PE-HEALTH: Enabling Fully Encrypted CNN for Health Monitor with Optimized Communication

Yang Liu[1], Yilong Yang[1], Zhuo Ma*[1], Ximeng Liu[2], Zhuzhu Wang[1] and Siqi Ma[3]
[1] Xidian University, [2] Fuzhou University, [3] The University of Queensland
bcds2018@foxmail.com, echotoken@gmail.com, mazhuo@mail.xidian.edu.cn, snbnix@gmail.com,
z.z.wang@foxmail.com, masiqi19910324@gmail.com

***Abstract*—Cloud-based Convolutional neural network (CNN) is a powerful tool for the healthcare center to provide health condition monitor service. Although the new service has future prospects in the medical, patient's privacy concerns arise because of the sensitivity of medical data. Prior works to address the concern have the following unresolved problems: 1) focus on data privacy but neglect to protect the privacy of the machine learning model itself; 2) introduce considerable communication costs for the CNN inference, which lowers the service quality of the cloud server. To push forward this area, we propose PE-HEALTH, a privacy-preserving health monitor framework that supports fully-encrypted CNN (both input data and model). In PE-HEALTH, the medical Internet of Things (IoT) sensor serves as the health condition data collector. For protecting patient privacy, the IoT sensor additively shares the collected data and uploads the shared data to the cloud server, which is efficient and suited to the energy-limited IoT sensor. To keep model privacy, PE-HEALTH allows the healthcare center to previously deploy, and then, use an encrypted CNN on the cloud server. During the CNN inference process, PE-HEALTH does not need the cloud servers to exchange any extra messages for operating the convolutional operation, which can greatly reduce the communication cost.**

***Index Terms*—Health Condition Monitor, CNN, Privacy-Preserving, Medical IoT Sensors**

I. INTRODUCTION

In these years, a phenomenal technique, machine learning, revolutionizes the health condition monitor field. Utilizing the emerging machine learning models, especially the convolutional neural network (CNN), the healthcare center can automatically and accurately monitor a patient's health condition in many aspects, like disease alerting [1], patient's activity recognition [2] and early detection of patient health deterioration [3]. Commonly, the health condition monitor system is based on the standard machine learning as a service (MLaaS) framework [4]–[6], shown in Fig. 1. In such an architecture, the Internet of Things (IoT) sensor is used to collect the physiological data of the patient. To guarantee service quality, the pre-trained machine learning model is deployed on the cloud server to process the data collected by the IoT sensors.

However, the problem is that the cloud server is usually provided by the third party and not always trustful. The collected physiological data of patients contain lots of valuable and private information, e.g., heartbeat, body temperature and blood pressure. If the data are directly transmitted in plaintext format, a malicious cloud server operator can use them to easily derive the patient's lifestyle, health status, and even personal preferences, which has a great attraction for the malevolent person and illegal market. In 2017, Amazon was pointed to leak more than 47GB personal health data, and the Health and Human Services apartment of USA reveals that only one person's data are worth at least 250 dollars[1]. Moreover, besides the patient data, a machine learning model is also a very precious resource. To obtain a well-trained model, the healthcare center may spend years and enormous amounts of money gathering valuable data and training the model. For example, to make sense of diabetes and insulin data in real time, IBM paid 2.6 billion dollars to buy out Truven Health, a healthcare analytics company in the last year. Therefore, a well-trained machine learning model can also be the "stealing target" of the malicious cloud server operator.

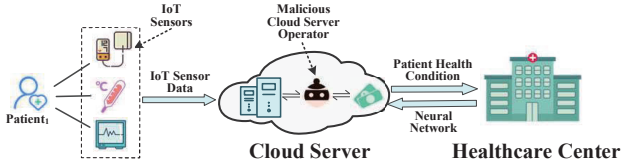


Fig. 1. A general framework for health condition monitor with IoT sensors.

Recent efforts to resolve the above problems are mainly based on three techniques, namely, homomorphic encryption (HE), secret sharing (SS), and differential privacy (DP). For HE, the exponentiation computations is the main burden for energy-limited IoT sensors in real-world applications. Compared with HE, the computations of SS are simpler for IoT sensors, which are just random value generation and computing some polynomials. Although complex computations are avoided, there is still a significant problem for SS, which is the high communication [7]. In the scenario where an emergency may occur (e.g., elder's activity monitor in the nursing home), the high communication can lead to the increased response latency of the cloud server and reduce the applicable range of the machine learning-based health monitor. DP is an emerg-

This work was supported by the National Natural Science Foundation of China (Grant No. 61872283, 61702105, U1764263, U1804263), the China 111 Project (No. B16037).

* Corresponding Author

978-1-7281-6887-6/20/$31.00 © 2020 IEEE

[1]https://www.slashgear.com/database-leaks-47gb-of-medical-records-including-names-and-test-results-10503457/

978-1-7281-6887-6/20/$31.00 ©2020 IEEE

ing privacy-preserving technique that can ideally overcome the drawbacks of the other two methods [8]. However, DP introduces random noise into the original data, and always causes noticeable performance loss [9], which is not tolerable in the medical field.

In this paper, we propose a privacy-preserving CNN based health condition monitor framework (PE-HEALTH). In order to avoid the high computation cost of HE and the high communication cost of SS, PE-HEALTH utilizes a series of hybrid protocols to process the physiological data collected by IoT sensors. Specifically, in PE-HEALTH, the IoT sensor does additive sharing on the collected data by generating random value, and then, uploads the shared data to two cloud servers, respectively. Using the CNN model encrypted by the healthcare center, the two cloud servers process the additively shared sensor data, judge a patient's health condition, and return the result to the healthcare center. Our contributions of this paper are summarized as follows.

- **Privacy-preserving Health Condition Monitor.** We propose PE-HEALTH, a privacy-preserving CNN based health condition monitor framework. In PE-HEALTH, the physiological data are processed in the third-party cloud servers in the additive secret share format, which avoids patient privacy breached by a malicious server.
- **Computable Encrypted CNN Model.** Besides patient privacy, PE-HEALTH can also protect healthcare center privacy, i.e., CNN model privacy. Specifically, a series of secure protocols are proposed to complete the inference of the convolutional layer, activation layer, pooling layer, and fully-connected layer of CNN. With these protocols, PE-HEALTH allows the healthcare center to deploy and use the encrypted CNN model on cloud servers.
- **Optimized Quality of Service.** Experiments conducted on three real-world datasets show that PE-HEALTH can achieve similar performance to the CNN model running without privacy protection (less than 0.1% accuracy loss). Moreover, compared with existing privacy-preserving CNN inference schemes, PE-HEALTH achieves great improvement in efficiency.

II. PRELIMINARY

In this section, we briefly review CNN and the concept of the cryptographic tool used in PE-HEALTH.

A. Convolutional Neural Network

PE-HEALTH aims at providing a privacy-preserving CNN scheme for health condition monitor. A standard CNN is composed of a sequence of layers that transform the feature vector into a fixed size output vector. There are four types of commonly used layers for CNN, convolutional layer, activation layer, pooling layer, and fully-connected layer.

- **Convolutional Layer.** The convolutional layer is the most important layer to extract features in a CNN model. A convolutional layer contains multiple filter kernels, each of which corresponds to a weight matrix W and a bias b. Suppose the kernel size to be $n \times n$. Every time the kernel slides on a feature map, it outputs a feature value $y = \sum_{i=0}^{n} \sum_{j=0}^{n} \omega_{i,j} x_{i,j} + b$, where $\omega_{i,j} \in W$.
- **Activation Layer.** Activation layer is mainly used after a convolutional or fully-connected layer to improve the nonlinear expressiveness of CNN. There are three kinds of commonly used activation functions in CNN, which are $sigmoid$, $tanh$, and the rectified linear unit (ReLU). Massive practices have proved that ReLU has the best performance in most applications and is recommended as the default activation function [10]. Therefore, PE-HEALTH deploys ReLU as the default activation layer.
- **Pooling Layer.** To reduce the size of the feature map and manifest the main features, the pooling layer is introduced into CNN. The core operation of the pooling layer is to partition a feature map into a set of non-overlapping sub-areas and performs downsampling on each sub-area. Two frequently used pooling functions are the max-pooling and mean-pooling, which output the maximum feature value and the average value of a sub-area, respectively.
- **Fully-connected Layer.** The neurons of a fully-connected layer are connected with a previous layer in an ordered manner. Define the weight and the bias of the connection between the i^{th} neuron of the previous layer and the j^{th} neuron of the current layer as $\omega_{i,j}$ and b_j. The output of the j^{th} neuron is $y_{i,j} = \sum_i \omega_{i,j} x_i + b_j$.

B. Cryptographic Primitives

Prior to introducing PE-HEALTH, we list the cryptographic primitives of PE-HEALTH below.

Homomorphic Encryption. Homomorphic encryption (HE) is a kind of encryption algorithm that supports computation over the encrypted data. PE-HEALTH uses HE to protect the privacy of CNN parameters. The HE algorithm used in PE-HEALTH is derived from [11], whose security is based on the DDH assumption [12]. Define the security parameter as 1^λ. To set up a pair of HE keys, we first run a PPT group generator $\mathcal{IG}$. The output of $\mathcal{IG}(1^\lambda)$ is $(\mathbb{G}, g)$, where $\mathbb{G}$ is a cyclic multiplicative group of a prime order q and g is the group generator of $\mathbb{G}$. Such a $\mathcal{IG}$ can be easily found according to [11]. Then, we can pick a random number $sk \in \mathbb{Z}_q$ as the private key and computes $pk = (\mathbb{G}, g, g^{sk})$ as the public key, where q is a big prime.

To encrypt an arbitrary integer $\omega \in \mathbb{Z}_q$, an encryption function $\texttt{Enc}(pk, \omega)$ is defined. In the function, $\texttt{Enc}$ first randomly select a number $r \in \mathbb{Z}_q$. Then, compute $\alpha = g^r$ and $\beta = g^\omega \cdot pk^r$. Finally, $\texttt{Enc}$ outputs (α, β), where $\alpha, \beta \in \mathbb{G}$. For brevity, we use $[\![\omega]\!]$ to represent (α, β) in the following paper. Notably, since the decryption function is not used in PE-HEALTH, we omit its definition.

Additive Secret Sharing. Additive secret sharing (ASS) is a kind of secret sharing method that is commonly used in the multi-party computation [13]. PE-HEALTH uses additive secret sharing to protect the privacy of the data collected by IoT sensors and the intermediate computation results. To additively share an arbitrary secret value $x \in \mathbb{Z}_q$, we define

a sharing function Share. To achieve Share, we sample a pseudo-random function ψ. Given an arbitrary seed, ψ can output a series of uniformly random values within $\mathbb{Z}_q$. We use ψ to randomly select a random value $x' \in \mathbb{Z}_q$, and compute $x'' = x - x' \mod q$. The output of Share is (x', x''), where $x', x'' \in \mathbb{Z}_q$ and $x' + x'' = x \mod q$. For brevity, $\langle x \rangle$ is used to represent (x', x'') in the following paper.

Data Format. From the above definitions, the valid input and output of the cryptographic primitives are both integers within a limited range. However, the intermediate computation results involved in CNN can be floating values. To resolve the problem, we adopt the fix-point data format. Assume a publicly known precision degree d. Given an arbitrary number $x_0 \in \mathbb{R}$ and $x_0 < \lfloor q/10^d \rfloor$, we represent it as $x = \lfloor x_0 \cdot 10^d \rfloor \mod q$. $\lfloor \cdot \rfloor$ is an integral function. Thus, x_0 is sampled into $\mathbb{Z}_q$. Also, we can recover the original addition result of two fixed-point values by computing $x_0 = x \cdot 10^{-d}$ (or multiplying 10^{-2d} for multiplication). For example, set $d = 1$, $x_0 = 1.23$ and $q = 127$. x_0 can represented as an integer $x = \lfloor 1.23 \cdot 10^1 \rfloor = 12 \mod 127$.

Remark: In PE-HEALTH, we assume that all sensor data, CNN parameters and intermediate results are stored, transmitted and computed in the above data format by default. The integral function used in the data format conversion may introduce some computation errors. To lower the computation error, we can sample a bigger prime q or precision degree d.

III. SYSTEM MODEL

In this section, we introduce the entities and the security model of PE-HEALTH.

A. System Model

As illustrated in Fig. 2, PE-HEALTH is composed of the following three kinds of entities.

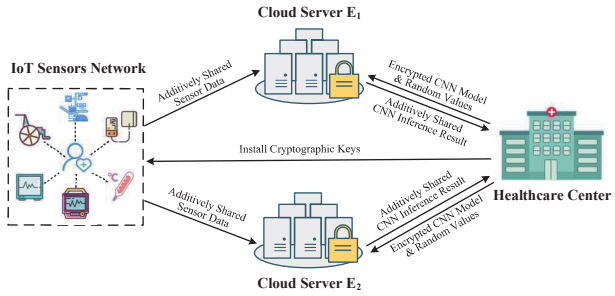


Fig. 2. System Model of PE-HEALTH.

- **Health Monitor Sensors (HMS).** HMS is the IoT sensor used to collect the physiological data of the patient, e.g., heart rate and blood pressure After conducting a simple additive secret sharing operation on the collected data, HMS sends the shared data to E_1 and E_2, respectively.
- **Healthcare Center (HC).** HC owns a pre-trained CNN model that can use the data collected from HMS to infer the health condition of a patient. In PE-HEALTH, the CNN model is previously and encrypted and deployed on the cloud servers. The inference result is returned to HC in the format of additive shares.
- **Cloud Servers.** There are two cloud servers (E_1 and E_2) involved in PE-HEALTH. Through operating the predefined secure protocols, E_1 and E_2 collaboratively finish the computations of CNN inference with the secretly shared sensor data and the encrypted CNN model.

B. Security Model

PE-HEALTH adopts the *curious-but-honest* security model. According to the model, PE-HEALTH follows the following assumptions. The adversary $\mathcal{A}$ can corrupt one of the cloud servers (E_1 or E_2). The corrupted cloud server honestly performs the secure protocols as promised, yet is also curious about the data collected by HMS and the CNN model parameters deployed by HC. Further, we assume that E_1 and E_2 cannot collude with each other, that is, E_1 cannot tell any information about the received messages to E_2, and vice versa. The assumption can be implemented in a real-world application by renting two cloud servers from two different cloud computing platforms, like AWS's Lambda@Edge and Microsoft Azure IoT Edge. HMS and HC are both *honest*. There are secure communication channels between any two *honest* entities. The assumption is reasonable in our application scenario because HMS is usually provided by HC and stays in a physically secure environment (like a nursing home). Moreover, PE-HEALTH is designed to achieve the following three security goals.

1) **Patient Privacy.** Since the physiological data is sensitive and valuable medical data for a patient, the first goal of PE-HEALTH is to prevent the physiological data collected by HMS or the features extracted from the data being revealed to an untrusted cloud server.
2) **Healthcare Center Privacy.** To get a well-trained CNN model, HC has to spend a lot of time and money on collecting training data and model training. Therefore, the second goal of PE-HEALTH is to prevent the model parameters of CNN from being revealed to an untrusted cloud server. Like all prior work [7], [14], [15], PE-HEALTH does not hide information about the structure of the network, such as the type of each layer in the network, which is independent of the patient data.
3) **CNN Inference Privacy.** The CNN inference result also contains patient private information. For example, knowing that the deployed model is used to detect Alzheimer's disease, a malicious server can use the inference result to judge whether a specific person has the disease. Therefore, the third goal of PE-HEALTH is to prevent the inference result of CNN from being revealed to an untrusted cloud server.

C. Design Rationale

Many prior work that protects the data privacy of CNN fall into the ASS based framework with two non-colluded cloud servers [16]. Compared with the pure HE based framework, the ASS based frameworks are much more lightweight and can better meet the practical requirement for the quality of service.

However, we observe that to implement the convolutional operation of CNN, all the ASS frameworks choose to use the Beaver's triplet [17] to implement secure multiplication. For such a design rationale, there are two drawbacks: 1) every time the multiplication operation is conducted, the two cloud servers have to exchange a couple of messages with each other, which greatly increases the communication cost; 2) the model privacy is ignored, which makes them not secure enough in our strengthened security model.

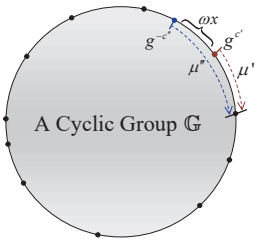


Fig. 3. The conversion procedure from the multiplicative shares $(g^{c'}, g^{c''})$ to additive shares (μ', μ''). The black dots are distinguished points.

To resolve the problem, PE-HEALTH introduces a hybrid method that combines HE and ASS to implement secure multiplication. The core of the method is using the HE algorithm to encrypt each layer's weight parameter $(\alpha, \beta) = [\![\omega]\!] \leftarrow \mathtt{Enc}(pk, \omega)$, and additively sharing the input $\langle x \rangle \leftarrow \mathtt{Share}(x)$ and $\langle sx \rangle \leftarrow \mathtt{Share}(sk \cdot x)$, where sx is the multiplication of x and sk. Thus, the two cloud servers can locally compute a pair of multiplicative shares $(g^{c'}, g^{c''}) \leftarrow \alpha^{\langle sx \rangle} \cdot \beta^{\langle x \rangle}$, where $g^{c'} \cdot g^{c''} = g^{\omega x}$. Now, what stops us from continuing the computation of a convolutional layer is to get $\langle \omega x \rangle$, i.e., converting the multiplicative shares to additive shares. PE-HEALTH achieves the conversion by referring to the idea of [18]. Pick a δ-sparse subset $\mathbb{G}' \subset \mathbb{G}$ by including each $h \in \mathbb{G}$ independently with probability δ. Sample a pseudo-random function $\varphi : \mathbb{G} \to \{0, ..., \lfloor 2M/\delta \rfloor\}$, where M is upper bound of the shared value and the element $h \in \mathbb{G}$ satisfying $\varphi(h) = 0$ is called the distinguished point. The cloud servers iteratively find $\mu', \mu'' \in \mathbb{Z}_q$ satisfying $\varphi(g^{\mu'} \cdot g^{c'}) = \varphi(g^{\mu''} \cdot g^{-c''}) = 0$, respectively (Protocol 2, line 3-9). Loosely speaking, the iteration returns the distance between $(g^{c'}, g^{c''})$ and a distinguished point in $\mathbb{G}$, shown in Fig. 3. Taking the steps, the cloud servers get (μ', μ'') (with good probability), where $\mu' + \mu'' = \omega \cdot x$. Failure occurs if the distinguished point lies between the two multiplicative shares. Without loss of generality, Proposition 1 is derived.

PROPOSITION 1. *Given a cyclic group $\mathbb{G}$, $\delta > 0$, $m \in \mathbb{N}$, $M < ord(\mathbb{G})$ and a uniformly sample pseudo-random function φ, we can convert the multiplicative shares $(g^{c'}, g^{c''})$ to the additive shares (c', c'') with a probability greater than $(1-\delta)$.*

Using the above method, PE-HEALTH reduces the communication cost by avoiding the message exchange between the cloud servers while performing the convolutional operation. Further, we notice that our method introduces a couple of extra computation cost and error probability. Nevertheless, compared with the communication resource, the cloud-computing service provider usually does not consider computation resource a serious problem.

IV. SYSTEM DESCRIPTION OF PE-HEALTH

In this section, we describe how to setup PE-HEALTH and present the detailed workflow of PE-HEALTH, shown in Fig. 4.

A. System Setup

Before being applied to applications, PE-HEALTH has to previously set up the cryptographic keys and deploy the pre-trained CNN model on the cloud server, illustrated in Protocol 1. Both the two steps can be finished *offline*, and thus, its computational cost is trivial for applications.

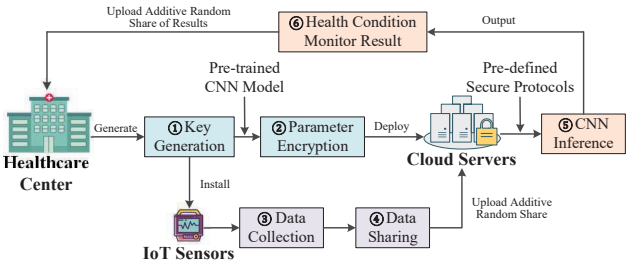


Fig. 4. High-level System Description of PE-HEALTH.

Key Setup. In this step, PE-HEALTH installs the cryptographic keys for operating HE. To set up a pair of HE keys, HC first defines a security parameter 1^λ. Then, HC runs the PPT group generator to sample the ElGamal parameters, $(\mathbb{G}, g) \leftarrow \mathcal{IG}(1^\lambda)$. Finally, HC picks a random number $sk \in \mathbb{Z}_q$ as the secret key and computes $pk \leftarrow (\mathbb{G}, g, g^{sk})$ as the public key. The public key is distributed to all parties. The secret key is installed on all HMSs, and cannot be known by E_1 and E_2.

CNN Parameter Setup. To avoid model privacy leakage, HC encrypts the parameters of the pre-trained CNN model with the public key generated in the key setup step. The encrypted model is sent to both E_1 and E_2.

Suppose HC pre-trains a CNN model for health condition monitor, which is expressed as $Net :=$ Input $\to$ [[Conv $\to$ ReLU] $\times\, u \to$ Pool] $\times\, v \to$ [FC $\to$ ReLU] $\times\, t \to$ FC $\to$ Output. Conv, ReLu, Pool and FC are corresponding to the convolutional layer, activation layer, pooling layer and fully-connected layer, respectively. The notation $\times$ indicates repetitions, and u, v and t represent the number of repetitions. Among the four kinds of layers, only Conv and FC have parameters, namely weights ω_l and bias b_l, where $l \in \{$Conv $\in Net$, FC $\in Net\}$. For different layers of Net, HC sends them to the cloud servers in different ways. Towards a Conv or a FC layer, HC first invokes the HE algorithm to encrypt the weight $[\![\omega_l]\!] \leftarrow \mathtt{Enc}(pk, \omega_l)$, and then, uses the secret sharing function to split the bias $\langle b_l \rangle \leftarrow \mathtt{Share}(b_l)$. Next, HC encrypts the product of each bit of the secret key $sk^{(\jmath)}$ and the weight, $[\![sk^{(\jmath)} \cdot \omega_l]\!] \leftarrow \mathtt{Enc}(pk, sk^{(\jmath)} \cdot \omega_l)$, where $sk^{(\jmath)}$ expresses the $\jmath$ bit of sk. $[\![sk^{(\jmath)} \cdot \omega_l]\!]$ are used in the subsequent computations of CNN inference. The bit-wise expression of sk is mainly used to keep the input of each layer in the same

encryption format, i.e., the additively shared format. All the encryption results are sent to E_1 and E_2 along with the layer type and its position in Net. In this way, PE-HEALTH avoids the CNN parameters to be revealed to the cloud servers in the plaintext format. For a ReLU or a Pool layer, HC directly sends the layer type and its position in Net to both E_1 and E_2, because they do not have parameters required to be protected.

Protocol 1 System Setup (SecSetup)

Input: A PPT group generator $\mathcal{IG}$.

Key Setup:

1: HC samples a DDH-hard cyclic group $(\mathbb{G}, g) \leftarrow \mathcal{IG}(1^\lambda)$ of prime order q.
2: From $\mathbb{Z}_q$, HC randomly selects secret key $sk \leftarrow \mathbb{Z}_q$ and computes its corresponding public key $pk \leftarrow g^{sk}$.

CNN parameter Setup:

1: HC pre-trains a CNN model Net, which can be used for health condition monitor with physiological data.
2: **for** each layer of Net **do**
3: If the current layer is ReLu or Pool, HC sends the layer type and position to E_1 and E_2.
4: If the current layer is Conv or FC, HC does the following computations.
5: Encrypt the weight matrix $[\![\omega_l]\!] \leftarrow \texttt{Enc}(pk, \omega_l)$.
6: **for** each bit of the secret key $sk^{(\jmath)}$ **do**
7: Encrypt $[\![sk^{(\jmath)} \cdot \omega_l]\!] \leftarrow \texttt{Enc}(pk, sk^{(\jmath)} \cdot \omega_l)$.
8: **end for**
9: Secretly share the bias $\langle b_l \rangle \leftarrow \texttt{Share}(b_l)$.
10: Besides the layer type and position, HC sends $[\![\omega_l]\!]$ and $[\![sk^{(\jmath)} \cdot \omega_l]\!]$ to both E_1 and E_2, and sends the two random shares of $\langle b_l \rangle$ to E_1 and E_2, respectively.
11: **end for**

B. Sensor Data Upload.

After the system setup is set up, HMS can perform the following computations to securely upload the collected data.

To preserve the privacy of patients, HMS locally operates additive secret sharing on the collected data. Assume that to monitor a patient's health, τ sensors sample the patient's physiological data. Thus, the physiological features of a specific patient u_i can be expressed as $\mathcal{X}_i \leftarrow (x_{i,1}, x_{i,2}, ..., x_{i,\tau})$, where the j dimension of $\mathcal{X}_i$ is the physiological data collected by the sensor $s_{i,j} \in$ HMS. To secretly share $\mathcal{X}_i$, each $s_{i,j} \in$ HMS first computes $\langle x_{i,j} \rangle \leftarrow \texttt{Share}(x_{i,j})$ and $\langle sk \cdot x_{i,j} \rangle \leftarrow \texttt{Share}(sk \cdot x_{i,j})$, where $\langle x_{i,j} \rangle \leftarrow (x'_{i,j}, x''_{i,j})$ and $\langle sk \cdot x_{i,j} \rangle \leftarrow (sx'_{i,j}, sx''_{i,j})$. Then, $(x'_{i,j}, sx'_{i,j})$ and $(x''_{i,j}, sx''_{i,j})$ are sent to E_1 and E_2, respectively. Thus, E_1 can construct the following two vectors of random shares about one patient.

$$\mathcal{X}'_i \leftarrow (x'_{i,1}, x'_{i,2}, ..., x'_{i,\tau}), \tag{1}$$

$$(sk \cdot \mathcal{X}_i)' \leftarrow (sx'_{i,1}, sx'_{i,2}, ..., sx'_{i,\tau}). \tag{2}$$

Similarly, E_2 can also obtain two vectors, which contains the other part of random shares.

C. Secure CNN Inference

After receiving the uploaded data, PE-HEALTH uses them as inputs to complete CNN inference. A typical CNN model usually comprises four types of layers (mentioned in Section II). To implement secure CNN inference, PE-HEALTH proposes a series of secure protocols to complete the computations of these layers. The secure protocols contain secure convolutional protocol (SecConv), secure fully-connected protocol (SecFC), secure pooling protocol (SecPool) and secure activation protocol (SecReLU), respectively. When all layers of a CNN model is completed, the inference result is returned to HC in the format of additive shares, and HC can recover the plaintext result by directly adding them together. The following are the details of the four protocols.

Protocol 2 Secure Convolution Protocol (SecConv)

Input: The additively shared feature map $\langle \mathcal{X}_{conv} \rangle$ and $\langle sk \cdot \mathcal{X}_{conv} \rangle$; an encrypted filter $[\![\omega_{conv}]\!]$ and $[\![sk^\jmath \cdot \omega_{conv}]\!]$ whose size is $W \times H$; the additively shared bias $\langle b_{conv} \rangle$; a pseudo-random function $\varphi(\cdot)$.

1: **for** $1 \le i \le W$ and $1 \le j \le H$ **do** # $\omega_{i,j} \in \omega_{conv}$
2: For each $[\![\omega_{i,j}]\!]$, E_1 and E_2 compute $g^{c'} \leftarrow \alpha_{i,j}^{-sx'_{i,j}} \cdot \beta_{i,j}^{x'_{i,j}}$ and $g^{c''} \leftarrow \alpha_{i,j}^{-sx''_{i,j}} \cdot \beta_{i,j}^{x''_{i,j}}$, respectively.
3: E_1 sets $\mu' \leftarrow 0$ and $\gamma' \leftarrow \varphi(g^{c'})$.
4: **while** $\gamma' \neq 0$ and $\mu' < \Theta$ **do**
5: Update $\mu' \leftarrow \mu' + 1$ and $\gamma' \leftarrow \varphi(g^{c'} \cdot g^{\mu'})$.
6: **end while**
7: E_1 computes $\mu' \leftarrow (-\mu') + b'_{conv}$.
8: Meanwhile, E_2 sets $\mu'' \leftarrow 0$, $\gamma'' \leftarrow \varphi(g^{-c''})$, and also operates the above iteration to obtain the updated μ''.
9: E_2 computes $\mu' \leftarrow \mu'' + b''_{conv}$.
10: For each $[\![sk^{(\jmath)} \cdot \omega_{i,j}]\!]$, E_1 and E_2 repeat the computations of **line 3-9** and obtain $(s\mu'_\jmath, s\mu''_\jmath)$, $1 \le \jmath \le \ell$.
11: E_1 and E_2 compute $s\mu' \leftarrow \sum_{\jmath=1}^{\ell} 2^{\jmath-1} \cdot s\mu'_\jmath$ and $s\mu'' \leftarrow \sum_{\jmath=1}^{\ell} 2^{\jmath-1} \cdot s\mu''_\jmath$, respectively.
12: **end for**
13: Return $\langle \mu \rangle \leftarrow (\mu', \mu'')$ and $\langle s\mu \rangle \leftarrow (s\mu', s\mu'')$.

Secure Convolutional Protocol. SecConv is mainly used to complete the convolutional operation between the filter (a weight matrix) and the input feature matrix. Define the inputs of SecConv to be $\langle \mathcal{X}_{conv} \rangle$ and $\langle sk \cdot \mathcal{X}_{conv} \rangle$, $[\![\omega_{conv}]\!]$, $[\![sk^{(\jmath)} \cdot \omega_{conv}]\!]$ and $\langle b_{conv} \rangle$. The additively shared feature map, $\langle \mathcal{X}_{conv} \rangle$ and $\langle sk \cdot \mathcal{X}_{conv} \rangle$, are from the previous layer or HMS. The encrypted filter $[\![\omega_{conv}]\!]$, $[\![sk^{(\jmath)} \cdot \omega_{conv}]\!]$ and the bias $\langle b_{conv} \rangle$ are uploaded in the system setup stage. Suppose the size of ω_{conv} to be $W_c \times H_c$. The computation goal of SecConv is to compute the convolution between each filter and different parts of the input feature map according to the following equation.

$$g^{c'} \leftarrow \alpha_{i,j}^{-sx'_{i,j}} \cdot \beta_{i,j}^{x'_{i,j}} \text{ and } g^{c''} \leftarrow \alpha_{i,j}^{-sx''_{i,j}} \cdot \beta_{i,j}^{x''_{i,j}}, \tag{3}$$

where $1 \le i \le W_c$, $1 \le j \le H_c$, $(x'_{i,j}, x''_{i,j}) \leftarrow \langle x_{i,j} \rangle$, $(sx'_{i,j}, sx''_{i,j}) \leftarrow \langle sk \cdot x_{i,j} \rangle$, $(\alpha_{i,j}, \beta_{i,j}) \leftarrow [\![\omega_{i,j}]\!]$, $\omega_{i,j}$ is

the weight value of the filter ω_{conv} and $x_{i,j}$ belongs to the region of $\mathcal{X}_{conv}$ that corresponds to the filter. After computing Eq. 3, the convolution result is expressed as a pair of multiplicative shares satisfying $g^{c'} \cdot g^{c''} \leftarrow g^{\omega \cdot x_{i,j}}$. Using the method mentioned in Section III-C, E_1 and E_2 iteratively convert the multiplicative shares to additive ones. $\Theta < ord(\mathbb{G})$ is the maximum iteration number. μ' and μ'' satisfy $\mu'' - \mu' = c' + c'' = \omega_{i,j} \cdot x_{i,j} \mod q$. Finally, E_1 and E_2 updates μ' and μ'' with the secretly shared bias $\mu' \leftarrow (-\mu') + b'_{conv}$ and $\mu'' \leftarrow \mu'' + b''_{conv}$ and obtain a pair of additive random shares satisfying $\mu' + \mu'' = \omega_{i,j} \cdot x_{i,j} \mod q$.. However, to provide valid inputs for the next layer, SecConv still has to compute $\langle sk \cdot \mu \rangle$. To achieve this, E_1 and E_2 utilize $\langle x_{i,j} \rangle$ and $[\![sk^{(\jmath)} \cdot \omega_{conv}]\!]$ as inputs and repeat the above process to get $(s\mu'_\jmath, s\mu''_\jmath) \leftarrow \langle sk^{(\jmath)} \cdot \omega_{i,j} \cdot x_{i,j} \rangle$. After this, E_1 and E_2 calculate:

$$s\mu' \leftarrow \sum_{\jmath=1}^{\ell} 2^{\jmath-1} \cdot s\mu'_\jmath \text{ and } s\mu'' \leftarrow \sum_{\jmath=1}^{\ell} 2^{\jmath-1} \cdot s\mu''_\jmath. \quad (4)$$

where ℓ is the data length of sk and $s\mu' + s\mu'' = sk \cdot \omega_{i,j} \cdot x_{i,j} \mod q$. As shown in Protocol 2, all computations of SecConv can be locally completed.

Secure Full-connected Protocol. The computation of the FC layer is the same as the Conv layer, which is the dot product between the weight matrix and the input matrix. Therefore, SecFC can be implemented in the same way as SecConv. For brevity, we omit the details of SecFC.

Secure Activation Protocol. As mentioned in Section II, the recommended activation function for CNN is ReLU. Other types of activation functions may suffer from serious vanishing gradient problem and reduce the learning ability of CNN. Therefore, we only give a detailed implementation of SecReLU in PE-HEALTH, outlined in Protocol 3. The ReLU layer is computed through the sign function $\texttt{Sign}(\cdot)$.

$$sign(x) \leftarrow \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases}. \quad (5)$$

To implement the comparison operation of $\texttt{Sign}(\cdot)$, we refer to the additive secret sharing based secure comparison protocol (SecCmp) in [19]. Given two pairs of additive random share $\langle \mu \rangle$ and $\langle \nu \rangle$, SecCmp outputs $\langle f \rangle$. If $\mu < \nu$, f is 1, and otherwise, f is 0. The following is the details of SecReLU.

Protocol 3 Secure ReLU Protocol (SecReLU)

Input: The feature map $\langle \mathcal{X}_{relu} \rangle$ and $\langle sk \cdot \mathcal{X}_{relu} \rangle$;

1: **for** each element of $\mathcal{X}_{relu}$, $x_{i,j} \in \mathcal{X}_{relu}$ **do**
2: E_1 and E_2 compute $comp \leftarrow \texttt{SecCmp}(\langle x_{i,j} \rangle, 0)$.
3: **if** $comp$ is 1 **then**
4: Set $x'_{i,j} \leftarrow 0$, $x''_{i,j} \leftarrow 0$, $sx'_{i,j} \leftarrow 0$ and $sx'_{i,j} \leftarrow 0$.
5: **else**
6: $\langle x_{i,j} \rangle$ and $\langle sk \cdot x_{i,j} \rangle$ retains its original value.
7: **end if**
8: **end for**
9: Return the updated $\langle x_{i,j} \rangle$ and $\langle sk \cdot x_{i,j} \rangle$.

Since ReLU does not have training parameters, the input of SecReLU only contains $\langle \mathcal{X}_{relu} \rangle$ and $\langle sk \cdot \mathcal{X}_{relu} \rangle$ from the previous layer. For each $x_{i,j} \in \mathcal{X}_{relu}$, SecReLU lets E_1 and E_2 compute $\textsf{SecCmp}(\langle x_{i,j} \rangle, 0)$. If the returned result is 1, $x_{i,j}$ is less than 0. $\langle x_{i,j} \rangle$ and $\langle sk \cdot x_{i,j} \rangle$ are both updated to be 0. Otherwise, $\langle x_{i,j} \rangle$ and $\langle sk \cdot x_{i,j} \rangle$ retain their original values. After all elements of $\langle \mathcal{X}_{relu} \rangle$ are traversed, SecReLU outputs the updated $\langle \mathcal{X}_{relu} \rangle$ and $\langle sk \cdot \mathcal{X}_{relu} \rangle$.

Secure Pooling Protocol. In the pooling layer, CNN partitions the feature map into many small sub-areas and conducts the average operation (mean pooling) or maximum operation (max pooling) on each sub-area. Suppose the size of the sub-area to be $W_p \times H_p$ and the inputs of SecPool are $\langle \mathcal{X}_{pool} \rangle$ and $\langle sk \cdot \mathcal{X}_{pool} \rangle$. For mean pooling, E_1 can locally complete the computations of each sub-area $\mathcal{X}_{sub} \in \mathcal{X}_{pool}$ by computing $y' \leftarrow (W_p \cdot H_p)^{-1} \sum_{i=1}^{W_p} \sum_{j=1}^{H_p} x'_{i,j}$ and $sy' \leftarrow (W_p \cdot H_p)^{-1} \sum_{i=1}^{W_p} \sum_{j=1}^{H_p} sx'_{i,j}$,, where $x_{i,j} \in \mathcal{X}_{sub}$. Meanwhile, E_2 does the same operations on $x''_{i,j}$ and $x''_{i,j}$ and obtain y'' and sy''. The output of SecPool for each sub-area is $\langle y \rangle = (y', y'')$ and $\langle sk \cdot y \rangle = (sy', sy'')$. For max pooling, SecPool can be implemented by invoking SecCmp for $W_p \times H_p - 1$ times and finding the index corresponding to the maximum feature value. The final output is $\langle y \rangle$ and $\langle sk \cdot y \rangle$, where $y \leftarrow \arg\max_{x_{i,j} \in \mathcal{X}_{sub}} x_{i,j}$.

V. SECURITY ANALYSIS

In this section, we prove the security of PE-HEALTH. Assume the adversary to be $\mathcal{A} \in \{E_1, E_2\}$. We say that PE-HEALTH is ideally secure if $\mathcal{A}$ cannot obtain more information from the received protocol messages than a series of random values with the same data length. Based on the above discussion, a formal definition of PE-HEALTH is given below.

DEFINITION 1. *A protocol π of* PE-HEALTH *is secure if there exists a simulator ξ that can simulate $Ideal(\pi, \xi)$ which is computationally indistinguishable from the real view $Real(\pi, \mathcal{A})$ of the adversary $\mathcal{A} = \{E_1, E_2\}$.*

Additionally, the pseudorandom functions and HE function used in PE-HEALTH satisfy the following two definitions.

DEFINITION 2. *There exist two pseudorandom functions ψ and φ for* PE-HEALTH. *ψ can ideally generate a series of uniformly random values within $\mathbb{Z}_q$. φ can ideally map DDH elements of $\mathbb{G}$ to random values within $\mathbb{Z}_q$.*

DEFINITION 3 [18] *Let $\mathbb{G}$ be a DDH-hard group of prime order q with generator $g \in \mathbb{G}$, and $\ell = \lfloor \log q \rfloor$. The HE used in* PE-HEALTH *is secure based on the DDH-hard assumption.*

Here, the Decisional Diffie-Hellman assumption (DDH) holds if there exists a PPT generator $\mathcal{G}(1^\lambda)$ that can sample a group generator and a corresponding cyclic group $\mathbb{G}$ with a prime order q. The sampled two variables satisfy that for every nonuniform polynomial-time algorithm $\mathcal{P}$, there is a negligible function ϵ such that:

$$|Pr[(\mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda); (a, b) \leftarrow \mathbb{Z}_q^2 : \mathcal{P}(\mathbb{G}, g, g^a, g^b, g^{ab}) = 1] - Pr[(\mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda); (a, b, c) \leftarrow \mathbb{Z}_q^3 : \mathcal{P}(\mathbb{G}, g, g^a, g^b, g^c) = 1]| \leq \epsilon(\lambda).$$

$\mathcal{G}(1^\lambda)$ is exactly $\mathcal{IG}(1^\lambda)$, defined for key generation of PE-HEALTH. g^a, g^b, g^c are corresponding to α, β and pk of Enc. Therefore, we say that the HE algorithm of PE-HEALTH conforms to the DDH assumption and Definition 3 holds. Moreover, we state several lemmas that are essential to the security proofs of the protocols in PE-HEALTH.

LEMMA 1 [20] *If a random element r is uniformly distributed on $\mathbb{Z}_q$ and independent from any variable $x \in \mathbb{Z}_q$, then $r \pm x$ is also uniformly random and independent from x.*

LEMMA 2 [21] *A protocol is perfectly simulatable if all its sub-protocols are perfectly simulatable.*

The detailed proofs for Lemma 1 and Lemma 2 can be referred to [20] and [21]. Based on the above lemmas, we prove the security of the protocols in PE-HEALTH as follows. Note that in the proofs, an ideal functionality $\mathcal{F}$ is assumed to be available for the simulator to operate all the cryptographic functions defined in PE-HEALTH.

THEOREM 1. *For SecSetup, there exists a simulator ξ that can generate an ideal view that is computationally distinguishable from the real view for the adversary $\mathcal{A} = \{E_1, E_2\}$.*

proof. In the key set step, no message is exchanged for E_1 and E_2. Therefore, the proof is focused on the CNN parameter setup step. From the protocol definition, the real view of E_1 is $Real = \{[\![\omega_l]\!], [\![sk^\jmath \cdot \omega_l]\!], b'_l\}$, $l \in Net$. $[\![\omega_l]\!]$ and $[\![sk^\jmath \cdot \omega_l]\!]$ are composed of four random DDH elements of $\mathbb{G}$. $\langle b_l \rangle$ comprises two random shares belonging to $\mathbb{Z}_q$. Therefore, ξ can simply simulate an ideal view of SecSetup by asking $\mathcal{F}$ to the following computations. First, sample three random values from $\mathbb{Z}_q$ as the dummy secret key sk, ω_l and b_l. Then, ξ can generate the elements corresponding to $Real$ by operating Enc and Share. Based on Definition 3, $\mathcal{A}$ cannot identify whether the encrypted CNN parameters are dummy or not in polynomial time. Based on Definition 2, the dummy secret shares and the real secret shares are both uniform random, and therefore, computationally indistinguishable for $\mathcal{A}$. For E_2, the only difference of the real view is that b'_l is changed to b''_l. ξ can use the same way to simulate it. Therefore, the simulator ξ defined in Theorem 1 exists and SecSetup is secure. □

THEOREM 2. *For SecConv or SecFC, there exists a simulator ξ that can generate an ideal view that is computationally distinguishable from the real view for the adversary $\mathcal{A} = \{E_1, E_2\}$.*

proof. Since SecConv and SecFC achieve the same kind of computation, we prove their security at the same time. From protocol definitions, the real view of E_1 for both the two protocols can be expressed as $Real = \{[\![\omega_l]\!], [\![sk^\jmath \cdot \omega_l]\!], b'_l\}$, which is the same as SecSetup. This is because there is no message exchanged in the two protocols and E_1 can locally complete the computations. The proof of Theorem 1 has shown that $[\![\omega_l]\!]$ and $[\![sk^\jmath \cdot \omega_l]\!]$ and b'_l are simulatable. And based on Definition 3 and Lemma 1, the further computation result about these values cannot reveal more information than the ciphertexts or secret shares themselves. Similarly, the simulated view for E_2 is also computationally indistinguishable from the real view o. In conclusion, the simulator ξ defined in Theorem 2 exists, SecConv and SecFC are secure. □

THEOREM 3. *For SecReLU or SecPool, there exists a simulator ξ that can generate an ideal view that is computationally distinguishable from the real view for the adversary $\mathcal{A} = \{E_1, E_2\}$.*

proof. If the pooling layer is max pooling, the involved computations of SecReLU and SecPool are identical, i.e., secure comparison. However, if the pooling layer is mean pooling, the computations of the two protocols make a difference. Therefore, we first prove that the mean-pooling is secure, and then, state the security of max-pooling and SecReLU. For mean pooling, the real view of E_1 contains a series of summing result, y' and sy', of secret shares $x'_{i,j} \in \mathbb{Z}_q$ and $sx'_{i,j} \in \mathbb{Z}_q$ for different sub-areas of the input, where $1 \le i \le W_p$, $1 \le i \le H_p$, W_p and H_p are the size of the sub-area. $x'_{i,j}$ and $sx'_{i,j}$ can be simulated in the same way as Theorem 1. Based on Lemma 1, y' is still uniformly random and cannot reveal any more information than $x'_{i,j}$ or $sx'_{i,j}$. Consequently, the simulated view of E_1 is indistinguishable from the real view of $\mathcal{A}$. Similarly, we can prove the simulated view of E_2 is also indistinguishable. For max pooling and SecReLU, the only involved operation is iteratively invoking the secure comparison protocol SecCmp. Based on Lemma 2, to prove the security of the two protocols, we just have to prove that SecCmp is secure, which has been proved in [10]. Therefore, it can be derived that Theorem 3 can hold. □

From Theorem 1-3, we prove that all protocols of PE-HEALTH are secure. Based on Lemma 2, it can be derived that PE-HEALTH is also simulatable in the *curious-but-honest* model and the two security goals are achieved.

VI. EXPERIMENT RESULTS OF PE-HEALTH

In this section, we mainly conduct two parts of experiments. First, we evaluate the effectiveness of PE-HEALTH for health condition monitor. Then, we comprehensively compare the efficiency of PE-HEALTH with the existing privacy-preserving CNN schemes.

Experiment Setting. Our experiments are performed with two desktops, equipped with Intel Core i7-9700 CPU @3.00Ghz and 16GB RAM. The computation time is an average over ten trials. Three opensource datasets are adopted to evaluate PE-HEALTH, namely the Post-operative Patient dataset (Post-operative), the Batteryless Wearable Sensor dataset (Wearable) and MNIST. The former two datasets describe the physiological state of the patient collected by body sensors, which can be found in the UCI repository[2]. The last dataset is commonly used to evaluate the performance of a machine learning scheme [7], [16], [22].

Network Architecture. We utilize six kinds of CNN architectures to evaluate the performance of PE-HEALTH, shown in Fig. 5. Network (a) and Network (b) are used to process the Post-operative Patient Dataset. Batteryless Wearable Sensor Dataset is evaluated with Network (c) and Network (d). The last two networks are used to process MNIST. The output of each network is input into a softmax function to get the

[2]https://archive.ics.uci.edu/ml/datasets/

classification result. The computation of the softmax function is completed by HC and operated in plaintext.

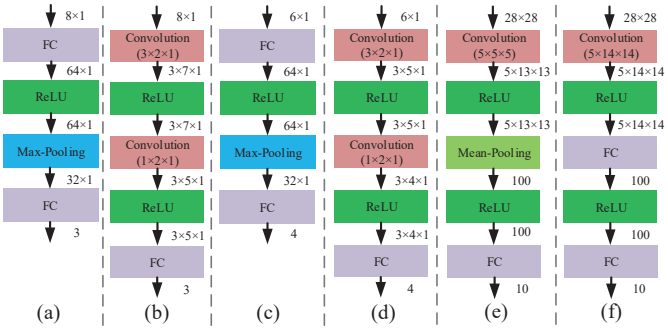


Fig. 5. Six CNN architectures used in the experiments: Network (a), Network (b), Network (c), Network (d), Network (e) and Network (f).

A. *Effectiveness of* PE-HEALTH

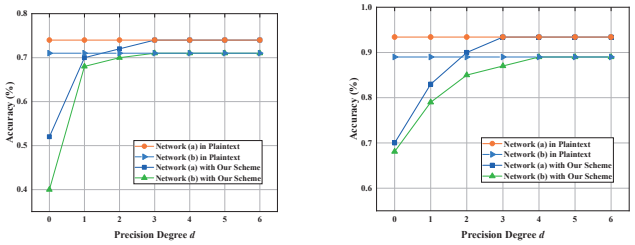
To evaluate the effectiveness of PE-HEALTH for health condition monitor, we test its accuracy on the Post-operative and Wearable datasets with Network (a), Network (b), Network (c) and Network (d). In the experiments, we partition each dataset into two sets, 70% used for training and 30% used for testing. The training set is used to get the pre-trained CNN model. The testing set is used to test the classification accuracy. Fig. 6 shows the testing accuracy of PE-HEALTH with different precision degrees d. It can be found that the accuracy of PE-HEALTH differs with the increase of d. This is because PE-HEALTH adopts the fixed-point format to ensure the security of the cryptographic tools, and d determines the computation error. According to our experiment results, when d is set to 4, PE-HEALTH can achieve similar accuracy as the CNN models without privacy preservation. As shown in Fig. 6(a), for Network (a) and Network (b), PE-HEALTH can separately reach 74.28% and 71.42% accuracies for the Post-operative dataset, which are as high as the CNN models in plaintext. Likewise, for the Wearable dataset, PE-HEALTH can also obtain similar accuracy to the plaintext CNN networks, shown in Fig. 6(b). Further, we also evaluate the computation error of the CNN model with different precision degrees, shown in Fig. 7(a) and Fig. 7(b). From the results, the computation error can reach as low as 10^{-3} as d is set to 4. The evaluation of the computation error for the ReLU and Pooling layers is ignored because the involved comparison operation in these two layers cannot cause computation error.

B. *Efficiency of* PE-HEALTH

Here, we theoretically and experimentally evaluate the efficiency of PE-HEALTH.

Theoretical Analysis. Prior to the experiment result, we list the theoretical computation and communication cost of PE-HEALTH in different stages.

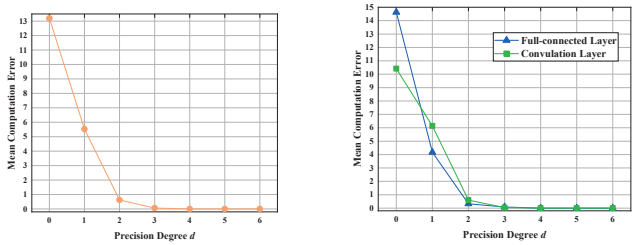
Computation. Suppose that the input size of an arbitrary layer is $m \times n$ and the maximum iteration number for downgrading the multiplicative shares is Θ. The time used for computing a pseudorandom function, a multiplication and a DDH exponentiation are $\mathcal{T}_{ran}$, $\mathcal{T}_{mul}$ and $\mathcal{T}_{exp}$, respectively. In



(a) Accuracy as the precision degree d increases for Post-operative.

(b) Accuracy as the precision degree d increases for Wearable.

Fig. 6. Effectiveness evaluation with different datasets and precision degrees.



(a) The computation error of the CNN inference output.

(b) The computation error of the Conv and FC outputs.

Fig. 7. The computation error with different precision degrees d.

PE-HEALTH, the sensor only conducts a simple additive secret sharing operation on the collected data, whose computation cost is $\mathcal{O}(1) \cdot \mathcal{T}_{ran}$. Once the sensor transmits the data of patients to the cloud servers, it can enter the standby state or perform the next stage of the monitor. The computation-intensive work for CNN inference is arranged for the cloud servers. For the Conv layer who has κ filters, PE-HEALTH needs to calculate $mn\kappa$ dot products, the computation complexity of which is $\mathcal{O}(1) \cdot \mathcal{T}_{exp} + \mathcal{O}(mn\kappa\Theta) \cdot \mathcal{T}_{ran}$. For the FC layer that has ι neurons, PE-HEALTH needs to calculate $ab\iota$ dot products, whose computation complexity is $\mathcal{O}(1) \cdot \mathcal{T}_{exp} + \mathcal{O}(mn\iota\Theta) \cdot \mathcal{T}_{ran}$. For the ReLU layer or Pooling layer, PE-HEALTH should only compare the input numbers using the SecCmp algorithm, whose computation complexity is $\mathcal{O}(mn\ell) \cdot \mathcal{T}_{mul}$, where ℓ is the data length.

Communication. Similar to the computation complexity, we can summarize the communication complexity of PE-HEALTH as follows. For data uploading, the sensor has to send $\mathcal{O}(\ell)$ bits messages. For both Conv and FC layer, PE-HEALTH can locally perform the dot product operation and does not have to exchange any messages. For the ReLU or Pooling layer, PE-HEALTH should send $\mathcal{O}(mn\ell)$ bits messages.

Experimental Analysis. In the experiments, we first compare the runtime and the massage size of PE-HEALTH to conduct CNN inference on MNIST with CryptoNets [22], one of the earliest privacy-preserving CNN schemes proposed by Microsoft, shown in Table I. The results show that the computation time and communication overhead of PE-HEALTH are much less than CryptoNets, which is completed based on HE. Besides, CrytoNets needs additional computations for

encoding and decoding every instance for data sharing and inference result recovery, given in Table II. To represent one pixel of an image, CrytoNets has to compute five polynomials, and each coefficient of the polynomial is expressed as 24 bytes bit-string. For PE-HEALTH, the encryption only needs the IoT sensor to generate random values and the decryption can be simply completed by the addition operation. Then, we compare the efficiency of PE-HEALTH to process one instance of MNIST with other state-of-the-art privacy-preserving schemes modified from CryptoNets. The comparison results are given in Table III. SecureML [7] achieves privacy-preserving CNN with a series of Garbled Circuit 2PC protocols but introduces some accuracy loss, which is usually intolerable for the medical application. MiniONN [16] and Chameleon [13] reduce the runtime by using HE and ASS; however, their communication overheads are still considerable. For PE-HEALTH, since the multiplication operation does not need to exchange any messages, its communication overhead is only produced by data uploading and the secure comparison operation. Therefore, the communication overhead is reduced in PE-HEALTH.

TABLE I
COMPARISON OF RUNTIME, MESSAGE SIZE FOR ONE IMAGE FROM MNIST NETWORK (E)

Approach	Runtime(s)	Message Size(MB)	Accuracy(%)
CryptoNets [22]	297.5	372.2	98.95
Our scheme	0.86	0.035	98.95

TABLE II
THE RUNTIME AND MASSAGE SIZE OF HMS AND HC FOR A BATCH OF 4096 IMAGES FROM MNIST

Approach	Runtime (s)		Massage size (MB)	
	Encryption (HMS)	Decryption (HC)	HMS $\rightarrow$ E_1/E_2	E_1/E_2 $\rightarrow$ HC
CryptoNets [22]	609.748	52.152	367.5	4.7
Our scheme	0.64	0.053	24.5	0.16

TABLE III
COMPARISON OF RUNTIME AND MESSAGE SIZE FOR ONE IMAGE FROM MNIST WITH NETWORK (F)

Approach	Runtime(s)	Message Size(MB)	Accuracy(%)
SecureML [7]	4.88	-	93.1
MiniONN [16]	9.32	657.5	99
Chameleon [13]	2.24	10.5	99
Our scheme	1.02	0.014	99

Moreover, although HE is introduced, PE-HEALTH avoids most of its complex exponentiation operations for ciphertext computations by combining it with additive SS, which makes it have much less runtime than other schemes. Moreover, to further evaluate the efficiency of PE-HEALTH, we also list the runtime of each layer to process one instance. Table IV shows that for all these layers, PE-HEALTH can process one instance in less than one second, which is quite efficient and totally satisfies the requirement for real-world applications. Specifically, for the Conv layer and the FC layer, the highest computation cost is spent on transforming the multiplicative shares into additive shares, i.e., Protocol 2, line 4-6. For the Pooling layer and the ReLU layer, the computation is mostly concentrated on performing SecCmp.

TABLE IV
THE RUNTIME (MS) OF DIFFERENT LAYERS TO PROCESS ONE INSTANCE

	Net(a)	Net(b)	Net(c)	Net(d)	Net(e)	Net(f)
Convolution	-	26.64	-	19.62	769	866
ReLU	0.52	0.36	0.53	0.29	8.24	10.31
Max-pooling	0.78	-	0.77	-	-	-
Mean-pooling	-	-	-	-	0.35	-
Fully-connected	60.3	3.28	61.3	4.1	9.37	103.5

VII. RELATED WORK

This paper emphatically resolves the privacy problem while applying CNN to health condition monitor. Thus, we review the related works about CNN based health condition monitor and privacy-preserving CNN.

Health Condition Monitor Health condition monitor is one of the most important applications for the artificial intelligence technique. By virtue of the IoT sensors and machine learning, the healthcare center can remotely monitor the health condition of the patient, and timely response to the emergency. Health condition monitor contains many parts, including activity recognition, disease alerting and health management. For example, Dawar *et al.* [23] and Huang *et al.* [19] used body sensors and CNN to recognize human activity for elderly monitoring and rehabilitation monitoring, which is quite helpful for the nursing home to look after the old man. Azimi *et al.* [6] designed a real-time remote IoT edging architecture that utilized CNN and patients' electrocardiography information to monitor patients' health. In [6], edge computing is introduced into the system architecture of health condition monitor. Edge computing is an emerging cloud computing technique that can greatly lower the communication latency by performing data processing at the edge of the network and close to the data provider [24]. Considering the limited communication ability of IoT sensors, the introduction of edge computing can significantly improves the quality of service for health condition monitor [25].

Privacy-preserving Convolutional Neural Network Benefited from the powerful data mining ability, CNN is widely used in current e-health systems. As mentioned above, given the physiological data collected by the IoT sensor, the healthcare center can use CNN to effectively infer the patient's health condition. However, a problem is that the input of CNN inference is usually physiological data or patient clinical images, which contains lots of privacy information of the patient. To resolve the problem, some privacy-preserving CNN schemes are proposed [7], [14], [15]. These schemes usually use three kinds of cryptographic tools to protect data privacy, namely DP, FHE and SS. DP is a research hotspot of the privacy protection field in recent years [8]. Abadi *et al.* [14] proposed a DP based CNN scheme and tested it on several standard datasets. However, the experiment results show that the noise introduced by DP can obviously influence the performance of CNN. FHE is a kind of encryption technique that supports computation over encrypted data. Until Gentry made a great breakthrough on FHE in 2009 [26], FHE had

been thought to be an overly idealistic cryptographic tool. Benefited from the breakthrough, FHE begins to be widely used in all kinds of security fields including privacy-preserving CNN [15], [16], [22]. The most serious problem that hinders a wider application of FHE is that compared with the other two methods, the involved computations of FHE are more complex, which makes it not very efficient. SS is first proposed by Shamir [27] to implement secure multi-party computation. Mohassel *et al.* [7] utilized SS to design a general privacy-preserving CNN scheme with two cloud servers. Nonetheless, SS requires to exchange massive data between data owners for data sharing, which is not appropriate for the IoT network.

VIII. Conclusion

Current IoT health condition monitor systems are mostly based on cloud computing and machine learning. Therefore, to provide data privacy for the systems, the core is the machine learning model deployed on the cloud, e.g., the convolutional neural network (CNN) in PE-Health. Inspired by the idea, PE-Health proposes a hybrid scheme that combines the advantages of homomorphic encryption (HE) and additive secret sharing (ASS). In PE-Health, ASS provides a lightweight but secure data uploading method for the IoT sensor. HE provides reliable security for the model parameters. A series of hybrid secure protocols are proposed to complete CNN inference and avoid the high communication overhead or complex computations brought by pure SS or HE schemes. In this way, PE-Health ensures that no patient data or the CNN parameters owned by the health center is revealed to the third-party cloud server while monitoring the patient's health. Further, experimental results show that the privacy-preserving CNN scheme used in PE-Health is more accurate and efficient than most existing schemes.

References

[1] S. S. Patil and S. A. Thorat, "Early detection of grapes diseases using machine learning and iot," in *2016 International Conference on Cognitive Computing and Information Processing (CCIP)*. IEEE, 2016.

[2] J. Qi, P. Yang, A. Waraich, Z. Deng, Y. Zhao, and Y. Yang, "Examining sensor-based physical activity recognition and monitoring for healthcare using internet of things: A systematic review," *Journal of biomedical informatics*, vol. 87, pp. 138–153, 2018.

[3] I. Azimi, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "Internet of things for remote elderly monitoring: a study from user-centered perspective," *Journal of Ambient Intelligence and Humanized Computing*, vol. 8, no. 2, pp. 273–289, 2017.

[4] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, and S. Andreescu, "Health monitoring and management using internet-of-things (iot) sensing with cloud-based processing: Opportunities and challenges," in *2015 IEEE International Conference on Services Computing*. IEEE, 2015, pp. 285–292.

[5] M. Chen, Y. Zhang, M. Qiu, N. Guizani, and Y. Hao, "Spha: Smart personal health advisor based on deep analytics," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 164–169, 2018.

[6] I. Azimi, J. Takalo-Mattila, A. Anzanpour, A. M. Rahmani, J.-P. Soininen, and P. Liljeberg, "Empowering healthcare iot systems with hierarchical edge-based deep learning," in *2018 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE, 2018, pp. 63–68.

[7] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 19–38.

[8] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2019, pp. 259–274.

[9] A. McGregor, I. Mironov, T. Pitassi, O. Reingold, K. Talwar, and S. Vadhan, "The limits of two-party differential privacy," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010, pp. 81–90.

[10] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, "A lightweight privacy-preserving cnn feature extraction framework for mobile sensing," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[11] E. Boyle, N. Gilboa, and Y. Ishai, "Group-based secure computation: optimizing rounds, communication, and computation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 163–193.

[12] H. Lipmaa, "On the cca1-security of elgamal and damgård's elgamal," in *International Conference on Information Security and Cryptology*. Springer, 2010, pp. 18–35.

[13] 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 *Proceedings of the 2018 Asia Conference on Computer and Communications Security (AsiaCCS)*, 2018, pp. 707–721.

[14] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[15] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2019, pp. 395–412.

[16] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 619–631.

[17] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Annual International Cryptology Conference*. Springer, 1991, pp. 420–432.

[18] E. Boyle, N. Gilboa, and Y. Ishai, "Breaking the circuit size barrier for secure computation under ddh," in *Annual International Cryptology Conference*. Springer, 2016, pp. 509–539.

[19] J. Huang, S. Lin, N. Wang, G. Dai, Y. Xie, and J. Zhou, "Tse-cnn: A two-stage end-to-end cnn for human activity recognition," *IEEE journal of biomedical and health informatics*, 2019.

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

[21] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *European Symposium on Research in Computer Security*. Springer, 2008, pp. 192–206.

[22] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.

[23] N. Dawar and N. Kehtarnavaz, "A convolutional neural network-based sensor fusion system for monitoring transition movements in healthcare applications," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*. IEEE, 2018, pp. 482–485.

[24] Z. Yan, J. Xue, and C. W. Chen, "Prius: Hybrid edge cloud and client adaptation for http adaptive streaming in cellular networks," *IEEE transactions on circuits and systems for video technology*, vol. 27, no. 1, pp. 209–222, 2016.

[25] J. Yu, B. Fu, A. Cao, Z. He, and D. Wu, "Edgecnn: A hybrid architecture for agile learning of healthcare data from iot devices," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2018, pp. 852–859.

[26] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices." in *Stoc*, vol. 9, no. 2009, 2009, pp. 169–178.

[27] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.