Securely Outsourcing Neural Network Inference to the Cloud With Lightweight Techniques

Xiaoning Liu, Yifeng Zheng, Xingliang Yuan, and Xun Yi

Abstract—Neural network (NN) inference services enrich many applications, like image classification, object recognition, facial verification, and more. These NN inference services are increasingly becoming an essential offering from cloud computing providers, where end-users' data are offloaded to the cloud for inference under a customized model. However, current cloud-based inference services operate on clear inputs and NN models, raising paramount privacy concerns. Individual user data may contain private information that should always remain confidential. Meanwhile, the NN model is deemed proprietary to the model owner as model training requires substantial resources. In this article, we present, tailor, and evaluate `Sonic`, a lightweight secure NN inference service delegated in the cloud. `Sonic` leverages the cloud computing paradigm to fully outsource the secure inference, freeing end devices and model owners from being actively online for assistance. `Sonic` guards both user input and model privacy along the whole service flow. We design a series of secure and efficient NN layer functions purely using lightweight cryptographic primitives. Extensive evaluations demonstrate that `Sonic` achieves up to $60\times$ bandwidth saving in online inference compared to prior art.

Index Terms—Secure outsourcing, cloud computing, privacy preservation, neural network inference

1 Introduction

Neural network inference services provide ready-made intelligence for end-user applications and help enterprises improve their business, ranging from image classification to face detection [1]. With the wide adoption of cloud computing in various domains (e.g., [2], [3], [4]), these inference services have rapidly become an appealing offering from cloud service providers to empower many applications. Consider a typical application of face detection, where an end-user wishes to evaluate a personal photo under an enterprise's neural network. Due to the well-understood benefits, the enterprise takes advantage of cloud computing service (like Google Vision [5]), where both the end-user photo and model are offloaded to the cloud. The inference service takes place on the cloud which then returns the result to the end-user.

Such practices seem appealing, yet raises critical privacy concerns, posing hurdles to the practical deployment of the service. The raw user data processed by these services often carry private information, such as biometrics and locations [6]. To ensure individuals' privacy, such data should be kept confidential at any time and not be disclosed to cloud in cleartext. Meanwhile, NN models are deemed as intellectual properties and embed traces of (sensitive) training data [7], [8]. Privacy protection of the private user data and proprietary models is thus of paramount importance and necessary in pushing forward the practical deployment of inference services in the cloud [9], [10].

Towards this pressing need, there have been growing research endeavors on leveraging generic cryptographic techniques (such as garbled circuits and homomorphic encryption) to build secure NN inference systems [11], [12], [13]. These systems proceed by encrypting user input data and/or an owner's model, and execute NN inference over encrypted data. Unfortunately, they are still unsatisfactory for practical deployment. Specifically, during the online secure inference, they all need to involve the aforementioned heavyweight cryptographic techniques, and some of them require active user aids throughout the whole inference procedure. Namely, end-users need to continually engage in online interactions and have *symmetric* computing capabilities to the server, in the case that the server holds the plaintext model. These limitations are further aggravated in resource-constrained mobile or embedded user devices.

In light of above observations, in this paper, we propose, implement, and evaluate `Sonic`, a lightweight secure NN inference system running in the cloud. `Sonic` is designed to support Convolutional Neural Networks (CNN), one of the most popular and powerful deep learning models. As shown in Fig. 1, `Sonic` offloads the entire secure inference computation to the cloud. Using such a system framework, `Sonic` mediates the resource limitation of the mobile user and the privacy-invasive cloud: it frees the resource-constrained mobile user from always staying online; and allows

- *Xiaoning Liu and Xun Yi are with the School of Computing Technologies, RMIT University, Melbourne, VIC 3001, Australia. E-mail: xiaoning.trust@gmail.com, xun.yi@rmit.edu.au.*
- *Yifeng Zheng is with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China. E-mail: yifeng.zheng@hit.edu.cn.*
- *Xingliang Yuan is with the Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia. E-mail: Xingliang.Yuan@monash.edu.*

Manuscript received 30 Apr. 2021; revised 11 Nov. 2021; accepted 3 Jan. 2022. Date of publication 7 Jan. 2022; date of current version 16 Jan. 2023.
This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2021A1515110027, in part by the Shenzhen High-Level Talents Research Start-up Funds and the HITSZ Research Start-up Funds, in part by the Australian Research Council (ARC) Discovery Projects under Grants DP200103308, DP180103251, and DP190102835, and in part by the ARC Linkage Project under Grant LP160101766.
(Corresponding author: Yifeng Zheng.)
Digital Object Identifier no. 10.1109/TDSC.2022.3141391

1545-5971 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

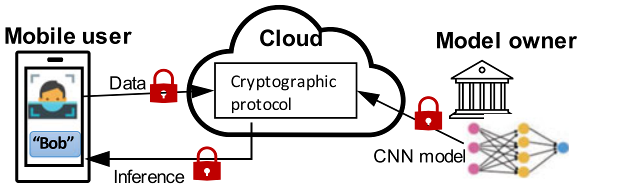


Fig. 1. A secure CNN-powered inference in the cloud. The lock indicates that the data is deployed in encrypted form.

the model owner to dynamically fine-tune their service with an updated NN model without reinstalling the application. At the same time, both the private user data and proprietary CNN model are well protected against the untrusted cloud. In particular, we design a delicate secure inference protocol purely from lightweight secret sharing techniques to foster a low-latency service with judicious usage of network resources. Our contributions can be summarized as follows.

- We design a secure cloud-based outsourced service `Sonic` that supports lightweight secure CNN inference, with protection of both the private user data and the proprietary CNN model. By combining the advancements from system, cryptography, and machine learning literature, `Sonic` enables a full-fledged cloud service framework that can support generic CNN models and provide high-performance inference service.
- We devise a series of efficient secure layer functions fully resorting to lightweight secret sharing. They are essential building blocks for CNN inference, including the secure convolutional layer (SCONV), secure batch normalization (SBN), secure ReLU activation (SReLU), and secure max pooling layer (SMP). We decompose these secure layer functions into smaller and composable cryptographic gadgets, and carefully devise each gadget to optimize the performance of secure NN inference on the cloud.
- We integrate the developed secure functionalities and apply them to four realistic CNN models over benchmarking datasets for extensive evaluation. Results validate the efficiency of `Sonic`, showing that `Sonic` produces a prediction in 0.7s on MNIST with 97% accuracy and in 1.5min on CIFAR-10 with 80% accuracy. Compared with prior art, `Sonic` can achieve up to 60× bandwidth saving in the online inference procedure.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries. Section 3 presents the system architecture and threat model. Section 4 introduces the proposed secure inference protocol. Section 5 analyzes the security. Section 6 presents the experiments. Section 7 investigates the related work. Section 8 concludes this paper.

2 Preliminary: Additive Secret Sharing

We now introduce the core cryptographic primitive used in `Sonic`— additive secret sharing. Additive secret sharing [14] protects an ℓ-bit secret value x by additively splitting it in the ring $\mathbb{Z}_{2^\ell}$ as $\langle x\rangle_0^A$ and $\langle x\rangle_1^A$ such that $\langle x\rangle_0^A + \langle x\rangle_1^A \equiv x \,(\mathrm{mod}\ 2^\ell)$. Each individual secret share is a uniformly distributed random value in $\mathbb{Z}_{2^\ell}$ and can perfectly hide the information about x. Given the shares of two secret values are x and y, and two parties (denoted as P_0 and P_1) join the secure computation that supply and obtain corresponding shares of the values. Some computation among two parties can be supported as follows. First, addition/subtraction over shares ($\langle z\rangle_i^A = \langle x\rangle_i^A \pm \langle y\rangle_i^A$) and multiplication by a public value ($\langle z\rangle_i^A = \eta \cdot \langle x\rangle_i^A$) can be efficiently evaluated by each party P_i ($i \in \{0,1\}$) at local with no interaction.

Multiplication over two shares ($\langle z\rangle^A = \langle x\rangle^A \cdot \langle y\rangle^A$) requires the assistance of a pre-computed secret-shared multiplication triple (a,b,c) [15], where $c = a \cdot b$. Such triples are data independent and can be generated offline by a third party [16], [17]. As such, `Sonic` assumes the secret-shared triples are already available during the online service. With the secret-shared triple, secure multiplication can be supported as follows. Each party P_i first sets $\langle e\rangle_i^A = \langle x\rangle_i^A - \langle a\rangle_i^A$ and $\langle f\rangle_i^A = \langle y\rangle_i^A - \langle b\rangle_i^A$. Then both parties interact to reconstruct e and f. Party P_i then sets $\langle z\rangle_i^A = i \cdot e \cdot f + f \cdot \langle a\rangle_i^A + e \cdot \langle b\rangle_i^A + \langle c\rangle_i^A$.

It is noted that for the special case $l = 1$, additive secret sharing is over $\mathbb{Z}_2$. In this case, the above addition and multiplication operations are replaced with boolean operations XOR ($\oplus$) and AND ($\wedge$) respectively. We denote the boolean sharing of a secret value x in $\mathbb{Z}_2$ as ($[\![x]\!]_0$, $[\![x]\!]_1$). We summarize the key notations in Table 1.

TABLE 1
Key Notations

Notation	Description
$\mathbf{X}, X, \mathbf{x}$	Input/activation tensor, matrix, and vector.
$\mathbf{W}, W, \mathbf{w}$	Weight tensor, matrix, and vector.
$\mu, \delta, \gamma, \beta$	Batch normalization parameters: the running mean, running variance, scale, and shift.
ℓ, n	Bit length ℓ and vector length n.
$x_k, \mathbf{x}_k$	The kth element of vector $\mathbf{x}$; The kth vector.
$\in_R$	Uniformly random sampling from a distribution
$\langle x\rangle_i^A$	Arithmetic shares of value x held by server i
$[\![x]\!]_i$	Boolean shares of value x held by server i
$\langle x\rangle_i^A \pm \langle y\rangle_i^A$	Addition/subtraction over arithmetic shares
$\langle x\rangle^A \cdot \langle y\rangle^A$	Multiplication over arithmetic shares
$[\![x]\!]_i + [\![y]\!]_i$	Bitwise XOR over Boolean shares
$[\![x]\!] \cdot [\![y]\!]$	Bitwise AND over Boolean shares

3 System Overview

3.1 Architecture

Fig. 2 shows `Sonic`'s system architecture. There is a cloud-based service platform operated by two distinct servers S_0 and S_1 from independent cloud providers (like Amazon Rekognition [18] and Google Vision [5]) to jointly provide efficient secure neural network inference services. Through the service, a *model owner* holding a proprietary CNN model can deploy the model in protected form on the cloud, which can then provide inference services without seeing the model in cleartext. In practice, the model owner can be a mobile application developing company who has spent massive resources in training the model for a certain application like image recognition with its private customers' data. `Sonic` supports a *mobile user*, who runs an ML-powered mobile application collecting user data, to get inference

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

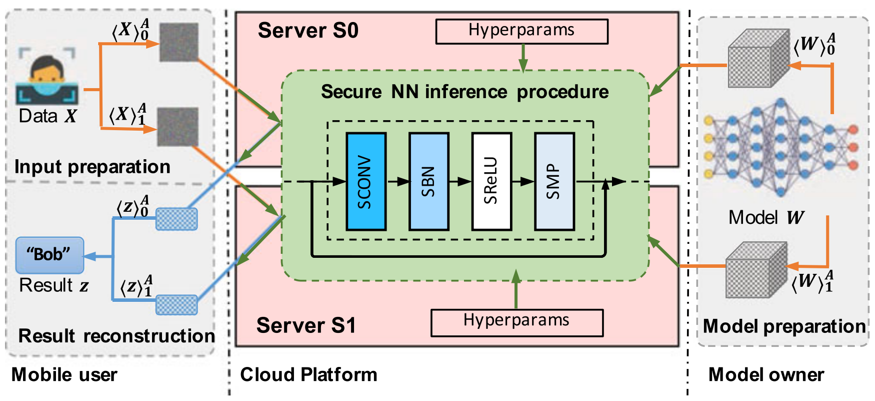


Fig. 2. System architecture.

results without disclosing the confidential or sensitive user data. Sonic's cloud-aided architecture frees the resource-constrained mobile user from storing large models on local device or continuously engaging in the secure inference. Meanwhile, it facilitates the model owner to dynamically fine-tune its service, where the neural network model can be regularly updated without republishing the mobile application.

Execution Flow. At a high level, Sonic works as follows: The model owner protects a pre-trained CNN model $\mathbf{W}$ by the secret sharing technique, generating secret shares $\langle\mathbf{W}\rangle_0^A$ and $\langle\mathbf{W}\rangle_1^A$. Then, $\langle\mathbf{W}\rangle_0^A$ and $\langle\mathbf{W}\rangle_1^A$ are deployed on the cloud servers S_0 and S_1, respectively. Once the mobile user's request arrives at the user interface (on the left of Fig. 2), Sonic protects the raw input (e.g., facial image) as secret-shared tensors $\langle\mathbf{X}\rangle_0^A$ and $\langle\mathbf{X}\rangle_1^A$ and feeds them into the secure service hosted in corresponding cloud servers. The cloud servers run Sonic's secure inference procedure collaboratively (green box in Fig. 2) and send the secret-shared output $\langle\mathbf{z}\rangle_0^A$ and $\langle\mathbf{z}\rangle_1^A$ to the mobile user. The mobile user then combines the shares and produces the inference result $\mathbf{z}$ labeling the class of user input.

3.2 Threat Model and Privacy Goals

We consider that the threats in Sonic mainly come from the engagement of the cloud servers that jointly provide the secure inference service. Following prior works in the two-server model [16], [19], [20], we assume that the cloud servers are semi-honest and non-colluding. They will honestly follow our protocol, yet attempt to infer private information beyond their access rights. The rationale behind such assumption is that cloud providers are well-established companies and not willing to harm their reputation, thus avoiding behaving maliciously and collusion [21]. It is noted that such multi-server model has become increasingly appealing in many industrial projects as well. Examples include privacy-preserving ML frameworks like Facebook's CrypTen [22] and Cape Privacy's TFEncrypted [23].

Sonic aims to ensure that both the private data of the mobile user and the (trained) parameters of the model are hidden from the cloud servers. Through interaction with the cloud servers, the mobile user only learns the inference result and nothing else. Consistent with prior works [11], [13], [19], [20], Sonic does not hide the architecture information (hyperparameters) of the model, such as dimension of weight tensor and number of layers. Last, like most prior works on secure inference [11], [12], [13], [19], [20], [24], Sonic does not aim to protect against adversarial machine learning attacks. Such attacks [7], [8], [25], [26] may attempt to exploit the inference procedure as a blackbox oracle to infer information about the training dataset or the model. Mitigating these adversarial ML attacks is orthogonal to Sonic's security scope, as per the definition of secure multiparty computation. Defenses against these attacks are complementary and active research areas, such as differentially private learning [27], [28], [29], [30], modification of NN models [25], [31], [32], and query auditing [33], [34]. We refer the readers to Section 7.3 for more detailed discussion.

4 Our Proposed Design

In this section, we present a series of secure layer functions as the main building blocks of secure NN inference in Sonic. Fig. 3 depicts the typical computational blocks in CNN and their secure counterparts, including the secure convolutional layer (SCONV), the secure batch normalization (SBN), the secure ReLU activation (SReLU), and the secure max pooling layer (SMP). These secure layers are organized in pipeline, where each layer receives an encrypted input and produces a desired encrypted output to the next layer. Sonic decomposes each layer into smaller and composable units (i.e., cryptographic gadgets) while preserving its security guarantees. Each unit is customized to be amiable for secret sharing techniques, so as to enable efficient and low-bandwidth execution of secure NN inference. For the most challenging ReLU and Max Pooling

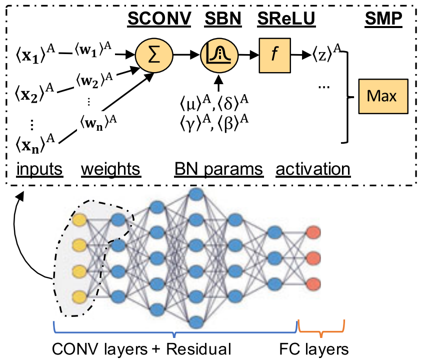


Fig. 3. Sonic's secure layer functions.

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

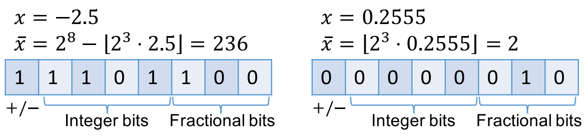


Fig. 4. Examples of the fixed-point representation for signed number, where $\ell = 8, q = 3$.

layers, we leverage insights from the digital circuit design literature and construct a highly efficient secure comparison gadget purely resort to lightweight boolean sharing techniques. With all these designs, our experiment results (Section 6) show that Sonic can achieve up to 60× bandwidth saving compared with prior art.

4.1 Setup

Sonic's secure designs proceed in the ring $\mathbb{Z}_{2^\ell}$. Initially, all mobile user input and model weights need to be converted to integers, so that they can be secretly shared over $\mathbb{Z}_{2^\ell}$. Sonic adopts the fixed-point representation to handle real-valued numbers. As Fig. 4 shows, each real-valued data x is rounded and scaled (with a factor q) into an integer $\bar{x} \bmod 2^\ell$, where a two's complement is used to represent the negative values. The most significant bit (MSB) indicates the sign (1→negative; 0→non-negative). Such a signed integer $\bar{x}$ can be shared over $\mathbb{Z}_{2^\ell}$, where the lower-half $[0, 2^{\ell-1} - 1]$ represents the non-negative values and the upper-half $[2^{\ell-1}, 2^\ell - 1]$ represents the negative values. Both data value and its sign are well protected. Multiplication over two fixed-point numbers can lead to a 2^{2q} scaling ($2q$ fractional bits) for the resulting product, which exceeds the bit-length ℓ of the ring $\mathbb{Z}_{2^\ell}$. To ensure the computation correct, all intermediate results should be scaled down by 2^q before proceeding the succeeding operation. Sonic adopts a secure local truncation scheme [19] which simply discards the last q-bit fractional part to adjust the product to ℓ bits. Similar treatment also appears in prior works [13], [35].

Based on this fixed-point representation, the client converts its task-specific raw input to a tensor $\mathbf{X}$ and produces additive shares of every data point. It then dispatches the corresponding shares to cloud servers S_0 and S_1 which then pad these shares with 0 to fit with the kernel size. The model owner produces additive shares of its CNN model tensor $\mathbf{W}$. To support efficient secure batch normal function, the model owner first derives two set of parameters $\epsilon_1 = \gamma/\delta, \epsilon_2 = \beta - \gamma\mu/\delta$ from the BN parameters: the running mean μ, the running variance δ, the scale γ, the shift β. It then produces additives shares of its CNN model tensor $\mathbf{W}$ (kernels and ϵ_1, ϵ_2). To this end, the model owner deploys the shares of weights to the cloud servers.

4.2 Secure Linear Layers

In this section, we present the secure realizations of linear layers, i.e., the secure convolutional layer (SCONV) and the secure fully connected layer (SFC). Their functionality can be expressed as

$$z = f(\mathbf{x}, \mathbf{w}) + bias;$$
$$f(\mathbf{x}, \mathbf{w}) = \mathsf{VDP}(\mathbf{x}, \mathbf{w}) = \Sigma_{k=1}^{n} w_k \cdot x_k, \tag{1}$$

which work over the n-dimensional layer input vector $\mathbf{x}$ (or the activation vector for hidden layers) and weight vector $\mathbf{w}$, plus the *bias* attached on each neuron. In Sonic, we introduce the main building block, i.e., the secure VDP function (SVDP), to realize the above Eq. (1) in the secret sharing domain. Here, we make an important observation from the known literature [36] and open source framework [37] that the bias can be removed if applying batch normalization, because the shift β in BN achieves the same effect as the bias. Likewise, we set the bias as 0 to avoid the involvement of real-valued bias and make our design more compatible with the secret sharing techniques.

Secure VDP. The secure VDP function (SVDP) computes the convolution operation for conventional CNNs in Eq. (1). It takes as input a set of secret shares of input vector as $\langle \mathbf{x} \rangle_i^A$ and weight vector as $\langle \mathbf{w} \rangle_i^A$, respectively, and outputs the secret shares of their VDP result $\langle z \rangle_i^A$, where $i \in \{0, 1\}$ is the identifier of the two cloud servers S_0, S_1. The vector $\mathbf{x}$ can be the activation output from previous ReLU function or represent the raw input of user. The two cloud servers S_0 and S_1 perform $\mathsf{SecVDP}(\langle \mathbf{x} \rangle_i^A, \langle \mathbf{w} \rangle_i^A)$ as follows:

1) For $k \in [1, n]$, S_0 and S_1 jointly compute the element-wise multiplication $\langle z_k \rangle^A = \langle x_k \rangle^A \cdot \langle w_k \rangle^A$.
2) S_i locally sums the products $\langle z \rangle_i^A = \sum_{k=1}^{n} \langle z_k \rangle_i^A$.

4.3 Secure Batch Normalization

Batch normalization [36] performs element-wise normalization on each neuron's feature a via

$$\hat{a} = (a - \mu)/\delta, z = \gamma \cdot \hat{a} + \beta, \tag{2}$$

where μ is the *running mean* and δ is the non-zero *running variance* of the training dataset; and γ, β are the *scale* and *shift* parameters. The secure batch normalization function (SBN) is applied on each neuron after the secure linear function. It takes as input the secret shares of feature $\langle x \rangle^A$ outputted from linear function, and outputs the shares of the normalized feature $\langle z \rangle^A$. We observe that its functionality in Eq. (2) can be transformed to a simpler problem

$$z = \epsilon_1 \cdot x + \epsilon_2,$$
$$\text{where } \epsilon_1 = \frac{\gamma}{\delta}, \epsilon_2 = \beta - \frac{\gamma\mu}{\delta}. \tag{3}$$

Since ϵ_1, ϵ_2 are derived from the BN parameters and independent of inference input, they can be *pre-calculated* by model owner in the setup phase. With preprocessing, only Eq. (3) is required to be calculated. Such a conversion circumvents the complex division operations over secret shares in Eq. (2), resulting in a functionality more compatible to secret sharing techniques. With above observation, given the pre-generated shares of two parameters $\langle \epsilon_1 \rangle^A, \langle \epsilon_2 \rangle^A$, the $\mathsf{SBN}(\langle x \rangle^A, \langle \epsilon_1 \rangle^A, \langle \epsilon_2 \rangle^A)$ performs as follows: S_0 and S_1 jointly compute the normalized feature on each neuron $\langle z \rangle^A = \langle x \rangle^A \cdot \langle \epsilon_1 \rangle^A + \langle \epsilon_2 \rangle^A$.

4.4 Secure ReLU Function

The secure ReLU function (SReLU) is applied on each neuron and proceeds with the functionality $\mathsf{ReLU}(x) = max(x, 0)$ in the encrypted domain. Prior work solving the problem is either relying on the heavyweight garbled circuit techniques [11], [19] or utilizing the linear activation functions (e.g., square function) [12], [13], [38]. In essence, the former requires intensive communication cost that is

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

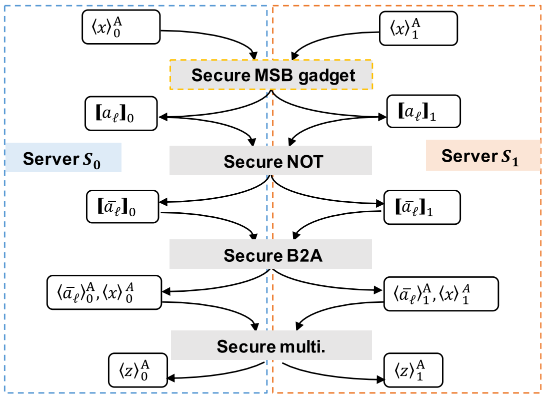


Fig. 5. An overview of the SReLU function.

unsuitable for real-world mobile application deployment. Using a linear activation could be problematic and violates the original intention to apply activation function, i.e., introducing non-linearity. This is because an NN with all linear functions has limited power to handle complex inference tasks and makes the training process hard to converge during backpropagation [39].

Overview. In Sonic, we rely on a new highly efficient realization of the secure ReLU function in the secret sharing domain. We first note that the MSB indicates the sign bit with the fixed-point representation, which would be one of a non-negative feature x, and would be 0 of a negative x. We then transform the ReLU function into an MSB extraction:

$$max(x,0) \xrightarrow{\text{trans}} \neg\mathsf{MSB}(x)\cdot x = \begin{cases} 1\cdot x & \text{if } x \geq 0 \\ 0\cdot x & \text{if } x < 0 \end{cases}. \tag{4}$$

Based on Eq. (4), we decompose the SReLU function into four atomic operations. As summarized in Fig. 5, it comprises the *secure* MSB$(\cdot)$ *gadget*, the *secure NOT gadget*, the *secure* B2A$(\cdot)$ *gadget*, and the *secure multiplication gadget*. The *secure* MSB *gadget* takes as input the arithmetic shares of normalized feature $\langle x\rangle^A$, and extracts the boolean-shared MSB $[\![a_\ell]\!]$. The *secure NOT* conducts the bitwise-NOT to produce the shares of NOT MSB $[\![\bar{a}_\ell]\!]$.

Prior to multiplying with normalized feature x, the *secure* B2A *gadget* needs to be applied, converting the boolean shared NOT MSB $[\![\bar{a}_\ell]\!]$ to its arithmetic form $\langle\bar{a}_\ell\rangle^A$. The reason is that $[\![\bar{a}_\ell]\!]$ is projected to ring $\mathbb{Z}_2$ via $[\![\bar{a}_\ell]\!]_0 + [\![\bar{a}_\ell]\!]_1 \pmod 2$, whereas the arithmetic-shared feature $\langle x\rangle^A$ is projected to $\mathbb{Z}_{2^\ell}$ via $\bmod\ 2^\ell$. These two shares cannot be naively multiplied with different moduli. The last *secure multiplication* produces the shares of ReLU result $\langle z\rangle^A$, given the shared NOT MSB and the feature $\langle x\rangle^A$.

Below we expatiate on the essential cryptographic gadgets of secure ReLU function in Sonic: the *secure* MSB *gadget* and the *secure* B2A *gadget*.

4.4.1 The Construction of Secure MSB Extraction

The secure MSB$(\cdot)$ gadget is used to securely extract the MSB of the arithmetic-shared data and to produce a boolean-shared MSB as the output. Suppose there are secret shares $\langle x\rangle_0^A$ and $\langle x\rangle_1^A$ of a secret value x, with the bit length ℓ. Let $x = \{x_\ell,\ldots,x_1\}$, $a = \{a_\ell,\ldots,a_1\}$, and $b = \{b_\ell,\ldots,b_1\}$ denote $x, \langle x\rangle_0^A$, $\langle x\rangle_1^A$ with their corresponding bit strings

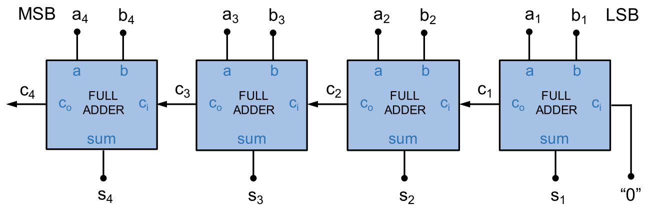


Fig. 6. An example of 4-bit ripple carry adder.

respectively, such that $x = a + b \pmod{2^\ell}$. The key observation is that the difference between the sum (+) of bit strings of a, b and the bitwise-XOR ($\oplus$) of the bit strings of a, b is equal to the carry bits $c_\ell,\ldots,c_1$. For example, given $x = 13, a = 6, b = 7$, the carry bits ("0110") are equal to bit strings of $(a+b)$ ("1101") XOR with $a \oplus b$ ("1011"). Then extracting MSB $x_\ell = c_\ell + a_\ell + b_\ell$ is converted to calculating the carry bit c_ℓ via an ℓ-bit full adder.

To do so, an effective and notable approach [16] is to use the ripple carry adder (RCA) to implement the ℓ-bit full adder in serial. As Fig. 6 demonstrates, the fan-out carry bit (c_o) of each full adder is propagated as fan-in (c_i) of the succeeding full adder. This serial implementation introduces an $\mathcal{O}(\ell)$ propagation delay, resulting in 2ℓ rounds of communication (computing AND operations over boolean shares) between the two cloud servers in the secret sharing domain. The linear round complexity could result in long processing latency when the two cloud servers are geographically separated and the network delay is high.

Instead, we observe an efficient realization of full adder logic from the digital circuit design literature [40], called the parallel prefix adder (PPA). In comparison to RCA, the PPA can extract the MSB in logarithm communication rounds $\mathcal{O}(\log\ell)$. We note that leveraging parallel adder (the tree-based PPA and its other variants) to efficiently realize the secure MSB extraction has also been explored in independent and concurrent works [41], [42], [43], under different contexts.

The usage of the PPA for secure MSB extraction in Sonic is introduced as follows. In the PPA, a signal tuple (G_i, P_i) can be pre-generated in parallel via

$$G_i = a_i\cdot b_i, P_i = a_i + b_i, \tag{5}$$

where G_i is called the carry generate signal and P_i is called the carry propagate signal. Given these two prefixes, PPA reformulates the full adder $c_{i+1} = (a_i\cdot b_i) + c_i\cdot(a_i+b_i)$ to compute the carry bit via $c_{i+1} = G_i + (c_i\cdot P_i)$. This reformulation allows a carry to be derived as follows:

$$c_\ell = G_{\ell-1} + (P_{\ell-1}\cdot G_{\ell-2}) + \cdots + (P_{\ell-1}\ldots P_2\cdot G_1), \tag{6}$$

without waiting for the previous carry propagated through all previous adders. Afterwards, PPA properly organizes the computation of Eq. (6) in parallel to reduce the latency in $\mathcal{O}(\log\ell)$ rounds. As demonstrated in Fig. 7, PPA forms a $\log\ell$-layer binary tree and attaches a binary operator $\diamond$ on each node. This binary operator is defined as: given two carry generate signal and propagate signal tuples $(G_{in_1}, P_{in_1}); (G_{in_2}, P_{in_2})$, and output signal tuple (G_{out}, P_{out}), it performs

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

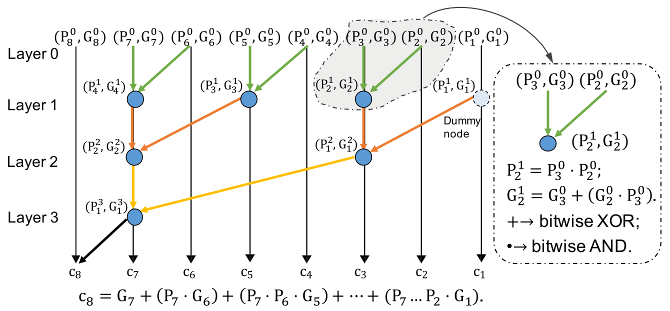

(P_8^0, G_8^0) (P_7^0, G_7^0) (P_6^0, G_6^0) (P_5^0, G_5^0) (P_4^0, G_4^0) (P_3^0, G_3^0) (P_2^0, G_2^0) (P_1^0, G_1^0)
Layer 0
Layer 1
Layer 2
Layer 3
(P_4^1, G_4^1)
(P_3^1, G_3^1)
(P_2^1, G_2^1)
(P_1^1, G_1^1)
Dummy node
(P_2^2, G_2^2)
(P_1^2, G_1^2)
(P_1^3, G_1^3)
c_8 c_7 c_6 c_5 c_4 c_3 c_2 c_1
(P_3^0, G_3^0) (P_2^0, G_2^0)
(P_2^1, G_2^1)
$P_2^1 = P_3^0 \cdot P_2^0$;
$G_2^1 = G_3^0 + (G_2^0 \cdot P_3^0)$.
$+ \rightarrow$ bitwise XOR;
$\bullet \rightarrow$ bitwise AND.
$c_8 = G_7 + (P_7 \cdot G_6) + (P_7 \cdot P_6 \cdot G_5) + \cdots + (P_7 \ldots P_2 \cdot G_1)$.

Fig. 7. An example of 8-bit parallel prefix adder.

$$
\begin{aligned}
&(G_{out}, P_{out}) = (G_{in_1}, P_{in_1}) \diamond (G_{in_2}, P_{in_2}) \\
&G_{out} = G_{in_2} + G_{in_1} \cdot P_{in_2} \\
&P_{out} = P_{in_2} \cdot P_{in_1}.
\end{aligned} \tag{7}
$$

PPA recursively computes the above binary operation over the signal tuples at the leaf nodes (layer 0), and propagates the resulting tuples as input to the next layer's nodes, until reaching the root node (the node corresponding to tuple (P_1^3, G_1^3) in Fig. 7). At the end, the most significant carry bit c_ℓ can be produced via Eq. (6) and the MSB is calculated by $x_\ell = c_\ell + a_\ell + b_\ell$. Based on above philosophy, the details of the secure MSB$(\cdot)$ gadget are presented in Fig. 8.

```
Input: Arithmetic shares of integer feature $\langle x \rangle^A \in \mathbb{Z}_{2^\ell}$.
Output: Boolean shares of MSB $[\![x_\ell]\!] \in \mathbb{Z}_2$.
Decompose $x$ to bit strings:
1) $S_0$ decomposes $\langle x \rangle_0^A$ to a bit string $a_\ell, \ldots, a_1$;
   $S_1$ decomposes $\langle x \rangle_1^A$ to a bit string $b_\ell, \ldots, b_1$;
2) For each $k \in [1, \ell]$:
   $S_0$ sets $[\![a_k]\!]_0 = a_k$, $[\![b_k]\!]_0 = 0$, $[\![w_k]\!]_0 = a_k$;
   $S_1$ sets $[\![a_k]\!]_1 = 0$, $[\![b_k]\!]_1 = b_k$, $[\![w_k]\!]_1 = b_k$;
Compute signal tuples $(G, P)$ in Eq. 5:
3) For each $k \in [1, \ell]$:
   $S_0$ and $S_1$ set $[\![G_k]\!] = [\![a_k]\!] \cdot [\![b_k]\!]$, $[\![P_k]\!] = [\![a_k]\!] \oplus [\![b_k]\!]$;
Compute PPA tree based on Eq. 7:
4) Layer $L = 0$
   For $k \in [1, \ell]$, $S_i$ sets $([\![G_k^0]\!], [\![P_k^0]\!]) = ([\![G_k]\!], [\![P_k]\!])$.
5) Layer $L = 1$
   $S_i$ sets dummy node $([\![G_k^1]\!], [\![P_k^1]\!]) = ([\![G_k^0]\!], [\![P_k^0]\!])$;
   For $k \in [2, \ell/2]$:
   Let $in_1 = 2k - 2$, $in_2 = 2k - 1$;
   $([\![G_k^1]\!], [\![P_k^1]\!]) = ([\![G_{in_1}^0]\!], [\![P_{in_1}^0]\!]) \diamond ([\![G_{in_2}^0]\!], [\![P_{in_2}^0]\!])$;
6) Layer $L = 2, \ldots, \log \ell$
   For $k \in [1, \ell/2^L]$:
   Let $in_1 = 2k - 1$, $in_2 = 2k$;
   $([\![G_k^L]\!], [\![P_k^L]\!]) = ([\![G_{in_1}^{L-1}]\!], [\![P_{in_1}^{L-1}]\!]) \diamond ([\![G_{in_2}^{L-1}]\!], [\![P_{in_2}^{L-1}]\!])$;
Compute MSB:
7) $S_0$ and $S_1$ set $[\![c_\ell]\!] = [\![G_1^{\log \ell}]\!]$, $[\![x_\ell]\!] = [\![w_\ell]\!] + [\![c_\ell]\!]$;
```

Fig. 8. The secure MSB$(\cdot)$ gadget.

4.4.2 Secure B2A Gadget

The secure B2A gadget converts a boolean-shared data $[\![x]\!]$ into its arithmetic share $\langle x \rangle^A$. Given two cloud servers S_0, S_1, the secure B2A$([\![x]\!])$ gadget performs as follow:

1) S_0 sets two variables $\langle a \rangle_0^A = [\![x]\!]_0$, $\langle b \rangle_0^A = 0$;
2) S_1 sets two variables $\langle a \rangle_1^A = 0$, $\langle b \rangle_1^A = [\![x]\!]_1$;
3) S_0 and S_1 set $\langle x \rangle^A = \langle a \rangle^A + \langle b \rangle^A - 2 \cdot \langle a \rangle^A \cdot \langle b \rangle^A$.

4.4.3 Realization of Secure ReLU Function

Given above secure MSB gadget and secure B2A gadget, and the arithmetic share of input feature $\langle x \rangle^A$, the secure ReLU function SReLU$(\langle x \rangle^A)$ performs as follows:

1) *Secure MSB extraction:* S_0 and S_1 run to get the MSB $[\![a_\ell]\!] \leftarrow$ MSB$(\langle x \rangle^A)$.
2) *Secure NOT:* S_i sets the NOT MSB $[\![\bar{a}_\ell]\!] = [\![a_\ell]\!] + i$.
3) *Secure B2A:* S_0 and S_1 run to get the arithmetic-shared NOT MSB $\langle \bar{a}_\ell \rangle^A \leftarrow$ B2A$([\![\bar{a}_\ell]\!])$.
4) *Secure multiplication:* S_0 and S_1 set $\langle z \rangle^A = \langle \bar{a}_\ell \rangle^A \cdot \langle x \rangle^A$.

4.5 Secure Max Pooling Layer

The max pooling layer is used to down sample the features by choosing the maximum values within a certain sliding window. It is typically applied after the ReLU function. The secure max pooling layer (SMP) function in Sonic securely realizes the $\max(x_1, \ldots, x_n)$ functionality over secret shares. In particular, we transform the pairwise maximum operation into the secure comparison with MSB extraction based on Eq. (8) and a secure linear branching based on Eq. (9) via

$$
b = \text{MSB}(a_1 - a_2) = \begin{cases} 0 & \text{if } a_1 \geq a_2 \\ 1 & \text{if } a_1 < a_2 \end{cases}; \tag{8}
$$

$$
\max(a_1, a_2) = (1 - b) \cdot a_1 + b \cdot a_2. \tag{9}
$$

With above insights in mind, we provide the details of the SMP realization. Given the secure MSB$(\cdot)$ gadget and secure B2A$(\cdot)$ gadget, and the secret shares of input features within each max pooling window $\langle x_1 \rangle^A, \ldots, \langle x_n \rangle^A$, the SMP$(\langle x_1 \rangle^A, \ldots, \langle x_n \rangle^A)$ performs as follows:

1) For $k \in [1, n-1]$:
2) S_i sets $\langle a_1 \rangle_i^A = \langle x_k \rangle_i^A$, $\langle a_2 \rangle_i^A = \langle x_{k+1} \rangle_i^A$.
3) *Secure comparison:* S_0 and S_1 run to get the comparison bit $[\![b]\!] \leftarrow$ MSB$(\langle a_1 \rangle^A - \langle a_2 \rangle^A)$.
4) *Secure B2A:* S_0 and S_1 run to get $\langle b \rangle^A \leftarrow$ B2A$([\![b]\!])$.
5) *Secure branching:* S_0 and S_1 set $\langle z \rangle^A = (i - \langle b \rangle^A) \cdot \langle a_1 \rangle^A + \langle b \rangle^A \cdot \langle a_2 \rangle^A$. S_i sets $\langle x_{k+1} \rangle_i^A = \langle z \rangle_i^A$.

4.5.1 Optimized Execution of ReLU and Max Pooling

We observe that reversing the execution order of ReLU and max pooling, i.e., maxpool(ReLU$(\cdot)$) $\rightarrow$ ReLU(maxpool$(\cdot)$) can greatly reduce the considerable workload to perform ReLU (75% saving on ReLU operations for a 2×2 max pooling window) [44]. This transformation is defined as

$$
\max(\max(x_1, 0), \ldots, \max(x_n, 0)) \xrightarrow{\text{transform}} \max(\max(x_1, \ldots, x_n), 0),
$$

where $x_1, \ldots x_n$ are the input features within the n-width pooling window. Such a conversion will reduce the number of secure comparison operations in the secret sharing domain and can significantly save the overall workloads. Because computing the non-polynomial ReLU over secret shares is knowingly complicated, the involved multiple interactions could lead to long processing latency.

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

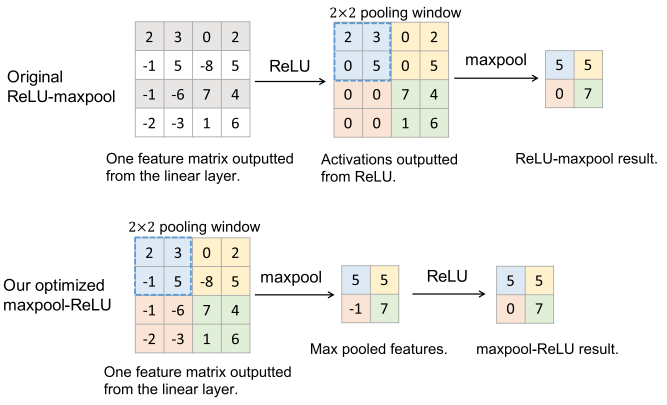


Fig. 9. An illustration of the optimized execution of ReLU and max pooling layer with typical 2×2 pooling window.

Consider a typical max pooling layer with 2×2 pooling window. In principle, a max pooling layer is applied directly after the ReLU layer, where the 4 inputs within the 2×2 pooling window are the outputs of 4 ReLU operations. Then, it performs a series of comparisons as follows:

$$\begin{aligned}&\mathrm{maxpool}(\mathrm{ReLU}(x_1), \mathrm{ReLU}(x_2), \mathrm{ReLU}(x_3), \mathrm{ReLU}(x_4))\\ =&\max(\max(x_1,0), \max(x_2,0), \max(x_3,0), \max(x_4,0)). \qquad (10)\end{aligned}$$

Since ReLU is monotonic (i.e., if $x_1 > x_2, \mathrm{ReLU}(x_1) >= \mathrm{ReLU}(x_2)$), the Eq. (10) is identical to

$$\begin{aligned}&\max(\max(x_1, x_2, x_3, x_4), 0)\\ &= \mathrm{ReLU}(\mathrm{maxpool}(x_1, x_2, x_3, x_4), 0), \qquad (11)\end{aligned}$$

where the internal maximum operation $\max(x_1, x_2, x_3, x_4)$ is regarded as the max pooling layer and the out-layer maximum operation $\max(\cdot, 0)$ is deemed as the ReLU activation. Through such a transformation, 75% of ReLU operations are saved given a typical max pooling with 2×2 window. A concrete illustration is given in Fig. 9

4.6 Complexity Analysis

We now provide an analysis of the complexity of Sonic's secure layer functions and secure gadgets regarding the number of communication rounds and the communication overhead. Table 2 summarizes the theoretical communication costs of non-linear layer functions (SReLU and SMP) and corresponding secure gadgets (secure MSB, secure B2A, and secure branching). We set the benchmark as the RCA based realization [16] for secure MSB. For SReLU and SMP, we set the benchmarks as the realizations based on RCA and GC [45], [46]. As defined, the bit length of an additive secret share is ℓ, and the max pooling window size is n (e.g., $n = 4$ if the pooling window is 2×2). Sonic's PPA based secure MSB gadget requires the communication as follows:

$$\begin{aligned}&4\ell + \left(\frac{\ell}{2} - 1\right)\cdot 8 + \frac{\ell}{2^2}\cdot 8 + \cdots + \frac{\ell}{2^{\log \ell}}\cdot 8\\ &= 4\ell - 8 + 8\ell \cdot \left(\frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^{\log \ell}}\right)\\ &= 12\ell - 16.\end{aligned}$$

For the GC benchmarks, the communication costs are summarized based on the ReLU and max pooling circuits shown in prior work [45], [46] which consist of the secure conversions between additive shares to Yao's shares and corresponding ReLU and max pooling functionalities. Note that the GC protocols have constant round complexity $\mathcal{O}(1)$ as the concrete number of communication rounds depending on the specific realizations. As shown, our PPA based secure MSB, SReLU, and SMP have logarithmic round trip costs to the RCA based realizations with the same communication costs. Meanwhile, our SReLU and SMP substantially save the communication than the GC based realizations.

We also analyze the communication complexity of secure linear layers (SCONV, SFC and SBN) and corresponding secure gadget (SVDP) in Table 3. Suppose a typical stride is 1. Suppose the number of padding p, the sizes of input tensor, kernel, and output tensor of secure convolutional layer are $c^{in} \times n^{in} \times n^{in}$, $c^{in} \times c^{out} \times n \times n$, and $c^{out} \times n^{out} \times n^{out}$ respectively; and the sizes of input vector and output vector of the fully-connected layer are c^{in}, c^{out}, respectively.

5 Security Analysis

Sonic properly encrypts the user input, the CNN model (i.e., the weights), and any intermediate results as secret shares based on the standard secret sharing techniques [14], [47]. During Sonic's execution, the two non-colluding cloud servers receive only their corresponding secret shares that are uniformly distributed randomnesses and reveals nothing about private data. Each cloud proceeds Sonic's secure layer functions on its local shares independently. Any interactions between the two cloud servers are supported by the standard Beaver's triples [15] that can offer

TABLE 2
Communication Complexity Analysis of Secure Non-Linear Layers

Secure Function	Benchmarks	Communication	Round Complexity
Secure MSB	RCA	$12\ell - 16$	$\mathcal{O}(\ell)$
	Sonic	$12\ell - 16$	$\mathcal{O}(\log \ell)$
Secure B2A	Sonic	4ℓ	$\mathcal{O}(1)$
Secure branching	Sonic	8ℓ	$\mathcal{O}(1)$
SReLU	GC [45], [46]	$30\ell\kappa$	$\mathcal{O}(1)$
	RCA	$20\ell - 16$	$\ell + 2$
	Sonic	$20\ell - 16$	$\log \ell + 2$
SMP	GC [45], [46]	$(n+7)3\ell\kappa$	$\mathcal{O}(1)$
	RCA	$(n-1)(24\ell - 16)$	$(n-1)(\ell + 2)$
	Sonic	$(n-1)(24\ell - 16)$	$(n-1)(\log \ell + 2)$

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

TABLE 3
Communication Complexity Analysis of Secure Linear Layers

Secure Function	SVDP	SCONV	SFC	SBN
Comm.	$4\ell n$	$c^{in}c^{out}(n^{in}-n+p+1)^2 4\ell n$	$c^{in}c^{out}4\ell n$	4ℓ
# of Rounds	n	$c^{out}(n^{in}-n+p+1)^2 n$	$c^{out}n$	1

provably security guarantees on the exchanged secret shares.

Formally, we define an ideal functionality $\mathcal{F}_{\mathsf{ONI}}$ for securely proceeding an outsourced neural network inference protocol targeted in this paper. On top of this ideal functionality, we formally present the security definition and prove that `Sonic`'s protocol Π securely realizes this functionality in the ideal/real world simulation paradigm. We consider an adversary $\mathcal{A}$ to be semi-honest and can statically compromise the model owner, the mobile user, or anyone of the two cloud servers S_0, S_1 independently and so to capture the non-colluding property. In the ideal world, parties directly input and interact with the trusted functionality $\mathcal{F}_{\mathsf{ONI}}$ which executes the entire inference computation faithfully on behalf of the parties. In the real world, parties directly interact with `Sonic`'s protocol Π in the presence of above adversary $\mathcal{A}$. We say that protocol Π securely realizes $\mathcal{F}_{\mathsf{ONI}}$ if for every adversary $\mathcal{A}$ from the real world execution of Π, there exists a probabilistic polynomial time (PPT) simulator $\mathcal{S}$ in the ideal world forging $\mathcal{A}$'s view such that $\mathcal{A}$ cannot distinguish the two scenarios.

We start with modeling the ideal functionality $\mathcal{F}_{\mathsf{ONI}}$.

Definition 1. *The ideal functionality $\mathcal{F}_{\mathsf{ONI}}$ of secure neural network inference outsourced to the cloud is modeled as follows:*

- Input. *The model owner $\mathcal{O}$ deploys the pre-trained CNN model $\mathbf{W}$ to $\mathcal{F}_{\mathsf{ONI}}$. The mobile user $\mathcal{U}$ submits the user input $\mathbf{X}$ to $\mathcal{F}_{\mathsf{ONI}}$. The two servers S_0 and S_1 inputs nothing to $\mathcal{F}_{\mathsf{ONI}}$.*
- Computation. *Upon receiving the model $\mathbf{W}$ from the model owner and the user input $\mathbf{X}$ from the mobile user, $\mathcal{F}_{\mathsf{ONI}}$ interacts in hybrid with the ideal functionalities of subroutines (i.e., layer-protocols) $\mathcal{F}_{\mathsf{SCONV}}$, $\mathcal{F}_{\mathsf{SBN}}$, $\mathcal{F}_{\mathsf{SReLU}}$, and $\mathcal{F}_{\mathsf{SMP}}$ to generate the prediction $\mathbf{z}$.*
- Output. *The $\mathcal{F}_{\mathsf{ONI}}$ outputs the prediction $\mathbf{z}$ to the mobile user and nothing to the model owner.*

We define the security guarantees.

Definition 2. *Protocol Π is said to securely evaluate the functionality $\mathcal{F}_{\mathsf{ONI}}$ in the presence of semi-honest adversaries in a static corruption, if for every PPT adversary $\mathcal{A}$ for the real model, there exits a PPT simulator $\mathcal{S}$ for the ideal model, such that for every party $\mathcal{P} \subset [\mathcal{O}, \mathcal{U}, S_0, S_1]$*

$$\{\textsc{Ideal}_{\mathcal{F}_{\mathsf{ONI}},\mathcal{P},\mathcal{S}(\bar{X})}(\mathbf{X},\mathbf{W})\}_{\mathbf{X},\mathbf{W},\bar{X}} \overset{c}{\equiv} \{\textsc{Real}_{\Pi,\mathcal{P},\mathcal{A}(\bar{X})}(\mathbf{X},\mathbf{W})\}_{\mathbf{X},\mathbf{W},\bar{X}},$$

where $\textsc{Ideal}_{\mathcal{F}_{\mathsf{ONI}},\mathcal{P},\mathcal{S}(\bar{X})}(\mathbf{X},\mathbf{W})$ is the output of joint execution of $\mathcal{F}_{\mathsf{ONI}}$ in the ideal world, $\textsc{Real}_{\Pi,\mathcal{P},\mathcal{S}(\bar{X})}(\mathbf{X},\mathbf{W})$ is the output of and $\bar{X}$ denotes the auxiliary input. We consider the following cases:

- Corrupted model owner. *A corrupted model owner is required to learn no private information about the values of user input $\mathbf{X}$. In this case, there should have a simulator $\mathcal{S}$ that interacts with $\mathcal{O}$ and extracts the model inputted from $\mathcal{A}$ to honest parties S_0, S_1. $\mathcal{O}$ cannot distinguish from the transcripts whether interacting with $\mathcal{S}$ from the ideal execution of $\mathcal{F}_{\mathsf{ONI}}$ or from the real execution of Π.*
- Corrupted mobile user. *A corrupted mobile user is required to learn no private information about the values of model weights and coefficients $\mathbf{W}$ beyond the hyper-parameters made public available. In this case, there should have a simulator $\mathcal{S}$ that interacts with $\mathcal{U}$, extracts the user input $\mathbf{X}$ inputted from $\mathcal{U}$ to honest parties S_0, S_1, and emulates the output $\mathbf{z}$. The corrupted mobile user $\mathcal{U}$ cannot distinguish from the transcripts whether interacting with $\mathcal{S}$ from the ideal execution of $\mathcal{F}_{\mathsf{ONI}}$ or from the real execution of Π.*
- Corrupted cloud server. *A corrupted cloud server $S_i \in [S_0, S_1]$ is required to learn no private information about the values of user input $\mathbf{X}$, model weights and coefficients $\mathbf{W}$ beyond the hyper-parameters. In this case, there should has a simulator $\mathcal{S}$ that interacts with the corrupted S_i and emulates the exact transcripts for interactions between the adversary S_i and honest parties $\mathcal{O}, \mathcal{U}, S_{1-i}$. The corrupted cloud server S_i cannot distinguish from the transcripts whether interacting with $\mathcal{S}$ from the ideal execution of $\mathcal{F}_{\mathsf{ONI}}$ or from the real execution of Π.*

Theorem 1. *`Sonic`'s protocol Π securely realizes $\mathcal{F}_{\mathsf{ONI}}$ (per Definition 2) in the presence of one semi-honest adversary $\mathcal{A}$ in the ($\mathcal{F}_{\mathsf{SCONV}}$, $\mathcal{F}_{\mathsf{SBN}}$, $\mathcal{F}_{\mathsf{SReLU}}$, $\mathcal{F}_{\mathsf{SMP}}$)-hybrid model.*

Proof. We begin by defining the simulator $\mathcal{S}$ based on the corrupted party. The simulator should be able to emulate the transcripts (including the joint distribution of the inputs and outputs) that are computationally indistinguishable from the ones in ideal world execution. That is, the transcripts during real interaction are exactly simulated and the honest parties should learn the correct outputs. In the hybrid argument, the executions of the subroutines (layer-protocols) are simulated by $\mathcal{S}$ which plays the roles of honest parties in corresponding ideal functionalities. We construct $\mathcal{S}$ for the following corruption parties:

- *Simulator for the corrupted model owner.* The only work where the corrupted model owner $\mathcal{O}$ engaged in protocol Π is inputting the secret-shared NN model $\mathbf{W}$. The simulator $\mathcal{S}$ plays the role of the corrupted model owner $\mathcal{O}$ in the ideal world by submitting the model $\mathbf{W}$ of $\mathcal{O}$ to the ideal functionality $\mathcal{F}_{\mathsf{ONI}}$. The remaining protocol Π runs by the honest two cloud servers by sequentially composing the ideal functionalities of subroutines. At the end of Π, only the shares of correct result $\mathbf{z}$

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

TABLE 4
Time and Bandwidth of Atomic Layer Functions

	SCONV		SBN	SReLU	SMP
	3×3	5×5			2×2
Time (ms)	5.04	7.07	15.35	22.3	43.1
Comm. (KB)	0.14	0.39	0.016	0.076	0.275

TABLE 5
Time and Bandwidth of the SFC Layer

Layer Size ($n \times n$)	64	128	256	512	1024
Time (ms)	0.12	0.19	0.234	0.35	0.65
Comm. (MB)	0.16	0.42	1.24	4.46	17.33

is returned to the honest mobile user $\mathcal{U}$ so to achieve the correctness, and nothing is outputted to $\mathcal{O}$. To emulate the view of $\mathcal{O}$, the simulator $\mathcal{S}$ simply outputs the model $\mathbf{W}$ in a dummy way. The $\mathcal{S}$'s output (i.e., $\mathcal{S}(\mathbf{W})$) is identically distributed to the view of $\mathcal{O}$ (i.e., $\mathsf{View}^{\Pi}_{\mathcal{U}}(\mathbf{W})$), and thus the real and ideal worlds are indistinguishable.

– *Simulator for the corrupted mobile user.* The corrupted mobile user $\mathcal{U}$ engages in protocol Π by submitting the secret shares of user input $\mathbf{X}$, and receives the two shares $\langle \mathbf{z} \rangle^A_0, \langle \mathbf{z} \rangle^A_1$ of inference result $\mathbf{z}$. The simulator $\mathcal{S}$ plays the role of the corrupted mobile user $\mathcal{U}$ in the ideal world by submitting the user input $\mathbf{X}$ to $\mathcal{F}_{\mathsf{ONI}}$. Upon receiving the inference result $\mathbf{z}$ from $\mathcal{F}_{\mathsf{ONI}}$, $\mathcal{S}$ generates a random value as $\langle \mathbf{z}^* \rangle^A_0 \in_R \mathbb{Z}_{2^\ell}$ and computes $\langle \mathbf{z}^* \rangle^A_1 = \mathbf{z} - \langle \mathbf{z}^* \rangle^A_0$. Then $\mathcal{S}$ outputs $\langle \mathbf{z}^* \rangle^A_0, \langle \mathbf{z}^* \rangle^A_1$. The security follows from the fact that the additive secret shares $\langle \mathbf{z}^* \rangle^A_0, \langle \mathbf{z}^* \rangle^A_1$ are uniformly random values in $\mathbb{Z}_{2^\ell}$, and are identically distributed to the transcripts from real world. Meanwhile, the result $\mathbf{z}$ can be reconstructed via $\langle \mathbf{z}^* \rangle^A_0 + \langle \mathbf{z}^* \rangle^A_1 \pmod{2^\ell}$. The output $\mathcal{S}(\mathbf{X}, \mathbf{z})$ and the view of $\mathcal{U}$ (i.e., $\mathsf{View}^{\Pi}_{\mathcal{U}}(\mathbf{X}, \mathbf{z})$) are identically distributed, and thus the two worlds are thus indistinguishable.

– *Simulator for a corrupted cloud server.* The two cloud servers S_0 and S_1 jointly proceed protocol Π in a symmetric way without submitting any inputs and receiving the final result. That is, they compute and interact with each other over all secret shares in different layers of Π. Given their symmetry, it is sufficient to construct a simulator $\mathcal{S}$ for one of the server, say S_0. In the subsequent protocol, $\mathcal{S}$ sequentially composes the simulators of subroutines, which play the role of honest cloud server S_1 to interact with S_0. All messages involved in the subroutines are random shares of matrices with the dimensions are consistent to the honest server's shares. The interactions between two servers are simulated by the simulator $\mathcal{S}_{\mathsf{BTri}}$ for the functionality of Beaver's multiplication $\mathcal{F}_{\mathsf{BTri}}$, which interacts with S_0 over random shares. The security follows from the fact that all transcripts are uniformly distributed secret shares that can be emulated with random values in $\mathbb{Z}_{2^\ell}$, and the view of S_0 in every interaction in Π is indistinguishable from the view emulated by $\mathcal{S}_{\mathsf{BTri}}$. Hence, we argue that $\mathcal{S}$ in the hybrid model can emulate the view of S_0, and thus the two worlds are indistinguishable. The above concludes the proof of Theorem 1. □

6 Performance Evaluation

6.1 System Implementation

We implement a prototype of `Sonic` in Java. Evaluations are executed on two servers, running CentOS 7 and equipped with Intel Xeon Gold 6150 CPU @2.7GHz processor, 192GB RAM, Mellanox Spectrum network. The reported measurements use the MNIST and CIFAR-10 machine learning benchmarking datasets with four CNNs, where M1, M2 are trained on MNIST and C1, C2 are trained on CIFAR-10. We refer the readers to Section 6.6 for the details of the model architectures. Consistent with prior art [11], [24], we evaluate the benchmarks in the LAN setting.

`Sonic`'s secure NN inference works in the secret sharing domain, where all data are represented in fixed-point integers and shared over ring $\mathbb{Z}_{2^\ell}$. In essence, we set the ring size as $\mathbb{Z}_{2^{32}}$. This setting of ring is consistent with prior work [13]. To present the weights to 32-bit fixed-point integers, we set the scaling factor q as 1024, 128, 64, and 128 for M1, M2, C1, and C2, respectively. Our implementation can further seamlessly support the 64-bit ring $\mathbb{Z}_{2^{64}}$. This choice of ring size enables us to losslessly embed deeper and more complex NNs, like the C1 and C2 for CIFAR-10. We vary the ring size of C1 and C2 to $\mathbb{Z}_{2^{64}}$ with the factors 65536 and 131070, and report the experiment results.

We implement the training procedure based on PyTorch backend and train the models over plaintext datasets on NVIDIA Tesla V100 GPU. All four networks are trained with standard SGD optimizer is adapted with cosine learning rate decay, and we set the training parameters as the initial learning rate 0.1, batch size 256, momentum 0.9, and the weight decay 5×10^{-4} for every 50 epochs. For both MNIST and CIFAR-10, all images are scaled to integers in [0, 255]. In this way, all user input images can be directly supplied to `Sonic`'s secure inference without data preprocessing.

6.2 Microbenchmarks

Secure Layer Functions. We benchmark the performance of secure layer functions, i.e., the building blocks used for `Sonic`'s secure inference. Table 4 summarizes the time and bandwidth consumptions of each layer function. For demonstration, we choose the commonly-used kernel sizes 3×3 and 5×5 for the SCONV layer, and 2×2 pooling window for the SMP layer. As shown, all functions can be accomplished within 50ms and require less than 0.5KB bandwidth regardless the window size. Table 5 depicts the performance of the SFC layer over a series of $n \times n$ layers, i.e., both input and output layers are with n neurons. With the growth of n, the time increases linearly attributed to our batch processing, and the bandwidth ascends quadratically.

ReLU and MaxPool Comparison With GC. Fig. 10 demonstrates the performance improvements for `Sonic`'s SReLU

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

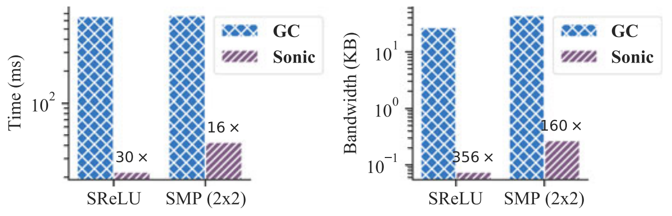


Fig. 10. Time and bandwidth comparison of the SReLU and SMP layers.

and SMP functions over a baseline realized based on GC. The GC baseline realizes equivalent functionalities to Sonic, is implemented on FlexSC with the free-XOR and half-AND optimizations. Compared with the GC baseline, Sonic requires 30× lower latency and 356× less communication for the SReLU function. Besides, Sonic saves 16× latency and 160× communication cost for the SMP function. The reported results show that our proposed secret sharing based designs are more lightweight and applicable for practical requirements, compared with the prior constructions relying on GC [12], [19], [20], [24].

6.3 Sonic's Protocol

Evaluation Over Ring $\mathbb{Z}_{2^{32}}$. We evaluate Sonic's secure NN inference protocol on four models over machine learning benchmarking datasets MNIST and CIFAT-10. Table 6 summarizes Sonic's overall performance running at the cloud. The models M1 with 3(FC-BN-ReLU) and M2 with 4 (CONV/FC-BN-ReLU)-2MP for MNIST dataset are relatively simple. Sonic performs high-quality predictions (97% for M1, 99% for M2) with 0.7s and 13.6s online processing time. The models C1 and C2 are trained for CIFAR-10. C1 uses the MiniONN architecture [11] with 8(CONV/FC-BN-ReLU)-2MP, which is commonly evaluated in prior secure NN inference works. C2's architecture is in line with the FitNet [48] with 10(CONV/FC-BN-ReLU)-3MP. For the more complex C1 and C2, Sonic's predictions require 2.68min and 1.58min respectively. The workloads on the mobile user (within 2.5ms) and model owner (within 6s) are light, which confirms that Sonic is suitable for end-users with resource constrained devices. Note that the workload of the model owner is one-time cost and determined by the NN's size.

We further report the performance breakdown of each network to have a more comprehensive understanding of resource consumptions. As shown in Figs. 11, 12, 13 and 14, the time (left/top figures) and bandwidth (right/bottom figures) costs for each stage in M1 (8 stages) and M2 (10 stages) on MNIST dataset, C1 (32 stages) and C2 (25 stages) on

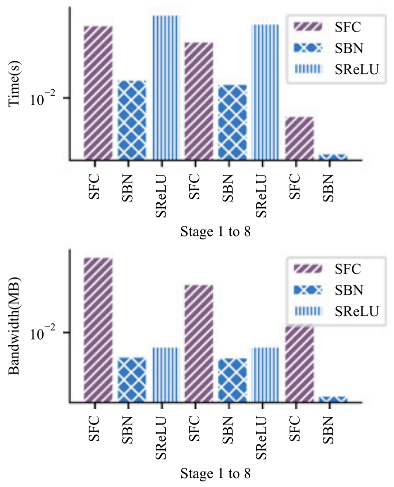


Fig. 11. Performance breakdown of M1. Left: time; Right: bandwidth.

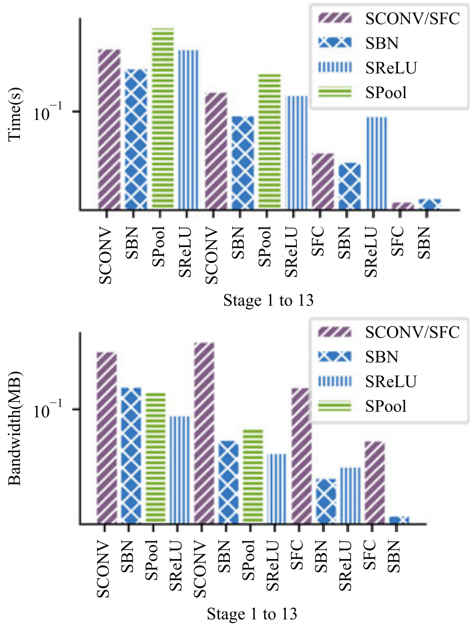


Fig. 12. Performance breakdown of M2. Top: time; Bottom: bandwidth.

CIFAR10 dataset, respectively. As seen, the linear layers occupy most of network resources, and non-polynomial functions usually require more computation.

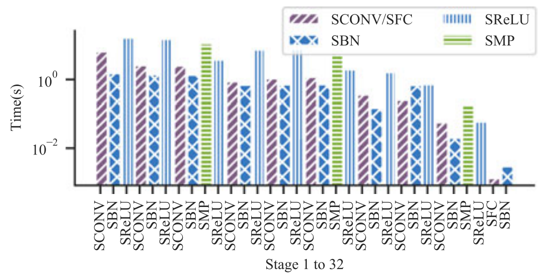
Evaluation Over Ring $\mathbb{Z}_{2^{64}}$. As above mentioned, Sonic provides a 64-bit implementation to support complex and deep networks. We evaluate Sonic with such a choice of ring size on CIFAR-10 and benchmark the results

TABLE 6
Performance Summary of the Benchmarking Models With 32-Bit Implementation

Dataset	Model	Cloud Time (s)	Cloud Comm. (MB)	Mobile User [a] Time (ms)	Model Owner [b] Time (s)	Layers
MNIST	M1	0.7	1.8	0.5	0.02	3SFC + SBN + SBA
	M2	13.6	10.8	0.7	0.11	4SCONV/SFC + SBN + SBA, 2SMP
CIFAR-10	C1	161.2	711.4	2.5	5.9	8SCONV/SFC + SBN + SBA, 2SMP
	C2	94.7	186.9	2.4	1.6	10SCONV/SFC + SBN + SBA, 3SMP

[a]*Cost of generating shares of an image during preprocessing.*
[b]*One-time cost of generating shares of the model during preprocessing.*

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app



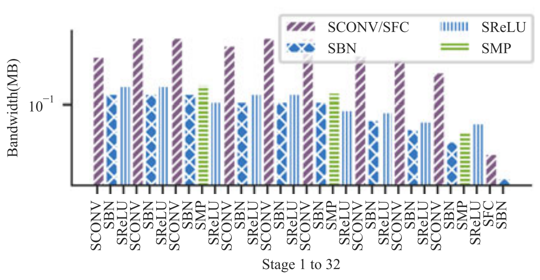
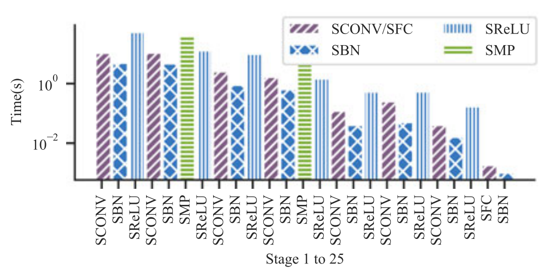


Fig. 13. Performance breakdown of C1. Left: time; Right: bandwidth.



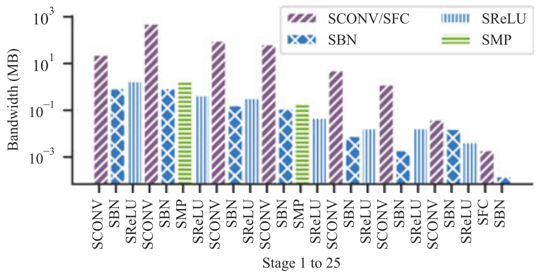


Fig. 14. Performance breakdown of C2. Left: time; Right: bandwidth.

TABLE 7
Performance Summary of the 64-bit Implementation of Models on CIFAR-10

Model	Cloud Time (s)	Cloud Comm. (MB)	Mobile User [a] Time (ms)	Model Owner [b] Time (s)
C1	227.0	1239.5	9.0	14.06
C2	123.1	373.7	9.6	5.86

[a]*Cost of generating shares of an image during preprocessing.*
[b]*One-time cost of generating shares of the model.*

in Table 7. For the models C1 and C2, `Sonic` performs the secure inference with 3.7min and 2.05min computational time, and using 1239MB and 373MB bandwidth costs, respectively. The workloads of the mobile user (within 10ms) and the model owner (within 15s) are lightweight. We note that enlarging the ring size increases their costs. This is because that our prototype is implemented in Java, the 64-bit numbers exceed the primitive data type (i.e., long), and the modular operations need to work over costly BigInteger data type.

6.4 Summary of Accuracy

We report the `Sonic`'s prediction accuracy by varying the ring sizes and compare the results with corresponding plaintext predictions in Table 8. As shown, `Sonic`'s secure inference of the M1 and M2 models on MNIST achieves the same accuracy with plaintext predictions, i.e., 97% and 99% accurate predictions, respectively. For CIFAR-10, `Sonic` varies the ring sizes. For the C1 and C2 over 32-bit ring, `Sonic` produces 75% and 80% accurate inference results, a slight and reasonable lower than the plaintext predictions accuracy 80.9% and 81%. For the C1 and C2 over 64-bit ring, `Sonic`'s inference achieves high-quality results with 84% and 88% accuracy. Such results are more accurate than corresponding plaintext predictions, as the quantization of model weights prevent the overfitting of models. We are aware a tension between the efficiency and utility, i.e., the 64-bit implementation produces more accurate predictions than the 32-bit one yet trading with additional computational and bandwidth overheads.

6.5 Comparison With Prior Art

To demonstrate `Sonic` practical and lightweight, we compare `Sonic`'s bandwidth with notable prior secure NN inference systems. All measurements are reported from their papers, and the cost of Gazelle with all-ReLU activations is given in Delphi. Tables 9 and 10 summarize the bandwidth costs on MNIST and CIFAR10 corresponding to each model. For MNIST, `Sonic` achieves up to 38× and 60× bandwidth savings over M1 and M2 compared with prior art except for Gazelle with the square activation. However, as aforementioned, Gazelle's client-server protocol aggravates the workload on client, including the homomorphic computation and interactions with the model owner.

Table 10 compares `Sonic`'s bandwidth costs over 32-bit and 64-bit rings with prior art on CIFAR-10. For XONN, it applies NN trimming and scaling techniques, which result

TABLE 8
Summary of Inference Accuracy

MNIST			CIFAR-10			
Model	`Sonic`	Plaintext	Model	`Sonic` 32-bit	`Sonic` 64-bit	Plaintext
M1	97.0%	97.0%	C1	75.0%	84.0%	80.9%
M2	99.12%	99.12%	C2	80.0%	88.0%	81%

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

TABLE 9
Bandwidth Comparison of Sonic With Prior Art on MNIST

Model	Prior Art	Bandwidth (MB)
M1	MiniONN [11]	15.8
	Chameleon [17]	10.5
	EzPC [49]	70
	Gazelle (square approx.) [12]	0.5
	XONN (trimmed BNN) [24]	4.29
	Sonic (our design)	**1.8**
M2	MiniONN [11]	657.5
	EzPC [49]	501
	Gazelle (ReLU) [12]	70
	XONN (trimmed BNN) [24]	32.13
	Sonic (our design)	**10.8**

in different accuracy and bandwidth for the same model, depending on the portion of neurons pruned off and the scaling factor. For C1, XONN-1 requires 1290MB bandwidth with 72% accuracy and XONN-2 requires 5099MB bandwidth with 80% accuracy. Sonic's 32-bit and 64-bit implementations achieve 75%, 84% accurate predictions with 711MB, 1239.5MB bandwidth, amounting to 1.8×, 4.1× lower costs and higher accuracy than XONN-1 and XONN-2, respectively. Compared with other works, Sonic's 32-bit implementation requires the least bandwidth. Sonic's 64-bit implementation achieves 7.5×, 2×, 32.8×, 4× bandwidth savings compared with MiniONN, Chameleon, EzPC, and Gazelle with all-ReLU activations, respectively.

For the model C2, XONN-1 consumes 2599MB bandwidth with 81% accuracy and XONN-2 consumes 3461MB bandwidth with 84% accuracy. Our 32-bit and 64-bit implementations cost 186.9MB, 373.7MB bandwidth with 75%, 84% accurate predictions, resulting in 13.9×, 9.3× savings and higher quality compared with XONN-1 and XONN-2. We note that Delphi [13] introduces an optimization on network architecture to improve the performance while preserving satisfactory accuracy. It trains a series of models with different architectures by replacing different portion of ReLU activations with quadratic polynomial approximations with tuned coefficients. However, this optimization has uncertainty in practice, as such process requires extra training costs which could be time-consuming [50].

TABLE 10
Bandwidth Comparison of Sonic's 32-Bit and 64-Bit Implementations With Prior Art on CIFAR10

Model	Prior Art	Bandwidth (MB)
C1	MiniONN [11]	9272
	Chameleon [17]	2650
	EzPC [49]	40683
	Gazelle (square approx.) [12]	1236
	Gazelle (ReLU) [12]	~5000
	XONN-1 [24]	1290
	XONN-2 [24]	5099
	Sonic (32-bit ring)	**711**
	Sonic (64-bit ring)	**1239.5**
C2	XONN-1 [24]	2599
	XONN-2 [24]	3461
	Sonic (32-bit ring)	**186.9**
	Sonic (64-bit ring)	**373.7**

TABLE 11
Time Comparison of Sonic With Prior Art on CNNs Over MNIST

Model	Prior Art	Time (s)	Implementation, Deployment
M1	SecureML [19]	4.88	C++, outsourced
	MiniONN [11]	1.04	C++, interactive
	Chameleon [17]	2.24	C++, 3PC
	EzPC [49]	0.7	C++, outsourced
	Sonic (our design)	**0.7**	Java, outsourced
M2	CryptoNets [38]	297.5	C++, interactive
	MiniONN [11]	9.32	C++, interactive
	EzPC [49]	5.1	C++, outsourced
	Gazelle (ReLU) [12]	1.37	C++, interactive
	Sonic (our design)	**13.6**	Java, outsourced

Tables 11 and 12 summarize the computational costs on the CNNs over MNIST and CIFAR-10 datasets. As shown, for the model M1, Sonic achieves 6.9×, 1.48×, 3.2× speedup over SecureML, MiniONN, Chameleon, respectively, and the time cost of Sonic is comparable to EzPC. For the model M2, Sonic is 21.8× and 1.1× faster than CryptoNets and EzPC, and the time cost of Sonic is comparable to MiniONN. For CIFAR-10, Sonic demonstrates 5.7 ×, 2.8 ×, 1.4× faster secure inference than the MiniONN, EzPC, and Gazelle, respectively. We are aware that some of prior works achieves better inference time than Sonic. However, we emphasize that the time comparison is not a fair comparison due to the following two-fold reasons. From the implementation perspective, most prior works are implemented in C++ whereas Sonic is implemented in Java. It is known that the computational performance of C++ programs are a few orders of magnitudes better than Java programs. For example, Gazelle is implemented in C++ with SIDM optimization and is running on more powerful computers with faster CPU than Sonic. Moreover, some of the prior systems consider the client-server deployment scenario, where the private model weights and the client input are hidden from the counterparty. The secure protocols of these systems *do not need* to work over encrypted model and the encrypted input at the same time. In comparison, the secure outsourced inference systems (including Sonic, SecureML) require to ensure the security of the model and the client input *simultaneously*, i.e., the secure protocols are executed over both encrypted model weights and the encrypted client input. Some other works require multiple servers that increase the deployment overhead than the two-

TABLE 12
Time Comparison of Sonic With Prior Art on CNNs Over CIFAR-10

Prior Art	Time (s)	Implementation, Deployment
MiniONN [11]	544	C++, interactive
Chameleon [17]	52.67	C++, 3PC
EzPC [49]	265.6	C++, outsourced
Gazelle (ReLU) [12]	140	C++, interactive
Sonic (our design)	**94.7**	Java, outsourced

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

TABLE 13
Model Architecture of M1

Layer	# SVDP	Padding, Stride
FC (784 $\rightarrow$ 128) - BN - ReLU	128	NA, NA
FC (128 $\rightarrow$ 128) - BN - ReLU	128	NA, NA
FC (128 $\rightarrow$ 10) - BN	10	NA, NA

server model. For example, the offline phase in Chameleon is designed under the three-server setting.

6.6 Model Architectures

This section presents the details of model architectures used in our evaluation. For MNIST, the model M1 (summarized in Table 13) is comprised of 3 fully-connected (FC) layers, each of the FC layer is followed by the batch normalization (BN) and ReLU. It has been used in prior works [11], [12], [17], [19], [24], [49]. The model M2 (reported in Table 14) has been used in prior works [11], [12], [24], [49], which consists of 3 convolutional layers followed by BN and ReLU, 2 max pooling (MP) layers and 1 FC. For CIFAR-10, we use two more complex models C1 and C2. The model C1 (reported in Table 15) has been used in prior works [11], [12], [17], [24] for a benchmarking evaluation, which consists of 7 CONV layers followed by BN and ReLU, 2 MP layers and 1 FC layer. The model C2 (reported in Table 16) is a variant of FitNets (a thin deep network) and has been adopted in the XONN [24] with its trimmed binarized version, which consists of 9 CONV layers followed by BN and ReLU, 3 MP layers and 1 FC layer.

7 Related Works

7.1 Secure Neural Network Inference

Secure neural network inference has received increasing attention in recent years. Most of prior works [11], [12], [13], [24], [38], [39], [41], [51] consider a scenario where the user directly interacts with the model owner, through cryptographic protocols to obtain the inference result. These protocols ensure that the private model parameters cannot be learned by the client and the private user input cannot be exposed to the server. These works require the user and the model owner to be actively online for synchronous interactions, and to have symmetric computing capabilities, which are hard to be satisfied in practice especially for mobile user clients. Furthermore, most of these works involve heavy cryptographic techniques such homomorphic encryption and/or garbled circuits in the online execution of the cryptographic protocols. When deployed for mobile users, they will result in significant performance overheads.

Some other works [19], [20], [52] leverage the two-server model to carry out the computation of secure inference through tailored protocols, freeing the model owner and the client from active online participation. These secure inference protocols outsourced to the two servers, compared to the above mentioned interactive protocol, require to protect the NN model and the client input simultaneously. However, these works still involve heavy cryptography during the secure inference procedure among the two servers. Specifically, they rely on expensive garbled circuit based approaches to support secure comparison as required in non-polynomial functions of NNs, such as the ReLU function and max pooling. Moreover, the works [20], [52] are designed for special binarized/ternarized NNs (BNN/TNN) with $\{0,1\}/\{-1,0,1\}$ weights, where their secure protocols cannot support generic CNNs with real-valued weights and more complicated layer functions. `Sonic` adopts a similar two-server model, yet fully relies on the lightweight secret sharing technique to build a customized protocol for efficient secure NN inference services in the cloud.

TABLE 14
Model Architecture of M2

Layer	# SVDP	Padding, Stride
CONV (input: $\mathbb{R}^{1\times28\times28}$, kernel: $\mathbb{R}^{1\times16\times5\times5}\rightarrow\mathbb{R}^{16\times24\times24}$) - BN - ReLU	1×9216	NA, 1
MP (input: $\mathbb{R}^{16\times24\times24}$, window: $\mathbb{R}^{16\times2\times2}\rightarrow\mathbb{R}^{16\times12\times12}$)	NA	NA, 2
CONV (input: $\mathbb{R}^{16\times12\times12}$, kernel: $\mathbb{R}^{16\times16\times5\times5}\rightarrow\mathbb{R}^{16\times8\times8}$) - BN - ReLU	16×1024	NA, 1
MP (input: $\mathbb{R}^{16\times8\times8}$, window: $\mathbb{R}^{16\times2\times2}\rightarrow\mathbb{R}^{16\times4\times4}$) - BN - ReLU	NA	NA, 2
FC (256 $\rightarrow$ 100) - BN - ReLU	100	NA, NA
FC (100 $\rightarrow$ 10) - BN	10	NA, NA

TABLE 15
Model Architecture of C1

Layer	# SVDP	Padding, Stride
CONV (input: $\mathbb{R}^{3\times32\times32}$, kernel: $\mathbb{R}^{3\times64\times3\times3}\rightarrow\mathbb{R}^{64\times30\times30}$) - BN - ReLU	3×57600	NA, 1
CONV (input: $\mathbb{R}^{64\times30\times30}$, kernel: $\mathbb{R}^{64\times64\times3\times3}\rightarrow\mathbb{R}^{64\times28\times28}$) - BN - ReLU	64×50176	NA, 1
MP (input: $\mathbb{R}^{64\times28\times28}$, window: $\mathbb{R}^{64\times2\times2}\rightarrow\mathbb{R}^{64\times14\times14}$)	NA	NA, 2
CONV (input: $\mathbb{R}^{64\times14\times14}$, kernel: $\mathbb{R}^{64\times64\times3\times3}$, feature: $\mathbb{R}^{64\times12\times12}$) - BN - ReLU	64×9216	NA, 1
CONV (input: $\mathbb{R}^{64\times12\times12}$, kernel: $\mathbb{R}^{64\times64\times3\times3}\rightarrow\mathbb{R}^{64\times10\times10}$) - BN - ReLU	64×6400	NA, 1
MP (input: $\mathbb{R}^{64\times10\times10}$, window: $\mathbb{R}^{64\times2\times2}\rightarrow\mathbb{R}^{64\times5\times5}$)	NA	NA, 2
CONV (input: $\mathbb{R}^{64\times5\times5}$, kernel: $\mathbb{R}^{64\times64\times3\times3}\rightarrow\mathbb{R}^{64\times3\times3}$) - BN - ReLU	64×576	NA, 1
CONV (input: $\mathbb{R}^{64\times3\times3}$, kernel: $\mathbb{R}^{64\times64\times3\times3}\rightarrow\mathbb{R}^{64\times3\times3}$) - BN - ReLU	64×576	0, 1
CONV (input: $\mathbb{R}^{64\times3\times3}$, kernel: $\mathbb{R}^{16\times64\times3\times3}\rightarrow\mathbb{R}^{16\times3\times3}$) - BN - ReLU	64×144	0, 1
FC (144 $\rightarrow$ 10) - BN	10	NA, NA

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

TABLE 16
Model Architecture of C2

Layer	# SVDP	Padding, Stride
CONV (input: $\mathbb{R}^{3\times32\times32}$, kernel: $\mathbb{R}^{3\times16\times3\times3} \rightarrow \mathbb{R}^{16\times32\times32}$) - BN - ReLU	3×16384	0, 1
CONV (input: $\mathbb{R}^{16\times32\times32}$, kernel: $\mathbb{R}^{16\times16\times3\times3} \rightarrow \mathbb{R}^{16\times32\times32}$) - BN - ReLU	16×16384	0, 1
CONV (input: $\mathbb{R}^{16\times32\times32}$, kernel: $\mathbb{R}^{16\times16\times3\times3} \rightarrow \mathbb{R}^{16\times32\times32}$) - BN - ReLU	16×16384	0, 1
MP (input: $\mathbb{R}^{16\times32\times32}$, window: $\mathbb{R}^{16\times2\times2} \rightarrow \mathbb{R}^{16\times16\times16}$)	NA	NA, 2
CONV (input: $\mathbb{R}^{16\times16\times16}$, kernel: $\mathbb{R}^{16\times32\times3\times3} \rightarrow \mathbb{R}^{32\times16\times16}$) - BN - ReLU	16×8192	0, 1
CONV (input: $\mathbb{R}^{32\times16\times16}$, kernel: $\mathbb{R}^{32\times32\times3\times3} \rightarrow \mathbb{R}^{32\times16\times16}$) - BN - ReLU	16×8192	0, 1
CONV (input: $\mathbb{R}^{32\times16\times16}$, kernel: $\mathbb{R}^{32\times32\times3\times3} \rightarrow \mathbb{R}^{32\times16\times16}$) - BN - ReLU	16×8192	0, 1
MP (input: $\mathbb{R}^{32\times16\times16}$, window: $\mathbb{R}^{32\times2\times2} \rightarrow \mathbb{R}^{32\times8\times8}$)	NA	NA, 2
CONV (input: $\mathbb{R}^{32\times8\times8}$, kernel: $\mathbb{R}^{32\times48\times3\times3} \rightarrow \mathbb{R}^{48\times6\times6}$) - BN - ReLU	32×1728	NA, 1
CONV (input: $\mathbb{R}^{48\times6\times6}$, kernel: $\mathbb{R}^{48\times48\times3\times3} \rightarrow \mathbb{R}^{48\times4\times4}$) - BN - ReLU	48×1728	NA, 1
CONV (input: $\mathbb{R}^{48\times4\times4}$, kernel: $\mathbb{R}^{48\times64\times3\times3} \rightarrow \mathbb{R}^{64\times2\times2}$) - BN - ReLU	48×2304	NA, 1
MP (input: $\mathbb{R}^{64\times2\times2}$, window: $\mathbb{R}^{64\times2\times2} \rightarrow \mathbb{R}^{64\times1\times1}$)	NA	NA, 2
FC (64 → 10) - BN	10	NA, NA

There have been some works [17], [35], [53], [54], [55] focusing on efficient secure inference with three or four servers. These works usually introduce higher real-world deployment complexity compared to the two-server model. Among all prior works, we are aware that some systems [35], [41], [55] also only uses secret sharing. However, their protocols are specially designed under a more complex *three-server* setting[35], [55] where three servers interact with each other in the online inference procedure, or for a non-outsourcing setting that requires *continuous interactions* between the client and the model owner [41]. Table 17 gives a high-level comparison of `Sonic` with prior works.

7.2 Privacy-Preserving Machine Learning in Cloud Computing

A rich body of work has explored privacy-preserving machine learning applications in cloud computing. Some of these works ensure the privacy resorting to cryptographic approaches (secure multiparty computation, homomorphic encryption) for various outsourced ML applications, like ridge regression and logistic regression [19], [56], decision trees [16], federated learning [57], video classification via CNNs [58]. A common rationale of these approaches is to design specialized cryptographic protocols to meet the certain needs of different applications. For example, POSEIDON [57] employs multiparty lattice-based fully homomorphic encryption for a quantum-resistant federated learning. Some other works utilize differential privacy techniques [27], [28], [29], [30] or rely on the trusted processors (e.g., SGX) [59], [60] for privacy preservation, such as the oblivious video analytics as a cloud service [59] and XGBoost learning [60]. Meanwhile, some other work [61] focuses on an orthogonal aspect, i.e., verifying the integrity of machine leaning computation, but it does not guarantee the user data and model confidentiality.

7.3 Adversarial Machine Learning Attacks

There have been adversarial machine learning attacks [7], [8], [25], [26] that attempt to infer private information about

TABLE 17
A High-Level Summary of Different Secure Inference Systems

			Heavy comp. cost	Heavy comm. cost[1]	Applicable for general CNNs[2]
Interactive 2PC Protocol (symmetric)	HE	CryptoNets [38], CryptoDL [51]	●	●	○
		XONN [24]	◐	○	○
	Mixed	MiniONN [11]	●	●	●
		Gazelle [12]	●	◐	◐
		Delphi [13]	◐	◐	●
	SS	MediSC [41]	○	○	●
Outsourced 2PC Protocol	Mixed	SecureML [19]	◐	-	●
		Quotient [20]	◐	-	○
		Leia [52]	○	○	○
	SS	`Sonic` (**our system**)	○	○	●
Using Multiple Servers[3]	Mixed	ABY^3 [53], Trident [54]	◐	○	●
		Chameleon [17]	◐	◐	●
	SS	SecureNN [35], CryptFlow [55]	○	○	●

[1]*The communication cost is evaluated based on the results reported in prior works, and [19], [20] have not reported their inference bandwidth costs.*
[2]*Not fully support common CNN model architectures, which could downgrade inference accuracy.*
[3]*Require three or more servers to assist the online inference procedure.*

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

the NN model and the training dataset via blackbox oracle access to the inference procedure. One major class of those attacks are the membership inference attacks [8], [26], where an attacker aims to discover the presence of a particular record in the training dataset via querying the NN models in a black-box way.[1] Another class of attacks are the model inversion attacks [7], [62], which can be abused to infer the (hidden) sensitive attributes of the input data through the API access to the model. Apart from the inference attacks, there exist other attacks that try to steal specific information about the NN models, such as the model stealing attack [25], and hyper-parameter stealing attack [63].

Defenses against those attacks are active research areas complementary to secure NN inference that we target in this paper. One mitigation approach is to limit the private information memorized by the NN model. A common strategy is to leverage differentially private learning [27], [28], [29], [30], so as to bound the privacy impact regarding whether or not a single data record is used to train the model. By injecting a calibrated amount of random noise to the NN model during training, the presence of a record in the training dataset and the effect of sensitive attribute on the prediction can be obfuscated [30]. Apart from using differential privacy techniques, some other ML methods have been proposed to reduce the private information given to the adversary via modifying the network and/or prediction, such as regularization [31], prediction API minimization [25], and perturbing the activation function [32]. Another approach is query auditing, where the feature space explored by queries [33], or the distribution of a batch of queries [34] is leveraged to detect adversarial ones. Indeed ensuring the confidentiality of the model weights could increase the overhead to successfully launch those attacks [34]. Such an approach compels an attacker to send more queries, and as a result, the adversarial queries are more likely to be detected.

8 Conclusion

In this paper, we present Sonic, an in-the-cloud lightweight secure neural network inference service. Sonic relaxes the end-user and model owner from being engaged in online secure inference procedure. Sonic is comprised of a series of secure layer functions purely relying on lightweight secret sharing techniques, each of which is highly customized for an efficient secure inference service in the cloud. Sonic provide strong guarantees on both user input and CNN model privacy. Experiments on common benchmark datasets demonstrates Sonic's practical performance.

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[2] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren, "Securing SIFT: Privacy-preserving outsourcing computation of feature extractions over encrypted image data," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3411–3425, Jul. 2016.

[3] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.

[4] H. Li, Y. Yang, Y. Dai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 484–494, Second Quarter 2020.

[5] Google Vision, 2021. [Online]. Available: Online at https://cloud.google.com/vision

[6] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical approximate k nearest neighbor queries with location and query privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1546–1559, Jun. 2016.

[7] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. ACM 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.

[8] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.

[9] R. Cammarota *et al.*, "Trustworthy AI inference systems: An industry research view," 2020, *arXiv:2008.04449*.

[10] X. Yi, E. Bertino, F.-Y. Rao, K.-Y. Lam, S. Nepal, and A. Bouguettaya, "Privacy-preserving user profile matching in social networks," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1572–1585, Aug. 2020.

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

[12] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proc. 27th USENIX Conf. Secur. Symp.*, 2018, pp. 1651–1668.

[13] M. Pratyush, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 2505–2522.

[14] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara, "Private collaborative forecasting and benchmarking," in *Proc. ACM Workshop Privacy Electron. Soc.*, 2004, pp. 103–114.

[15] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proc. 11th Annu. Int. Cryptol. Conf. Advances Cryptol.*, 1991, pp. 420–432.

[16] Y. Zheng, H. Duan, and C. Wang, "Towards secure and efficient outsourcing of machine learning classification," in *Proc. 24th Eur. Symp. Res. Comput. Secur.*, 2019, pp. 22–40.

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

[18] Amazon Rekognition, 2021. [Online]. Available: https://aws.amazon.com/rekognition

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

[20] N. Agrawal, A. Shahin Shamsabadi , M. J. Kusner, and A. Gascón, "QUOTIENT: Two-party secure neural network training and prediction," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1231–1247.

[21] Y. Zheng, H. Duan, and C. Wang, "Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2475–2489, Oct. 2018.

[22] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," in *Proc. NeurIPS Workshop Privacy-Preserving Mach. Learn.*, 2020, p. 1.

[23] Cape Privacy, "TF encrypted: Encrypted deep learning in tensorflow," 2020. [Online]. Available: https://tf-encrypted.io/

[24] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "XONN: XNOR-based oblivious deep neural network inference," in *Proc. 28th USENIX Conf. Secur. Symp.*, 2019, pp. 1501–1518.

[25] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 601–618.

1. Membership inference attacks via white-box access [62] are out of the consideration as they requires auxiliary information (e.g., training losses) about training process.

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

[26] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 587–601.
[27] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321.
[28] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 332–349.
[29] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang, "Towards practical differentially private convex optimization," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 299–316.
[30] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 1895–1912.
[31] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 267–284.
[32] T. Lee, B. Edwards, I. Molloy, and D. Su, "Defending against neural network model stealing attacks using deceptive perturbations," in *Proc. IEEE Secur. Privacy Workshops*, 2019, pp. 43–49.
[33] M. Kesarwani, B. Mukhoty, V. Arya, and S. Mehta, "Model extraction warning in MLaaS paradigm," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, 2018, pp. 371–380.
[34] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "PRADA: Protecting against DNN model stealing attacks," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2019, pp. 512–527.
[35] S. Wagh, D. Gupta, and N. Chandran, "Securenn: 3-party secure computation for neural network training," *Proc. Privacy Enhancing Technol.*, vol. 2019, pp. 26–49, 2019.
[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proc. Int. Conf. Mach. Learn., vol. 37, 2015, pp. 448–456.
[37] Pytorch for densenet, 2021. [Online]. Available: https://github.com/pytorch/vision/blob/master/torchvision/models/densenet.py
[38] 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. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[39] Q. Lou and L. Jiang, "SHE: A fast and accurate deep neural network for encrypted data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 10 035–10 043.
[40] D. Harris, "A taxonomy of parallel prefix networks," in *Proc. 37th Asilomar Conf. Signals Syst. Comput.*, 2003, pp. 2213–2217.
[41] X. Liu, Y. Zheng, X. Yuan, and X. Yi, "MediSC: Towards secure and lightweight deep learning as a medical diagnostic service," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2021, pp. 519–541.
[42] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2. 0: Improved mixed-protocol secure two-party computation," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2165–2182.
[43] Y. Zheng, C. Wang, R. Wang, H. Duan, and S. Nepal, "Optimizing secure decision tree inference outsourcing," 2021, *arXiv:2111.00397*.
[44] Execution order of ReLU and Max-Pooling, 2016. [Online]. Available: https://github.com/tensorflow/tensorflow/issues/3180
[45] F. Boemer, R. Cammarota, D. Demmler, T. Schneider, and H. Yalame, "MP2ML: A mixed-protocol machine learning framework for private inference," in *Proc. 15th Int. Conf. Availability Rel. Secur.*, 2020, pp. 1–10.
[46] D. Demmler, T. Schneider, and M. Zohner, "ABY-A framework for efficient mixed-protocol secure two-party computation," in *Proc. Annu. Netw. Distrib. Syst. Secur. Symp.*, 2015, p. 1.
[47] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," in *Proc. 19th Annu. ACM Symp. Theory Comput.*, 1987, pp. 218–229.
[48] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv: 1412.6550*.
[49] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "EzPC: Programmable, efficient, and scalable secure two-party computation for machine learning," *ePrint Rep.*, vol. 1109, pp. 496–511, 2017.
[50] P. Ren *et al.*, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv. (CSUR)*, ACM New York, NY, USA, vol. 54, no. 4, pp. 1–34, 2021.
[51] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep neural networks over encrypted data," 2017, *arXiv:1711.05189*.
[52] X. Liu, B. Wu, X. Yuan, and X. Yi, "Leia: A lightweight cryptographic neural network inference system at the edge," *IACR Cryptol. ePrint Arch.*, vol. 2020, 2020, Art. no. 463.
[53] P. Mohassel and P. Rindal, "ABY3: A mixed protocol framework for machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 35–52.
[54] R. Rachuri and A. Suresh, "Trident: Efficient 4PC framework for privacy preserving machine learning," in *Proc. Annu. Netw. Distrib. Syst. Security Symp.*, 2020, p. 1.
[55] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow: Secure tensorflow inference," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 336–353.
[56] 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. IEEE Symp. Secur. Privacy*, 2013, pp. 334–348.
[57] S. Sav *et al.*, "POSEIDON: Privacy-preserving federated neural network learning," *Proc. 28th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2022, p. 1.
[58] S. Pentyala, R. Dowsley, and M. De Cock, "Privacy-preserving video classification with convolutional neural networks," 2021, *arXiv:2102.03513*.
[59] R. Poddar, G. Ananthanarayanan, S. Setty, S. Volos, and R. A. Popa, "Visor: Privacy-preserving video analytics as a cloud service," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1039–1056.
[60] A. Law *et al.*, "Secure collaborative training and inference for XGBoost," in *Proc. Workshop Privacy-Preserving Mach. Learn. Pract.*, 2020, pp. 21–26.
[61] L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, and B. Feng, "VeriML: Enabling integrity assurances and fair payments for machine learning as a service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2524–2540, OCt. 2021.
[62] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Found. Symp.*, 2018, pp. 268–282.
[63] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 36–52.

Xiaoning Liu received the BEng degree in computer science and technology from Henan University, Kaifeng, China, in 2012, and the MSc degree in computer science from the City University of Hong Kong, Hong Kong, in 2013. She is currently working toward the PhD degree in the School of Computing Technologies, RMIT University, Melbourne, Australia. Her research interests include pivots on data privacy and security related to machine learning, data mining, cloud computing, and digital health, with the current focus on designing practical secure computation systems for neural network powered applications.

Yifeng Zheng received the PhD degree in computer science from the City University of Hong Kong, Hong Kong, in 2019. He is currently an assistant professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. He worked as a postdoctoral with the Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia, and the City University of Hong Kong, respectively. His work has appeared in prestigious venues such as ESORICS, ACM AsiaCCS, DSN, IEEE INFOCOM, IEEE ICDCS, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, and *IEEE Transactions on Multimedia*, and *IEEE Transactions on Services Computing*. His current research interests include focused on security and privacy related to cloud computing, IoT, machine learning, and multimedia.

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app

Xingliang Yuan is currently a senior lecturer (aka U.S. associate professor) with the Department of Software Systems and Cybersecurity, Faculty of Information Technology, Monash University, Australia. His research interests include data security and privacy, secure networked system, machine learning security and privacy, confidential computing. His research has been supported by Australian Research Council, CSIRO Data61, and Oceania Cyber Security Centre. In the past few years, his work has appeared in prestigious venues in cybersecurity, computer networks, and distributed systems, such as ACM CCS, NDSS, IEEE INFOCOM, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, and *IEEE Transactions on Parallel and Distributed Systems*. He was the recipient of Dean's Award for Excellence in Research by an Early Career Researcher at Monash Faculty of IT in 2020. He received the Best Paper Award in the European Symposium on Research in Computer Security (ESORICS) 2021, the IEEE Conference on Dependable and Secure Computing (IDSC) 2019, and the IEEE International Conference on Mobility, Sensing and Networking (MSN) 2015.

Xun Yi is currently a professor with the School of Computing Technologies, RMIT University, Australia. His research interests include applied cryptography, computer and network security, mobile and wireless communication security, and privacy-preserving data mining. He has published more than 150 research papers in international journals, such as the *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Circuits and Systems*, *IEEE Transactions on Vehicular Technology*, *IEEE Communication Letters*, *IEE Electronic Letters*, and conference proceedings. He has ever undertaken program committee members for more than 30 international conferences. Recently, he has led a few of Australia Research Council (ARC) Discovery Projects. Since 2014, he has been an associate editor of the *IEEE Transactions on Dependable and Secure Computing*.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

thorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on December 27,2025 at 15:31:57 UTC from IEEE Xplore. Restrictions app