beta_i^{(j)}, \alpha_i^{(j)}), 0 \leq j \leq l$. Here, $v_i^{(j)}$ and $r_i^{(j)}$ are uniformly random. Meanwhile, $\beta_i^{(j)}$ and $\alpha_i^{(j)}$ are obtained through the `SecMul` protocol that has been proven secure in the honest-but-curious model. Therefore, view_i is simulatable by the simulator $\mathcal{S}$. Besides, the output of $\mathcal{S}_i$ will be $\mathsf{output}_i = (u_i^{(j)} = v_i^{(j)} \oplus r_i^{(j)} \oplus c_i^{(j)})$, where $c_i^{(j)} = \alpha_i^{(j-1)} \oplus \beta_i^{(j-1)}$. Since the operations are performed locally by $\mathcal{S}_1$ and $\mathcal{S}_2$, output_i is also simulatable by the simulator $\mathcal{S}$. Therefore, our protocol `BitAdd` is secure in the honest-but-curious model. □

Theorem 3. *The protocol `BitExtra` is secure in the honest-but-curious model.*

Proof. Before performing the `BitAdd`, the views of $\mathcal{S}_1$ and $\mathcal{S}_2$ will be $\mathsf{view}_1 = (u_1, r_1, v_2)$ and $\mathsf{view}_2 = (u_2, r_2, v_2^{(l-1)} \cdots v_2^{(0)})$, respectively. It's trivial to see that they are uniformly random and can be simulated by the simulator $\mathcal{S}$. Since the security of the protocol `BitAdd` has been proven above, it's trivial to prove the security of the whole protocol `BitExtra` as well. Correspondingly, the `MSB` protocol is also secure in the honest-but-curious model. □

Theorem 4. *The CNN feature extraction protocol in our scheme is secure in the honest-but-curious model.*

Proof. For the convolutional layer and the fully-connected layer, $\mathcal{S}_i$ received no output. Thus, it's trivial for a simulator $\mathcal{S}$ to generate the view of the incoming messages received by $\mathcal{S}_i$. For the ReLU layer and the Max-pooling layer, the only interactions between $\mathcal{S}_1$ and $\mathcal{S}_2$ occur when comparing two numbers. Note that our comparison protocol is based on the `MSB` protocol that is actually the simulatable building block `BitExtra`. For the other non-linear activation layers, they are solved by representing them as polynomials, which is also based on the simulatable building blocks `BitExtra` and `SecMul`. They are simulatable by the simulator $\mathcal{S}$. Therefore, we conclude that our CNN feature extraction protocol is secure in the honest-but-curious model. □

7.3 Efficiency

Although there can be some trade-off between privacy and efficiency, use of the additive secret sharing in our scheme leads to substantial efficiency gains over the use of homomorphic encryption in previous work. On the one hand, the image sensor just needs to perform random number generation and simple subtraction to encrypt the images before moving the images to the edge servers. The computation time will be only $O(1)$. This greatly reduces the overhead on the end devices. On the other hand, during the online phase of the whole process of CNN feature extraction in our scheme, all communication takes place between the two edge servers. The end devices do not have to participate in an online phase with unacceptable amounts of work and communication.

As to the communication cost between the two edge servers, it relies on the network architecture to some extent. However, communication takes place only when the non-linearity happens. Since we don't rely on any computation-intensive cryptographic primitives like homomorphic encryption and garbled circuits, our approach has a very small communication overhead. With regard to the number of rounds of communication between the two edge servers, we take full advantage of the nature of convolutional neural networks and secret sharing techniques, and the operations in our scheme can be performed in parallel. Correspondingly, the number of rounds of communication in our scheme can be reduced to $O(l)$, where l is the bit-width of the inputs. In the following experiments, we will show that the latency

TABLE 1
Comparison of the Protocol Complexities
(Here, l is the bit-width, and $m = log_2 l$.)

Approach	SecMul		SecCmp	
	Rounds	Comm (bits)	Rounds	Comm (bits)
[37]	1	$15l$	$\frac{1}{2}lm$	$30lm + 32l$
[43]	1	$6l$	-	-
Ours	1	$2l$	$l + 1$	$10l - 4$

would be very low since the message size in each round of communication is very small.

Besides, we restrict the number of the participants and put the computation-intensive work on just two edge servers in our scheme. Correspondingly, on the mobile sensors and the user side, we only need two thirds of the storage and bandwidth resources in previous work [37], [39], [43], [44], [45] that is designed for three parties. The decrease of the required storage and transmission resources is considerable for the resource-constrained end devices. Moreover, this can avoid frequent interactions between the multi-parties, which would bring extra delay and reduce the efficiency. We can see from Table 1 that the rounds of communication and the total amount of communication for the basic secure computation protocols (including SecMul and SecCmp) in our scheme are also much less than previous work.

8 Experimental Results

In this section, we present the experimental results of our scheme with regard to the secret-sharing based secure comparison protocols and the privacy-preserving CNN feature extraction protocol.

Our mobile application was implemented in Java and was run on a mobile phone (HUAWEI Honor 8 Lite) running Android 7.0 with an octa-core Kirin 655 processor @ 1.7 GHz and 4 GB of RAM. Our privacy-preserving CNN feature extraction protocol was implemented in Python 3. The package NumPy was used as a multi-dimensional container of the numbers to implement our secret-sharing based secure computation protocols in parallel. The CNN networks were trained with the Caffe framework [58]. And we run the privacy-preserving CNN feature extraction protocol on two servers running Ubuntu 18.04 in the LAN setting. Each server was equipped with a 4-core Intel Core i7-6700 CPU @ 3.40 GHz and 16 GB of RAM. The times reported are an average over 10 trials.

8.1 Performance of Secure Comparison

In our scheme, the operations in the non-linear ReLU layers and the Max-pooling layers are both reduced to the secure comparison of numbers. Since other linear layers that can be performed locally and efficiently, the secure comparison becomes the main part of the computation and communication. Therefore, we first evaluate the performance of the secure comparison in our scheme.

Since we implement our secure comparison protocol based on the bit-decomposition method, we evaluated the impact of different bit-width l of the inputs on the runtime of doing one secure comparison. The offline phase includes the generation

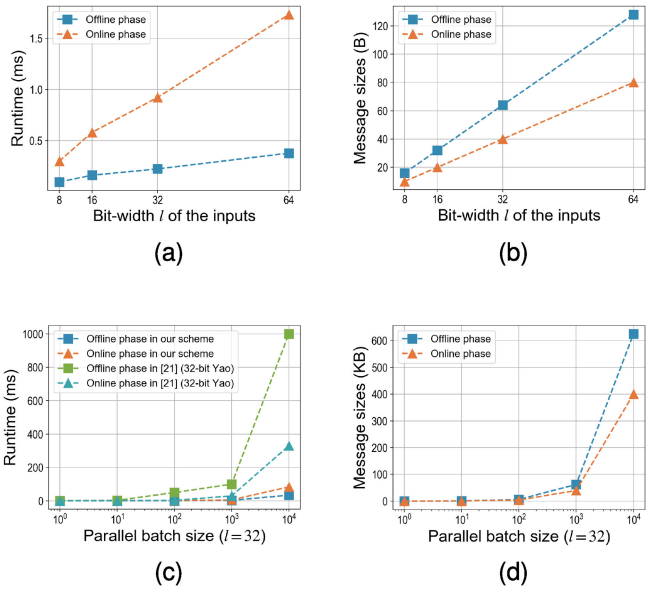


Fig. 7. Performance of secure comparison: (a) Runtime for different bit-width l of the inputs; (b) Communication overhead between the two edge servers for different bit-width l of the inputs; (c) Runtime for different parallel batch size; (d) Communication overhead between the two edge servers for different parallel batch size.

of random numbers and random multiplication triplets by the trusted third party. And the online phase includes the comparison of two l-bit numbers by the two edge servers. As shown in Figs. 7a and 7b, the runtime and the communication overhead both go up in the bit-width l. However, they are in a matter of "milliseconds" and "bytes". In particular, when we choose the security parameter $n < 30$, it is enough to set $l = 32$. At this point, the runtime and the communication overhead are less than 1 ms and 100 B, respectively. Therefore, to make a tradeoff between privacy and efficiency, we'll use $l = 32$ in the following.

Instead of doing secure comparisons one by one, we put the numbers in the NumPy ndarrays and make use of vectorization to apply operations in parallel. This is consistent with the nature of CNN and secret sharing. Fig. 7c depicts the runtime in ms when doing comparisons with different parallel batch sizes. We can see that the runtime has not changed much as the parallel batch size grows. Even when the batch size is 10^4, the runtimes in the offline and online phase are just 34 ms and 83 ms, respectively. We also compared our results with the well known Yao's GC implemented in [59]. Its runtime grows in proportion to the parallel batch size, which is much less efficient than ours. Besides, we can see from Fig. 7d that the communication overhead among the parties are just several hundred KB when the batch size is 10^4, which ensures the high efficiency of the communication in our scheme.

8.2 Performance of Privacy-Preserving CNN Feature Extraction Protocol

To evaluate the performance of our scheme, we compare our results with previous work including (1) CryptoNets [8], which adopted the homomorphic encryption; (2) MiniONN [9], which combined the garbled circuits with homomorphic encryption; and (3) Chameleon [10], which proposed a more complicated and mixed protocol framework combining the garbled circuits, GMW, and additive secret-sharing.

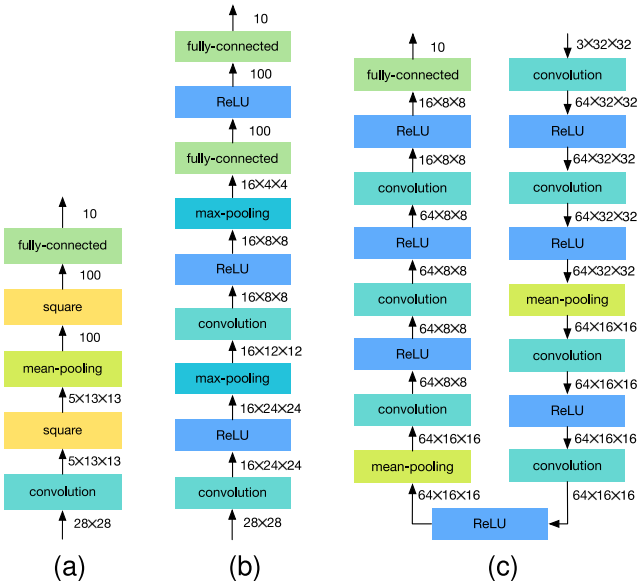


Fig. 8. Our CNN architectures: (a) Network I, (b) Network II, and (c) Network III.

Network Architecture. As shown in Fig. 8, we tested our approach on three kinds of network architectures. We represent them by Network I, II, and III, which include 5, 9, and 17 layers, respectively. Note that we use the output of the last layer as our feature vector and meanwhile use the softmax to further handle the extracted feature vector. Here, Network I and II are trained for the MNIST recognition task. The MNIST dataset comprises 60,000 training and 10,000 test greyscale images of size 28×28. Network III is trained for the image classification task on the CIFAR-10 dataset, which consists of 50,000 training and 10,000 test RGB images of size $3 \times 32 \times 32$.

Accuracy. For Network I, II, and III, we can obtain the accuracy of 98.27, 99.14, and 82.45 percent, respectively, which are consistent with the networks we trained in plaintext. Essentially, this is due to the fact that we don't need to make any approximations for the common layers during the design of the network architecture. Thus, our scheme can be applied to the network that has any number of layers and can obtain comparable accuracy to the CNN model in plaintext. More importantly, to make the network compatible with homomorphic encryption, [8] simply replaces the activation function by the lowest-degree non-linear polynomial function, i.e., the squared function. As admitted by the authors

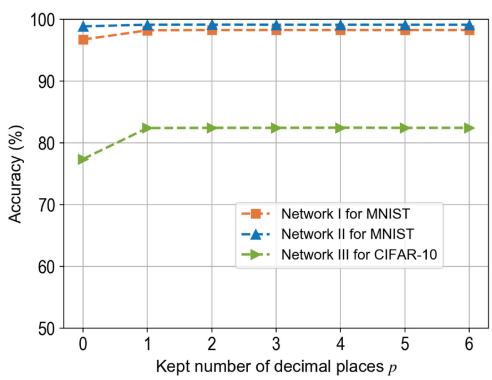


Fig. 9. Relationship between accuracy and the kept number of decimal places p.

TABLE 2
The Runtimes of the Encryption Phase for Different Batch Sizes in the Mobile Device

Batch Size	MNIST		CIFAR-10	
	Runtime (s)	Power usage (mAh)	Runtime (s)	Power usage (mAh)
1	0.009	0.0429	0.015	0.0538
10	0.012	0.0487	0.018	0.0548
100	0.024	0.0515	0.052	0.0588
1000	0.103	0.0637	0.346	0.0740
10000	0.796	0.140	3.158	0.345

themselves, the unbounded derivative of the square functions leads to a strange behavior during training. As a result, their protocols can only be applied to a shallow network and its accuracy can be very low for networks with more than 2 non-linear layers.

We also evaluate the impact of the kept number of decimal places p when performing secure comparison on the accuracy of our networks. As illustrated in Fig. 9, when $p > 0$, we can maintain the accuracy that is obtained in plaintext. It is consistent with our accuracy analysis that converting the decimals into integers when performing comparison has an impact on the accuracy only when the two numbers are very close to each other.

Efficiency. We first test the cost of the end devices for image encryption. In particular, the power consumption (measured in mAh) is collected with the tool *Batterystats* included in the Android framework. We can see from Table 2 that the runtime and the power consumption for encryption increase almost linearly with the image size and the batch size. However, both the runtime and the power consumption are totally acceptable and the encryption process does not cause any noticeable impact on the performance. We also compare with CryptoNets the cost of the end devices and the communication overhead between the end devices and the servers for processing a batch of 4096 images from the MNIST. The results in Table 3 show that, compared with CryptoNets, the time costs during the encryption and decryption phases in our implementation are almost negligible. Besides, we don't need additional time for every parallel instance to be encoded and decoded. This is because the end devices just need to perform the random number generation and simple subtraction to encrypt the images and decrypt the final results. Thus, the overhead of the end devices can be greatly reduced. With regard to the communication costs in Table 4, CryptoNets need to encrypt each pixel as five polynomials and each coefficient

TABLE 3
Comparison of the Runtimes of the Encryption and Decryption Phases for a Batch of 4096 Images from the MNIST

Approach	Encoding +Encryption (s)		Decryption +Decoding (s)	
	Runtime	Additional latency	Runtime	Additional latency
CryptoNets [8]	44.5	565.248	3	49.152
Our scheme	0.344	0	0.056	0

TABLE 4
Comparison of the Communication Overhead Between the End Devices and the Servers for a Batch of 4096 Images from the MNIST

Approach	Sensor → Server		Server → User	
	Message size (MB)	Size per instance (KB)	Message size (MB)	Size per instance (KB)
CryptoNets [8]	367.5	91.975	4.70	1.17
Our scheme	12.25	3.063	0.16	0.04

TABLE 5
Comparison of the Runtimes (ms) of Different Layers

Layers		Network I	Network II	Network III
Convolution	Online	0.62	2.01	17.49
Square	Offline	0.839	-	-
	Online	0.82	-	-
ReLU	Offline	-	34.3	611.36
	Online	-	78.71	1541.44
Max-pooling	Offline	-	50.13	-
	Online	-	121.58	-
Mean-pooling	Online	0.37	-	1.85
FC	Online	0.15	0.34	0.29

TABLE 6
Comparison of the Runtimes and Message Sizes with Network I

Approach	Runtime (s)		Message sizes (MB)	
	Offline	Online	Offline	Online
CryptoNets [8]	0	297.5	0	372.2
MiniONN [9]	0.88	0.4	3.6	44
Our scheme	0.0009	0.002	0.022	0.015

in the polynomial requires 24 bytes, while each pixel just requires 4 bytes in our scheme. Therefore, we can greatly bring down the communication overhead between the end devices and the edge servers.

We then test the performance of our privacy-preserving feature extraction scheme that is performed by the two edge servers. Table 5 shows the runtimes of different layers for processing one instance with the three networks. We can see that most layers are very efficient since they can be performed by the servers locally, and the cost of our scheme is led by the activation and max-pooling layers. Here, the runtimes in the online phase mainly include performing the secure comparison protocols, while the generation of random numbers and multiplication triplets are performed by the trusted third party in the offline phase.

We compare the runtime and communication overhead of processing one instance from Tables 6, 7, and 8. We can see that our scheme outperforms previous work by several orders of magnitude in terms of the runtime. This is mainly due to the fact that we don't rely on any heavy cryptographic primitives. Thus, we avoid the heavy homomorphic computation over the encrypted data. In addition, generation and transmission of garbled circuits are time-consuming, particularly for such data and computation intensive task. More importantly, by using the vectorization technique, we can maintain the data structure of the network and perform the operations in parallel to the maximum. The tables also exhibit a very low communication overhead between the parties in our scheme. This is for the same reason that we don't need to transmit large ciphertexts and garbled circuits.

TABLE 7
Comparison of the Runtimes and Message Sizes with Network II

Approach	Runtime (s)		Message sizes (MB)	
	Offline	Online	Offline	Online
MiniONN [9]	3.58	5.74	20.9	636.6
Our scheme	0.09	0.21	1.57	0.99

TABLE 8
Comparison of the Runtimes and Message Sizes with Network III

Approach	Runtime (s)		Message sizes (MB)	
	Offline	Online	Offline	Online
MiniONN [9]	472	72	3046	6226
Chameleon [10]	22.97	29.7	1210	1440
Our scheme	0.62	1.55	10.57	6.61

9 Conclusions

When designing a privacy-preserving CNN feature extraction scheme for mobile sensing, it is not easy to simultaneously satisfy the three requirements: privacy, accuracy, and efficiency. To achieve the privacy requirement, previous work mostly relies on the heavy cryptographic primitives. Due to the complexity of deep CNN, the accuracy and efficiency of their schemes were reduced inevitably.

In this paper, we proposed a novel lightweight framework integrating mobile sensing and edge computing. The privacy-preserving CNN feature extraction could be performed at the edge of the network. We first randomly split the images into shares and outsourced them to two edge servers respectively. By utilizing the secret-sharing based secure computation, we designed a series of secure interaction protocols corresponding to the different layers in CNN. Thus, we implemented the CNN feature extraction over the encrypted data. Moreover, we could guarantee low overhead on the mobile devices and low latency of the network by moving the data and processing to the edge. Through theoretical analysis and empirical experiments, we demonstrated the security, effectiveness, and efficiency of our scheme.

Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grant No. 61572026, 61672195, 61702105, 61772544, and 61872372, the Open Foundation of State Key Laboratory of Cryptology, and the Research Project of National University of Defense Technology.

REFERENCES

[1] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2014, pp. 806–813.

[2] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 991–999.

[3] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4565–4574.

[4] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. S. Shen, "Catalyzing cloud-fog interoperation in 5G wireless networks: An SDN approach," *IEEE Netw.*, vol. 31, no. 5, pp. 14–20, Sep. 2017.

[5] Z. Yan, J. Xue, and C. W. Chen, "Prius: Hybrid edge cloud and client adaptation for HTTP adaptive streaming in cellular networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 1, pp. 209–222, Jan. 2017.

[6] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-nets: Neural networks over encrypted data," *arXiv:1412.6181*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6181

[7] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," *IACR Cryptology ePrint Archive*, vol. 2017, 2017, Art. no. 35.

[8] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.

[9] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 619–631.

[10] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Security*, 2018, pp. 707–721.

[11] Z. Qin, J. Yan, K. Ren, and C. W. Chen, "Privacy-preserving outsourcing of image global feature detection," in *Proc. Global Commun. Conf.*, 2014, pp. 710–715.

[12] S. Wang, M. Nassar, M. Atallah, and Q. Malluhi, *Secure and Private Outsourcing of Shape-Based Feature Extraction*. Berlin, Germany: Springer, 2013.

[13] C.-Y. Hsu, C.-S. Lu, and S.-C. Pei, "Image feature extraction in encrypted domain with privacy-preserving SIFT," *IEEE Trans. Image Process.*, vol. 21, no. 11, pp. 4593–4607, Nov. 2012.

[14] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Towards efficient privacy-preserving image feature extraction in cloud computing," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 497–506.

[15] Q. Wang, S. Hu, K. Ren, J. Wang, Z. Wang, and M. Du, "Catch me in the dark: Effective privacy-preserving outsourcing of feature extractions over image data," in *Proc IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[16] Y. Bai, L. Zhuo, B. Cheng, and Y. F. Peng, "Surf feature extraction in encrypted domain," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2014, pp. 1–6.

[17] Q. Wang, S. Hu, J. Wang, and K. Ren, "Secure surfing: Privacy-preserving speeded-up robust feature extractor," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst.*, 2016, pp. 700–710.

[18] Q. Wang, J. Wang, S. Hu, Q. Zou, and K. Ren, "Sechog: Privacy-preserving outsourcing computation of histogram of oriented gradients in the cloud," in *Proc. 11th ACM Asia Conf. Comput. Commun. Security*, 2016, pp. 257–268.

[19] B. Ferreira, J. Rodrigues, J. Leitão, and H. Domingos, "Privacy-preserving content-based image retrieval in the cloud," in *Proc. Symp. Reliable Distrib. Syst.*, 2015, pp. 11–20.

[20] W. Lu, A. L. Varna, and M. Wu, "Confidentiality-preserving image search: A comparative study between homomorphic encryption and distance-preserving randomization," *IEEE Access*, vol. 2, pp. 125–141, 2014.

[21] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2594–2608, Nov. 2017.

[22] X. Yuan, X. Wang, C. Wang, A. C. Squicciarini, and K. Ren, "Towards privacy-preserving and practical image-centric social discovery," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 868–882, Sep./Oct. 2018.

[23] C. Hu, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang, "Outsourceable two-party privacy-preserving biometric authentication," in *Proc. 9th ACM Symp. Inf. Comput. Commun. Security*, 2014, pp. 401–412.

[24] L. Zhu, C. Zhang, C. Xu, X. Liu, and C. Huang, "An efficient and privacy-preserving biometric identification scheme in cloud computing," *IEEE Access*, vol. 6, pp. 19025–19033, Mar. 2018.

[25] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *Proc. Privacy Enhancing Technol.*, vol. 2018, no. 3, pp. 123–142, 2018.

[26] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *Proc. 38th IEEE Symp. Security Privacy*, 2017, pp. 19–38.

[27] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," *arXiv:1705.08963*, 2017. [Online]. Available: https://arxiv.org/abs/1705.08963

[28] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "Gazelle: A low latency framework for secure neural network inference," *arXiv:1801.05507*, 2018. [Online]. Available: https://arxiv.org/abs/1801.05507

[29] P. Mohassel and P. Rindal, "ABY 3: A mixed protocol framework for machine learning," in *Proc. Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 35–52.

[30] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 212–221, Jan. 2014.

[31] Q. Zhang, L. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1351–1362, May 2016.

[32] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. Allerton Conf. Commun. Control Comput.*, 2015, pp. 909–910.

[33] N. H. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: an application of human behavior prediction," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1309–1316.

[34] M. Abadi, A. Chu, I. Goodfellow, H. B. Mcmahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. ACM Sigsac Conf. Comput. Commun. Security*, 2016, pp. 308–318.

[35] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Foundations Comput. Sci.*, 1982, pp. 160–164.

[36] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen, "Asynchronous multiparty computation: Theory and implementation," in *Proc. Int. Workshop Public Key Cryptography*, 2009, pp. 160–179.

[37] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemson, "High-performance secure multi-party computation for data mining applications," *Int. J. Inf. Security*, vol. 11, no. 6, pp. 403–418, 2012.

[38] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: A system for secure multi-party computation," in *Proc. 15th ACM Conf. Comput. Commun. Security*, 2008, pp. 257–266.

[39] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *Proc. Eur. Symp. Res. Comput. Security*, 2008, pp. 192–206.

[40] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical covertly secure MPC for dishonest majority–or: Breaking the SPDZ limits," in *Proc. Eur. Symp. Res. Comput. Security*, 2013, pp. 1–18.

[41] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[42] G. R. Blakley, et al., "Safeguarding cryptographic keys," in *Proc. National Comput. Conf.*, vol. 48, 1979, pp. 313–317.

[43] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 805–817.

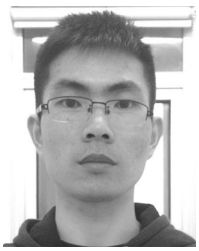
[44] J. Furukawa, Y. Lindell, A. Nof, and O. Weinstein, "High-throughput secure three-party computation for malicious adversaries and an honest majority," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2017, pp. 225–255.

[45] T. Araki, A. Barak, J. Furukawa, T. Lichter, Y. Lindell, A. Nof, K. Ohara, A. Watzman, and O. Weinstein, "Optimized honest-majority MPC for malicious adversaries?breaking the 1 billion-gate per second barrier," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 843–862.

[46] J. Wang, S. Hu, Q. Wang, and Y. Ma, "Privacy-preserving outsourced feature extractions in the cloud: A survey," *IEEE Netw.*, vol. 31, no. 5, pp. 36–41, Sep. 2017.

[47] J. Ning, Z. Cao, X. Dong, and L. Wei, "White-box traceable CP-ABE for cloud storage service: How to catch people leaking their access credentials effectively," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 5, pp. 883–897, Sep./Oct. 2018.

[48] J. Ning, J. Xu, K. Liang, F. Zhang, and E.-C. Chang, "Passive attacks against searchable encryption," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 789–802, Mar. 2019.
[49] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proc. Annu. Int. Cryptology Conf.*, 1991, pp. 420–432.
[50] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation," in *Proc. Theory Cryptography Conf.*, 2006, pp. 285–304.
[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, [Online]. Available: http://www.deeplearningbook.org.
[52] X. Liu, R. H. Deng, Y. Yang, H. N. Tran, and S. Zhong, "Hybrid privacy-preserving clinical decision support system in fog–cloud computing," *Future Generation Comput. Syst.*, vol. 78, pp. 825–837, 2018.
[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556, 2014.
[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[55] E. Orhan and X. Pitkow, "Skip connections eliminate singularities," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HkwBEMWCZ
[56] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. 42nd IEEE Symp. Foundations Comput. Sci.*, 2001, pp. 136–145.
[57] R. Canetti, A. Cohen, and Y. Lindell, "A simpler variant of universally composable security for standard multiparty computation," in *Proc. Annu. Cryptology Conf.*, 2015, pp. 3–22.
[58] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[59] D. Demmler, G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, and S. Zeitouni, "Automated synthesis of optimized circuits for secure computation," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1504–1517.



Kai Huang received the BS degree in network engineering and the MS degree in cryptography from the Naval University of Engineering, Wuhan, China, in 2007 and 2009, respectively. Currently, he is working toward the PhD degree in computer science at the College of Computer, National University of Defense Technology, Changsha, China. His research interests include cloud computing, machine learning, and network security.

Ximeng Liu (S'13-M'16) received the BSc degree in electronic engineering from Xidian University, Xi'an, China, in 2010, and the PhD degree in cryptography from Xidian University, China, in 2015. He was a research fellow at School of Information System, Singapore Management University, Singapore. Now, he is a full professor with the College of Mathematics and Computer Science, Fuzhou University, China. He has published more than 100 research articles include the *IEEE Transactions on Information Forensics and Security*, the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Industrial Informatics*, the *IEEE Transactions on Services Computing*, and the *IEEE Transactions on Cloud Computing*. His research interests include cloud security, applied cryptography and big data security. He is a member of the IEEE.

Shaojing Fu received the PhD degree in applied cryptography from the National University of Defense Technology, in 2012, and has studied at University of Tokyo for one year during his PhD He is currently an associate professor with the College of Computer, National University of Defense Technology. His research interests include cryptography theory and application, cloud storage security. He is a member of the IEEE.

Deke Guo (SM'17) received the BS degree in industry engineering from the Beijing University of Aeronautic and Astronautic, Beijing, China, in 2001, and the PhD degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. Currently, he is a professor with the College of Systems Engineering, National University of Defense Technology, Changsha, China. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE.

Ming Xu received the PhD degree from the National University of Defense Technology, Changsha, China, in 1995. Currently, he is a professor and the director of the Network Engineering Department in the College of Computer, National University of Defense Technology, Changsha, China. His research interests include mobile computing, wireless network, cloud computing and network security. He is a member of the IEEE.