

Deep Learning-Based Medical Diagnostic Services: A Secure, Lightweight, and Accurate Realization

Xiaoning Liu ^a, Yifeng Zheng ^{b,*}, Xingliang Yuan ^c and Xun Yi ^a

^a *School of Computing Technologies, RMIT University, Melbourne, VIC 3001, Australia*

E-mail: xiaoning.trust@gmail.com

E-mail: xun.yi@rmit.edu.au

^b *School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China*

E-mail: yifeng.zheng@hit.edu.cn

^c *Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia*

E-mail: xingliang.yuan@monash.edu

Abstract. In this paper, we propose CryptMed, a system framework that enables medical service providers to offer secure, lightweight, and accurate medical diagnostic service to their customers via an execution of neural network inference in the ciphertext domain. CryptMed ensures the privacy of both parties with cryptographic guarantees. Our technical contributions include: 1) presenting a secret sharing based inference protocol that can well cope with the commonly-used linear and non-linear NN layers; 2) devising optimized secure comparison function that can efficiently support comparison-based activation functions in NN architectures; 3) constructing a suite of secure smooth functions built on precise approximation approaches for accurate medical diagnoses. We evaluate CryptMed on 6 neural network architectures across a wide range of non-linear activation functions over two benchmark and four real-world medical datasets. We comprehensively compare our system with prior art in terms of end-to-end service workload and prediction accuracy. Our empirical results demonstrate that CryptMed achieves up to respectively 413×, 19×, and 43× bandwidth savings for MNIST, CIFAR-10, and medical applications compared with prior art. For the smooth activation based inference, the best choice of our proposed approximations preserve the precision of original functions, with less than 1.2% accuracy loss and could enhance the precision due to the newly introduced activation function family.

Keywords: Secure computation, Privacy-preserving medical service, Neural network inference, Secret sharing

1. Introduction

Recent thriving deep learning techniques have been fueling a wide spectrum of medical endeavors, ranging from radiotherapy [1], clinical trial and research [2], to medical imaging diagnostics [3]. Enterprises capitalize on neural networks (NNs) to offer medical diagnostic services, facilitating hospitals and researchers to produce faster and more accurate decisions over their medical data. With the growth in such offerings comes rapidly growing awareness of daunting privacy concerns. The medical data is of sensitive nature and must be always kept confidential [4–7]. Meanwhile, NN models used in these services are seen as lucrative intellectual properties and encode knowledge of private training data [8].

*Corresponding author. E-mail: yifeng.zheng@hit.edu.cn.

Addressing these privacy concerns in the above deep learning powered service scenario generally fits within paradigm of secure multi-party computation (MPC). A rich body of work [9–13] proposes hand-tuning MPC protocols for secure inference, whereby the service provider and the customer interact to produce an inference over encrypted/secret-shared NN model and individual data. We note that prior designs are still facing obstacles regarding inference efficiency and accuracy in the ciphertext domain, and do not appear to be able to fulfill practical requirements of real-world medical diagnostic scenarios.

Firstly, they all require customers to conduct heavy cryptographic computations like homomorphic encryption (HE) and garbled circuits (GC), imposing intensive computational and communication overheads during inference. These overheads are further exacerbated when, e.g., the service is deployed to a hospital with resource-constrained devices (like portable medical imaging scanners [14]).

Secondly, existing protocols are either not compatible with non-linear (activation) functions [9, 11] or only focus on the simple ReLU function [12, 13, 15], causing limitations of applicability for medical diagnoses. There are limited works investigating other essential (smooth) activation functions, like sigmoid. Among them, most of existing designs use high-degree polynomials [16, 17] or ad-hoc piecewise polynomials [10, 18–20] to approximate those activation functions. However, the former approach incurs heavy costs when evaluating repeated multiplications. The latter one relies on intervention and expertise on models and training datasets for fine-tuning [10], or encounters the severe ‘vanishing gradient problem’ making the NNs imprecise [18–20].

To address the above challenges, we design, implement, and evaluate CryptMed, a lightweight and secure NN inference system tailored for medical diagnostic services. CryptMed proceeds by having the hospital and the medical service engage in a tailored secure inference protocol over their secret-shared inputs. Only the hospital learns the diagnostic result; and the privacy of the medical data and model is ensured against each other. In particular, we combine insights from cryptography, digital circuit design, and deep learning literature, fostering an efficient, low-interaction, and accurate deep learning service suitable for realistic medical scenarios. Our contributions are summarized as follows.

- We propose a new secure NN inference system framework CryptMed relying only on the lightweight secret sharing techniques, which requires neither heavy cryptographic computation nor large-size ciphertext transmission.
- We present a hybrid protocol design that consists of a preprocessing phase and an online phase where the preprocessing phase conducts as much computation as possible to ease the online phase. Moreover, the preprocessing only involves lightweight computation in the secret sharing domain.
- We devise an efficient and communication-optimized secure comparison function harnessing the insights from cryptography and the field of digital circuit design. Our proposed secure comparison function can efficiently support the widely adopted comparison-based non-linear functions, including ReLU, ReLU6, Leaky ReLU, Binary activation, and MaxPool/MinPool. Compared to the commonly-used GC solutions, CryptMed’s secure ReLU is $36\times$ faster and requires $398\times$ less communication, and the secure MaxPool is $20\times$ faster and uses $192\times$ less communication.
- We devise secure smooth activation functions (i.e., tanh, sigmoid, ELU) from newly proposed precise and cryptography-friendly approximations in the field of digital circuit design and deep learning literature. Our introduced approximations are non-linear and low-degree piecewise polynomial approximations with quantitative performance and demonstrate promising accuracy through comprehensive empirical. They are not only approximations but also new activation function family that are natural replacements of the smooth functions. With such approximations, CryptMed reformulates the challenging support for secure smooth activation functions into the comparison-based construction that can be efficiently and accurately evaluated in secure domain.

- We conduct formal security analysis. We implement a prototype of CryptMed. We extensively train varying neural network models based on 6 architectures across a wide range of non-linear activation functions. We conduct comprehensive experiments over two benchmarking datasets and four real-world medical datasets. Our experiment results show that CryptMed requires the least network resources compared to prior works with up to $413\times$, $19\times$ and $43\times$ bandwidth savings for MNIST, CIFAR-10, and the medical applications, respectively.

The rest of this paper is organized as follows. Section 2 investigates the related work. Section 3 introduces the necessary preliminaries. After an overview of our system in Section 4, we present our proposed construction in Section 5. Section 6 presents our empirical evaluation on microbenchmarks and end-to-end secure inference service. Finally, Section 7 concludes this paper.

2. Related Works

Secure Neural Network Inference. Secure neural network inference has drawn much attention in the emerging field of privacy-preserving machine learning. Our design is closely related to a line of studies on secure NN inference. These studies [9–13, 21–23] mostly propose an *interactive* protocol for secure inference running between the service provider and the customer. Among others, there are designs [15, 18, 24] that consider an outsourced scenario, where two *non-colluding cloud servers* collaboratively perform NN inference over the encrypted/secret-shared model and data. Apart from different system models, they commonly rely on heavy cryptographic techniques (like HE and GC) during the latency-sensitive online inference procedure. Very recently, Delphi [12] proposes a hybrid and interactive inference protocol, which preprocesses some cryptographic operations to accelerate the online inference execution. However, this work still demands intensive workloads on the customer to conduct heavy cryptographic computations during preprocessing, and relies on expensive GC based approach to evaluate the basic ReLU function. CryptMed adopts a similar hybrid setting yet only involves the lightweight secret sharing techniques during the entire secure inference procedure, which has a prominent advantage of *rather simplified* implementation for easy real-world deployment, compared to the SOTA which requires heavy optimization in GC and homomorphic encryption implementation.

We emphasize that most prior works only support the basic ReLU activation [12, 13, 15]. Other essential activation functions commonly-used in deep learning based medical diagnoses are unexplored, such as ReLU6, Leaky ReLU, and the exponential linear unit (ELU) [25]. Even worse, some works can not fully cope with the non-linearity [9, 11]. Instead, they use the square function for approximation, resulting in an imprecise prediction [26, 27] that could cause impactful consequences in the medical diagnostic applications.

In the literature, only limited works explore the smooth sigmoid activation function via polynomial approximations. These works either resort to high-degree polynomials [16, 17] or ad-hoc piecewise polynomials [10, 18–20]. The first approach suffers from substantial costs to evaluate a large amount of secure multiplications. The second approach heavily relies on intervention from developers to fine-tune the piecewise polynomials (coefficients and segments) [10], or runs into the severe vanishing gradient problem making the NNs imprecise [18–20]. All those solutions do not appear to be competent for practical deep learning based medical diagnoses deployment. A detailed comparison between our work and prior works is summarised in Table 1.

Table 1
 Limitations of prior secure neural network inference systems and comparison with our systems.

	No heavy crypto. for linear layers?	No heavy crypto. for nonlinear layers?	No large ciphertext transmissions?	Applicable for generic NNs?	Support smooth activations?
CryptoNets [9]	✗	✗	✗	✗ ¹	✗
SecureML [18]	✓	✗	✓	✓	✗
MiniONN [10]	✓	✗	✓	✓	✓
Gazelle [11]	✗	✗	✗	✗ ¹	✗
Chameleon [15]	✓	✗	✓	✓	✗
XONN [28]	✗	✗	✓	✗ ²	✗
Quotient [29]	✗	✗	✓	✗ ²	✗
Falcon [30]	✗	✗	✗	✓	✗
Delphi [12] (SOTA)	✗	✗	✓	✓	✗
CryptMed (our system)	✓	✓	✓	✓	✓

¹ These systems requires NN model architecture modification, such as polynomial approximation of ReLU activation. This setting could downgrade the prediction accuracy.

² These systems are designed for quantized NNs. Quotient [29] uses Ternarized NN, and XONN [28] is designed for Binarized NN.

Secure Machine Learning MPC Frameworks. A number of generic MPC frameworks are designed and implemented for complicated computation tasks like machine learning. Noteworthy examples include the two-party framework with a trusted party to generate correlated randomness proposed in Chameleon [15], the framework proposed by Reza Sadeghi *et al.* [31], TAPAS [32], FHE DiNN [33], the framework proposed by Dalskov *et al.* [34], MP2ML [35]; the three-party frameworks proposed in ABY3 [19], SecureNN [36], CryptTFlow [37]; the four-party frameworks in Trident [38], FLASH [39]; and multi-party frameworks in TFEncrypted [40], PySyft [41, 42], FALCON [43]. Note that a handful of latest privacy-preserving machine learning systems opt for specialized and optimized designs rather than direct application of generic MPC frameworks [12, 28, 29, 44] for performance consideration. Our design also follows such trend.

Privacy-Preserving Medical Diagnosis. This work also relates to the designs of privacy-preserving medical diagnosis. There is a plethora of work proposed on privacy-preserving medical imaging based diagnostic applications. Some works strive to enhance the reliability of image-centric diagnoses via privacy-preserving image denoising [45–48]. These works resort to DNNs [45, 46] or reference image patches [47, 48] to devise image denoising protocols that can privately reduce the noises and deliver high-quality medical image content. Meanwhile, a line of work aims to explore privacy-preserving machine learning for various medical diagnostics, like medical image classification [24, 49–52], tumor segmentation [53–56], and genomic data regression [57–59]. Some of them utilize cryptographic privacy-enhancing techniques (e.g., homomorphic encryption, secure multiparty computation) to protect the privacy of machine learning models and medical data [24, 49, 50, 57–59]. Others resort to differential privacy techniques to train a private model that can be used to conduct private inference over medical images [51–56]. We emphasize that these works focus on different problems and their designs are highly different from ours. A few of them consider secure neural network inference based medical diagnosis, yet requiring to use heavy cryptography [24, 50] or not focusing on supporting smooth activation functions [49]. Many others explore different problems, like the relatively simpler regression models [57–59], or federated learning [51–56]. Besides, those works relying on differential privacy techniques require to perturb the data with noise, and thus would downgrade the utility of models [51–56].

Table 2
Table of notation.

Notation	Description
\mathbf{X}, x	Input/feature tensor and element.
\mathbf{W}, w	Weight tensor and element.
$\mu, \delta, \gamma, \beta$	Batch normalization parameters: the running mean, running variance, scale, and shift.
L	Number of neural network layers
ℓ, n	Bit length ℓ and vector length n .
i	Identifier of a party i , where $i \in \{0, 1\}$.
\mathbf{b}, \mathbf{c}	Control bits
x_ℓ	The most significant bit of element x
x_k, \mathbf{x}_k	The k -th element of vector \mathbf{x} ; The k -th vector.
$\langle x \rangle_i^A$	Additive secret shares of value x in ring \mathbb{Z}_{2^ℓ} held by the party i
$\llbracket x \rrbracket_i$	Boolean shares of value x in ring \mathbb{Z}_2 held by the party i
$\langle x \rangle_i^A \pm \langle y \rangle_i^A$	Addition/subtraction over additive secret shares
$\langle x \rangle_i^A \cdot \langle y \rangle_i^A$	Multiplication over additive secret shares
$\llbracket x \rrbracket_i + \llbracket y \rrbracket_i$	Bitwise XOR over Boolean shares
$\llbracket x \rrbracket_i \cdot \llbracket y \rrbracket_i$	Bitwise AND over Boolean shares

Apart from the machine learning based approaches, works adopt data mining techniques to securely analyze the medical data for diagnostics. Applications include similarity analysis of human genome sequences [60–62], genome-wide association studies [63–66], biometric identification [67], pharmacology and medicine [68, 69], and medical time-series data analytics [6, 7, 70]. A common paradigm is to customize secure computation protocols to meet certain requirements for different medical diagnostic applications.

Differences from the Conference Version. Portions of this paper have been presented in [49]. We have revised the preliminary work [49] with substantial new contributions and improvements, as summarized below. Firstly, we have proposed a number of new efficient, lightweight, and accurate secure non-linear activation functions in Section 5, including the secure comparison-based activation functions ReLU6, Leaky ReLU, Binary activation, and the secure smooth activation functions tanh, sigmoid, and ELU with different approximation approaches. Secondly, we have made a full-fledged implementation of the new realizations of our security design and conducted comprehensive performance evaluation and comparisons. The overall experiment results have demonstrated the prominent performance advantage of our new design. Finally, we have refined the previous work significantly to reflect our new contributions and latest understanding on the topic, as well as improve the clarity.

3. Preliminaries

In this section, we introduce the core primitives and background used in CryptMed. We summarize the key notations used in this paper in Table 2.

Secret Sharing. We now present the key cryptographic primitive used in our design: *additive secret sharing*. Additive secret sharing [71] protects an ℓ -bit value $x \in \mathbb{Z}_{2^\ell}$ as two secret shares $\langle x \rangle_0 = r \pmod{2^\ell}$ and $\langle x \rangle_1 = x - r \pmod{2^\ell}$ such that $\langle x \rangle_0^A + \langle x \rangle_1^A \equiv x \pmod{2^\ell}$, where \mathbb{Z}_{2^ℓ} is a ring and r is a random value from \mathbb{Z}_{2^ℓ} ($r \in_R \mathbb{Z}_{2^\ell}$). It perfectly hides x as each share is a random value and

reveals no information of x . Given two parties P_0 and P_1 , each party holds corresponding shares of two secret values x and y . Additive secret sharing supports efficient local addition and subtraction over shares $\langle z \rangle_i = \langle x \rangle_i \pm \langle y \rangle_i$ and scalar multiplication $\langle z \rangle_i = \eta \cdot \langle x \rangle_i$ (η is a public value). They are calculated by each party P_i ($i \in \{0, 1\}$) locally without interaction. Multiplication over two shares $\langle z \rangle = \langle x \rangle \cdot \langle y \rangle$ is realized by the secret-shared Beaver's triple [72], i.e., P_i holds $(\langle t_1 \rangle_i, \langle t_2 \rangle_i, \langle t_3 \rangle_i)$ in a way that $t_3 = t_1 \cdot t_2$. Such a multiplication operation with Beaver's triple is a standard secure computation protocol, whereby P_i obtains the shares $\langle z \rangle_i$ of xy at the end. Note that Beaver's triples are data independent and can be efficiently generated via one-off computation by a third party [15, 73]. In addition, additive secret shares can readily support boolean operations over binary values. Given the bit length $\ell = 1$ and the ring \mathbb{Z}_2 , a secret binary value x is shared as $\llbracket x \rrbracket_0 = r \in \mathbb{Z}_2$ and $\llbracket x \rrbracket_1 = r \oplus \llbracket x \rrbracket_0$. The bitwise XOR (\oplus) and AND (\wedge) over shares are calculated in the same way as the above addition and multiplication, respectively.

Deep Neural Networks. A typical Deep Neural Network (DNN) comprises two types of layers in sequence: *linear* layers and *non-linear* layers. Linear layers include convolutional layers (CONV), fully-connected layers (FC), batch normalization (BN) layers, and average pooling layers (AvgPool). The functionalities of these layers in cleartext can be formulated as a bunch of additions, multiplications, and flattened operations over *kernels* (partial model weights for a certain function) and *features* (user inputs and intermediate results). Specifically, the underlying functionality of CONV and FC is the vector-wise dot product $\text{VDP}(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^n w(i) \cdot x(i)$ between a kernel vector $\mathbf{w} \in \mathbb{R}^n$ and a feature vector $\mathbf{x} \in \mathbb{R}^n$ within a sliding window $n \times n$. Given a kernel $\mathbf{W} \in \mathbb{R}^{c_{in} \times c_{out} \times n \times n}$, CONV transforms an input feature $\mathbf{X} \in \mathbb{R}^{c_{in} \times h_{in} \times w_{in}}$ into an output feature $\mathbf{Y} \in \mathbb{R}^{c_{out} \times h_{out} \times w_{out}}$ via

$$Y(t, m, n) = \sum_{k=1}^{c_{in}} \sum_{i=1}^n \sum_{j=1}^n \text{VDP}(X(k, m+i-1, n+j-1), W(k, t, i, j)),$$

where $t \in [c_{out}]$, $m \in [h_{out}]$, $n \in [w_{out}]$. That is, any data point in \mathbf{Y} is produced by applying a sliding kernel tensor $\mathbf{W}^{c_{in} \times n \times n}$ over the entire feature tensor \mathbf{X} , and performing cross-channel VDP operations repeatedly. Such a function indicates the FC layer when $n = 1$. *Batch normalization* is used to regularize the model. It is applied after CONV/FC layers and performs $z = \gamma(x - \mu)/\delta + \beta$ over the feature x on each neuron, where μ , δ , γ , and β are BN parameters: the running mean, the running variance, the scale, and the shift. Note that the BN parameters are part of the model weights and should be protected properly.

As summarized in Table 3, non-linear functions in DNNs can broadly be classified into comparison-based activation functions, smooth activation functions, and pooling layers. *Comparison-based activation functions* (ReLU, ReLU6, LeakyReLU, and Binary activation) have been demonstrated with superior performance in deep learning applications for rapid learning and high prediction accuracy. They are essential building blocks in neural networks to introduce non-linearity, particularly in image classifications. These functions alleviate the well-known 'vanishing gradient problem' (neural networks could not converge) encountered when the sigmoid and tanh functions are leveraged for training. The ReLU function is the most widely adopted activation function in CNNs. The ReLU6 activation function is a variant of the ReLU that clips weights between 0 and 6. The Leaky ReLU function adopts a linear function for negative features. The Binary activation function is usually adopted in quantized NNs.

Smooth activation functions make non-trivial usage in deep learning. Similar to comparison-based activation functions, smooth activation functions introduce non-linearity to NNs. Additionally, these functions have properties of continuity, smoothness, and monotonicity to empower NNs with complex capabilities [74, 75]. CryptMed focuses on three widely-adopted smooth activation functions, i.e., sigmoid, hyperbolic tangent (tanh), and ELU. They are vital building blocks in a variety of machine learning and deep learning paradigms for medical diagnoses, like medical time series predictions [74], tumor

Table 3
Typical non-linear layers in DNNs.

Layer	Description	Function
Comparison-based activation function	Binary activation function	$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$
	ReLU activation function	$f(x) = \max(0, x)$
	ReLU6 activation function	$f(x) = \min(\max(0, x), 6)$
	Leaky ReLU activation function	$f(x) = \max(0.1x, x)$
Smooth activation functions	Sigmoid function	$f(x) = \frac{1}{1+e^{-x}}$
	Hyperbolic tangent (tanh) function	$f(x) = \frac{2^x - e^{-x}}{e^x + e^{-x}}$
	Exponential linear units (ELU) activation function	$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha \cdot (e^x - 1) & \text{if } x < 0 \end{cases}$
Pooling layers (window size n)	max pooling	$\max(x_1, \dots, x_n)$
	min pooling	$\min(x_1, \dots, x_n)$

segmentation, and medical imaging denoising [46]. The tanh function and the sigmoid function are preferable over logistic regression, LSTM, and RNN for sequential and time series data prediction. The ELU function [76] is a noise-robust and precise activation function compared to the conventional ReLU activation and its variants (e.g., Leaky ReLU, Parametrized ReLU). Besides, ELU effectively diminishes the ‘vanishing gradient problem’ by setting the identity for positive features.

Secure Computation over Fixed-Point Ring. Deep learning inference operates over real-valued numbers, i.e., DNN weights and user inputs. In cleartext, they are represented in floating-point numbers. To allow CryptMed to operate in the secret sharing domain, we leverage the fixed-point representation to project values to the underlying ring \mathbb{Z}_{2^ℓ} . Such fixed-point representation is a common paradigm adapted in prior work [10, 12, 18, 20]. Specifically, given a floating-point number x , we first convert it to a signed fixed-point integer $\bar{x} = \lfloor x \cdot 2^s \rfloor$ with a scaling factor s embedding the fractional part. Afterwards, we project such an integer to the ring \mathbb{Z}_{2^ℓ} via $\bar{x} \bmod 2^\ell$. To represent the sign, we leverage the two’s complement representation, where the most significant bit (MSB) represents the sign. In this way, non-negative values are mapped to the lower-half ring $[0, 2^{\ell-1} - 1]$, while negative values are mapped to the upper-half ring $[2^{\ell-1}, 2^\ell - 1]$. Then, the MSB will be ‘0’ for a non-negative value, and ‘1’ for a negative value. In fixed-point representation, repeated multiplications may lead to integer overflow due to the excess of fractional bits (from s to $2s$ bits). A common treatment is to use a secure local truncation [12, 18, 20], where the least s bits are chopped off ahead of subsequent multiplications.

4. System Overview

4.1. Architecture

Figure 1 illustrates the system architecture of CryptMed which enables secure deep learning based medical diagnostic service. CryptMed operates between two parties: the *hospital* and the *medical service* provider. On the one hand, we consider that the *medical service* as an enterprise which deploys an NN powered medical diagnoses service offering through a proprietary NN model. On the other hand, we consider that the *hospital* as a customer intends to take advantage of the deep learning service to

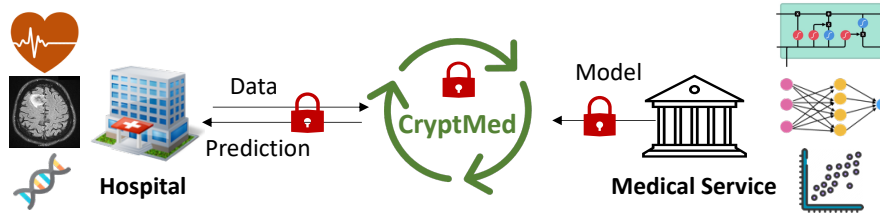


Fig. 1. System architecture.

facilitate an accurate medical conclusion, while protecting its own confidential medical records (e.g., CT image, physiological data). Note that the role of the hospital in the real world can be any healthcare institutes, such as clinics, biomedical research centers, or life-science institutes. To launch a secure medical diagnostic service, the above two parties execute CryptMed’s secure NN inference protocol over the secret-shared model and secret-shared medical record. At the end, only the hospital can obtain the secret shares prediction result, which is then recovered to get the cleartext diagnosis. CryptMed provides a cryptographic guarantee such that the hospital obtains the inference result only and nothing else, while the medical service learns no information about the hospital’s medical records.

4.2. Threat Model

CryptMed designs two-party inference secure against *semi-honest* adversaries. In CryptMed, the hospital and the medical service will honestly follow the prescribed protocol, yet trying to deduce auxiliary information about each other’s private input beyond what revealed from the protocol result. It is noted that such an assumption is practical. Nowadays the behavior of hospital is widely enforced by ethics, law and privacy regulations [4, 5]. In the meantime, the medical service is usually offered by well-established vendors (e.g., Microsoft Project InnerEye [3], Google DeepMind Health [1]), that do not have strong incentives to risk their business model and publicity for malicious behaviors [70]. Due to the above facts and observations, such a threat model is commonly adopted in prior secure NN inference work [10, 12] as well. CryptMed strives to ensure the privacy of the hospital’s medical records and the NN model (values of trained weights). Consistent with prior art [10, 12, 24], CryptMed does not hide the data-independent model architecture, e.g., the model size and number of layers. Lastly, CryptMed deems thwarting adversarial machine learning attacks orthogonal, which attempt to exploit the inference procedure as a blackbox oracle to extract private information. Mitigation strategies can be differentially private learning [77].

5. Our Proposed Design

In this section, we introduce CryptMed’s secure NN inference protocol for medical diagnostic applications. At a high level, our design consists of two types of secure layer evaluations: secure *linear* layers and secure *non-linear* layers. CryptMed efficiently supports a suite of secure *linear* layers, including the secure convolutional layers, secure fully-connect layers, secure batch normalization, and secure average pooling layers. For secure *non-linear* layers, CryptMed efficiently realizes a series of comparison-based non-linear layers, i.e., ReLU, ReLU6, Leaky ReLU, Binary activation, MaxPool, and MinPool. Besides, CryptMed enables rich non-linear functionalities by supporting lightweight and accurate evaluations of secure smooth activation functions, including tanh, sigmoid, and ELU. All these layers are vital building blocks in deep learning based medical diagnostic services.

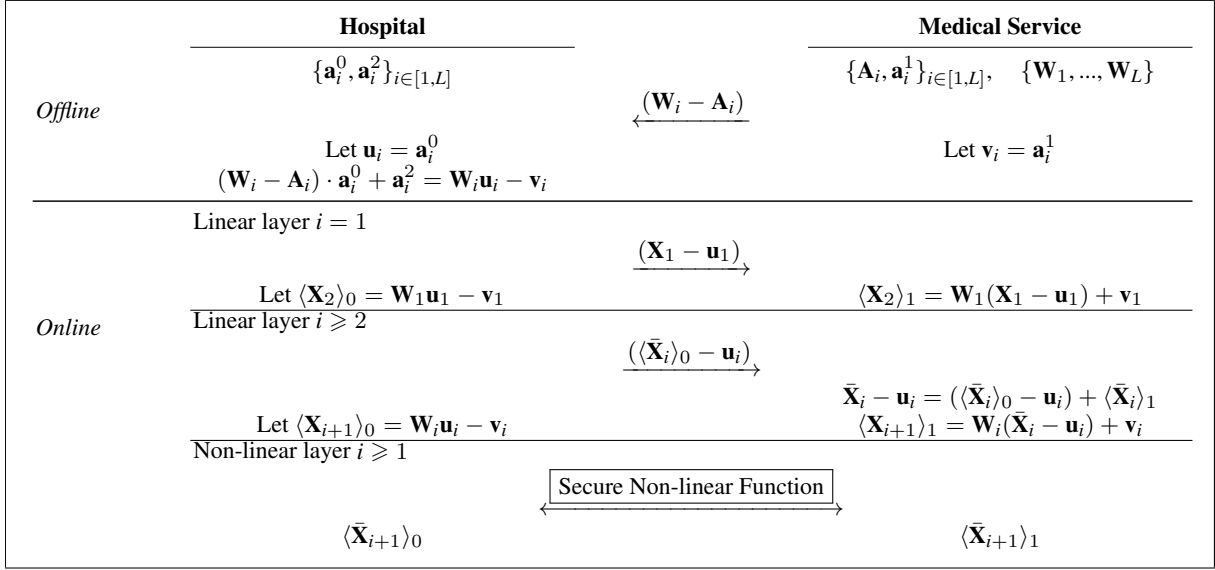


Fig. 2. CryptMed's secure inference protocol.

In CryptMed, each secure layer securely evaluates a certain cleartext functionality in the secret sharing domain, which proceeds by taking the secret-shared inputs (features and/or kernels) and producing the secret-shared outputs passed to the succeeding secure layer. Our overarching goal is to devise a lightweight protocol for secure neural network inference with optimized interactions, while empowering rich and accurate secure functionalities for deep learning based medical services. Atop such goal, we have three prominent design insights.

Supporting lightweight secure linear layers. We first split CryptMed's protocol into a preprocessing phase and an online phase, so as to shift as much computation as possible to preprocessing phase. Inspired by [12], we preprocess the model as secret shares and deliver corresponding shares to the hospital before medical record becomes available. So, the online phase can directly work over secret shares without any heavy cryptographic techniques (like HE) or multi-round ciphertext transmissions. Yet we are aware that the protocol in [12] involves heavy HE during preprocessing to produce and send the model shares as ciphertexts, which may not be amiable for the resource-limited hospital, like COVID-19 pandemic screening centers with handheld medical imaging scanners [14]). Instead, our protocol delicately leverages the insight from Chameleon [15] and enables the preprocessing to be purely based on lightweight computation in the secret sharing domain. As a result, our entire protocol works only with small shares, which immediately gains $20\times$ improvement on preprocessing and $10\times$ on overall communication costs over [12].

Supporting lightweight non-linear layers. For secure evaluation of non-linear layers, prior works either resort to the heavy cryptographic techniques (i.e., garbled circuits) [10, 12], or circumvent the non-linearities with the square function approximations [9, 11]. Unfortunately, such methods may introduce high communication overheads or induce instabilities of NN when handling complex tasks [26, 27]. In CryptMed, we make observations from the field of digital circuit design [78] and present a secure comparison function that can efficiently evaluate comparison-based non-linear layers, including ReLU, ReLU6, Leaky ReLU, and Binary activation functions. At the core, this function is fully based on lightweight

secret sharing with optimized interactions between the hospital and the medical service. With these designs, our experiment demonstrates a $413\times$ bandwidth reduction compared with prior works.

Supporting accurate activation functions over secret sharing domain. Most existing works [12, 13, 15] focus only on the simple ReLU function. Other essential activation functions remain under exploration, including Leaky ReLU, ReLU6, and ELU. These activation are fundamental building blocks in modern NN architectures for medical applications, such as medical image classification [25], image denoising [46], and medical times series (physiological data) prediction [74, 79]. The most challenging task is to accurately and efficiently evaluate the smooth activation functions (e.g., sigmoid, tanh, and ELU) in secure domain. Such functions involve exponentiation and division operations that are knowingly expensive to be computed over secret-shared data.

Prior art tackling such challenges mainly falls into two categories. Works in the first category resort to function approximations based on polynomials. Some of them use the high-degree polynomials [10, 16, 17], which require heavy computational and communication costs to evaluate repeated multiplications. Others leverage the ad-hoc piecewise polynomial approximation [10, 18–20]. These works either heavily rely on intervention and expertise on DNN model and training dataset to fine-tune the coefficients [10] or encounter severe ‘vanishing gradient problem’ making the NNs quite imprecise [18–20]. The work [80] in the other category uses GC to evaluate the smooth functions in a blackbox manner, suffering from prohibitively performance overheads.

Instead of the strawman approximations, CryptMed proposes secure smooth activation functions that are accurate while keeping the cryptographic costs in mind. The core idea is to make use of advancements from the field of digital circuit design [46, 81–83] and the machine learning literature [74, 84] so as to propose more precise and cryptography-friendly approximations. These approximations are non-linear and low-degree piecewise polynomials that have *quantitative* performance demonstrating promising accuracy over comprehensive empirical evaluations. With such approximations, CryptMed reformulates the smooth activation functions into comparison-based constructions, and thus circumvents the obstacles coming from exponentiation and division. As a result, CryptMed empowers accurate and efficient secure realizations of smooth activation functions over secret sharing domain.

5.1. Secure Linear Layers

The subsequent section presents CryptMed’s secure inference protocol, which is comprised of the *preprocessing* phase and the *online inference* phase as shown in Fig. 2.

Preprocessing phase. During preprocessing, the hospital and the medical service pre-generate custom secret shares of the NN model in an appropriate form which are to be used during online inference. This is a one-off computation and conducted independent of the hospital’s medical record. Let L be number of layers. The hospital takes as input the L sets of randomnesses (in tensor form) $\{\mathbf{a}_i^0, \mathbf{a}_i^2\}$, where $i \in [1, L]$. Similarly, the medical service takes as input the tensors of model weights for each layer $\mathbf{W}_1, \dots, \mathbf{W}_L$ and randomnesses tensors $\{\mathbf{A}_i, \mathbf{a}_i^1\}$. Such randomness tensors $\{\mathbf{a}_i^0, \mathbf{a}_i^1, \mathbf{a}_i^2, \mathbf{A}_i\}$ are independent to any party’s input and can be pre-distributed to the parties. They satisfy the relationship: $\mathbf{a}_i^1 = \mathbf{A}_i \cdot \mathbf{a}_i^0 - \mathbf{a}_i^2$. Note that the dimension of each randomness tensor is in line with the dimension of each layer’s filter. Given these inputs, the two parties perform the following steps.

- (1) For each $i \in [1, L]$, the medical service computes $\mathbf{W}_i - \mathbf{A}_i$ over the weight tensors and sends to the hospital.
- (2) The hospital computes $(\mathbf{W}_i - \mathbf{A}_i) \cdot \mathbf{a}_i^0 + \mathbf{a}_i^2 = \mathbf{W}_i \mathbf{a}_i^0 - \mathbf{A}_i \mathbf{a}_i^0 + \mathbf{a}_i^2$ for each layer.

(3) Let \mathbf{u}_i denote \mathbf{a}_i^0 , and \mathbf{v}_i denote \mathbf{a}_i^1 . The medical service thus holds \mathbf{v}_i , and the hospital holds $\mathbf{W}_i\mathbf{u}_i - \mathbf{v}_i$, i.e., an additively secret-shared weight tensor $\mathbf{W}_i\mathbf{u}_i$.

Online inference phase. During online inference, the hospital takes as input the tensor of a medical record \mathbf{X}_1 , the randomnesses \mathbf{u}_i , and weight shares $\mathbf{W}_i\mathbf{u}_i - \mathbf{v}_i$. The medical service takes as input the weight tensors $\mathbf{W}_1, \dots, \mathbf{W}_L$ and the randomnesses \mathbf{v}_i . They then perform the secure layer function in pipeline as follows.

The first linear layer $i = 1$:

1. The hospital computes and sends $\mathbf{X}_1 - \mathbf{u}_1$ to the medical service, and uses $\langle \mathbf{X}_2 \rangle_0$ to denote $\mathbf{W}_1\mathbf{u}_1 - \mathbf{v}_1$.
2. The medical service computes $\langle \mathbf{X}_2 \rangle_1 = \mathbf{W}_1(\mathbf{X}_1 - \mathbf{u}_1) + \mathbf{v}_1 = \mathbf{W}_1\mathbf{X}_1 - \mathbf{W}_1\mathbf{u}_1 + \mathbf{v}_1$.
3. At this point, the hospital and the medical service hold the additive secret shares (i.e., $\langle \mathbf{X}_2 \rangle_0, \langle \mathbf{X}_2 \rangle_1$) of features¹ outputted from the first linear layer $\mathbf{W}_1\mathbf{X}_1$.

Remaining linear layers $i \geq 2$:

1. Similar to the first layer, the hospital computes $\langle \bar{\mathbf{X}}_i \rangle_0 - \mathbf{u}_i$ over its share $\langle \bar{\mathbf{X}}_i \rangle_0$ of activation produced from the secure ReLU evaluation (which we will detail later), and sends it to the medical service. Such a treatment can perfectly hide the hospital's share, and protect the activation $\bar{\mathbf{X}}_i$ against the medical service. It then sets $\langle \mathbf{X}_{i+1} \rangle_0 = \mathbf{W}_i\mathbf{u}_i - \mathbf{v}_i$.
2. The medical service computes $\mathbf{X}_i - \mathbf{u}_i = \langle \bar{\mathbf{X}}_i \rangle_0 - \mathbf{u}_i + \langle \bar{\mathbf{X}}_i \rangle_1$. Then it gets $\langle \mathbf{X}_{i+1} \rangle_1 = \mathbf{W}_i(\mathbf{X}_i - \mathbf{u}_i) + \mathbf{v}_i$, ensuring both parties hold additive secret shares (i.e., $\langle \mathbf{X}_{i+1} \rangle_0, \langle \mathbf{X}_{i+1} \rangle_1$) of layer result $\mathbf{W}_i\mathbf{X}_i$.

Non-linear layers: The shares from secure linear layer evaluation can be fed into the secure non-linear layer, which outputs shares $\langle \bar{\mathbf{X}}_{i+1} \rangle_0, \langle \bar{\mathbf{X}}_{i+1} \rangle_1$ of activations to each party.

Output layer: The medical service sends $\langle \mathbf{X}_L \rangle_1$ to the hospital, who can then integrate $\langle \mathbf{X}_L \rangle_0$ for reconstruction of the final inference result \mathbf{X}_L .

5.2. Secure Non-linear Layers

CryptMed supports highly efficient evaluation of the secure non-linear layers in the secret sharing domain. We observed that most of the non-linear activation functions and pooling layers can be decomposed into a series of comparison operations along with some linear operations (i.e., addition and multiplication). Besides, as mentioned above, the smooth activation functions will be delicately reformulated to the comparison-based piecewise non-linear polynomials, and thus relying on the comparison as well. With such an observation in mind, we reformulate each comparison to the MSB extraction defined as follows. Suppose we have two features x_1, x_2 . The MSB extraction is defined as

$$\mathbf{b} \leftarrow \text{MSB}(x_1 - x_2) = \begin{cases} 0 & \text{if } x_1 \geq x_2 \\ 1 & \text{if } x_1 < x_2 \end{cases}.$$

Then, finding the maximum $\max(x_1, x_2)$ (or the minimum $\min(x_1, x_2)$) is reformulated as

$$\max(x_1, x_2) = \mathbf{b} \cdot (x_2 - x_1) + x_1; \quad \min(x_1, x_2) = \mathbf{b} \cdot (x_1 - x_2) + x_2.$$

¹Biases can be added to the medical service's shares locally.

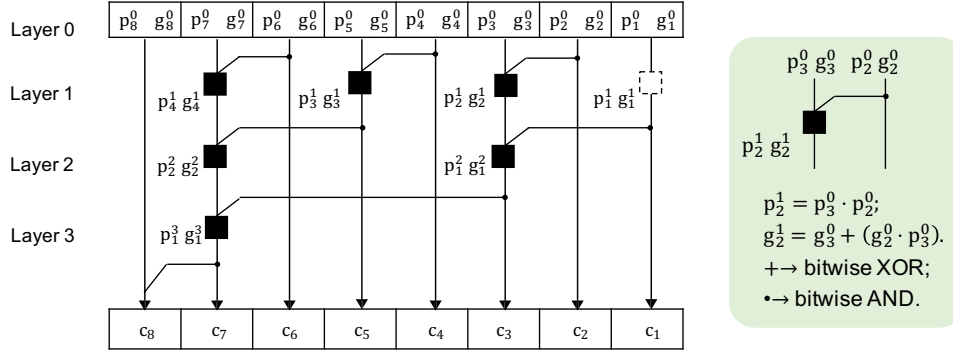


Fig. 3. A concrete example of 8-bit parallel prefix adder and the corresponding binary operator.

The sign of a feature x is calculated via $\text{sign}(x) = 1 - 2\text{MSB}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$.

Note that similar philosophy has also been adopted in the preliminary version of our paper [49], yet prior solution requires two multiplications for each comparison (i.e., $\max(x_1, x_2) = (1 - \mathbf{b}) \cdot x_1 + \mathbf{b} \cdot x_2$), whereas our newly proposed reformulation involves only one multiplication. Such an improvement is *non-trivial*, since a typical neural network usually requires a million and even billion scale of the number of comparison operations. With the above reformulation, the most challenging computation is how to securely extract the MSB in the secret sharing domain. We resort to a communication-optimized construction of the secure MSB extraction function. Details are presented in this subsequent section.

Communication-Optimized Secure MSB Extraction. The secure $\text{MSB}(\cdot)$ extraction function is used to securely extract the MSB of an additive-shared data $\langle x \rangle$ and generate a boolean-shared MSB $[[x_\ell]]$, where ℓ is the bit length. The key idea is to efficiently extract the MSB in the secret sharing domain. CryptMed's proposed design is built upon an practical and communication-optimized construction [49], which realizes the MSB extraction via carry look-ahead adder logic. By taking advantage of the parallel prefix adder [78] (PPA), such a construction can securely produce the MSB in logarithm round complexity $O(\log \ell)$ in the secret sharing domain.

The construction of the PPA-based MSB extraction is introduced as follows. The starting point is to view the two shares of an ℓ -bit value x as two inputs of the PPA. To do so, we decompose the secret shares $\langle x \rangle_0, \langle x \rangle_1$ as two bit strings $e = \{e_\ell, \dots, e_1\}$ and $f = \{f_\ell, \dots, f_1\}$, respectively. Afterwards, an ℓ -bit PPA is used to calculate $x = e + f \pmod{2^\ell}$ via a series of binary additions $\{e_i\} + \{f_i\}$ and pop out the carry bits c_ℓ, \dots, c_1 . In this way, the MSB can be produced as $x_\ell = c_\ell \oplus e_\ell \oplus f_\ell$. We provide a concrete example of 8-bit PPA in Fig. 3 and introduce the details of an ℓ -bit PPA realization as follows.

1. The first step is to calculate the initial signal tuple (p_i^0, g_i^0) : the carry generate signal g_i^0 and the carry propagate signal p_i^0 in parallel via $g_i^0 = e_i \cdot f_i$ and $p_i^0 = e_i + f_i$.
2. The second step is to produce the rest signal tuples $(p_i^1, g_i^1) \dots (p_i^{\log \ell}, g_i^{\log \ell})$ via a binary operator \odot . Given $(g_{in_1}, p_{in_1}), (g_{in_2}, p_{in_2})$ are the inputted two adjacent signal tuples, and (g_{out}, p_{out}) is the outputted single tuple. Each binary operator is defined as $(g_{out}, p_{out}) = (g_{in_1}, p_{in_1}) \odot (g_{in_2}, p_{in_2})$, where $g_{out} = g_{in_2} + g_{in_1} \cdot p_{in_2}$ and $p_{out} = p_{in_2} \cdot p_{in_1}$. Such a binary operation is recursively performed over the input tuples, and the outputted signal tuples is propagated to the next layer's nodes as inputs, until reaching the $\log \ell$ layer.
3. The third step is to calculate the carry bits via $c_{i+1} = (e_i \cdot f_i) + c_i \cdot (e_i + f_i) = c_{i+1} = g_i + c_i \cdot p_i$.

4. Finally, the most significant carry bit can be generated via $c_\ell = g_{\ell-1} + (p_{\ell-1} \cdot g_{\ell-2}) + \dots + (p_{\ell-1} \dots p_2 \cdot g_1)$. The MSB is calculated as $x_\ell = c_\ell \oplus e_\ell \oplus f_\ell$.

With the above PPA-based MSB construction in mind, we present details of the secure MSB extraction function in what follows. On each neuron, it takes as input the arithmetic shared integer feature $\langle x \rangle \in \mathbb{Z}_{2^\ell}$, and produces the boolean shared MSB $\llbracket x_\ell \rrbracket \in \mathbb{Z}_2$ as output. The hospital (denoted as P_0) and the medical service (denoted as P_1) jointly compute the function $\llbracket x_\ell \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$ as follows:

(1) Decompose $\langle x \rangle$ into bit strings:

(a) P_0 sets e as $\langle x \rangle_0$, and decomposes it to bit string $e \rightarrow e_\ell, \dots, e_1$;

P_1 sets f as $\langle x \rangle_1$, and decomposes it to bit string $f \rightarrow f_\ell, \dots, f_1$.

(b) For each $k \in [1, \ell]$, P_0 sets $\llbracket e_k \rrbracket_0 = e_k$ and $\llbracket f_k \rrbracket_0 = 0$; P_1 sets $\llbracket e_k \rrbracket_1 = 0$ and $\llbracket f_k \rrbracket_1 = f_k$.

(2) Compute signal tuples (g, p) : $\llbracket g_k^0 \rrbracket = \llbracket e_k \rrbracket \cdot \llbracket f_k \rrbracket$, $\llbracket p_k^0 \rrbracket = \llbracket e_k \rrbracket + \llbracket f_k \rrbracket$.

(3) Compute PPA:

(a) **Round $\mathcal{R} = 1$:**

P_0 and P_1 set $(\llbracket g_1^1 \rrbracket, \llbracket p_1^1 \rrbracket) = (\llbracket g_1^0 \rrbracket, \llbracket p_1^0 \rrbracket)$ as a dummy node.

For each $k \in [2, \ell/2]$, let $in_1 = 2k - 2$, $in_2 = 2k - 1$. P_0 and P_1 compute $(\llbracket g_k^1 \rrbracket, \llbracket p_k^1 \rrbracket) = (\llbracket g_{in_1}^0 \rrbracket, \llbracket p_{in_1}^0 \rrbracket) \odot (\llbracket g_{in_2}^0 \rrbracket, \llbracket p_{in_2}^0 \rrbracket)$.

(b) **Round $\mathcal{R} = 2, \dots, \log \ell$:**

For each $k \in [1, \ell/2^{\mathcal{R}}]$, let $in_1 = 2k - 1$, $in_2 = 2k$. P_0 and P_1 compute $(\llbracket g_k^{\mathcal{R}} \rrbracket, \llbracket p_k^{\mathcal{R}} \rrbracket) = (\llbracket g_{in_1}^{\mathcal{R}-1} \rrbracket, \llbracket p_{in_1}^{\mathcal{R}-1} \rrbracket) \odot (\llbracket g_{in_2}^{\mathcal{R}-1} \rrbracket, \llbracket p_{in_2}^{\mathcal{R}-1} \rrbracket)$.

(4) Compute MSB: P_0 and P_1 set $\llbracket c_\ell \rrbracket = \llbracket g_1^{\log \ell} \rrbracket$, $\llbracket x_\ell \rrbracket = \llbracket p_\ell^0 \rrbracket + \llbracket c_\ell \rrbracket$.

Secure B2A function. Recall that in CryptMed, our proposed secure comparison function is formulated via the above proposed secure MSB extraction. For example, finding the minimum between two features x_1, x_2 is formulated as $\min(x_1, x_2) \rightarrow \text{MSB}(x) \cdot (x_1 - x_2) + x_2$. The secure realization of such a formula requires secure share conversion when securely multiplying $\text{MSB}(x)$ with $(x_1 - x_2)$. Namely, the boolean-shared $\llbracket \text{MSB}(x) \rrbracket \in \mathbb{Z}_2$ needs to be firstly converted into its corresponding additive shares $\langle \text{MSB}(x) \rangle \in \mathbb{Z}_{2^\ell}$. Afterwards, the additively-shared MSB can multiply with the additive shares $\langle (x_1 - x_2) \rangle \in \mathbb{Z}_{2^\ell}$.

CryptMed resorts to the standard secure boolean-to-additive shares conversion construction (i.e., B2A) [24, 73]. The aim is to convert any boolean shares $\llbracket x \rrbracket \in \mathbb{Z}_2$ to its additive shares $\langle x \rangle \in \mathbb{Z}_{2^\ell}$. Given two parties, the hospital (denoted as P_0) and the medical service (denoted as P_1), the secure B2A function is computed as follow:

(1) P_0 sets $\langle e \rangle_0 = \llbracket x \rrbracket_0$, $\langle f \rangle_0 = 0$, and P_1 sets $\langle e \rangle_1 = 0$, $\langle f \rangle_1 = \llbracket x \rrbracket_1$;

(2) P_0 and P_1 compute $\langle x \rangle = \langle e \rangle + \langle f \rangle - 2 \cdot \langle e \rangle \cdot \langle f \rangle$.

5.3. Secure Comparison Based Activation Functions

CryptMed targets on four popular comparison-based activation functions, i.e., ReLU and its variants ReLU6 and Leaky ReLU, and the conventional Binary activation function, as summarized in Table 3. CryptMed manages to convert their cleartext functionalities into the MSB extraction based constructions. Through careful customizing, we propose efficient and lightweight realizations of secure *ReLU* function, secure *ReLU6* function, secure *LeakyReLU* function, and secure *Binary* function that are purely based on secret sharing techniques. In what follows, we present the details of their secure constructions.

5.3.1. Secure ReLU Activation Function

In CryptMed, we reformulate the ReLU activation to a simpler MSB extraction problem that can be efficiently evaluated in the secret sharing domain. Given the feature x on each neuron outputted from the preceding linear layer, it is reformulated into an MSB extraction based construction via

$$ReLU(x) = \max(x, 0) \xrightarrow{\text{transform}} \neg\text{MSB}(x) \cdot x = \begin{cases} 1 \cdot x & \text{if } x \geq 0 \\ 0 \cdot x & \text{if } x < 0 \end{cases},$$

Such a reformulated construction comprises of four atomic steps: the *secure MSB(x) extraction*, the *secure NOT* (i.e., \neg operation), the *secure B2A* to convert the boolean-shared MSB into additive shares, and the *secure multiplication*. All these steps can be efficiently realized by CryptMed's secure linear and non-linear functions. Without loss of generality, we demonstrate the secure *ReLU* function over single feature element x on each neuron. Given the shares of a single input feature $\langle x \rangle$, the hospital (denoted as P_0) and the medical service (denoted as P_1) jointly compute the secure *ReLU* function as follows:

- (1) P_0 and P_1 run to get $\llbracket \mathbf{b} \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$.
- (2) P_i computes the NOT operation $\llbracket \mathbf{c} \rrbracket = \llbracket \mathbf{b} \rrbracket + i$.
- (3) P_0 and P_1 run to get the additively shared NOT MSB $\langle \mathbf{c} \rangle \leftarrow \text{B2A}(\llbracket \mathbf{c} \rrbracket)$.
- (4) P_0 and P_1 produce the *ReLU* activation $\langle z \rangle = \langle \mathbf{c} \rangle \cdot \langle x \rangle$.

5.3.2. Secure ReLU6 Activation Function

The ReLU6 activation function is a variant of the ReLU that clips the weights between 0 and 6. Given the feature x on each neuron, the MSB extraction based ReLU6 is converted via

$$ReLU6(x) = \min(\max(0, x), 6) = \begin{cases} x & \text{if } 6 \geq x > 0 \\ 0 & \text{if } x \leq 0 \\ 6 & \text{if } x > 6. \end{cases}$$

$$\rightarrow \text{transform} \left(\underbrace{\neg\text{MSB}(x-6) \cdot 6}_{\mathbf{c}_1} + \underbrace{(\text{MSB}(x-6) \cdot \neg\text{MSB}(x) \cdot x)}_{\mathbf{c}_2} \right).$$

Similar to ReLU, such a construction can be efficiently realized in the secret sharing domain via CryptMed's secure linear and non-linear functions. Given the additive shares of each neuron's feature $\langle x \rangle$, the hospital P_0 and the medical service P_1 jointly compute the secure *ReLU6* function as follows:

- (1) P_0 and P_1 run to get the MSB $\llbracket \mathbf{b}_1 \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$ and $\llbracket \mathbf{b}_2 \rrbracket \leftarrow \text{MSB}(\langle x \rangle - 6)$.
- (2) P_0 and P_1 compute the control bits $\llbracket \mathbf{c}_1 \rrbracket = \llbracket \mathbf{b}_2 \rrbracket + i$ and $\llbracket \mathbf{c}_2 \rrbracket = (\llbracket \mathbf{b}_1 \rrbracket + i) \cdot \llbracket \mathbf{b}_2 \rrbracket$.
- (3) P_0 and P_1 run to get the additively shared control bits $\langle \mathbf{c}_1 \rangle \leftarrow \text{B2A}(\llbracket \mathbf{c}_1 \rrbracket)$ and $\langle \mathbf{c}_2 \rangle \leftarrow \text{B2A}(\llbracket \mathbf{c}_2 \rrbracket)$.
- (4) P_0 and P_1 produce the *ReLU6* activation $\langle z \rangle = 6 \cdot \langle \mathbf{c}_1 \rangle + \langle \mathbf{c}_2 \rangle \cdot \langle x \rangle$.

5.3.3. Secure Leaky ReLU Function

Given the feature x on each neuron, CryptMed reformulates the Leaky ReLU into the MSB extraction based construction via

$$LeakyReLU(x) = \max(0.01x, x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{if } x \leq 0 \end{cases}$$

$$\xrightarrow{\text{transform}} (\text{MSB}(x) \cdot (x - 100x) + 100x) \cdot 1/100.$$

Similar to other comparison-based activation functions, the reformulated Leaky ReLU can be realized via CryptMed’s secure functions. Specifically, given the additive shares of each neuron’s feature $\langle x \rangle$, the hospital P_0 and the medical service P_1 jointly compute the secure *LeakyReLU* function as follows:

- (1) P_0 and P_1 run to get the MSB $\llbracket \mathbf{b} \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$.
- (2) P_0 and P_1 run to get the additively shared MSB $\langle \mathbf{b} \rangle \leftarrow \text{B2A}(\llbracket \mathbf{b} \rrbracket)$.
- (3) P_0 and P_1 produce the *LeakyReLU* activation $\langle z \rangle = \lfloor (\langle \mathbf{b} \rangle \cdot \langle x \rangle \cdot (-99) + \langle x \rangle \cdot 100) / 100 \rfloor$.

5.3.4. Secure Binary Activation Function

Given the feature x on each neuron, CryptMed converts the Binary activation function into an MSB extraction based construction as follows

$$\text{Binary}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \xrightarrow{\text{transform}} \neg \text{MSB}(x).$$

Such a construction can be efficiently realized via CryptMed’s secure linear and non-linear functions. Given the additive shares of each neuron’s feature $\langle x \rangle$, the hospital P_0 and the medical service P_1 jointly compute the secure *Binary* function as follows:

- (1) P_0 and P_1 run to get $\llbracket -\mathbf{b} \rrbracket \leftarrow \text{MSB}(\langle x \rangle) + i$.
- (2) P_0 and P_1 run to get the additively shared NOT MSB as the *Binary* activation $\langle z \rangle \leftarrow \text{B2A}(\llbracket -\mathbf{b} \rrbracket)$.

5.4. Secure Smooth Activation Functions

The smooth activation functions make non-trivial usages in deep learning. As shown in Table 3, CryptMed focuses on three widely-adopted smooth activation functions, i.e., sigmoid, tanh, and ELU. They are vital building blocks in a variety of machine learning and deep learning paradigms for medical diagnoses, like medical time series predictions [74] and medical imaging denoising [46].

CryptMed’s secure smooth activation functions are tailored from two precise and cryptography-friendly approximations: the polynomial piecewise approximations (PLAs) and the SQNL activation function family. We provide a high-level overview of our insights below.

Piecewise Linear Approximations. Our first insight is to make use of the PLAs from the field of digital circuit design [46, 81–83]. They are non-linear and low-degree polynomials with quantitative performance. With these approximations, we propose the secure tanh_{PLA} function and secure sigmoid_{PLAN2} function. They are efficient and accurate realizations of the tanh and the sigmoid functions in the secret sharing domain. We note that prior work [18–20] also makes use of the PLA approximation. However, during our experiments, we observed that their proposed approximation could induce the severe ‘vanishing gradient problem’, which makes the NNs quite imprecise. More details are given later.

Replacements with SQNL-Family. Our second insight is to leverage the SQNL-family [74, 84] from deep learning literature. Instead of a vanilla approximation, the SQNL-family is a suite of new activation functions demonstrating promising accuracy over comprehensive empirical evaluations. It utilizes the universal approximation theorem [85, 86] and introduces a quadratic non-linearity. With such an observation, CryptMed devises the secure tanh_{SQNL} function, secure $\text{sigmoid}_{\text{LogSQNL}}$ function, and the secure ELU_{SQNL} function for the tanh, sigmoid, and ELU activation functions. In this subsequent sections, we provide the details of their constructions.

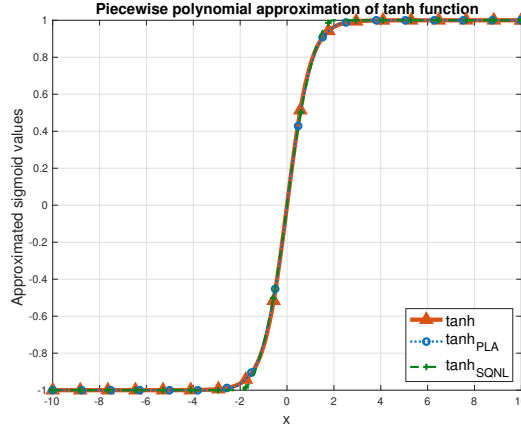


Fig. 4. The tanh function and its approximation.

5.4.1. Secure Tanh Function

As defined in Table 3, the tanh function involves exponentiation and division operations that are prohibitively expensive to be evaluated in secure domain. In CryptMed, we propose the secure \tanh_{PLA} function and the secure \tanh_{SQNL} function that demonstrate promising precision as plotted in Fig. 4.

Secure \tanh_{PLA} Function. The \tanh_{PLA} function is a piecewise polynomial approximation resorting to the non-linear and low-degree approximation [46, 83]. Such a PLA has *quantitative* performance, where the average error and the maximum error are $E_{ave} = 0.41\%$ and $E_{max} = 2.2\%$, respectively. Given the feature on each neuron x , the cleartext functionality can be reformulated to an MSB extraction based construction via

$$\tanh_{PLA}(x) = \begin{cases} \text{sign}(x) \cdot (-0.2716x^2 + |x| + 0.016) & \text{if } |x| \leq 0.016 \\ \text{sign}(x) \cdot (-0.0848x^2 + 0.42654|x| + 0.4519) & \text{if } 0.016 \leq |x| < 2.57 \\ \text{sign}(x) & \text{if } |x| \geq 2.57 \end{cases}$$

$$\xrightarrow{\text{transform}} \underbrace{\text{MSB}(100x - 257)}_{c_1} \cdot \underbrace{\neg \text{MSB}(125x - 2)}_{b_3} \cdot \underbrace{\text{sign}(x)}_c \cdot (-10x^2 + 51x + 54)/120 \quad (1)$$

$$+ \underbrace{\neg \text{MSB}(100x + 257)}_{b_2} \cdot \underbrace{\text{MSB}(125x + 2)}_{b_4} \cdot \text{sign}(x) \cdot (-10x^2 - 51x + 54)/120 \quad (2)$$

$$+ \underbrace{\neg \text{MSB}(125x + 2) \cdot \text{MSB}(125x - 2)}_{c_3} \cdot \text{sign}(x) \cdot (-27x^2 + 125\text{sign}(x) \cdot x + 2)/125 \quad (3)$$

$$+ \underbrace{\neg \text{MSB}(100x - 257)}_{c_4} \cdot \text{sign}(x) + \text{MSB}(100x + 257) \cdot \text{sign}(x). \quad (4)$$

Such a reformulated construction can be viewed as five polynomials, and each of them is triggered by a control bit. Each control bit indicates the certain threshold x located in and is derived from the MSBs of the comparison results. For example, the first polynomial in Eq. 1 is defined as $\text{sign}(x) \cdot (-10x^2 + 51x +$

54)/120 and is triggered by the control bit c_1 . That is to say, when $0.016 \leq x < 2.57$, then $c_1 = 1$, and the other control bits are '0's, and thus the tanh activation of x is produced based on above polynomial. In addition, the control bit c_1 stems from the MSBs of two comparison results: the MSB $b_1 = 1$ indicates $x < 2.57$ and the MSB $b_3 = 0$ indicates $x \geq 0.016$. Note that the control bit of the last polynomial in Eq. 4 is the MSB b_2 .

To this end, we present the secure realization of the secure tanh_{PLA} function based on above reformulation. Given the additive shares of each neuron's feature $\langle x \rangle$, the hospital P_0 and the medical service P_1 jointly compute the secure tanh_{PLA} function as follows:

- (1) P_0 and P_1 run to get the MSB $\llbracket b \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$, $\llbracket b_1 \rrbracket \leftarrow \text{MSB}(100\langle x \rangle - 257)$, $\llbracket b_2 \rrbracket \leftarrow \text{MSB}(100\langle x \rangle + 257)$, $\llbracket b_3 \rrbracket \leftarrow \text{MSB}(125\langle x \rangle - 2)$, $\llbracket b_4 \rrbracket \leftarrow \text{MSB}(125\langle x \rangle + 2)$.
- (2) P_0 and P_1 compute the control bits $\llbracket c_1 \rrbracket = \llbracket b_1 \rrbracket \cdot (\llbracket b_3 \rrbracket + i)$, $\llbracket c_2 \rrbracket = \llbracket b_4 \rrbracket \cdot (\llbracket b_2 \rrbracket + i)$, $\llbracket c_3 \rrbracket = \llbracket b_3 \rrbracket \cdot (\llbracket b_4 \rrbracket + i)$, and $\llbracket c_4 \rrbracket = \llbracket b_1 \rrbracket + i$.
- (3) P_0 and P_1 run to get the additively shared bits $\langle b \rangle \leftarrow \text{B2A}(\llbracket b \rrbracket)$, $\langle c_1 \rangle \leftarrow \text{B2A}(\llbracket c_1 \rrbracket)$, $\langle c_2 \rangle \leftarrow \text{B2A}(\llbracket c_2 \rrbracket)$, $\langle c_3 \rangle \leftarrow \text{B2A}(\llbracket c_3 \rrbracket)$, $\langle c_4 \rangle \leftarrow \text{B2A}(\llbracket c_4 \rrbracket)$, and $\langle b_2 \rangle \leftarrow \text{B2A}(\llbracket b_2 \rrbracket)$.
- (4) P_0 and P_1 compute the sign bit $\langle c \rangle = 1 - 2 \cdot \langle b \rangle$.
- (5) P_0 and P_1 compute the Eq. 1 via $\langle z_1 \rangle = \langle c_1 \rangle \cdot \langle c \rangle \cdot \lfloor (-10\langle x \rangle \cdot \langle x \rangle + 51\langle x \rangle + 54)/120 \rfloor$.
- (6) P_0 and P_1 compute the Eq. 2 via $\langle z_2 \rangle = \langle c_2 \rangle \cdot \langle c \rangle \cdot \lfloor (-10\langle x \rangle \cdot \langle x \rangle - 51\langle x \rangle + 54)/120 \rfloor$.
- (7) P_0 and P_1 compute the Eq. 3 via $\langle z_3 \rangle = \langle c_3 \rangle \cdot \langle c \rangle \cdot \lfloor (-27\langle x \rangle \cdot \langle x \rangle + 125 \cdot \langle c \rangle \langle x \rangle + 2)/125 \rfloor$.
- (8) P_0 and P_1 compute the Eq. 4 via $\langle z_4 \rangle = \langle c_4 \rangle \cdot \langle c \rangle + \langle b_2 \rangle \cdot \langle c \rangle$.
- (9) P_0 and P_1 produce the tanh_{PLA} activation $\langle z \rangle = \langle z_1 \rangle + \langle z_2 \rangle + \langle z_3 \rangle + \langle z_4 \rangle$.

Secure tanh_{SQL} Function. The tanh_{SQL} function uses the SQLN-family [74] demonstrating promising accuracy over comprehensive empirical evaluations. Given the feature on each neuron x , the cleartext functionality can be reformulated to an MSB extraction based construction via

$$\text{tanh}_{SQL}(x) = \begin{cases} 1 & \text{if } x > 2 \\ x - 0.25 \cdot \text{sign}(x) \cdot x^2 & \text{if } -2 \leq x < 2 \\ -1 & \text{if } x < -2 \end{cases} \quad (5)$$

$$\xrightarrow{\text{transform}} \underbrace{\text{MSB}(x-2) \cdot \neg \text{MSB}(x+2)}_{c_1} \cdot \underbrace{\lfloor (4x - \text{sign}(x) \cdot x^2)/4 \rfloor}_{c} + \underbrace{\neg \text{MSB}(x-2) - \text{MSB}(x+2)}_{c_2}$$

Such a construction consists of three polynomials that are triggered by the control bits c_1, c_2 , and the $\text{MSB}(x+2)$ indicating the thresholds $-2 \leq x < 2$, $x > 2$, and $x < -2$, respectively. Each control bit is derived from the MSBs of comparing x with the threshold boundaries.

With the above reformulation and the additive shares of each neuron's feature $\langle x \rangle$, the hospital P_0 and the medical service P_1 jointly compute the secure tanh_{SQL} function as follows:

- (1) P_0 and P_1 run to get the MSB $\llbracket b \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$, $\llbracket b_1 \rrbracket \leftarrow \text{MSB}(\langle x \rangle - 2)$, and $\llbracket b_2 \rrbracket \leftarrow \text{MSB}(\langle x \rangle + 2)$.
- (2) P_0 and P_1 compute the control bits $\llbracket c_1 \rrbracket = \llbracket b_1 \rrbracket \cdot (\llbracket b_2 \rrbracket + i)$ and $\llbracket c_2 \rrbracket = \llbracket b_1 \rrbracket + i$.
- (3) P_0 and P_1 run to get the additively shared bits $\langle b \rangle \leftarrow \text{B2A}(\llbracket b \rrbracket)$, $\langle c_1 \rangle \leftarrow \text{B2A}(\llbracket c_1 \rrbracket)$, $\langle c_2 \rangle \leftarrow \text{B2A}(\llbracket c_2 \rrbracket)$, and $\langle b_2 \rangle \leftarrow \text{B2A}(\llbracket b_2 \rrbracket)$.
- (4) P_0 and P_1 compute the sign bit $\langle c \rangle = 1 - 2 \cdot \langle b \rangle$.
- (5) P_0 and P_1 produce the tanh_{SQL} activation $\langle z \rangle = \langle c_1 \rangle \cdot \lfloor (4\langle x \rangle - \langle c \rangle \cdot \langle x \rangle \cdot \langle x \rangle)/4 \rfloor + \langle c_2 \rangle - \langle b_2 \rangle$.

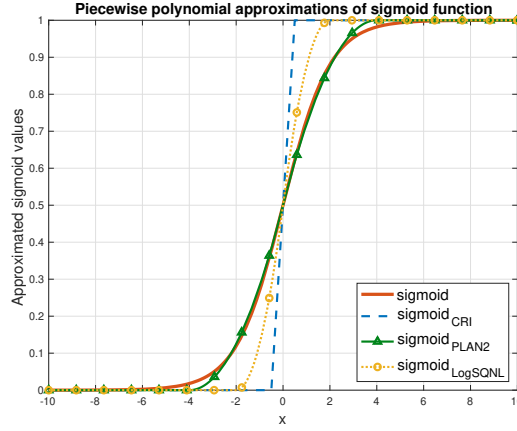


Fig. 5. The sigmoid function and its approximations.

5.4.2. Secure Sigmoid Function

The sigmoid function, as defined in Table 3, comprises the exponentiation and division operations which are knowingly computational extensive by a secure two-party computation protocol [10, 79, 87].

Prior work [18–20] leverages the first-order PLA to approximate the sigmoid function denoted as

$$\text{sigmoid}_{CRI}(x) = \begin{cases} 0 & \text{if } x < -1/2 \\ x + 1/2 & \text{if } -1/2 \leq x < 1/2 \\ 1 & \text{if } 1/2 \leq x \end{cases}$$

in our experiments, we observed that the $\text{sigmoid}_{CRI}(x)$ approximation could introduce severe ‘vanishing gradient problem’ making the NNs hardly converge during training. As a consequence, the prediction accuracy of the trained NNs is quit undesired, particularly when dealing with complex datasets. Table 4 showcases the vanishing gradient problem of the $\text{sigmoid}_{CRI}(x)$ approximation in our experiments.

Instead, CryptMed takes advantage of the insights from the field of digital circuit design [81, 82] and the machine learning literature [74, 84] to propose more precise approximations, i.e., the second-order PLA (sigmoid_{PLAN2}) and the SQLN-family replacement (sigmoid_{LogSQL}). Fig. 5 illustrates the approximations of sigmoid used in CryptMed and the sigmoid_{CRI} used in prior art. Note that, all these approximations can be converted to the MSB extraction based functions and can be efficiently evaluated in the secret sharing domain. In this subsequent section, we expatiate on the secure sigmoid_{PLAN2} function and secure sigmoid_{LogSQL} function.

Secure sigmoid_{PLAN2} Function. The sigmoid_{PLAN2} function is a non-linear and second-order approximation [81, 82] of the sigmoid function, where the average error and the maximum error are quantified as $E_{ave} = 0.41\%$ and $E_{max} = 2.2\%$, respectively. Given the feature on each neuron x , the cleartext

Table 4
Vanishing gradient problem of the sigmoid_{CRI} approximation.

	sigmoid	sigmoid_{CRI}	sigmoid_{PLAN2}	sigmoid_{LogSQL}
Thyroid	81.47%	<u>5.16 ~ 9.88%</u>	86.55%	85.85%
MNIST	96.44%	<u>11.35%</u>	91.41%	96.75%
CIFAR-10	47.87%	<u>10.03%</u>	47.87%	59.08%

functionality can be converted to an MSB extraction based construction via

$$\begin{aligned}
 sigmoid_{PLAN2}(x) &= \begin{cases} \neg\text{MSB}(x) & \text{if } |x| \geq 4 \\ -\text{sign}(x) \cdot 0.03125x^2 + 0.25x + 0.5 & \text{if } 0 \leq |x| < 4 \end{cases} \\
 \xrightarrow{\text{transform}} & \underbrace{\text{MSB}(x-4) \cdot \neg\text{MSB}(x+4)}_{c_1} \cdot \underbrace{[-\text{sign}(x) \cdot x^2 + 8x + 16]/32}_c + \underbrace{\neg\text{MSB}(x-4)}_{c_2}. \quad (6)
 \end{aligned}$$

It comprises two polynomials that are triggered by the control bits c_1 and c_2 indicating the thresholds $-4 < x < 4$, and $x \geq 4$, respectively. Note that here we prune off the threshold $x \leq -4$, since the polynomial keeps zero ($\neg\text{MSB}(x) \cdot c_2 = 0$) in such a threshold.

Given above reformulation and the additive shares of each neuron's feature $\langle x \rangle$, the hospital P_0 and the medical service P_1 jointly compute the secure $sigmoid_{PLAN2}$ function as follows:

- (1) P_0 and P_1 run to get the MSB $\llbracket b \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$, $\llbracket b_1 \rrbracket \leftarrow \text{MSB}(\langle x \rangle - 4)$, and $\llbracket b_2 \rrbracket \leftarrow \text{MSB}(\langle x \rangle + 4)$.
- (2) P_0 and P_1 compute the control bits $\llbracket c_1 \rrbracket = \llbracket b_1 \rrbracket \cdot (\llbracket b_2 \rrbracket + i)$ and $\llbracket c_2 \rrbracket = \llbracket b_1 \rrbracket + i$.
- (3) P_0 and P_1 run to get $\langle b \rangle \leftarrow \text{B2A}(\llbracket b \rrbracket)$, $\langle c_1 \rangle \leftarrow \text{B2A}(\llbracket c_1 \rrbracket)$, and $\langle c_2 \rangle \leftarrow \text{B2A}(\llbracket c_2 \rrbracket)$.
- (4) P_0 and P_1 compute the sign bit $\langle c \rangle = 1 - 2 \cdot \langle b \rangle$.
- (5) P_0 and P_1 produce the $sigmoid_{PLAN2}$ activation $\langle z \rangle = \langle c_1 \rangle \cdot [(8\langle x \rangle - \langle c \rangle \cdot \langle x \rangle \cdot \langle x \rangle + 16)/32] + \langle c_2 \rangle$.

Secure $sigmoid_{LogSQNL}$ Function. The $sigmoid_{LogSQNL}$ function resorts to a new activation function LogSQNL from the SQNL-family [74, 84], which is a precise replacement of the sigmoid function. Given the feature on each neuron x , the cleartext functionality and corresponding MSB extraction based construction are defined as follows

$$\begin{aligned}
 sigmoid_{LogSQNL}(x) &= \begin{cases} 1 & \text{if } x > 2 \\ 0.5x - 0.125 \cdot \text{sign}(x) \cdot x^2 + 0.5 & \text{if } -2 \leq x < 2 \\ 0 & \text{if } x < -2 \end{cases} \\
 \xrightarrow{\text{transform}} & \underbrace{\text{MSB}(x-2) \cdot \neg\text{MSB}(x+2)}_{c_1} \cdot \underbrace{[(4x - \text{sign}(x) \cdot x^2 + 4)/8]}_c + \underbrace{\neg\text{MSB}(x-2)}_{c_2} \quad (7)
 \end{aligned}$$

The MSB extraction based construction consists of two polynomials. They are activated by the control bits c_1 and c_2 when x in the thresholds $-2 \leq x < 2$, and $x \geq 2$, respectively. Similar to the $sigmoid_{PLAN2}$ function, the polynomial in threshold $x < -2$ is trimmed due to its zero value. Given the additive shares of each neuron's feature $\langle x \rangle$, the hospital P_0 and the medical service P_1 jointly compute the secure $sigmoid_{LogSQNL}$ function as follows:

- (1) P_0 and P_1 run to get the MSB $\llbracket b \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$, $\llbracket b_1 \rrbracket \leftarrow \text{MSB}(\langle x \rangle - 2)$, and $\llbracket b_2 \rrbracket \leftarrow \text{MSB}(\langle x \rangle + 2)$.
- (2) P_0 and P_1 compute the control bits $\llbracket c_1 \rrbracket = \llbracket b_1 \rrbracket \cdot (\llbracket b_2 \rrbracket + i)$ and $\llbracket c_2 \rrbracket = \llbracket b_1 \rrbracket + i$.
- (3) P_0 and P_1 run to get $\langle b \rangle \leftarrow \text{B2A}(\llbracket b \rrbracket)$, $\langle c_1 \rangle \leftarrow \text{B2A}(\llbracket c_1 \rrbracket)$, and $\langle c_2 \rangle \leftarrow \text{B2A}(\llbracket c_2 \rrbracket)$.
- (4) P_0 and P_1 compute the sign bit $\langle c \rangle = 1 - 2 \cdot \langle b \rangle$.
- (5) P_0 and P_1 produce the $sigmoid_{LogSQNL}$ activation $\langle z \rangle = \langle c_1 \rangle \cdot [(4\langle x \rangle - \langle c \rangle \cdot \langle x \rangle \cdot \langle x \rangle + 4)/8] + \langle c_2 \rangle$.

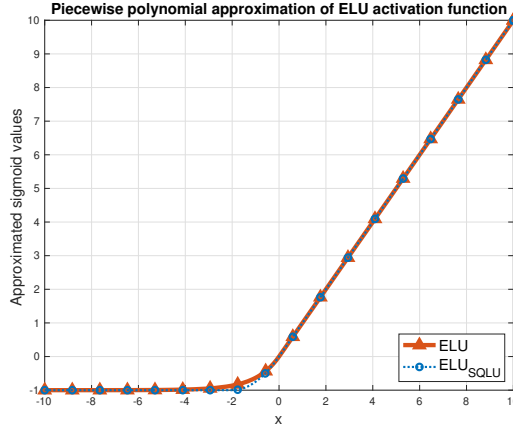


Fig. 6. The ELU activation function and its approximation.

5.4.3. Secure Exponential Linear Units Function

As Fig. 6 illustrates, CryptMed proposes the ELU_{SQLU} function taking advantages of the SQLU function from the SQLN-family [74, 84]. It is defined as follows

$$ELU_{SQLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ x + 0.25x^2 & \text{if } -2 \leq x \leq 0 \\ -1 & \text{if } x < -2 \end{cases}$$

$$\xrightarrow{\text{transform}} \underbrace{\text{MSB}(x) \cdot \neg \text{MSB}(x+2)}_{c_1} \cdot [(4x + x^2)/4] + \underbrace{\neg \text{MSB}(x)}_{c_2} \cdot x - \text{MSB}(x+2) \quad (8)$$

Given the additive shares of each neuron's feature $\langle x \rangle$, the hospital P_0 and the medical service P_1 jointly compute the secure ELU_{SQLU} activation function as follows:

- (1) P_0 and P_1 run to get the MSB $\llbracket b_1 \rrbracket \leftarrow \text{MSB}(\langle x \rangle)$, $\llbracket b_2 \rrbracket \leftarrow \text{MSB}(\langle x \rangle + 2)$.
- (2) P_0 and P_1 compute the control bits $\llbracket c_1 \rrbracket = \llbracket b_1 \rrbracket \cdot (\llbracket b_2 \rrbracket + i)$ and $\llbracket c_2 \rrbracket = \llbracket b_1 \rrbracket + i$.
- (3) P_0 and P_1 run to get the additively shared bits $\langle b_2 \rangle \leftarrow \text{B2A}(\llbracket b_2 \rrbracket)$, $\langle c_1 \rangle \leftarrow \text{B2A}(\llbracket c_1 \rrbracket)$, and $\langle c_2 \rangle \leftarrow \text{B2A}(\llbracket c_2 \rrbracket)$.
- (4) P_0 and P_1 produce the ELU_{SQLU} activation $\langle z \rangle = \langle c_1 \rangle \cdot [(4\langle x \rangle + \langle x \rangle \cdot \langle x \rangle)/4] + \langle c_2 \rangle \cdot \langle x \rangle - \langle b_2 \rangle$.

5.5. Secure Pooling Layers

The MaxPool layer $\max(x_1, \dots, x_n)$ (or MinPool $\min(x_1, \dots, x_n)$) can be views as the pairwise maximum (or minimum) over features within the n -width pooling window. They can be realized based on the secure $\text{MSB}(\cdot)$ extraction as follows:

$$\mathbf{b} \leftarrow \text{MSB}(x_1 - x_2); \quad // \mathbf{b} = 0, x_1 \geq x_2; \mathbf{b} = 1, x_1 < x_2$$

$$z = \max(x_1, x_2) = \mathbf{b} \cdot (x_2 - x_1) + x_1; \quad z = \min(x_1, x_2) = \mathbf{b} \cdot (x_1 - x_2) + x_2.$$

Based on above equations, given the secret shares of features $\langle x_1 \rangle, \dots, \langle x_n \rangle$, the hospital P_0 and the medical service P_1 perform the secure MaxPool (or MinPool) function as follows:

- 1 (1) For $k \in [1, n - 1]$:
- 2 (2) P_0 and P_1 run to get $\llbracket \mathbf{b} \rrbracket \leftarrow \text{MSB}(\langle x_k \rangle - \langle x_{k+1} \rangle)$.
- 3 (3) P_0 and P_1 run to get additively shared MSB $\langle \mathbf{b} \rangle \leftarrow \text{B2A}(\llbracket \mathbf{b} \rrbracket)$.
- 4 (4) P_0 and P_1 compute the maximum value $\langle z_k \rangle = \langle \mathbf{b} \rangle \cdot (\langle x_{k+1} \rangle - \langle x_k \rangle) + \langle x_k \rangle$; or the minimum value
- 5 $\langle z_k \rangle = \langle \mathbf{b} \rangle \cdot (\langle x_k \rangle - \langle x_{k+1} \rangle) + \langle x_{k+1} \rangle$. P_i sets $\langle x_{k+1} \rangle := \langle z_k \rangle$.
- 6 (5) Finally, P_i outputs the shares $\langle z_n \rangle_i$ as MaxPool (or MinPool) result.

7 The average pooling layer $\lfloor (x_1 + \dots + x_n)/n \rfloor$ can be directly computed over additive secret shares via
 8 secure addition, where n is a cleartext hyper-parameter.
 9

10 5.6. Security Analysis

11 In this section, we present a comprehensive security analysis of CryptMed's secure inference proto-
 12 col in the semi-honest adversary model. CryptMed's core secure inference protocol is devised based on
 13 standard additive secret sharing techniques [71, 88], where all the input data (i.e., medical data, neural
 14 network model) are perfectly protected as additive secret shares that are uniformly distributed in ring \mathbb{Z}_{2^ℓ} .
 15 Besides, during CryptMed's protocol execution, any message transcriptions are supported by standard
 16 Beaver's multiplication/AND procedure [72] as uniformly distributed secret shares. CryptMed provides
 17 stringent cryptographic guarantees throughout the service procedure, where only the prescribed infer-
 18 ence result can be learned by the hospital. Nothing else about each party's private input can be deduced
 19 from the counterparty beyond what is revealed from the inference result. In what follows, we formally
 20 define the ideal functionality $\mathcal{F}^{\text{CryptMed}}$, security definition, and the security proof of CryptMed's secure
 21 NN inference protocol under the ideal/real world paradigm.
 22

23 **Definition 1.** *The ideal functionality $\mathcal{F}^{\text{CryptMed}}$ of CryptMed's secure deep neural network inference con-*
 24 *sists of the following parts:*

- 25 - **Input.** *The medical service submits the DNN model \mathcal{W} and the hospital submits the medical record \mathbf{X}*
 26 *to $\mathcal{F}^{\text{CryptMed}}$.*
- 27 - **Computation.** *Upon receiving \mathcal{W} and \mathbf{X} , $\mathcal{F}^{\text{CryptMed}}$ conducts DNN inference and produces the inference*
 28 *result $\mathcal{W}(\mathbf{X})$.*
- 29 - **Output.** *$\mathcal{F}^{\text{CryptMed}}$ outputs $\mathcal{W}(\mathbf{X})$ only to the hospital, and returns no information to the medical service.*
 30

31 Given the ideal functionality, we formally define the security definition.
 32

33 **Definition 2.** *A protocol Π securely realizes the $\mathcal{F}^{\text{CryptMed}}$ if it provides the following guarantees in the*
 34 *presence of a probabilistic polynomial time (PPT) semi-honest adversary with static corruption:*

- 35 - **Corrupted hospital.** *A corrupted semi-honest hospital H should learn no information about the medi-*
 36 *cal service's DNN model weights except the hyper-parameters of model architecture. Formally, there*
 37 *should exist a PPT simulator Sim_{H} that can simulate the view $\text{View}_{\mathsf{H}}^{\Pi}$ of the corrupted hospital in real-*
 38 *world protocol execution: $\text{View}_{\mathsf{H}}^{\Pi} \stackrel{c}{\approx} \text{Sim}_{\mathsf{H}}(\mathbf{X}, \mathcal{W}(\mathbf{X}))$.*
- 39 - **Corrupted medical service.** *A corrupted semi-honest medical service S should learn no information*
 40 *about the medical record \mathbf{X} submitted by the hospital. Formally, there should exist a PPT simula-*
 41 *tor Sim_{S} that can simulate the view $\text{View}_{\mathsf{S}}^{\Pi}$ of the corrupted medical service in real-world protocol*
 42 *execution: $\text{View}_{\mathsf{S}}^{\Pi} \stackrel{c}{\approx} \text{Sim}_{\mathsf{S}}(\mathcal{W})$.*

Theorem 1. *CryptMed's secure deep neural network inference protocol securely emulates the ideal functionality $\mathcal{F}^{\text{CryptMed}}$ under the Security Definition 2.*

Proof. We present a simulator for the corrupted medical service or the corrupted hospital, in a way that the distribution of real-world protocol execution is computationally indistinguishable to the simulated distribution under our security definition.

- **Simulator for the corrupted hospital:** Define the simulator of Beaver's multiplication procedure is Sim_{BM} . The emulated view is indistinguishable from the real-world view of the corrupted hospital H in the multiplication procedure.

- i) In the preprocessing phase of Π , the simulator of the corrupted hospital Sim_{H} generates the randomness $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_{2^\ell}$ to emulate the message $\mathcal{W} - \mathbf{a}^1$ in real-world protocol execution. Both messages are uniformly distributed in ring \mathbb{Z}_{2^ℓ} . Given the security of additive secret sharing, H cannot distinguish the simulated message with the message received from real-world protocol execution. H computes $\langle \tilde{\mathbf{X}}_2 \rangle_0 = \mathbf{a}^0 \cdot \mathbf{r} + \mathbf{a}^2$ and $\mathbf{u} = \mathbf{a}^0$.
- ii) In the online phase of Π , H inputs the secret shares of medical record $\mathbf{X} - \mathbf{u}$ or the activation $\langle \tilde{\mathbf{X}}_i \rangle_0 - \mathbf{u}$ for secure linear layers and receives no messages for the linear layers. Sim_{H} works in a dummy way by directly outputting inputs of H. Thus, the output of Sim_{H} is identically distributed to the view $\text{View}_{\text{H}}^{\Pi}$. For the non-linear layers, Sim_{H} generates $\langle \tilde{\mathbf{X}}_{i+1} \rangle_1 \xleftarrow{\$} \mathbb{Z}_{2^\ell}$ and runs Sim_{BM} to compute secure multiplication over $\langle \mathbf{X}_{i+1} \rangle_1$ and $\langle \mathbf{X}_{i+1} \rangle_0$ whenever interactions are required. Sim_{H} outputs the simulated shares of activation returned from the secure non-linear layer. Sim_{H} conducts the above computations for every layer. At the end, Sim_{H} outputs the simulated shares of the last layer's result $\langle \tilde{\mathbf{X}}_L \rangle_0, \langle \tilde{\mathbf{X}}_L \rangle_1$. The reconstructed value of these two shares is uniformly distributed in ring \mathbb{Z}_{2^ℓ} , same as the result from the real-world protocol execution. Therefore, the output of $\text{Sim}_{\text{H}}(\mathbf{X}, \mathcal{W}(\mathbf{X}))$ is computationally indistinguishable to $\text{View}_{\text{H}}^{\Pi}$ the view of the corrupted hospital.

- **Simulator for the corrupted medical service:**

- i) In the preprocessing phase of Π , the corrupted medical service S only inputs the secret shares of model $\mathcal{W} - \mathbf{A}^1$ and receives no message. Sim_{S} works in a dummy way by directly outputting the inputs of S $\mathbf{v} = \mathbf{a}^1$. In this way, the output of Sim_{S} is identically distributed to the view $\text{View}_{\text{S}}^{\Pi}$.
- ii) In the online phase of Π , Sim_{S} generates and outputs the randomness $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_{2^\ell}$ to emulate the message $\mathbf{X} - \mathbf{u}$ (or $\tilde{\mathbf{X}} - \mathbf{u}$) in the real-world protocol execution. Given the security of additive secret sharing, S cannot distinguish the emulated message with the one received from real protocol. For the non-linear layers, Sim_{S} generates $\langle \tilde{\mathbf{X}}_{i+1} \rangle_0 \xleftarrow{\$} \mathbb{Z}_{2^\ell}$. Whenever interactions happen in the secure activation function, Sim_{S} runs Sim_{BM} to perform the Beaver's secure multiplication procedure over $\langle \tilde{\mathbf{X}}_{i+1} \rangle_0$ and $\langle \mathbf{X}_{i+1} \rangle_1$ received from the corrupted medical service S. Sim_{S} outputs the simulated shares of activation outputted from the secure activation function. Sim_{S} conducts the above computations for every layer. Because all simulated intermediate messages are uniformly distributed in \mathbb{Z}_{2^ℓ} , and given the security of additive secret sharing and Beaver's secure multiplication procedure, the output of $\text{Sim}_{\text{S}}(\mathcal{W})$ is computationally indistinguishable to $\text{View}_{\text{S}}^{\Pi}$ the view of the corrupted medical service.

□

Table 5
Performance of secure layer functions.

Secure layer	Conv.		FC	BN	ReLU	MaxPool	AvgPool
	3×3	5×5	16×16				
Time (ms)	1.25	2.16	8.44	1.74	22.7	31.2	0.05
Comm. (Bytes)	36	100	943	4	78	234	0

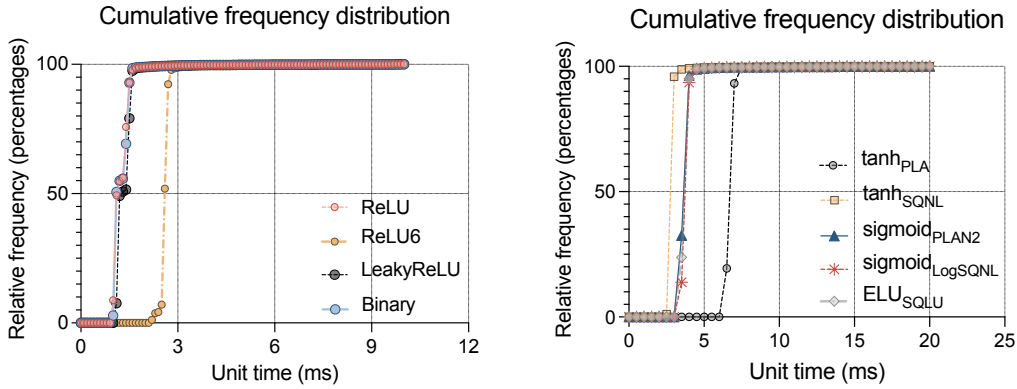


Fig. 7. Unit time of the secure activation functions.

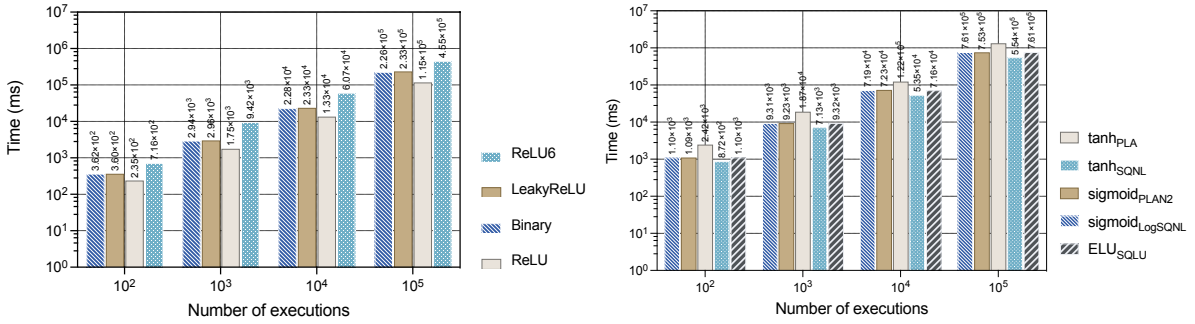


Fig. 8. Computational overhead of the secure activation functions.

6. Performance Evaluation

We implement a proof-of-concept prototype of CryptMed in Java. We deploy our prototype to Australian MASSIVE M3 using m3i computation nodes. Each computation node has 2.7GHz Intel Xeon Gold 6150 CPU and 384GB RAM, running CentOS Linux 7 system. In our experiment, we choose to protect the data as additive secret shares in 32-bit ring \mathbb{Z}_{2^ℓ} . In line with prior secure inference systems [9, 10, 28], we deploy the computation nodes representing the medical service and the hospital in a dedicated network. For cleartext neural networks implementation and training, we use the Pytorch backend and train our models on a NVIDIA Tesla V100 GPU.

We evaluate CryptMed's performance in terms of performance and accuracy. To evaluate the performance, we train 78 neural network models based on six architectures (see appendix Sec. A) across

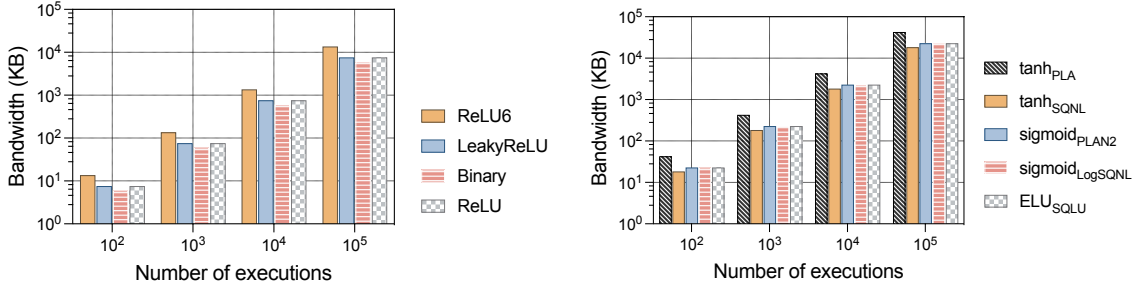


Fig. 9. Communication overhead of the secure activation functions.

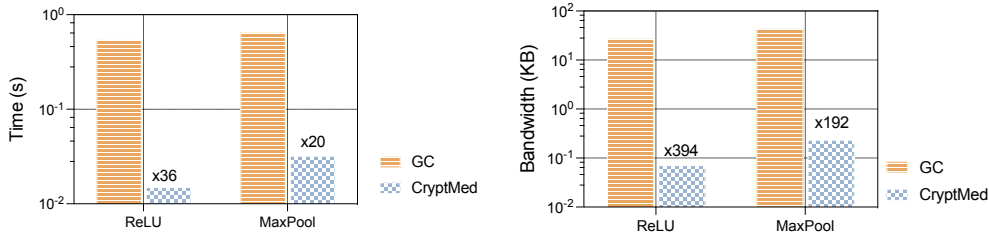


Fig. 10. Performance comparison of the secure ReLU and MaxPool layers with GC baseline.

13 different activation functions. These activation functions include the secure comparison-based activations (ReLU, ReLU6, Leaky ReLU, Binary activation), the original smooth activations (tanh, sigmoid, ELU) and the secure smooth activations based on different approximation approaches (\tanh_{PLA} , \tanh_{SQNL} , sigmoid_{CRI} , sigmoid_{PLAN2} , $\text{sigmoid}_{LogSQNL}$, and ELU_{SQLU}). They are trained over MNIST, CIFAR-10, Breast Cancer, Diabetes, Liver Disease, and Thyroid. The goal of our evaluation is to answer two questions:

- 1) Whether CryptMed's secure inference protocol lightweight?
- 2) Is CryptMed practical and accurate for secure medical diagnostic service?

To answer above questions, we evaluate microbenchmarks of CryptMed's secure layer functions and a series of end-to-end system-level inference performance.

6.1. Microbenchmarks

Secure Layer Functions. Table 5 summarizes the performance of commonly-used secure layer functions, including a series of secure linear layers (CONV, FC, BN, AvgPool) and the secure non-linear layers ReLU and MaxPool. For demonstration purpose, we set the parameters of the sliding windows as 3×3 and 5×5 for the CONV kernels, the pooling window as 2×2 for the AvgPool and MaxPool, and the 16×16 vectors for FC. As reported, all secure layers in CryptMed are lightweight and can be efficiently evaluated within 35ms consuming at most 1KB bandwidth.

For secure non-linear activation functions, we first investigate their performance in terms of execution time and network consumption, including secure comparison-based activations $ReLU$, $ReLU6$, $LeakyReLU$, $Binary$, and secure smooth activations \tanh_{PLA} , \tanh_{SQNL} , sigmoid_{PLAN2} , $\text{sigmoid}_{LogSQNL}$, ELU_{SQLU} . Fig. 8 and Fig. 9 illustrate the computational and communication overheads of secure comparison-based activations (in respective left figures) and secure smooth activations (in respective right figures). As shown, all secure smooth activations require around $10\times$ more resources than secure comparison-based activations. Nevertheless, CryptMed's secure non-linear activation functions can

Table 6
Prediction workload of NNs over different secure comparison-based activation functions.

Dataset	overhead	Binary	ReLU	ReLU6	LeakyReLU
Breast Cancer	time (s)	0.28	0.22	0.35	0.28
	comm. (KB)	8.14	8.64	10.56	8.64
Diabetes	time (s)	0.24	0.39	0.33	0.24
	comm. (KB)	40.34	40.97	43.37	40.97
Liver Disease	time (s)	0.19	0.19	0.34	0.19
	comm. (KB)	8.22	9.22	13.07	9.22
Thyroid	time (s)	0.88	0.94	1.34	0.89
	comm. (KB)	110.46	113.59	125.60	113.59
MNIST	time (s)	0.82	0.64	1.41	0.84
	comm. (MB)	0.91	0.91	0.93	0.91
CIFAR-10	time (s)	423.58	146.9	820.07	436.05
	comm. (MB)	496.18	498.83	508.98	498.83

be accomplished within 10ms for a single execution, as plotted in Fig. 7 by grabbing 10^4 unit executions. In particular, for the secure smooth functions, the SQNL-family based approximations (\tanh_{SQNL} , $\text{sigmoid}_{\text{LogSQNL}}$, ELU_{SQLU}) are relatively lightweight than the PLA based approximations (\tanh_{PLA} and $\text{sigmoid}_{\text{PLAN2}}$). In combination with their superior prediction accuracy, as shown later in Table 10, the SQNL-family based secure smooth activations would be more desirable in practical medical diagnostic applications.

Secure Non-linear Layers Comparison with GC. We compare CryptMed’s secure realizations of the secure ReLU and MaxPool with the common GC-based solutions, since prior works target only on these two non-linear layers. The GC baseline is implemented with FlexSC [89], a Java based two-party GC framework in the semi-honest setting. In our implementation, we adopt the free-XOR and half-AND optimizations for GC. All GC-based realizations are implemented with equivalent functionalities to ours. As Fig. 10 depicts, CryptMed’s realizations are $36\times$, $20\times$ faster than the GC baseline for secure ReLU and MaxPool, respectively. For bandwidth consumption, CryptMed’s secure ReLU and MaxPool achieve respective $394\times$, $192\times$ bandwidth savings compared with the GC baseline. Such improvements demonstrate that CryptMed’s secret sharing based design is indeed lightweight and much more efficient than the prior works relying on GC [10, 12, 15, 28].

6.2. CryptMed’s Protocol Performance

Prediction Workload. Table 6 and Table 7 summarize the end-to-end prediction workload of CryptMed’s system with 9 different activation functions over real-world medical applications and the commonly-used machine learning benchmarks. In summary, CryptMed’s secure NN inference system is lightweight and low-latency over all medical datasets. By using the secure comparison-based activations, CryptMed can produce deep learning based medical conclusions within 1s consuming less than 130KB bandwidth for all evaluated medical datasets. For more complex secure smooth activations, CryptMed requires less than 31s and 200KB to produce a deep learning based medical conclusion. For the ML benchmarks, CryptMed evaluates MNIST within 4s and 1MB network resources. For the CIFAR-10 dataset on a complicated 10 layers NN, CryptMed produces an inference within 14min, 509MB using secure comparison-based activations, and 38min, 558MB using secure smooth activations. Furthermore,

Table 7
Prediction workload of NNs over different secure smooth activation functions.

Dataset	overhead	sigmoid_{PLAN2}	$\text{sigmoid}_{LogS_{QNL}}$	tanh_{PLA}	$\text{tanh}_{S_{QNL}}$	$\text{ELU}_{S_{QLU}}$
Breast Cancer	time (s)	0.45	0.45	0.63	0.38	0.45
	comm. (KB)	13.48	13.48	19.83	12.06	13.48
Diabetes	time (s)	0.45	0.45	0.67	0.37	0.45
	comm. (KB)	47.02	47.02	54.95	45.24	47.02
Liver Disease	time (s)	0.53	0.53	0.89	0.40	0.53
	comm. (KB)	18.91	18.91	31.60	16.07	18.91
Thyroid	time (s)	19.37	19.54	30.56	15.39	19.54
	comm. (KB)	143.86	143.86	183.51	134.97	143.86
MNIST	time (s)	2.17	2.20	3.61	1.66	2.19
	comm. (MB)	0.95	0.95	1.00	0.94	0.95
CIFAR-10	time (s)	1336.20	1350.77	2304.31	991.88	1350.39
	comm. (MB)	524.41	524.41	557.91	516.90	524.41

Table 8
Preprocessing overhead of secure NNs.

Overhead	Breast Cancer	Diabetes	Liver Disease	Thyroid	MNIST	CIFAR-10
Time (s)	0.1	0.03	0.06	0.28	0.07	243
Comm. (MB)	0.003	0.0022	0.018	0.048	0.45	246.4

we report the performance of secure preprocessing overhead in Table 8. Note that, the costs of preprocessing are only related to the NN sizes, i.e., the amount of weights in linear layers (non-linear layers do not produce any weights).

Prediction Accuracy. Table 9 and Table 10 demonstrate the prediction accuracy of our system over different activation functions. We note that, for the original smooth activation functions sigmoid, tanh, ELU, and the sigmoid_{CRI} used in prior work [18, 20], we evaluate their accuracy in cleartext domain. For each family of the smooth functions, we highlight the most precise functions and underline the one facing with gradient vanishing problem. As shown, our proposed approximations do not introduce many losses to prediction accuracy. In particular, some of them from the SQNL-family can even enhance the accuracy. Consider both the evaluation costs and the accuracy, the secure smooth functions based on the SQNL-family would be better suitable for deep learning based medical diagnostic services.

Comparison with Prior Art. Table 11 compares CryptMed’s end-to-end inference performance with notable prior secure NN inference works. As shown, CryptMed requires the least network resources among all other prior works with up to $413\times$ bandwidth saving for MNIST and up to $19\times$ bandwidth saving for CIFAR-10. For the medical applications, CryptMed improves up to $43\times$ communication over XONN, the notable work investigating medical scenario with smaller quantized NNs and network trimming optimization.

For the state-of-the-art work Delphi [12], their all ReLU version consumes total 5100MB, whereas CryptMed only requires 498MB, with a $10\times$ enhancement². Such significant improvement stems from

²Preprocessing: 243MB in CryptMed and 4915MB in Delphi.

Table 9

Prediction accuracy of different comparison-based activation functions.

Dataset	Binary	ReLU	ReLU6	LeakyReLU
Breast Cancer	88.81%	93.0%	93.7%	91.61%
Diabetes	68.75%	71.87%	73.43%	73.43%
Liver Disease	71.72%	72.0%	66.89%	71.03%
Thyroid	48.77%	86.31%	86.20%	86.11%
MNIST	75.78%	97%	96.97%	97.57%
CIFAR-10	13.2%	81%	80%	82.68%

Table 10

Prediction accuracy of different approximated smooth activation functions.

Dataset	sigmoid family				tanh family			ELU family	
	sigmoid	sigmoid_{CRI}	sigmoid_{PLAN2}	$\text{sigmoid}_{LogS_{QNL}}$	tanh	tanh_{PLA}	$\text{tanh}_{S_{QNL}}$	ELU	$\text{ELU}_{S_{QNL}}$
Breast Cancer	92.3%	86.01%	93.01%	95.8%	92.3%	72.02%	93.01%	95.1%	95.8%
Diabetes	68.75%	67.7%	74.47%	72.39%	70.31%	69.79%	70.31%	72.91%	71.87%
Liver Disease	71.72%	65.51%	71.72%	71.03%	69.65%	71.03%	69.65%	70.34%	69.65%
Thyroid	81.47%	<u>5.16 ~ 9.88%</u>	86.55%	85.85%	85.56%	64.2%	85.88%	85.5%	85.88%
MNIST	96.44%	<u>11.35%</u>	91.41%	96.75%	96.03%	38.62%	96.93%	97.58%	97.67%
CIFAR-10	47.87%	<u>10.03%</u>	47.87%	59.08%	76.0%	<u>19.71%</u>	74.19%	81.27%	83.87%

Table 11

Bandwidth (MB) comparison of CryptMed with prior art.

	MNIST	CIFAR-10		Breast Cancer		Diabetes		Liver Disease	
MiniONN	15.8	MiniONN	9272	XONN	0.35	XONN	0.16	XONN	0.3
CryptoNets	372.2	FALCON	1278						
XONN	4.29	XONN	2599						
Chameleon	10.5	Chameleon	2650						
		Gazelle (ReLU)	~5000						
		Delphi (ReLU)	~5100						
CryptMed	0.9	CryptMed	498	CryptMed	0.008	CryptMed	0.04	CryptMed	0.009

the fact that CryptMed only involves lightweight secret sharing based secure computation through out the whole service procedure. In comparison, Delphi relies on the usages of heavy cryptography, i.e., homomorphic encryption for secure preprocessing and garbled circuits for secure non-linear layers. Regarding the overall runtime, we emphasize that it is not a fair comparison to directly compare the experiment results reported in [12]. The reason is that Delphi is implemented in a different programming language (Rust) with significant optimizations and acceleration from GPU computing. Our evaluation results are not based on such optimizations.

We note that secure evaluation of non-linear layers is the *performance bottleneck* in secure neural network inference [12]. To support the non-linearity introduced by the original ReLU, Delphi adopts a GC-based realization. For a fair comparison, we have reported in above Fig. 10 the performance of our design and the GC-based realization. The results have validated a significant performance boost of our

design over the GC-based realization ($36\times$ in runtime and $398\times$ in communication). Even with a direct (*unfair*) comparison, CryptMed’s overall inference time, with much simplified implementations, is still comparable to Delphi’s reported runtime (147s in CryptMed against 140s in Delphi) which is driven by aforementioned significantly *optimized* and *sophisticated* implementations.

7. Conclusion

In this paper, we present CryptMed, a new secure and lightweight NN inference system towards secure intelligent medical diagnostic services. Our protocol fully resorts to the lightweight additive secret sharing techniques, free of heavy cryptographic operations as seen in prior art. The commonly-used non-linear comparison-based and smooth activation functions are well supported in a secure, efficient, and accurate manner. With CryptMed, the privacy of the medical record of the hospital and the NN model of the medical service is provably ensured with practical performance.

Acknowledgment

This work was supported in part by the Australian Research Council (ARC) Discovery Projects (No. DP200103308, No. DP180103251, and No. DP190102835), by the ARC Linkage Project (No. LP160101766), by the HITSZ Start-up Research Grant (No. BA45001023), by the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2021A1515110027, and by the Shenzhen Science and Technology Program under Grant No. RCBS20210609103056041.

References

- [1] Google DeepMind Health, *Online at <https://deepmind.com/blog/announcements/deepmind-health-joins-google-health>* (2020).
- [2] PathAI, *Online at <https://www.pathai.com/>* (2020).
- [3] Microsoft Project InnerEye, *Online at <https://www.microsoft.com/en-us/research/project/medical-image-analysis/>* (2020).
- [4] European Parliament and the Council, The General Data Protection Regulation (GDPR), 2016.
- [5] 104th United States Congress, Health Insurance Portability and Accountability Act of 1996 (HIPPA), 1996.
- [6] X. Liu and X. Yi, Privacy-preserving collaborative medical time series analysis based on dynamic time warping, in: *European Symposium on Research in Computer Security*, Springer, 2019, pp. 439–460.
- [7] X. Liu, Y. Zheng, X. Yi and S. Nepal, Privacy-Preserving Collaborative Analytics on Medical Time Series Data, *IEEE Transactions on Dependable and Secure Computing* (2020), 1–1. doi:10.1109/TDSC.2020.3035592.
- [8] M. Fredrikson, S. Jha and T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: *Proc. of ACM CCS*, 2015.
- [9] 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: *Proc. of ICML*, 2016.
- [10] J. Liu, M. Juuti, Y. Lu and N. Asokan, Oblivious neural network predictions via miniomn transformations, in: *Proc. of ACM CCS*, 2017.
- [11] C. Juvekar, V. Vaikuntanathan and A. Chandrakasan, GAZELLE: A low latency framework for secure neural network inference, in: *Proc. of 27th USENIX Security*, 2018.
- [12] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng and R.A. Popa, DELPHI: A cryptographic inference service for neural networks, in: *USENIX Security Symposium*, 2020.
- [13] S. Li, K. Xue, B. Zhu, C. Ding, X. Gao, D. Wei and T. Wan, FALCON: A fourier transform based approach for fast and secure convolutional neural network predictions, in: *Proc. of IEEE/CVF CVPR*, 2020.
- [14] A. Jacobi, M. Chung, A. Bernheim and C. Eber, Portable chest X-ray in coronavirus disease-19 (COVID-19): A pictorial review, *Clinical Imaging* (2020).
- [15] 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. of AsiaCCS*, 2018.
- [16] M. Keller, MP-SPDZ: A versatile framework for multi-party computation, in: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1575–1590.

- [17] A. Aly and N.P. Smart, Benchmarking privacy preserving scientific operations, in: *International Conference on Applied Cryptography and Network Security*, Springer, 2019, pp. 509–529.
- [18] P. Mohassel and Y. Zhang, SecureML: A system for scalable privacy-preserving machine learning, in: *Proc. of IEEE S&P*, 2017.
- [19] P. Mohassel and P. Rindal, ABY3: A mixed protocol framework for machine learning, in: *Proc. of ACM CCS*, 2018.
- [20] A. Patra, T. Schneider, A. Suresh and H. Yalame, ABY2. 0: Improved mixed-protocol secure two-party computation, in: *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.
- [21] A. Brutzkus, R. Gilad-Bachrach and O. Elisha, Low latency privacy preserving inference, in: *International Conference on Machine Learning*, PMLR, 2019, pp. 812–821.
- [22] Q. Zhang, C. Wang, H. Wu, C. Xin and T.V. Phuong, GELU-Net: A Globally Encrypted, Locally Unencrypted Deep Neural Network for Privacy-Preserved Learning., in: *Proc. of IJCAI*, 2018, pp. 3933–3939.
- [23] P. Xie, B. Wu and G. Sun, BAYHENN: Combining Bayesian Deep Learning and Homomorphic Encryption for Secure DNN Inference, in: *Proc. of IJCAI*, 2019, pp. 4831–4837.
- [24] X. Liu, B. Wu, X. Yuan and X. Yi, Leia: A Lightweight Cryptographic Neural Network Inference System at the Edge, *IEEE Transactions on Information Forensics and Security* **17** (2022), 237–252. doi:10.1109/TIFS.2021.3138611.
- [25] W. Chen, Y. Zhang, J. He, Y. Qiao, Y. Chen, H. Shi, E.X. Wu and X. Tang, Prostate segmentation using 2D bridged U-net, in: *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–7.
- [26] Q. Lou and L. Jiang, SHE: A Fast and Accurate Deep Neural Network for Encrypted Data, in: *Proc. of NeurIPS*, 2019, pp. 10035–10043.
- [27] M. Leshno, V.Y. Lin, A. Pinkus and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, *Neural networks* **6**(6) (1993), 861–867.
- [28] M.S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter and F. Koushanfar, XONN: XNOR-based Oblivious Deep Neural Network Inference, in: *Proc. of 28th USENIX Security*, 2019.
- [29] N. Agrawal, A. Shahin Shamsabadi, M.J. Kusner and A. Gascón, QUOTIENT: two-party secure neural network training and prediction, in: *Proc. of ACM CCS*, 2019.
- [30] Q. Lou, W.-j. Lu, C. Hong and L. Jiang, Falcon: Fast Spectral Inference on Encrypted Data, *Proc. of NeurIPS* **33** (2020).
- [31] A.-R. Sadeghi and T. Schneider, Generalized universal circuits for secure evaluation of private functions with application to data classification, in: *International Conference on Information Security and Cryptology*, Springer, 2008, pp. 336–353.
- [32] A. Sanyal, M. Kusner, A. Gascon and V. Kanade, TAPAS: Tricks to accelerate (encrypted) prediction as a service, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 4490–4499.
- [33] F. Bourse, M. Minelli, M. Minihold and P. Paillier, Fast homomorphic evaluation of deep discretized neural networks, in: *Annual International Cryptology Conference*, Springer, 2018, pp. 483–512.
- [34] A.P. Dalskov, D. Escudero and M. Keller, Secure Evaluation of Quantized Neural Networks., *Proc. Priv. Enhancing Technol.* **2020**(4) (2020), 355–375.
- [35] F. Boemer, R. Cammarota, D. Demmler, T. Schneider and H. Yalame, MP2ML: a mixed-protocol machine learning framework for private inference, in: *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–10.
- [36] S. Wagh, D. Gupta and N. Chandran, Securenn: 3-party secure computation for neural network training, *Proc. of PETS* (2019).
- [37] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi and R. Sharma, Cryptflow: Secure tensorflow inference, in: *2020 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2020, pp. 336–353.
- [38] R. Rachuri and A. Suresh, Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning, 2020.
- [39] M. Byali, H. Chaudhari, A. Patra and A. Suresh, FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning., *Proc. Priv. Enhancing Technol.* **2020**(2) (2020), 459–480.
- [40] M. Dahl, J. Mancuso, Y. Dupis, B. Decoste, M. Giraud, I. Livingstone, J. Patriquin and G. Uhma, Private machine learning in tensorflow using secure computation, *arXiv preprint arXiv:1810.08130* (2018).
- [41] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose et al., PySyft: A Library for Easy Federated Learning, in: *Federated Learning Systems*, Springer, 2021, pp. 111–139.
- [42] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert and J. Passerat-Palmbach, A generic framework for privacy preserving deep learning, *arXiv preprint arXiv:1811.04017* (2018).
- [43] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal and T. Rabin, Falcon: Honest-majority maliciously secure framework for private deep learning, *arXiv preprint arXiv:2004.02229* (2020).
- [44] W. Zheng, R. Popa, J.E. Gonzalez and I. Stoica, Helen: Maliciously Secure Cooperative Learning for Linear Models, in: *Proc. of IEEE S&P*, 2019.
- [45] Y. Zheng, C. Wang and J. Zhou, Toward secure image denoising: A machine learning based realization, in: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 6936–6940.

- [46] Y. Zheng, H. Duan, X. Tang, C. Wang and J. Zhou, Denoising in the dark: privacy-preserving deep neural network-based image denoising, *IEEE Transactions on Dependable and Secure Computing* **18**(3) (2019), 1261–1275.
- [47] X. Hu, W. Zhang, K. Li, H. Hu and N. Yu, Secure nonlocal denoising in outsourced images, *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* **12**(3) (2016), 1–23.
- [48] Y. Zheng, H. Cui, C. Wang and J. Zhou, Privacy-preserving image denoising from external cloud databases, *IEEE Transactions on Information Forensics and Security* **12**(6) (2017), 1285–1298.
- [49] X. Liu, Y. Zheng, X. Yuan and X. Yi, MediSC: Towards Secure and Lightweight Deep Learning as a Medical Diagnostic Service, in: *European Symposium on Research in Computer Security*, Springer, 2021, pp. 519–541.
- [50] J. Alvarez-Valle, P. Bhatu, N. Chandran, D. Gupta, A. Nori, A. Rastogi, M. Rathee, R. Sharma and S. Ugare, Secure medical image analysis with cryptflow, *arXiv preprint arXiv:2012.05064* (2020).
- [51] G.A. Kaissis, M.R. Makowski, D. Rückert and R.F. Braren, Secure, privacy-preserving and federated machine learning in medical imaging, *Nature Machine Intelligence* **2**(6) (2020), 305–311.
- [52] G. Kaissis, A. Ziller, J. Passerat-Palmbach, T. Ryffel, D. Usynin, A. Trask, I. Lima, J. Mancuso, F. Jungmann, M.-M. Steinborn et al., End-to-end privacy preserving deep learning on multi-institutional medical imaging, *Nature Machine Intelligence* **3**(6) (2021), 473–484.
- [53] X. Li, Y. Gu, N. Dvornek, L.H. Staib, P. Ventola and J.S. Duncan, Multi-site fMRI analysis using privacy-preserving federated learning and domain adaptation: ABIDE results, *Medical Image Analysis* **65** (2020), 101765.
- [54] M.Y. Lu, R.J. Chen, D. Kong, J. Lipkova, R. Singh, D.F. Williamson, T.Y. Chen and F. Mahmood, Federated learning for computational pathology on gigapixel whole slide images, *Medical image analysis* **76** (2022), 102298.
- [55] W. Li, F. Milletari, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M.J. Cardoso et al., Privacy-preserving federated brain tumour segmentation, in: *International workshop on machine learning in medical imaging*, Springer, 2019, pp. 133–141.
- [56] M.J. Sheller, G.A. Reina, B. Edwards, J. Martin and S. Bakas, Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation, in: *International MICCAI Brainlesion Workshop*, Springer, 2018, pp. 92–104.
- [57] R. Bost, R.A. Popa, S. Tu and S. Goldwasser, Machine learning classification over encrypted data, *Cryptology ePrint Archive* (2014).
- [58] M. De Cock, R. Dowsley, A.C. Nascimento, D. Railsback, J. Shen and A. Todoki, High performance logistic regression for privacy-preserving genome analysis, *BMC Medical Genomics* **14**(1) (2021), 1–18.
- [59] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh and N. Taft, Privacy-preserving ridge regression on hundreds of millions of records, in: *Proc. of IEEE S&P*, 2013.
- [60] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti and G. Tsudik, Countering gattaca: efficient and secure testing of fully-sequenced human genomes, in: *Proc. of ACM CCS*, 2011.
- [61] A. Salem, P. Berrang, M. Humbert and M. Backes, Privacy-Preserving Similar Patient Queries for Combined Biomedical Data, *Proc. of PETS* (2019).
- [62] X.S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang and D. Bu, Efficient genome-wide, privacy-preserving similar patient query based on private edit distance, in: *Proc. of ACM CCS*, 2015.
- [63] O. Tkachenko, C. Weinert, T. Schneider and K. Hamacher, Large-scale privacy-preserving statistical computations for distributed genome-wide association studies, in: *Proc. of ACM AsiaCCS*, 2018.
- [64] H. Cho, D.J. Wu and B. Berger, Secure Genome-Wide Association Analysis using Multiparty Computation, *Nature Biotechnology* **36**(6) (2018), 547–551.
- [65] S. Jha, L. Kruger and V. Shmatikov, Towards practical privacy for genomic computation, in: *2008 IEEE Symposium on Security and Privacy (sp 2008)*, IEEE, 2008, pp. 216–230.
- [66] K.A. Jagadeesh, D.J. Wu, J.A. Birgeimer, D. Boneh and G. Bejerano, Deriving genomic diagnoses without revealing patient genomes, *Science* **357**(6352) (2017), 692–695.
- [67] Y. Huang, L. Malka, D. Evans and J. Katz, Efficient privacy-preserving biometric identification, in: *Proc. of NDSS*, 2011.
- [68] E. Check Hayden, Extreme cryptography paves way to personalized medicine, *Nature News* **519**(7544) (2015), 400.
- [69] B. Hie, H. Cho and B. Berger, Realizing private and practical pharmacological collaboration, *Science* **362**(6412) (2018), 347–350.
- [70] M. Barni, P. Failla, R. Lazeretti, A.-R. Sadeghi and T. Schneider, Privacy-preserving ECG classification with branching programs and neural networks, *IEEE Trans. on Information Forensics and Security* (2011).
- [71] M. Atallah, M. Bykova, J. Li, K. Frikken and M. Topkara, Private collaborative forecasting and benchmarking, in: *Proc. of WPES*, 2004.
- [72] D. Beaver, Efficient multiparty protocols using circuit randomization, in: *Proc. of Crypto*, 1991.
- [73] Y. Zheng, H. Duan and C. Wang, Towards Secure and Efficient Outsourcing of Machine Learning Classification, in: *Proc. of ESORICS*, Springer, 2019.
- [74] A. Wuraola and N. Patel, SQLN: A new computationally efficient activation function, in: *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018, pp. 1–7.

- [75] B. DasGupta and G. Schnitger, The Power of Approximation: A Comparison of Activation Functions., in: *NIPS*, Denver, CO, 1992, pp. 615–622.
- [76] D. Clevert, T. Unterthiner and S. Hochreiter, Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), in: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
- [77] L. Yu, L. Liu, C. Pu, M.E. Gursoy and S. Truex, Differentially private model publishing for deep learning, in: *Proc. of S&P*, IEEE, 2019.
- [78] D. Harris, A taxonomy of parallel prefix networks, in: *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2, IEEE, 2003, pp. 2213–2217.
- [79] D. Rathee, M. Rathee, R.K.K. Goli, D. Gupta, R. Sharma, N. Chandran and A. Rastogi, SiRnn: A Math Library for Secure RNN Inference, in: *IEEE Symposium on Security and Privacy*, IEEE, 2021, pp. 1003–1020.
- [80] B.D. Rouhani, M.S. Riazi and F. Koushanfar, Deepsecure: Scalable provably-secure deep learning, in: *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.
- [81] M. Tommiska, Efficient digital implementation of the sigmoid function for reprogrammable logic, *IEE Proceedings-Computers and Digital Techniques* **150**(6) (2003), 403–411.
- [82] I. Tsmots, O. Skorokhoda and V. Rabyk, Hardware implementation of sigmoid activation functions using FPGA, in: *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, IEEE, 2019, pp. 34–38.
- [83] C.-W. Lin and J.-S. Wang, A digital circuit design of hyperbolic tangent sigmoid function for neural networks, in: *2008 IEEE International Symposium on Circuits and Systems*, IEEE, 2008, pp. 856–859.
- [84] A. Wuraola, N. Patel and S.K. Nguang, Efficient activation functions for embedded inference engines, *Neurocomputing* **442** (2021), 73–88.
- [85] K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural networks* **2**(5) (1989), 359–366.
- [86] K. Hornik, Approximation capabilities of multilayer feedforward networks, *Neural networks* **4**(2) (1991), 251–257.
- [87] D. Demmler, T. Schneider and M. Zohner, ABY-A Framework for Efficient Mixed-Protocol Secure Two-Party Computation., in: *Proc. of NDSS*, 2015.
- [88] O. Goldreich, S. Micali and A. Wigderson, How to play any mental game or a completeness theorem for protocols with honest majority, in: *Proc. of STOC*, 1987.
- [89] X. Wang, FlexSC, Online at <https://github.com/wangxiao1254/FlexSC> (2018).

Appendix A. More Details of Model Architecture

In this section, we present the detailed model architectures used in our paper.

Table 12
Model architecture of MNIST.

Layers	Padding	Stride
FC (input: 784, output: 128)+ ReLU	-	-
FC (input: 128, output: 128)+ ReLU	-	-
FC (input: 128, output: 10)	-	-

Table 13
Model architecture of CIFAR-10.

Layers	Padding	Stride
CONV (input: $3 \times 32 \times 32$, kernel: $3 \times 64 \times 3 \times 3$ feature: $64 \times 32 \times 32$) + ReLU	0	1
CONV (input: $64 \times 32 \times 32$, kernel: $64 \times 64 \times 3 \times 3$ feature: $64 \times 32 \times 32$) + ReLU	0	1
AP (input: $64 \times 32 \times 32$, window: $64 \times 2 \times 2$ output: $64 \times 16 \times 16$)	-	2
CONV (input: $64 \times 16 \times 16$, kernel: $64 \times 64 \times 3 \times 3$ feature: $64 \times 16 \times 16$) + ReLU	0	1
CONV (input: $64 \times 16 \times 16$, kernel: $64 \times 64 \times 3 \times 3$ feature: $64 \times 16 \times 16$) + ReLU	0	1
AP (input: $64 \times 16 \times 16$, window: $64 \times 2 \times 2$ output: $64 \times 8 \times 8$)	-	2
CONV (input: $64 \times 8 \times 8$, kernel: $64 \times 64 \times 3 \times 3$ feature: $64 \times 8 \times 8$) + ReLU	0	1
CONV (input: $64 \times 8 \times 8$, kernel: $64 \times 64 \times 3 \times 3$ feature: $64 \times 8 \times 8$) + ReLU	0	1
CONV (input: $64 \times 8 \times 8$, kernel: $16 \times 64 \times 3 \times 3$ feature: $16 \times 8 \times 8$) + ReLU	0	1
FC (input: 1024, output: 10)	-	-

Table 14
Model architecture of Breast Cancer.

Layers	Padding	Stride
FC (input: 30, output: 16) + BN + ReLU	-	-
FC (input: 16, output: 16) + BN + ReLU	-	-
FC (input: 16, output: 2) + BN	-	-

Table 15
Model architecture of Diabetes.

Layers	Padding	Stride
FC (input: 8, output: 20) + ReLU	-	-
FC (input: 20, output: 20) + ReLU	-	-
FC (input: 20, output: 2)	-	-

Table 16
Model architecture of Liver Disease.

Layers	Padding	Stride
FC (input: 10, output: 32) + ReLU	-	-
FC (input: 32, output: 32) + ReLU	-	-
FC (input: 32, output: 2)	-	-

Table 17
Model architecture of Thyroid.

Layers	Padding	Stride
FC (input: 21, output: 100) + BN + ReLU	-	-
FC (input: 100, output: 100) + BN + ReLU	-	-
FC (input: 100, output: 3) + BN	-	-