MediSC: Towards Secure and Lightweight Deep Learning as a Medical Diagnostic Service

Xiaoning Liu[1], Yifeng Zheng[2,⋆], Xingliang Yuan[3], and Xun Yi[1]

[1] RMIT University, Melbourne Australia {maggie.liu, xun.yi}@rmit.edu.au
[2] Harbin Institute of Technology, Shenzhen, China yifeng.zheng@hit.edu.cn
[3] Monash University, Clayton, Australia xingliang.yuan@monash.edu

Abstract. The striking progress of deep learning paves the way towards intelligent and quality medical diagnostic services. Enterprises deploy such services via the neural network (NN) inference, yet confronted with rising privacy concerns of the medical data being diagnosed and the pretrained NN models. We propose MediSC, a system framework that enables enterprises to offer secure medical diagnostic service to their customers via an execution of NN inference in the ciphertext domain. MediSC ensures the privacy of both parties with cryptographic guarantees. At the heart, we present an efficient and communication-optimized secure inference protocol that purely relies on the lightweight secret sharing techniques and can well cope with the commonly-used linear and non-linear NN layers. Compared to the garbled circuits based solutions, the latency and communication of MediSC are 24× lower and 868× less for the secure ReLU, and 20× lower and 314× less for the secure Max-pool. We evaluate MediSC on two benchmark and four real-world medical datasets, and comprehensively compare it with prior arts. The results demonstrate the promising performance of MediSC, which is much more bandwidth-efficient compared to prior works.

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 the radiotherapy [5], clinical trial and research [7], to medical imaging diagnostics [6]. 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 [13, 8, 24, 25]. Meanwhile, NN models used in these services are seen as lucrative intellectual properties and encode knowledge of private training data [14].

⋆ Corresponding author.

The general setup of the above NN-powered service scenario fits within the field of secure multi-party computation (MPC). By designing specialized MPC protocols, recent works [15, 22, 19, 28, 21] enable the joint execution of secure NN inference systems over encrypted customer's data and/or the service provider's model. Nevertheless, these systems still come at a steep performance overhead that may not be amiable for the real-world medical scenario. During inference, they all require customers to conduct heavy cryptographic computations like homomorphic encryption (HE) and garbled circuits (GC), imposing intensive computational and communication overheads. These performance hurdles are further exacerbated when, e.g., the service is deployed to a hospital with resource-constrained devices (like portable medical imaging scanners [18]). Furthermore, some of their protocols [15, 19] are not directly compatible with the widely-adopted non-linear functions (like ReLU), causing limitations of applicability for modern NN architectures.

We design, implement, and evaluate `MediSC`, a lightweight and secure NN inference system tailored for medical diagnostic services. `MediSC` proceeds by having the hospital and the medical service engage in a tailored secure inference protocol over their encrypted 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 and digital circuit design, making an efficient and low-interaction service suitable for realistic medical scenarios. Our contributions are summarized as follows.

- We propose a secure NN inference system framework `MediSC` relying only on the lightweight secret sharing techniques, which requires neither heavy cryptographic computation nor large-size ciphertext transmissions.
- 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 to support the most challenging and widely adopted non-linear functions (ReLU and Max-pool), harnessing the insights from cryptography and the field of digital circuit design. Compared to the commonly-used GC solutions, `MediSC`'s secure ReLU is $24\times$ faster and requires $868\times$ less communication, and the secure Max-pool is $20\times$ faster and uses $314\times$ less communication.
- We conduct formal security analysis. We implement a prototype of `MediSC` and conduct comprehensive evaluations over two benchmarking datasets and four realistic medical datasets. Our experiment results show that `MediSC` requires the least network resources compared to prior works with up to $413\times$, $19\times$ and $10\times$ bandwidth savings for MNIST, CIFAR-10, and the medical applications, respectively. `MediSC` outperforms the state-of-the-art (SOTA) [28] by $10\times$ in bandwidth cost, with comparable latency[4].

[4] From a direct comparison with results reported in SOTA which demands highly optimized implementations with GPU acceleration. Our performance result is not based on such optimization.

2 Related Works

The past few years have seen an increased interest in secure neural network inference. A plethora of prior works [15, 22, 19, 21, 12, 28, 36, 34] focus on a scenario where an *interactive* protocol is run between the service provider and the customer. Some other works [29, 31, 23] rely on a security assumption where two *non-colluding cloud servers* are employed to jointly conduct secure inference over outsourced model and data. Despite their different system models, these works require to use heavy cryptographic tools (like HE and GC) during the latency-sensitive online inference procedure. Moreover, some of these works do not fully support the modern NN models [15, 19]. Instead, these works approximate the non-linear functions into the crypto-friendly polynomials, trading the accuracy and applicability for efficiency [26, 20], which could cause critical consequences in the medical scenario.

The SOTA work [28] presents a hybrid and interactive inference protocol, preprocessing 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 ReLU. `MediSC` adopts a similar hybrid setting yet only involves the lightweight secret sharing techniques during the entire secure inference procedure, which has an 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.

3 Preliminaries on Additive Secret Sharing

Additive secret sharing [9] protects an ℓ-bit value $x \in \mathbb{R}_{2^\ell}$ 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 $\mathbb{Z}_{2^\ell}$ is a ring and r is a random value from the ring ($r \in_R \mathbb{Z}_{2^\ell}$). 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 the corresponding shares of two secret values x and y. Additive secret sharing supports efficient local addition/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 ($i \in \{0, 1\}$) without interactions. Multiplication over two shares $\langle z\rangle = \langle x\rangle \cdot \langle y\rangle$ is enabled with the secret-shared Beaver's triple [11], 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 with Beaver's triple is a standard secure 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 [37, 31]. Additive secret shares can 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 $[\![x]\!]_0 = r \in \mathbb{Z}_2$ and $[\![x]\!]_1 = r \oplus [\![x]\!]_0$. The bitwise XOR ($\oplus$) and AND ($\wedge$) over shares are calculated in the same way as the above addition and multiplication, respectively.

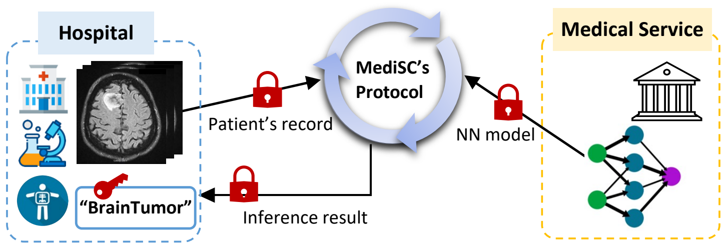

Fig. 1. System architecture.

4 System Overview

4.1 Architecture

`MediSC` targets a typical scenario of secure NN inference based medical diagnostic service. As shown in Fig. 1, `MediSC` operates between two parties: the *hospital* (customer) and the *medical service* provider. The *medical service* holds a proprietary NN model that is pre-trained on medical datasets. The *hospital* holds confidential medical records (e.g., brain MRI images) and intends to leverage the deep learning service to facilitate a medical conclusion. In practice, the role of hospital in `MediSC` can actually be any healthcare institutes, medical research laboratories, or life-science organizations. To initiate a secure medical diagnostic service, the two parties execute `MediSC`'s secure NN inference protocol over their encrypted model and encrypted medical record. At the end, an encrypted inference result is returned to the hospital which can then decrypt to get the plaintext inference result. `MediSC` ensures that the hospital learns the inference result and nothing else, while the medical service learns no information about the hospital's medical records.

4.2 Threat Model

`MediSC` is designed for the *semi-honest* two-party model: the hospital and the medical service will faithfully follow the protocol, yet attempting to deduce information about the counterparty's private input from the messages seen from protocol execution. It is noted that the behavior of hospital is enforced by the ethics, law and privacy regulations [8, 13]. The medical service is usually offered by well-established companies (e.g., Microsoft Project InnerEye [6], Google DeepMind Health [5]) and would not take their business model and reputation at risk to act maliciously [10]. Such an adversarial model is also adopted in prior secure NN inference works [28, 22]. `MediSC` strives to ensure the privacy of the hospital's medical records and the NN model (values of trained weights). Like prior work [28, 22, 23], `MediSC` does not hide the data-independent model architecture, such as kernel size and number of layers. Lastly, `MediSC` 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 the differentially private learning [35].

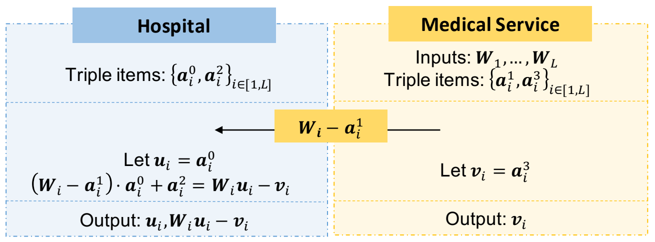
Hospital
Medical Service
Triple items: $\{\boldsymbol{a}_i^0, \boldsymbol{a}_i^2\}_{i\in[1,L]}$
Inputs: $\boldsymbol{W}_1, \dots, \boldsymbol{W}_L$
Triple items: $\{\boldsymbol{a}_i^1, \boldsymbol{a}_i^3\}_{i\in[1,L]}$
$\boldsymbol{W}_i - \boldsymbol{a}_i^1$
Let $\boldsymbol{u}_i = \boldsymbol{a}_i^0$
$(\boldsymbol{W}_i - \boldsymbol{a}_i^1) \cdot \boldsymbol{a}_i^0 + \boldsymbol{a}_i^2 = \boldsymbol{W}_i \boldsymbol{u}_i - \boldsymbol{v}_i$
Let $\boldsymbol{v}_i = \boldsymbol{a}_i^3$
Output: $\boldsymbol{u}_i, \boldsymbol{W}_i \boldsymbol{u}_i - \boldsymbol{v}_i$
Output: $\boldsymbol{v}_i$

Fig. 2. MediSC's preprocessing phase.

5 Our Proposed Design

In this section, we introduce MediSC'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* and *non-linear* layers, which can well support the typical NN layers, i.e., the convolutional/fully-connected/batch normalization/average pooling layers (linear), and the ReLU activation/max pooling layers (non-linear). Each secure layer evaluation conducts a certain function on the encrypted inputs (features) and yields encrypted outputs passed to the next secure layer. Our overarching goal is to devise a lightweight protocol for secure neural network inference, while minimizing interactions in between the hospital and the medical service for a low-latency diagnosis. Atop such goal, we have two prominent design insights.

Eliminating heavy cryptography for linear layers. We first split MediSC's protocol into a preprocessing phase and an online phase, and shift as much computation as possible to preprocessing phase. Inspired by [28], 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 [28] 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 [18]). Instead, our protocol delicately leverages the insight from Chameleon [31] 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 [28].

Eliminating the usages of GC for non-linear layers. For secure evaluation of non-linear layers, prior works either resort to the heavy cryptographic techniques (i.e., garbled circuits) [22, 28], or circumvent the non-linearities with their polynomial approximations [19, 15]. Unfortunately, such methods may introduce high communication overheads or introduce instabilities of NN when handling complex tasks [26, 20]. In MediSC, we make observations from the field of digital circuit design [17] and present a secure comparison function that can efficiently evaluate comparison-based non-linear layers like ReLU. At the core, this function is fully based on lightweight secret sharing with optimized interactions between

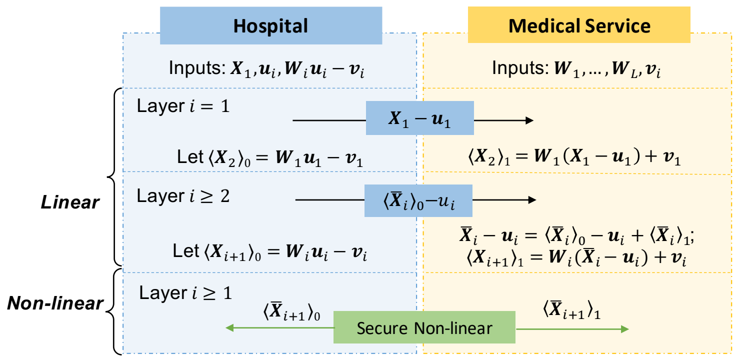
Hospital
Medical Service
Inputs: $X_1, u_i, W_i u_i - v_i$
Inputs: $W_1, \ldots, W_L, v_i$
Linear
Layer $i = 1$
$X_1 - u_1$
Let $\langle X_2 \rangle_0 = W_1 u_1 - v_1$
$\langle X_2 \rangle_1 = W_1(X_1 - u_1) + v_1$
Layer $i \geq 2$
$\langle \bar{X}_i \rangle_0 - u_i$
Let $\langle X_{i+1} \rangle_0 = W_i u_i - v_i$
$\bar{X}_i - u_i = \langle \bar{X}_i \rangle_0 - u_i + \langle \bar{X}_i \rangle_1$;
$\langle X_{i+1} \rangle_1 = W_i(\bar{X}_i - u_i) + v_i$
Non-linear
Layer $i \geq 1$
$\langle \bar{X}_{i+1} \rangle_0$
Secure Non-linear
$\langle \bar{X}_{i+1} \rangle_1$

Fig. 3. `MediSC`'s online phase.

the hospital and the medical service. With these designs, our experiment demonstrates a 413× bandwidth reduction compared with prior works.

5.1 Secure Linear Layers

The subsequent section presents `MediSC`'s secure inference protocol, which comprises two phases: the *preprocessing* phase and the *online inference* phase. Before diving in to the details, we first note that `MediSC` works over the secret sharing domain. That is, any float-point number v (i.e., model weights and medical record values) is projected to a signed fixed-point integer $\bar{v} = \lfloor v \cdot 2^s \rfloor \bmod 2^\ell$ in ring $\mathbb{Z}_{2^\ell}$, where 2^s is the scaling factor. The most significant bit (MSB) indicates the sign ($1 \rightarrow$negative; $0 \rightarrow$non-negative)[5]. Through such a conversion, both the value and sign information are perfectly hidden.

Preprocessing phase. The preprocessing phase is illustrated in Fig. 2. The hospital and the medical service pre-generate 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 $\mathbf{W}_1, \ldots, \mathbf{W}_L$ and randomnesses tensors $\{\mathbf{a}_i^1, \mathbf{a}_i^3\}$. Such randomness tensors $\{\mathbf{a}_i^0, \mathbf{a}_i^1, \mathbf{a}_i^2, \mathbf{a}_i^3\}$ are independent to any party's input and can be pre-distributed to the parties, and satisfy the relationship: $\mathbf{a}_i^3 = \mathbf{a}_i^0 \cdot \mathbf{a}_i^1 - \mathbf{a}_i^2$. 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 $\mathbf{W}_i - \mathbf{a}_i^1$ over the weight tensors and sends to the hospital.
2. The hospital computes $(\mathbf{W}_i - \mathbf{a}_i^1) \cdot \mathbf{a}_i^0 = \mathbf{W}_i \mathbf{a}_i^0 - \mathbf{a}_i^0 \mathbf{a}_i^1 + \mathbf{a}_i^2$ for each layer.
3. Let $\mathbf{u}_i$ denote $\mathbf{a}_i^0$, and $\mathbf{v}_i$ denote $\mathbf{a}_i^3$. 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 tensors $\mathbf{W}_i \mathbf{u}_i$.

[5] We refer the readers to more details in Appendix Sec. A.

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 $\mathbf{W}_i\mathbf{u}_i - \mathbf{v}_i$, as shown in Fig. 3. The medical service takes as input the weight tensors $\mathbf{W}_1, ..., \mathbf{W}_L$ 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 features[6] 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 form secure linear layer evaluation can be fed into the secure non-linear layer (e.g., ReLU), 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 the final inference result $\mathbf{X}_L$.

5.2 Secure Non-linear Layers

`MediSC` supports highly efficient evaluation of the secure non-linear layers in the secret sharing domain. As mentioned above, `MediSC` denotes all values as the signed fixed-point integers with MSB indicating the sign, i.e., the MSB would be '0' for a non-negative value and '1' for a negative value. With such a representation, we observe that all non-linear layers mainly relying on the comparison operation can be simplified to an MSB extraction problem along with some linear operations (addition and multiplication). For the ease of presentation, we focus on the most-widely adopted ReLU function. The ReLU function can be converted to a simplified MSB extraction problem via

$$max(x, 0) \rightarrow \neg \mathsf{MSB}(x) \cdot x = \begin{cases} 1 \cdot x & \text{if } x \geq 0 \\ 0 \cdot x & \text{if } x < 0 \end{cases}, \quad (1)$$

[6] Biases can be added to the medical service's shares locally.

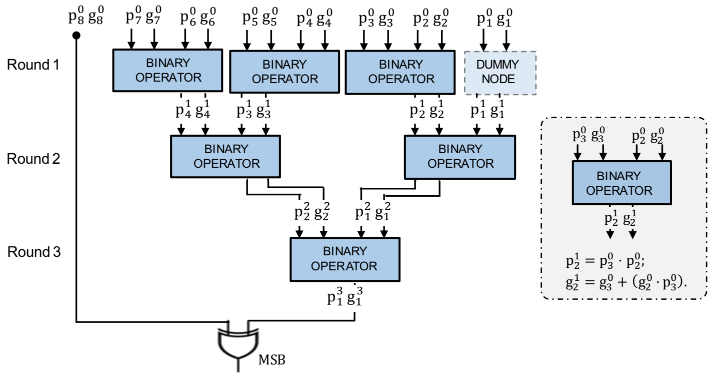

Fig. 4. An illustration of 8-bit parallel prefix adder (PPA).

where x is the feature on each neuron outputted from previous linear layer. Through such a conversion, we observe that it consists of four atomic steps: the *secure* $\mathsf{MSB}(\cdot)$ *extraction*, the *secure NOT*, the *secure* $\mathsf{B2A}$ (i.e., Boolean-to-Additive shares conversion), and the *secure multiplication*. The most challenging computation is the secure $\mathsf{MSB}(\cdot)$ extraction, of which we propose an efficient and communication-optimized construction and present the details in the subsequent section. For the rest steps, the secure $\mathsf{B2A}$ converts boolean shares $[\![x]\!]$ in ring $\mathbb{Z}_2$ to additive shares in $\mathbb{Z}_{2^\ell}$, i.e., $\langle x\rangle \leftarrow \mathsf{B2A}([\![x]\!])$. Meanwhile, the *secure NOT* and the *secure multiplication* are linear operations and well handled by additive secret sharing.

Communication-optimized secure MSB extraction. The secure $\mathsf{MSB}(\cdot)$ 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 idea is that extracting the MSB in the secret sharing domain can be performed via binary addition over two secret shares' bit strings by an ℓ-bit full adder, as expatiated below. Suppose an ℓ-bit value x with its decomposed bit string $x = \{x_\ell, ..., x_1\}$ and the secret shares $\langle x\rangle_0, \langle x\rangle_1$. Let $e = \{e_\ell, ..., e_1\}$ and $f = \{f_\ell, ..., f_1\}$ denote the bit strings of $\langle x\rangle_0$ and $\langle x\rangle_1$, respectively. In this way, $x = e + f \pmod{2^\ell}$. Then, an ℓ-bit full adder is used to perform the binary addition $(\{e_k\} + \{f_k\})$ in the secret sharing domain to produce the carry bits $c_\ell, \dots, c_1$, and finally the MSB is calculated via $x_\ell = c_\ell \oplus e_\ell \oplus f_\ell$, where $k \in [1, \ell]$. The key takeaway to extract the MSB is producing the most significant carry bit c_ℓ via the full adder logic. In the following section, without specifically mentioned, the operator '+' over two binary values (including boolean shares) denotes the bitwise-XOR operation for the ease of demonstration.

We make an observation from the field of digital circuit design that the parallel prefix adder [17] (PPA) offers an efficient realization of the full adder logic in logarithm round complexity $O(\log \ell)$. To construct PPA, we introduce a *signal tuple* (g_i, p_i): the carry generate signal g_i and the carry propagate signal p_i, which can be derived in parallel via

$$g_i = e_i \cdot f_i;\ p_i = e_i + f_i. \tag{2}$$

Algorithm 1 Secure $\mathsf{MSB}(\cdot)$ Extraction Function.

```
Input: Arithmetic shared integer feature $\langle x \rangle \in \mathbb{Z}_{2^\ell}$.
Output: Boolean shared MSB $[\![x_\ell]\!] \in \mathbb{Z}_2$.
Decompose $\langle x \rangle$ into bit strings:
 1: Let $e$ denote $\langle x \rangle_0$ and $f$ denote $\langle x \rangle_1$.
 2: Decompose to bit strings $e \to e_\ell, ..., e_1$ and $f \to f_\ell, ..., f_1$.
 3: for $k \in [1, \ell]$ do
 4:     Set $[\![e_k]\!]_0 = e_k$, $[\![e_k]\!]_1 = 0$ and $[\![f_k]\!]_0 = 0$, $[\![f_k]\!]_1 = f_k$.
Compute signal tuples $(g, p)$ in Eq. 2:
 5:     $[\![g_k^0]\!] = [\![e_k]\!] \cdot [\![f_k]\!]$, $[\![p_k^0]\!] = [\![e_k]\!] + [\![f_k]\!]$.
 6: end for
Compute PPA tree based on Eq. 3:
Round $\mathcal{R} = 1$:
 7: for $k \in [2, \ell/2]$ do
 8:     Set $([\![g_1^1]\!], [\![p_1^1]\!]) = ([\![g_1^0]\!], [\![p_1^0]\!])$ as a dummy node.
 9:     Let $in_1 = 2k - 2$, $in_2 = 2k - 1$.
10:     $([\![g_k^1]\!], [\![p_k^1]\!]) = ([\![g_{in_1}^0]\!], [\![p_{in_1}^0]\!]) \odot ([\![g_{in_2}^0]\!], [\![p_{in_2}^0]\!])$.
11: end for
Round $\mathcal{R} = 2, ..., \log \ell$:
12: for $k \in [1, \ell/2^{\mathcal{R}}]$ do
13:     Let $in_1 = 2k - 1$, $in_2 = 2k$.
14:     $([\![g_k^{\mathcal{R}}]\!], [\![p_k^{\mathcal{R}}]\!]) = ([\![g_{in_1}^{\mathcal{R}-1}]\!], [\![p_{in_1}^{\mathcal{R}-1}]\!]) \odot ([\![g_{in_2}^{\mathcal{R}-1}]\!], [\![p_{in_2}^{\mathcal{R}-1}]\!])$.
15: end for
Compute MSB:
16: Set $[\![c_\ell]\!] = [\![g_1^{\log \ell}]\!]$, $[\![x_\ell]\!] = [\![p_\ell^0]\!] + [\![c_\ell]\!]$.
```

Then the full adder logic $c_{i+1} = (e_i \cdot f_i) + c_i \cdot (e_i + f_i)$ is reformulated as $c_{i+1} = g_i + c_i \cdot p_i$, and the ℓ-th carry bit can be generated via $c_\ell = g_{\ell-1} + (p_{\ell-1} \cdot g_{\ell-2}) + ... + (p_{\ell-1} ... p_2 \cdot g_1)$. Through this reformulation, PPA can extract the MSB in $O(\log \ell)$ communication round latency.

A concrete illustration of 8-bit PPA is given in Fig. 4. As shown, it constructs a $\log \ell$-depth (3-depth) binary tree with a binary operator $\odot$ adhering to each node. Each layer of the tree indicates one round of communication. The binary operator $\odot$ takes as inputs the two adjacent signal tuples $(g_{in_1}, p_{in_1}), (g_{in_2}, p_{in_2})$, performs the following computations:

$$\begin{aligned} &(g_{out}, p_{out}) = (g_{in_1}, p_{in_1}) \odot (g_{in_2}, p_{in_2}); \\ &g_{out} = g_{in_2} + g_{in_1} \cdot p_{in_2}; \;\; p_{out} = p_{in_2} \cdot p_{in_1}, \end{aligned} \tag{3}$$

and outputs a signal tuple (g_{out}, p_{out}). PPA iteratively performs the above binary operation over the input tuples associated with each leaf node, and propagates the outputted signal tuples to the next layer's nodes as inputs. Such computations are terminated until the root node is reached, i.e., the node with (p_1^3, g_1^3) in Fig. 4. To this end, the carry bit c_ℓ is obtained and the MSB is calculated via $x_\ell = c_\ell + p_\ell = c_\ell + (e_\ell + f_\ell)$. In light of above philosophy, we present details of the secure MSB extraction function in Algorithm 1.

Secure B2A function. Given a secret value x, the secure $\mathsf{B2A}$ function is used to convert its boolean shares $[\![x]\!]$ in ring $\mathbb{Z}_2$ to the corresponding additive secret shares $\langle x \rangle$ in ring $\mathbb{Z}_{2^\ell}$. Recall that our proposed secure ReLU function over each feature x is formulated as follows: $max(x, 0) \rightarrow \neg\mathsf{MSB}(x) \cdot x$. The secure $\mathsf{B2A}$ function is invoked after securely extracting the boolean shares of NOT MSB $[\![\neg x_\ell]\!]$. However, the produced boolean shares cannot be directly multiplied with the additively-shared feature $\langle x \rangle$ as they are shared with different moduli, i.e., $[\![\neg x_\ell]\!] = [\![\neg x_\ell]\!]_0 + [\![\neg x_\ell]\!]_1 \pmod 2$ and $\langle x \rangle = \langle x \rangle_0 + \langle x \rangle_1 \pmod{2^\ell}$. So we need to convert $[\![\neg x_\ell]\!]$ to its additive form $\langle \neg x_\ell \rangle$.

Our secure $\mathsf{B2A}$ function follows the standard realization [37]. Given two parties the hospital (denoted as P_0) and the medical service (denoted as P_1), the secure $\mathsf{B2A}([\![x]\!])$ function is performed as follow:

1. P_0 sets $\langle e \rangle_0 = [\![x]\!]_0$, $\langle f \rangle_0 = 0$, and P_1 sets $\langle e \rangle_1 = 0$, $\langle f \rangle_1 = [\![x]\!]_1$;
2. P_0 and P_1 compute $\langle x \rangle_i = \langle e \rangle_i + \langle f \rangle_i - 2 \cdot \langle e \rangle \cdot \langle f \rangle$.

Secure ReLU function. For the ease of presentation, we show the secure ReLU function attached on each neuron over single feature element x. Given the above secure MSB extraction function and the shares of a single input feature $\langle x \rangle$, the hospital (denoted as P_0) and the medical service (denoted as P_1) perform the secure ReLU function as follows:

1. <u>Secure MSB extraction:</u> P_0 and P_1 invoke Algorithm 1 to get $[\![x_\ell]\!] \leftarrow \mathsf{MSB}(\langle x \rangle)$.
2. <u>Secure NOT:</u> P_i computes NOT MSB $[\![\neg x_\ell]\!] = [\![x_\ell]\!] + i$.
3. <u>Secure B2A:</u> P_0 and P_1 run $\langle \neg x_\ell \rangle \leftarrow \mathsf{B2A}([\![\neg x_\ell]\!])$ to convert the boolean-shared NOT MSB into additive shares.
4. <u>Secure multiplication:</u> P_0 and P_1 compute to produce the activation on each neuron $\langle \bar{x} \rangle = \langle \neg x_\ell \rangle \cdot \langle x \rangle$.

Secure pooling layer. Within an n-width pooling window, the max pooling layer $\max(x_1, \cdots, x_n)$ can be transformed to the pairwise maximum operation and realized based on the secure $\mathsf{MSB}(\cdot)$ extraction via $\mathfrak{b} \leftarrow \mathsf{MSB}(x_1 - x_2)$ and $\max(x_1, x_2) = (1 - \mathfrak{b}) \cdot x_1 + \mathfrak{b} \cdot x_2$. Given the above secure MSB extraction function, the secure $\mathsf{B2A}$ function, and the shares of a set of activations $\langle x_1 \rangle, ..., \langle x_n \rangle$ within the n-width pooling window, the hospital (denoted as P_0) and the medical service (denoted as P_1) perform the secure MaxPool function as follows:

1. For $k \in [1, n-1]$:
2. <u>Secure MSB extraction:</u> P_0 and P_1 invoke Algorithm 1 to get the boolean shares of MSB $[\![\mathfrak{b}]\!] \leftarrow \mathsf{MSB}(\langle x_k \rangle - \langle x_{k+1} \rangle)$.
3. <u>Secure B2A:</u> P_0 and P_1 run $\langle \mathfrak{b} \rangle \leftarrow \mathsf{B2A}([\![\mathfrak{b}]\!])$ to convert the boolean-shared MSB into additive shares.
4. <u>Secure branching:</u> P_0 and P_1 use the MSB to choose the maximum value as follows: $\langle \mathfrak{b}' \rangle_i = i - \langle \mathfrak{b} \rangle_i$, where $i \in \{0, 1\}$ is the identifier of party P_i, and then compute $\langle z_k \rangle = \langle \mathfrak{b}' \rangle \cdot \langle x_k \rangle + \langle \mathfrak{b} \rangle \cdot \langle x_{k+1} \rangle$. P_i sets $\langle x_{k+1} \rangle := \langle z_k \rangle$.
5. Finally, P_i outputs $\langle z_n \rangle_i$ as the shares of MaxPool result.

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

5.3 Security Analysis

`MediSC` properly encrypts the medical data, NN model (i.e., the weights), and any intermediate values as secret shares uniformly distributed in ring $\mathbb{Z}_{2^\ell}$ via standard secret sharing techniques [9, 16]. It ensures that throughout the service procedure, the hospital only learns the inference result and nothing else, while the medical service learns nothing. Formally, we present the ideal functionality and the formal security definition, and then prove the security of `MediSC`'s secure NN inference protocol under the ideal/real world paradigm. We first define the ideal functionality $\mathcal{F}^{\mathsf{SNNI}}$ capturing our targeted security properties.

Definition 1. *The ideal functionality $\mathcal{F}^{\mathsf{SNNI}}$ of secure neural network inference comprises the following parts:*

- ***Input.*** *The medical service submits the neural network model $\mathcal{W}$ to $\mathcal{F}^{\mathsf{SNNI}}$. The hospital submits the medical record $\mathbf{X}$ to $\mathcal{F}^{\mathsf{SNNI}}$.*
- ***Computation.*** *Upon receiving the neural network model $\mathcal{W}$ from the medical service and the medical record $\mathbf{X}$ from the hospital, $\mathcal{F}^{\mathsf{SNNI}}$ performs neural network inference and generates the prediction $\mathcal{W}(\mathbf{X})$.*
- ***Output.*** *The $\mathcal{F}^{\mathsf{SNNI}}$ returns the prediction $\mathcal{W}(\mathbf{X})$, and outputs nothing to the medical service.*

Given above ideal functionality, we formally provide the security definition.

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

- ***Corrupted hospital.*** *A corrupted and semi-honest hospital should learn nothing about the service's model weights and coefficients beyond the generic architecture hyper-parameter. Formally, there should exit a PPT simulator $\mathsf{Sim}_{\mathsf{H}}$ that $\mathsf{View}_{\mathsf{H}}^{\Pi} \overset{c}{\approx} \mathsf{Sim}_{\mathsf{H}}(\mathbf{X}, \mathcal{W}(\mathbf{X}))$, where H is the hospital and $\mathsf{View}_{\mathsf{H}}^{\Pi}$ indicates the view of the semi-honest hospital in real-world protocol execution.*
- ***Corrupted medical service.*** *A corrupted and semi-honest medical service should learn nothing about the values of the medical record $\mathbf{X}$ inputted by the hospital. Formally, there should exist a PPT simulator $\mathsf{Sim}_{\mathsf{S}}$ that $\mathsf{View}_{\mathsf{S}}^{\Pi} \overset{c}{\approx} \mathsf{Sim}_{\mathsf{S}}(\mathcal{W})$, where S is the medical service and $\mathsf{View}_{\mathsf{S}}^{\Pi}$ indicates the view of the semi-honest medical service in the real-world protocol execution.*

Theorem 1. *`MediSC`'s secure neural network inference protocol securely realizes the ideal functionality $\mathcal{F}^{\mathsf{SNNI}}$ under Definition 2.*

Proof. We show a simulator for the corrupted medical service or hospital, such that the distribution of real protocol execution is computationally indistinguishable to the simulated distribution according to our security definition.

- **Simulator for the corrupted hospital:** Let $\mathsf{Sim}_{\mathsf{BM}}$ denote the simulator of Beaver's multiplication procedure. Its emulated view is indistinguishable from the real view of hospital H in the multiplication procedure. $\mathsf{Sim}_{\mathsf{H}}$ chooses an uniform random tape for the hospital.

i) During preprocessing of $\mathit{\Pi}$, $\mathsf{Sim}_{\mathsf{H}}$ produces and outputs the randomness $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_{2^\ell}$ to emulate the message in real protocol, i.e., $\mathcal{W} - \mathbf{a}^1$. As both messages are uniformly distributed in ring $\mathbb{Z}_{2^\ell}$ and given the security of additive secret sharing, the hospital cannot distinguish the simulated message with the one received from real protocol. The hospital calculates $\langle \tilde{\mathbf{X}}_2 \rangle_0 = \mathbf{a}^0 \cdot \mathbf{r} + \mathbf{a}^2$ and sets $\mathbf{u} = \mathbf{a}^0$.

ii) During online inference of $\mathit{\Pi}$, the corrupted hospital inputs the shares of medical record $\mathbf{X} - \mathbf{u}$ or protected activation shares $\langle \bar{\mathbf{X}}_i \rangle_0 - \mathbf{u}$ and the shares $\langle \mathbf{X}_{i+1} \rangle_0$ and receives no messages for the linear layers. $\mathsf{Sim}_{\mathsf{H}}$ works in a dummy way by directly outputting inputs of the hospital, and thus the output of $\mathsf{Sim}_{\mathsf{H}}$ is identically distributed to the view of the semi-honest hospital. For the non-linear layers, $\mathsf{Sim}_{\mathsf{H}}$ produces $\langle \tilde{\mathbf{X}}_{i+1} \rangle_1 \xleftarrow{\$} \mathbb{Z}_{2^\ell}$ and invokes $\mathsf{Sim}_{\mathsf{BM}}$ to conduct secure multiplication over $\langle \tilde{\mathbf{X}}_{i+1} \rangle_1$ and $\langle \mathbf{X}_{i+1} \rangle_0$ whenever interactions are involved in the secure ReLU function. $\mathsf{Sim}_{\mathsf{H}}$ outputs the simulated shares of activation returned from the secure ReLU function. $\mathsf{Sim}_{\mathsf{H}}$ performs the above operations for each layer. At the end, $\mathsf{Sim}_{\mathsf{H}}$ outputs the simulated last layer's result shares $\langle \tilde{\mathbf{X}}_L \rangle_0, \langle \tilde{\mathbf{X}}_L \rangle_1$. The combination of these two shares is uniformly distributed in ring $\mathbb{Z}_{2^\ell}$, same as the result from the real protocol execution. Thus, the output of $\mathsf{Sim}_{\mathsf{H}}(\mathbf{X}, \mathcal{W}(\mathbf{X}))$ is indistinguishable to the view $\mathsf{View}_{\mathsf{H}}^{\mathit{\Pi}}$ of the semi-honest hospital.

- **Simulator for the corrupted medical service:** $\mathsf{Sim}_{\mathsf{S}}$ chooses an uniform random tape for the medical service.

i) During preprocessing of $\mathit{\Pi}$, the medical service inputs only the shares of model $\mathcal{W} - \mathbf{a}^1$ and does not receive any messages. $\mathsf{Sim}_{\mathsf{S}}$ works in a dummy way by directly outputting inputs of the medical service $\mathbf{v} = \mathbf{a}^1$. Thus, the output of $\mathsf{Sim}_{\mathsf{S}}$ is identically distributed to the view $\mathsf{View}_{\mathsf{S}}^{\mathit{\Pi}}$ of the semi-honest medical service.

ii) During online inference of $\mathit{\Pi}$, $\mathsf{Sim}_{\mathsf{S}}$ produces and outputs the randomness $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_{2^\ell}$ to simulate the real world message $\mathbf{X} - \mathbf{u}$ (or $\bar{\mathbf{X}} - \mathbf{u}$). Given the security of additive secret sharing, the medical service cannot distinguish the simulated message with the one received from real protocol. For the non-linear layers, $\mathsf{Sim}_{\mathsf{S}}$ produces $\langle \tilde{\mathbf{X}}_{i+1} \rangle_0 \xleftarrow{\$} \mathbb{Z}_{2^\ell}$. Whenever interactions are involved in the secure ReLU function, $\mathsf{Sim}_{\mathsf{S}}$ invokes $\mathsf{Sim}_{\mathsf{BM}}$ to conduct secure multiplication over $\langle \tilde{\mathbf{X}}_{i+1} \rangle_0$ and $\langle \mathbf{X}_{i+1} \rangle_1$ received from the medical service. $\mathsf{Sim}_{\mathsf{S}}$ outputs the simulated shares of activation returned from the secure ReLU function. $\mathsf{Sim}_{\mathsf{S}}$ performs the above operations for each layer. Since all simulated intermediary messages are uniformly distributed in ring $\mathbb{Z}_{2^\ell}$, and given the security of additive secret sharing and Beaver's secure multiplication procedure, the output of $\mathsf{Sim}_{\mathsf{S}}(\mathcal{W})$ is indistinguishable to the view $\mathsf{View}_{\mathsf{S}}^{\mathit{\Pi}}$ of the corrupted medical service.

6 Performance Evaluation

We implement a prototype of `MediSC` in Java and evaluate the prototype to two computational nodes emulating the hospital and the medical service. Each

Table 1. Performance of secure layer functions.

Secure layer	Conv.		BN	ReLU	MaxPool	AvgPool
	3×3	5×5			2×2	2×2
Time (ms)	1.25	2.16	1.74	22.7	31.2	0.05
Comm. (Bytes)	36	100	4	32	144	0

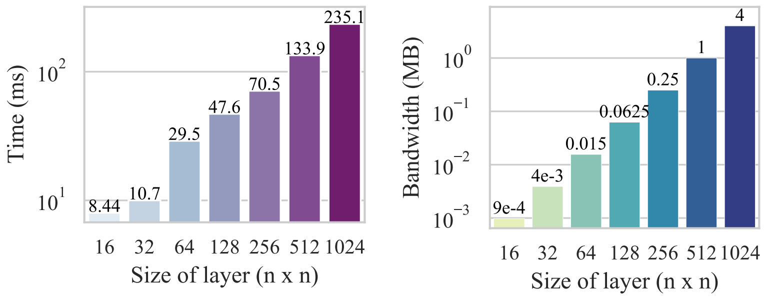

Fig. 5. Performance of the secure FC layer. Left: time (ms). Right: bandwidth (MB).

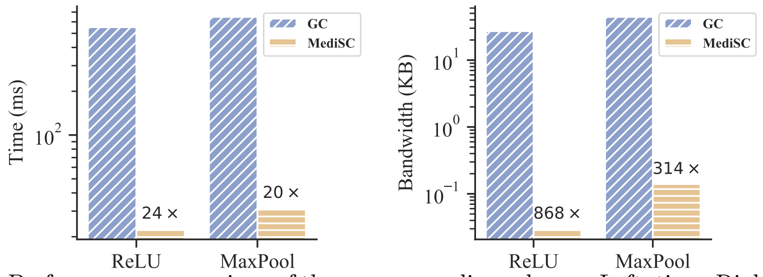

Fig. 6. Performance comparison of the secure non-linear layers. Left: time. Right: bandwidth. Baseline: GC realizations.

computational node runs CentOS Linux 7 with Intel Xeon Gold 6150 CPU at 2.7GHz, 384GB RAM, Mellanox Spectrum network. In our experiment, we set the data filed size to be 32-bit integers, i.e., the additively secret shared data in ring $\mathbb{Z}_{2^{32}}$. We follow most of the prior secure inference works [30, 22, 15] to evaluate `MediSC` in fast networks, as the hospital and the medical service can communicate via dedicated connections. We evaluate `MediSC` with two benchmarking datasets (MNIST and CIFAR-10) with three NN models, and four real-world medical datasets (Breast Cancer, Diabetes, Liver Disease, and Thyroid). For training, we use PyTorch backend on a NVIDIA Tesla V100 GPU. More implementation details and model architectures are available in Appendix Sec. A.

6.1 Microbenchmarks

Secure layer functions. We evaluate `MediSC`'s secure layer functions: the secure convolutional (Conv.), fully connected (FC), batch normalization (BN), ReLU, max pooling (MaxPool), and average pooling (AvgPool) layers. They are the main building blocks in `MediSC`'s secure inference. For demonstration, we choose to evaluate the Conv. with the commonly-used 3×3 and 5×5 filter

Table 2. Performance summary of the benchmarking networks.

Dataset	Model	Time (s)		Comm. (MB)		Accu.	Layers
		Preprocess[a]	Online	Preprocess[a]	Online		
MNIST	M1	0.07	0.57	0.45	0.45	98%	3FC-2ReLU
	M2	1.21	4.42	2.54	2.62	99%	4CONV/FC-8ReLU-2AP
CIFAR-10	C1	17.01	130.02	243.0	246.4	81%	8CONV/FC-8ReLU-2AP

[a] One-time cost during preprocessing.

Table 3. Performance summary of the medical applications.

Dataset/model	Time (s)		Comm. (KB)		Accu.	Layers
	Preprocess	Online	Preprocess	Online		
Breast Cancer	0.10	0.20	3.13	4.13	93%	3FC-2ReLU-3BN
Diabetes	0.03	0.16	2.34	3.59	74%	3FC-2ReLU
Liver Disease	0.06	0.32	19	23	72%	3FC-2ReLU
Thyroid	0.28	0.64	49.2	55.5	98%	3FC-2ReLU-3BN

Table 4. Bandwidth (MB) comparison of `MediSC` with prior art.

Model M1		Model M2		Model C1	
MiniONN	15.8	MiniONN	657.5	MiniONN	9272
CryptoNets	372.2	FALCON	62.1	FALCON	1278
XONN	4.29	XONN	32.13	XONN	2599
Chameleon	10.5	Gazelle (ReLU)	70	Chameleon	2650
				Gazelle (ReLU)	~5000
				Delphi (ReLU)	~5100
MediSC	**0.9**	**MediSC**	**5.16**	**MediSC**	**489**
Breast Cancer		Diabetes		Liver Disease	
XONN	0.35	XONN	0.16	XONN	0.3
MediSC	**0.007**	**MediSC**	**0.005**	**MediSC**	**0.04**

sizes, and the MaxPool and AvgPool with 2×2 pooling window. As Table 1 benchmarks, all functions are demonstrated lightweight, where the linear and non-linear layers can be finished within 2.5ms and 35ms respectively, consuming less than 150 Bytes bandwidth. Fig. 5 plots the performance of the $n \times n$ fully connected layer. The time (left figure) and bandwidth (right figure) ascend in linear with the growth of the input and output feature size n.

Non-linear layers comparison with GC. Fig. 6 demonstrates that `MediSC`'s design achieves $24\times, 20\times$ speedup and consumes $868\times, 314\times$ less communication for the ReLU and MaxPool over the GC-based approaches. This GC baseline realizes equivalent functionalities to ours. For a fair comparison, we use the Java based GC framework [33] which integrates modern free-XOR and half-AND optimizations. Such achievements validate that `MediSC`'s purely secret sharing based design is lightweight and much more practical, compared with the prior works involving GC [30, 31, 22, 28].

Table 5. Performance breakdown of M1.

Layers	Preprocess	FC1	ReLU1	FC2	ReLU2	FC3
Time (s)	0.072	0.123	0.198	0.055	0.198	0.001
Comm. (MB)	0.45	0.383	0.0039	0.0625	0.0039	0.0048

Table 6. Performance breakdown of M2.

Layers	Preprocess	CONV1	ReLU1	AP1	CONV2	ReLU2	AP2	FC3	ReLU3	FC4
Time (s)	1.21	0.80	3.04	0.002	0.15	0.28	1.4E-4	0.029	0.087	0.024
Comm. (MB)	2.54	0.87	0.07	0	1.56	0.008	0	0.097	0.003	0.0038

Table 7. Performance breakdown of C1.

Layers	Preprocess	CONV1	ReLU1	CONV2	ReLU2	AP1	CONV3	ReLU3	CONV4
Time (s)	17.0	6.99	68.32	6.45	14.2	0.008	1.104	17.29	1.07
Comm. (MB)	243.0	6.75	2.0	144.0	0.5	0.0	36.0	0.5	36.0
Layers	ReLU4	AP2	CONV5	ReLU5	CONV6	ReLU6	CONV7	ReLU7	FC1
Time (s)	4.3	0.002	0.21	4.35	0.17	4.35	0.056	1.08	0.003
Comm. (MB)	0.125	0.0	9.0	0.125	9.0	0.125	2.25	0.03	0.04

6.2 MediSC's Protocol Performance

Evaluations on MNIST and CIFAR-10. We evaluate MediSC's secure inference protocol on MNIST and CIFAR-10 datasets with three models, and summarize the performance in Table 2. For MNIST, MediSC produces high-quality predictions with 0.47s (98%) and 4.42s (99%) online processing time for M1 (3FC-ReLU) and M2 (4CONV/FC-ReLU-2AP). For CIFAR-10, MediSC consumes 2.1min to produce a 81% accurate prediction for the model C1 (8CONV/FC-ReLU-2AP). Note that the costs of preprocessing are one-time overhead and are determined by the model size.

Evaluations on medical datasets. To showcase MediSC's applicability for the real-world medical diagnostic applications, we deploy and evaluate our secure NN inference protocol over the publicly available healthcare datasets. As shown in Table 3, MediSC produces the robust diagnoses within 1s for all medical applications and consumes <60KB. Besides, the workload during preprocessing conducted mainly at the hospital side is light (within 0.3s and 50KB), which confirms that MediSC is amiable for resource constrained devices.

Comparison with prior art. We compare MediSC's performance with notable prior secure NN inference works in Table 4 to demonstrate MediSC's performance efficiency. MediSC requires the least network resources among all other prior works with up to 413× bandwidth saving for MNIST and up to 19× bandwidth saving for CIFAR-10. For the medical datasets, MediSC achieves at least 10× improvement over XONN, the notable prior work considering medical scenario.

For the SOTA - Delphi [28] (all ReLU version for keeping accuracy), it consumes overall 5100MB while MediSC only needs 489MB, with a 10× improvement[7]. Such significant improvement stems from the fact that MediSC only involves lightweight secret sharing based secure computation through out the whole service procedure, while Delphi involves the use of heavy homomorphic encryption and garbled circuits. Regarding the overall runtime, we note that it is not fair to make a direct comparison with results reported in [28] as Delphi is

[7] Preprocessing: 243MB in MediSC and 4915MB in Delphi.

Table 8. Performance breakdown of Breast Cancer.

Layers	Preprocess	FC1	BN1	ReLU1	FC2	BN2	ReLU2	FC3	BN3
Time (ms)	100	7	27	110	75	1.8	5.4	0.2	0.2
Comm. (KB)	3.13	1.84	0.062	0.49	1	0.0625	0.5	0.12	0.007

Table 9. Performance breakdown of Diabetes.

Layers	Preprocess	FC1	ReLU1	FC2	ReLU2	FC3
Time (ms)	62.2	38.5	146.3	47.8	92.9	0.3
Comm. (KB)	18.94	2.50	2.00	16.00	16.00	0.50

Table 10. Performance breakdown of Liver Disease.

Layers	Preprocess	FC1	ReLU1	FC2	ReLU2	FC3
Time (ms)	34.8	6.1	102.4	4.1	47.3	0.2
Comm. (KB)	2.05	0.63	0.62	1.54	0.63	0.16

Table 11. Performance breakdown of Thyroid.

Layers	Preprocess	FC1	BN1	ReLU1	FC2	BN2	ReLU2	FC3	BN3
Time (ms)	287.2	46.1	21.7	248.8	28.0	21.1	262.8	27.3	0.3
Comm. (KB)	49.23	8.20	0.39	3.13	39.06	0.39	3.13	1.17	0.01

implemented in a different programming language (Rust) with significant optimizations and acceleration from GPU computing. Our performance results are not based on such optimizations.

It is worth noting that secure evaluation of non-linear layers is the *performance bottleneck* in secure neural network inference [28]. For evaluation of the original ReLU function, Delphi adopts a GC-based approach. Note that we have provided above in Fig. 6 a (fair) comparison between our design and the GC-based approach, which has demonstrated a significant performance boost of our design over the GC-based approach (24× in runtime and 868× in communication). On another hand, even when a direct (unfair) comparison is made with their reported runtime driven by aforementioned significantly *optimized* and *sophisticated* implementations, the overall runtime of `MediSC` with much simplified implementations is still comparable (147s in `MediSC` against 140s in Delphi).

Performance breakdown. The breakdown of time and bandwidth costs of the preprocessing and each layer during online inference are given in this section. Table 5 and Table 6 report the performance breakdown of M1, M2 for MNIST. Table 7 reports the performance breakdown of C1 for CIFAR-10. Table 8, Table 9, Table 10 and Table 11 report the performance breakdown of Breast Cancer, Diabetes, Liver Disease, and Thyroid.

7 Conclusion

In this paper, we present `MediSC`, a 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 ReLU and max pooling layer functions are well supported in a secure and efficient manner. With `MediSC`, 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 Australian Research Council (ARC) Discovery Projects (No. DP200103308, No. DP180103251, and No. DP190102835), ARC Linkage Project (No. LP160101766), and HITSZ Start-up Research Grant (No. BA45001023).

References

1. Breast cancer. `https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/`
2. Diabetes. `https://www.kaggle.com/uciml/pima-indians-diabetes-database`
3. Liver disease. `https://www.kaggle.com/uciml/indian-liver-patient-records`
4. Thyroid. `https://archive.ics.uci.edu/ml/datasets/Thyroid+Disease`
5. Google DeepMind Health. Online at `https://deepmind.com/blog/announcements/deepmind-health-joins-google-health` (2020)
6. Microsoft Project InnerEye. Online at `https://www.microsoft.com/en-us/research/project/medical-image-analysis/` (2020)
7. PathAI. Online at `https://www.pathai.com/` (2020)
8. 104th United States Congress: Health Insurance Portability and Accountability Act of 1996 (HIPPA). online at `https://www.hhs.gov/hipaa/index.html` (1996)
9. Atallah, M., Bykova, M., Li, J., Frikken, K., Topkara, M.: Private collaborative forecasting and benchmarking. In: Proc. of WPES (2004)
10. Barni, M., Failla, P., Lazzeretti, R., Sadeghi, A.R., Schneider, T.: Privacy-preserving ecg classification with branching programs and neural networks. IEEE Trans. on Information Forensics and Security (2011)
11. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Proc. of Crypto (1991)
12. Brutzkus, A., Gilad-Bachrach, R., Elisha, O.: Low latency privacy preserving inference. In: Proc. of ICML. pp. 812–821. PMLR (2019)
13. European Parliament and the Council: The General Data Protection Regulation (GDPR). online at `http://data.europa.eu/eli/reg/2016/679/2016-05-04` (2016)
14. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proc. of ACM CCS (2015)
15. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: Proc. of ICML (2016)
16. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proc. of STOC (1987)
17. Harris, D.: A taxonomy of parallel prefix networks. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. vol. 2, pp. 2213–2217. IEEE (2003)
18. Jacobi, A., Chung, M., Bernheim, A., Eber, C.: Portable chest x-ray in coronavirus disease-19 (covid-19): A pictorial review. Clinical Imaging (2020)
19. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: Gazelle: A low latency framework for secure neural network inference. In: Proc. of 27th USENIX Security (2018)
20. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural networks **6**(6), 861–867 (1993)
21. Li, S., Xue, K., Zhu, B., Ding, C., Gao, X., Wei, D., Wan, T.: Falcon: A fourier transform based approach for fast and secure convolutional neural network predictions. In: Proc. of IEEE/CVF CVPR (2020)
22. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via minionn transformations. In: Proc. of ACM CCS (2017)
23. Liu, X., Wu, B., Yuan, X., Yi, X.: Leia: A lightweight cryptographic neural network inference system at the edge. IACR Cryptol. ePrint Arch. **2020**, 463 (2020)

24. Liu, X., Yi, X.: Privacy-preserving collaborative medical time series analysis based on dynamic time warping. In: European Symposium on Research in Computer Security. pp. 439–460. Springer (2019)
25. Liu, X., Zheng, Y., Yi, X., Nepal, S.: Privacy-preserving collaborative analytics on medical time series data. IEEE Transactions on Dependable and Secure Computing (2020)
26. Lou, Q., Jiang, L.: She: A fast and accurate deep neural network for encrypted data. In: Proc. of NeurIPS. pp. 10035–10043 (2019)
27. Lou, Q., Lu, W.j., Hong, C., Jiang, L.: Falcon: Fast spectral inference on encrypted data. Proc. of NeurIPS **33** (2020)
28. Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., Popa, R.A.: Delphi: A cryptographic inference service for neural networks. In: USENIX Security Symposium (2020)
29. Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: Proc. of IEEE S&P (2017)
30. Riazi, M.S., Samragh, M., Chen, H., Laine, K., Lauter, K., Koushanfar, F.: Xonn: Xnor-based oblivious deep neural network inference. In: Proc. of 28th USENIX Security (2019)
31. Riazi, M.S., Weinert, C., Tkachenko, O., Songhori, E.M., Schneider, T., Koushanfar, F.: Chameleon: A hybrid secure computation framework for machine learning applications. In: Proc. of AsiaCCS (2018)
32. Wagh, S., Gupta, D., Chandran, N.: Securenn: 3-party secure computation for neural network training. Proc. of PETS (2019)
33. Wang, X.: Flexsc. `https://github.com/wangxiao1254/FlexSC` (2018)
34. Xie, P., Wu, B., Sun, G.: Bayhenn: Combining bayesian deep learning and homomorphic encryption for secure dnn inference. In: Proc. of IJCAI. pp. 4831–4837 (2019)
35. Yu, L., Liu, L., Pu, C., Gursoy, M.E., Truex, S.: Differentially private model publishing for deep learning. In: Proc. of S&P. IEEE (2019)
36. Zhang, Q., Wang, C., Wu, H., Xin, C., Phuong, T.V.: Gelu-net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning. In: Proc. of IJCAI. pp. 3933–3939 (2018)
37. Zheng, Y., Duan, H., Wang, C.: Towards secure and efficient outsourcing of machine learning classification. In: Proc. of ESORICS. Springer (2019)

A Further Implementation Details

A.1 More Details of Implementation Setting

`MediSC` is implemented in Java. Recall that `MediSC`'s secure NN inference protocol is computed in the secret sharing domain over ring $\mathbb{Z}_{2^\ell}$, i.e., all real-valued model weights are converted into ℓ-bit signed fixed-point integers and secretly shared in $\mathbb{Z}_{2^\ell}$. In `MediSC`, we follow the state-of-the-art work [28] to choose the ring size as $\mathbb{Z}_{2^{32}}$, a 32-bit ring by the modulus 4294967296. To represent the signed integers, we split the ring into two halves, where the lower-half ring $[0, 2^{31}-1]$ represents the non-negative values and the upper-half ring $[2^{31}, 2^{32}-1]$ represents the negative values. In this way, both the sign and the secret value is well protected. Besides, to convert the real-valued model weights to 32-bit fixed-point integers, we scale and quantize the weight with a scaling factor s to represent the bit length of the fractional part. For M1, M2, and C1, the factor is set as 1024, 128, and 64, respectively. For all medical datasets, the factor is set as 1024.

Multiplication over two fixed-point integers can overflow the capacity of the ring $\mathbb{Z}_{2^\ell}$, since the fractional part is increased to $2s$ bits in the resulting product. To assure the correctness, all intermediate results after multiplying over two shares should be rescaled down by 2^s before subsequent operation. We follow prior works [28, 32] to adopt a secure local truncation scheme proposed in the work [29], which simply discard the last s fractional bits to adjust the product to ℓ bits.

A.2 Training Details

We provide the detailed setting of training over plaintext datasets. Recall that we train the models M1 and M2 on MNIST, the model C1 on CIFAR-10, and the models over four publicly available medical datasets: Breast Cancer [1], Diabetes [2], Liver Disease [3] and Thyroid Disease [4]. Our training procedure is executed on NVIDIA Tesla V100 GPU with PyTorch backend. We adopt the SGD for M1, M2, C1, and Breast Cancer, and Adam optimizer for Diabetes, Liver Disease, and Thyriod. They are with adaptive learning rate with cosine learning rate decay every 50 epoches. For all datasets, all image pixels and the medical features are normalized to integers in $[0, 255]$. In this way, the hospital's inputs do not need to be preprocessed in our secure NN inference protocol.

Table 12. Summary of training settings.

Learning rate	Weight decay	Momentum	Optimizer	Epoch	Batch size
MNIST (M1, M2)					
1×10^{-3}	5×10^{-4}	0.9	SGD	600	128
CIFAR-10 (C1)					
1×10^{-3}	5×10^{-4}	0.9	SGD	600	128
Breast Cancer					
1×10^{-3}	5×10^{-4}	0.9	SGD	9000	453
Diabetes					
1×10^{-5}	-	-	Adam	50000	615
Liver Disease					
1×10^{-4}	-	-	Adam	50000	467
Thyroid					
1×10^{-5}	-	-	Adam	30000	3772

A.3 More Details of Model Architecture

In this section, we present the detailed model architectures used in our paper. The models M1 and M2 are trained on MNIST. In general, M1 is a Multi-Layer Perception consisting of 3 fully connected (FC) layers with ReLU activation, which has been used in prior works [19, 30, 15, 22, 31]. The architecture of M1 is summarized in Table 13. As shown in Table 14, M2 comprises 3 convolutional (CONV) layers with ReLU, 2 average pooling (AP) layers and an FC layer, which has been adopted in prior works [22, 19, 30, 27]. For CIFAR-10, the model

C1 (minionn network) consists of 7 CONV layers with ReLU, 2 AP layers and an FC layer as shown in Table 15. Is has been adopted in prior works [22, 31, 19, 30, 27] for a benchmarking evaluation. Table 16, table 17, and table 18 report the architectures of the models on Breast Cancer, Diabetes, and Liver Disease, respectively. They have been adopted in prior work [30]. Table 19 reports the model architecture evaluating on Thyroid disease.

Table 13. Model architecture of M1.

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

Table 14. Model architecture of M2.

Layers	Padding	Stride
CONV (input: $1 \times 28 \times 28$, kernel: $1 \times 16 \times 5 \times 5$ feature: $16 \times 24 \times 24$) + ReLU	-	1
AP (input: $16 \times 24 \times 24$, window: $16 \times 2 \times 2$ output: $16 \times 12 \times 12$)	-	2
CONV (input: $16 \times 12 \times 12$, kernel: $16 \times 16 \times 5 \times 5$ feature: $16 \times 8 \times 8$) + ReLU	-	1
AP (input: $16 \times 8 \times 8$, window: $16 \times 2 \times 2$ output: $16 \times 4 \times 4$)	-	2
FC (input: 256, output: 100) + ReLU	-	-
FC (input: 100, output: 10)	-	-

Table 15. Model architecture of C1.

Layers	Padding	Stride
CONV (input: $3 \times 32 \times 32$, kernel: $3 \times 64 \times 3 \times 3$ feature: $64 \times 30 \times 30$) + ReLU	0	1
CONV (input: $64 \times 32 \times 32$, kernel: $64 \times 64 \times 3 \times 3$ feature: $64 \times 32 \times 23$) + 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 16. 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 17. 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 18. 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 19. 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	-	-