HeSUN: Homomorphic Encryption for Secure Unbounded Neural Network Inference

Duy Tung Khanh Nguyen(✉), Dung Hoang Duong, Willy Susilo, and Yang-Wai Chow

Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Northfields Avenue, Wollongong, NSW 2522, Australia
dtkn354@uowmail.edu.au, {hduong,wsusilo,caseyc}@uow.edu.au

Abstract. In recent years, homomorphic encryption (HE) has become a crucial tool for secure neural network inference (SNNI), which enables the server to classify encrypted data of clients while guaranteeing privacy. However, current HE-based frameworks limit the depth of neural networks. The main reason for the limitation is the noise and scaling factor growth in ciphertext after successive homomorphic operators. Gentry's bootstrapping is normally the solution for addressing noise growth. However, bootstrapping is a costly procedure and requires the circular security assumption. For scaling factor growth, it remains a challenging problem because rescaling is based on division, which is not natively supported by current HE schemes. This paper proposes a double ciphertext refreshing protocol called DoubleR, which refreshes noise and scaling factor growth at the same time. Our protocol is proven secure in the semi-honest model without introducing additional assumptions. The experimental results show that our protocol outperforms bootstrapping by 300× in running time. Based on DoubleR, we build a versatile framework for SNNI called HeSUN, which significantly accelerates the inference time with comparable communication costs.

Keywords: Privacy preserving machine learning · Homomorphic encryption

1 Introduction

Machine Learning as a Service (MLaaS) has become a popular computing paradigm that uses robust cloud infrastructures to provide machine learning inference services to clients. The paradigm can be described as follows: A server holds a pre-trained neural network model F. A client wants to use F to predict on data x, but the client is hesitant to send x to the server because x may contain the client's sensitive information, e.g., healthcare records and financial data. In a secure setting, called secure neural network inference (SNNI), the client can

H. Duan et al. (Eds.): SecureComm 2023, LNICST 567, pp. 413–438, 2025.
https://doi.org/10.1007/978-3-031-64948-6_21

learn the inference result $F(x)$ with the guarantee that the server learns nothing about x and the client learns nothing about F. Homomorphic encryption (HE) is a potential tool for this privacy requirement. HE is an emerging technology that enables computing on encrypted data. In SNNI, the client can send the data x in encrypted form; the server runs inference on the encrypted data and returns the encrypted result to the client; the client decrypts the message and gets the prediction result. Unfortunately, current HE schemes suffer from inherent drawbacks that hinder applying HE in SNNI.

Normally, HE ciphertexts contain noises. The ciphertext can be decrypted correctly if the noise is small within a bound. Running homomorphic operations on ciphertexts increases the noise in the resulting ciphertext. At some point, the noise will become too large, and the ciphertext cannot be decrypted correctly. HE schemes that allow evaluation on circuits of arbitrary depth and have no restrictions on the required operations are called fully homomorphic encryption (FHE). A crucial question for cryptographers is how to achieve FHE. The current answer is bootstrapping, an innovative procedure proposed by Gentry [20]. This ciphertext refreshing technique produces a new ciphertext that encrypts the same message at a lower noise level so that more homomorphic operations can be performed. Gentry's paper [20] triggered significant interest in HE, and novel constructions on FHE have been proposed [11,15,16,19] which follows Gentry's bootstrapping idea. Roughly speaking, bootstrapping homomorphically decrypts the ciphertext using encryption of the secret key, called the bootstrapping key, then re-encrypts the message with the bootstrapping key. This requires an extra security prerequisite, termed the circular security assumption. The assumption implies that disclosing the encryption of the secret key is presumed to be secure. The circular security assumption remains not well studied and understood. Proving circular security for HE schemes remains an open problem. Besides, bootstrapping is generally considered an expensive operation, so one usually tries to avoid it as much as possible [36].

Besides noise growth, scaling factor growth is another crucial reason that hinders applying HE for SNNI. Neural networks typically operate on real numbers. However, most current HE schemes are integer-HE schemes that work on integers. Hence, we need to convert a real number to an integer, which is done by multiplying a real number x by some scaling factor Δ, resulting in $\lfloor x.\Delta \rceil$ where $\lfloor . \rceil$ denotes rounding to the nearest integer. The scaling factor increases tremendously after homomorphic multiplication and the final calculations will need to operate on very large integers. An example of scaling factor growth is shown in Table 3 where the initial scaling factor Δ, after 5 layers, increases to Δ^{11} which requires plaintext modulus of the corresponding HE scheme greater than 2^{80} [21]. For more extensive networks, HE is unusable. In non-HE integer-only networks, as computations are carried out, the integer variables tend to expand in size. Typically, after each layer, these intermediate values are commonly scaled down to a smaller number to make these integer-only networks manageable. Essentially, after every operator, the most significant bits are retained, while the least significant bits are omitted. Unfortunately, these reduction procedure involves

division or shift operations, which aren't inherently compatible with integer-HE schemes. Hence, downscaling in integer-HE schemes remains a challenging problem.

Besides integer-HE schemes, CKKS [15] is a HE scheme that supports computation on real numbers by relaxing the definition of HE. The standard definition of HE requires that computations on encrypted and plain data should yield identical outcomes. However, CKKS conducts approximate calculations. Thus, outcomes from encrypted and plain data are nearly identical, with a small approximation error. Notably, CKKS is the only HE scheme that solves the problem of scaling factor growth by introducing a rescaling procedure. The rescaling procedure in CKKS can be thought of as one that homomorphically removes the inaccurate least significant bits in the ciphertext. The procedure is run on ciphertext directly. Unfortunately, this method cannot be applied to integer-HE schemes because of the difference in decryption structures of CKKS and integer-HE schemes (please refer to [15] for the details). Furthermore, rescaling is an expensive operation in CKKS, which is $\approx 9\times$ more expensive than ciphertext-plaintext multiplication [7,8]. On the other hand, our proposed framework does the rescaling operation in plaintext space, which speeds up the performance of the rescaling procedure.

In summary, this paper proposes a novel protocol that simultaneously refreshes noise and scaling factors. Our main contributions are as follows:

- We proposed a novel double refreshing protocol, called DoubleR, which simultaneously refreshes noise and scaling factor. For noise refreshing, our protocol deviates from Gentry's bootstrapping technique, such that we bypass circular security. The experiments show that the refreshing noise time is $300\times$ faster than current bootstrapping techniques [3]. For scaling factor refreshing, to the best of our knowledge, this work is the first work that refreshes the scaling factor for integer-HE schemes in SNNI, which is a step towards unbounded SNNI for integer-HE schemes. For the CKKS scheme, our protocol replaces the expensive rescaling operators on ciphertexts with our rescaling algorithm, which works in plaintext space. Hence, we speed up the rescaling procedure.
- We build a versatile framework for SNNI, called HeSUN, which supports both integer-HE schemes (with BFV [19] as an instantiation) and the CKKS scheme. The idea of HeSUN is to use DoubleR to refresh intermediate neural network layers after some homomorphic evaluation. Thanks to DoubleR, we can evaluate deep neural networks using HE schemes with smaller parameters than the current HE-based approach, which efficiently speeds up the inference time. Our protocol requires communication between client and server, while previous HE-based frameworks are non-interactive approaches. However, because HeSUN can significantly reduce the size of a HE scheme's parameters, the size of messages exchanged during communication is even smaller than current HE-base frameworks. The details are presented in Sect. 6. As shown in Table 1, compared to current approaches, our framework satisfies both security, efficiency, and unbounded HE criteria.

2 Related Works

This section presents an overview of current approaches for SNNI using HE.

LHE-Based. This approach sidesteps bootstrapping by using leveled homomorphic encryption (LHE). LHE allows a predefined number of additions or multiplications and requires knowing the depth of the neural network in advance. CryptoNets [21] is the first work that applies LHE for SNNI, which performed inference on a 5-layer neural network on the MNIST dataset. Subsequent work, CryptDL [24], CHET [18], RAMPARTS [5], and nGraph-HE [9] improved the performance, scaled to larger networks, and integrated with deep neural network compiler technologies. These frameworks support Single Instruction Multiple Data (SIMD) operations [40], which pack multiple plaintexts into a ciphertext and perform an operation on the ciphertext equivalent to operating on multiple plaintexts. The packing method is the crucial property of the LHE scheme, which increases inference throughput. However, LHE-based frameworks have two shortcomings. First, these frameworks are based on LHE, so it limits the depth of the neural networks. So, this approach does not scale well and is not adaptable to deep learning, where neural networks can contain tens, hundreds, or sometimes thousands of layers [23,43]. Secondly, the computational cost is prohibitively large. This is because inference on deep neural networks requires a large multiplicative level, leading to large LHE schemes' parameters. As shown in Table 1, the LHE-based approach provides the highest security level. It also limits the depth of the neural network and requires a huge computation overhead. Compared to the LHE-based approach, HeSUN allows unbounded SNNI with smaller parameters.

n-Graph-HE2 [8] aims to achieve unbounded SNNI with LHE schemes. It is based on client-aided architecture where the server sends encrypted layer value to the client, and the client decrypts and evaluates the activation function. This procedure can be considered to be a ciphertext refreshing process. However, n-Graph-HE2 leaks the network's information. Compared to n-Graph-HE2, HeSUN does not leak the network's information. Hence, HeSUN provides a higher level of security than n-Graph-HE2.

FHE-Based. Another approach is based on the HE schemes equipped with a fast bootstrapping technique, e.g., TFHE [16], that supports unbounded SNNI. However, since the packing technique isn't readily adaptable with current TFHE schemes, it might introduce extra inefficiencies in terms of running time and memory overhead when we want to process large amounts of data concurrently. Furthermore, the TFHE-based frameworks usually work on quantized neural networks, where accuracy is lower than the original neural network [10,33]. For example, [17] compares current HE schemes for SNNI, while CKKS and BFV achieve comparable accuracy with plain data, i.e., 96.34%, TFHE only achieves 82.70%. As shown in Table 1, while other frameworks achieve comparable accuracy with the original model, the TFHE-based approach, i.e., FHE-DiNN and SHE, have lower accuracy. Therefore, compared to the FHE-based approach, HeSUN supports SIMD and provides higher accuracy.

HE-MPC Hybrid. Besides pure HE-based, GAZELLE [27] and MP2ML [7] are hybrid frameworks combining HE with MPC, which compute linear layer using HE and non-linear activation functions using MPC. Although hybrid approaches can evaluate the non-linear functions precisely, the privacy of model information can be leaked. In particular, the client must know the type of activation function used in the model, which is undesirable for the servers in SNNI. In contrast, HeSUN keeps this information secret. Furthermore, hybrid approaches require conversion between HE ciphertexts and MPC values. HeSUN is merely based on HE schemes. Hence, no data conversion is required during the inference process. Besides, Gazzelle works on an additive integer-HE scheme, which incurs the problem of scaling growth as integer-HE schemes, so it cannot achieve unbounded SNNI. MP2ML simply uses CKKS as a building block, so the rescaling procedure is automatically run on ciphertexts after homomorphic multiplications. Compared to MP2ML, HeSUN provides a faster rescaling procedure because the rescaling is run in plaintext space.

Another popular approach for SNNI in literature is merely based on MPC protocol [32,35,38]. Although this approach is widely applied in SNNI, it reveals all network architecture, which is less secure than HE-based approaches.

Our protocol, HeSUN, overcomes the shortcomings of previous approaches by introducing a novel refreshing protocol, DoubleR, that simultaneously refreshes the noise and scaling factor in the SNNI process. Table 1 shows a comparison between HeSUN and previous works. HeSUN satisfies both security, efficiency, and unbounded HE criteria. HeSUN is an interactive approach. However, by using smaller parameters for HE schemes, HeSUN achieves comparable communication costs compared to non-interactive approaches. The details are presented in Sect. 6. Notably, HeSUN is the first framework that provides unbounded SNNI for integer-HE schemes thanks to the scaling factor refresh algorithm in DoubleR protocol.

Table 1. Comparison of secure neural network inference frameworks.

		Security		Efficiency				Unbounded depth
		Weight	Activation function	SIMD	Small Parameters	Comparable accuracy	Non-interactive	
LHE-based	CryptoNets [21]	✓	✓	✓	✗	✓	✓	✗
	CryptoDL [24]	✓	✓	✓	✗	✓	✓	✗
	CHET [18]	✓	✓	✓	✗	✓	✓	✗
	RAMPARTS [5]	✓	✓	✓	✗	✓	✓	✗
	nGraph-HE [9]	✓	✓	✓	✗	✓	✓	✗
	nGraph-HE2 [8]	✗	✗	✓	✓	✓	✗	✓
FHE-based	FHE-DiNN [10]	✓	✓	✗	✓	✗	✓	✓
	SHE [33]	✓	✓	✗	✓	✗	✓	✓
Hybrid-based	GAZELLE [27]	✓	✗	✓	✓	✓	✗	✗
	MP2ML [7]	✓	✗	✓	✓	✓	✗	✓
	HeSUN (this work)	✓	✓	✓	✓	✓	✗	✓

3 Preliminaries

3.1 Homomorphic Encryption

Homomorphic encryption (HE) is essential for privacy-preserving machine learning. HE supports computation on encrypted data without decryption. The data used for encryption is assumed to be elements in a polynomial ring $\mathcal{R}$. Given two plaintext m_1, $m_2 \in \mathcal{R}$ and corresponding two ciphertexts $[m_1]$ and $[m_2]$ respectively, the HE scheme provides two additional operators $\oplus$ and $\otimes$ so that:

$$Decrypt([m_1] \oplus [m_2]) = m_1 + m_2, \text{ and}$$
$$Decrypt([m_1] \otimes [m_2]) = m_1 \times m_2$$

where $Decrypt(.)$ is the decryption function.

HE schemes can be categorized based on the computational operations they accommodate. Leveled HE (LHE) schemes allow a restricted number of additions and multiplications. Fully HE (FHE) schemes support unlimited additions and multiplications. Our framework uses two LHE schemes; namely, BFV [19] and CKKS [15]. A full description of these two schemes is beyond the scope of this paper; we refer to [19] and [15] for the details. Below, we briefly review relevant points related to this paper.

Besides supporting operators between ciphertexts, BFV and CKKS are equipped with ciphertext-plaintext operators.

$$Decrypt([m_1] \oplus m_2) = m_1 + m_2 \text{ and}$$
$$Decrypt([m_1] \otimes m_2) = m_1 \times m_2$$

While plaintext and ciphertext in the CKKS scheme are in the same ring $\mathcal{R}_q$, in BFV, plaintext space is $\mathcal{R}_t$ and ciphertext space is $\mathcal{R}_q$, where $\mathcal{R}_t$ ($\mathcal{R}_q$ respectively) denotes the set of polynomials in $\mathcal{R}$ with coefficients in $\mathbb{Z}_t$ ($\mathbb{Z}_q$ respectively). In BFV, if plaintext polynomial wraps around t at any point during the computation, we are no longer doing integer arithmetic but modulo t arithmetic, and decoding yields an unexpected result. In the context of SNNI, it usually requires t to be extremely big for the needed precision [21]. An exciting result of our proposed protocol for BFV is that we can achieve unbounded SNNI with much smaller t compared to current approaches.

In CKKS, each ciphertext goes along with a non-negative integer, called level, which is the number of homomorphic multiplication operations consumed. The initial ciphertext (after encryption) has level 0. After each homomorphic multiplication, the level is increased by one. If the level achieves L, we no longer perform computation on the ciphertext. Ciphertext refreshing in CKKS means converting a L-level ciphertext to a zero-level ciphertext to enable further homomorphic multiplications.

3.2 Adversarial Model

Security proof of our protocol is based on the secure two-party computation framework for semi-honest adversaries [22]. At a high level, consider two parties, A and B, jointly compute a function f by running a protocol Π. Suppose an adversary compromises A. This adversary attempts to use the received information while running Π to learn private information about the input of B. Notably, this adversary always follows the protocol, which is different from a malicious adversary, where a malicious adversary can arbitrarily deviate from the specified protocol. Our framework (Sect. 5) is created by composing sequential secure protocols. To prove the security of our framework, we invoke modular sequential composition, as introduced in [13]. Besides, the protocol security is also based on the semantic security of HE schemes. The preliminaries for security proof and the proof are detailed in Appendix A and B, respectively.

4 Double Refreshing Protocol

This section presents our proposed double refreshing protocol (DoubleR) that refreshes noise and the scaling factor in LHE schemes. We assume a party B holds a ciphertext (with large noise and large scaling factor) encrypted under another party's (A) public key and wants to reduce the noise and scaling factor of the ciphertext. In particular, consider a LHE scheme, HE, which is semantically secure. Let [.] be an encryption using HE. Consider that party A holds the secret key sk, and both parties have the public key pk of the scheme. Party B also has a ciphertext $[m]$. Our protocol enables B to obtain a new ciphertext $[m']$ ($Decrypt([m'] \approx Decrypt([m])$ with less noise and a smaller scaling factor. Our protocol guarantees that neither party learns anything about message m.

4.1 DoubleR-CKKS

DoubleR-CKKS protocol is shown in Protocol 1. In CKKS, plaintext and ciphertext spaces are the same, i.e., $\mathcal{R}_q$. In the CKKS scheme, after encryption, the ciphertext is at level 0. After each multiplication, the level increases until the maximum level L is reached; we no longer compute encrypted data. Addition and subtraction do not change the level of ciphertext. In our protocol, refreshing a ciphertext means reducing the level of ciphertext. We denote a ciphertext by $[m, \Delta_L, L]$ where m is the message, Δ_L, and L are the scaling factor and level of ciphertext, respectively. We now provide intuition about the protocol and its security.

A simple solution to refresh the ciphertext (with large noise) is to let party B send the ciphertext to party A. Then, A decrypts the ciphertext, re-encrypts the message and returns it to B. However, this solution leaks the message m to A. Our protocol aims at keeping m secret from both A and B. The idea is that B homomorphically masks the ciphertext $[m] \in \mathcal{R}_q$, with large noise, by a randomness $r \in \mathcal{R}_q$ and sends it (i.e., $[m] + r$) to A. Then, A decrypts the

received message and obtains the masked message, i.e., $m + r \in \mathcal{R}_q$, which hides m in an information-theoretic way (it is a one-time pad). This is fundamental for the protocol's security. The decryption can be considered a noise remover, which refreshes the noise in the message. Then, A encrypts the masked message $m + r$ and sends [m+r], which has small noise, to B. Ultimately, B homomorphically removes the randomness r, which yields encrypted message $[m]$, with small noise, under A's public key, so m is kept secret from B.

Along with noise reduction, we aim to reduce the scaling factor in the ciphertext. We use Algorithm 2 to reduce the scaling factor in the message. This procedure is similar to the rescaling procedure in CKKS. A difference is that we reduce the scaling in plaintext space while CKKS runs the rescale in ciphertext space. Hence, we reduce the inference time for the SNNI system. In the protocol, the input ciphertext $[m, \Delta_L, L]$ is the encryption of m with scaling factor Δ_L. At the end of the protocol, B receives a new ciphertext $[m', \Delta_0, 0]$, which is the encryption of m' with scaling factor Δ_0. Note that $\frac{m}{\Delta_L} \approx \frac{m'}{\Delta_0}$. However, CKKS inherently works on approximate numbers, so our rescale algorithm can be considered equivalent to CKKS in this scenario. However, the approximation may be a problem in the BFV scheme. We discuss the problem in the next section. The security of the DoubleR-CKKS protocol follows from Lemma 1. See Appendix B for the details of the proof.

As can be seen, DoubleR leaks Δ_L and Δ_0. It means we reveal the number of consecutive multiplication (L) runs in the server before refreshing. Nevertheless, the server can randomly choose the time to refresh the ciphertext, so this number is random from the view of the client. Besides, DoubleR leaks the number of neurons in the refreshing layer. However, at a computational expense, we can pad the layer with zero neurons, which hides the exact number of neurons in the layer. Our padding technique is similar to the technique used in Gazelle [27].

Protocol 1. DoubleR-CKKS protocol

Input A: Secret key sk, public key pk, scaling factor Δ_0, Δ_L
Input B: Public key pk, ciphertext $[m, \Delta_L, L]$, scaling factor Δ_0, Δ_L
Output B: ciphertext $[m', \Delta_0, 0]$, where $m' \approx m$, $e_0 < e_L$, and $\Delta_0 < \Delta_L$

Party B runs following steps:
1: Uniformly picks $r \leftarrow \mathcal{R}_q$ ▷ r is considered as a message with scaling factor Δ_L
2: Computes $[mask, \Delta_L, L] \leftarrow [m, \Delta_L, L] \oplus (r, \Delta_L)$
3: B sends $[mask, \Delta_L, L]$ to A
Party A runs following steps:
4: Computes $(mask, \Delta_L) \leftarrow Decrypt([mask, \Delta_L, L], sk)$
5: Computes $(mask', \Delta_0) \leftarrow PRescale((mask, \Delta_L), \Delta_0)$
6: Computes $[mask', \Delta_0, 0] \leftarrow Encrypt((mask', \Delta_0), pk)$
7: A sends $[mask', \Delta_0, 0]$ to B
Party B runs following steps:
8: Computes $(r', \Delta_0) \leftarrow \text{PRescale}((r, \Delta_L), \Delta_0)$
9: Computes $[m', \Delta_0, 0]_A \leftarrow [mask', \Delta_0, 0] \ominus (r', \Delta_0)$

Lemma 1. *Protocol 1 is secure in the semi-honest model.*

Algorithm 2. PRescale

Input: message (m, Δ_L), and new scaling factor Δ_0
Output: new message (m', Δ_0)

1: $m' \leftarrow \left\lfloor \left(m . \frac{\Delta_0}{\Delta_L} \right) \right\rceil$
2: **return** (m', Δ_0)

4.2 DoubleR-BFV

Compared to the CKKS scheme, BFV and other integer-HE schemes do not support the rescale algorithm in ciphertext space because they are not equipped with a divisor operator. To the best of our knowledge, DoubleR-BFV is the first method to solve this problem for the BFV scheme. We denote a ciphertext by $[m, \Delta_L, e_L]$ where m is the message, Δ_L, and e_L is the scaling factor and noise in ciphertext, respectively. The DoubleR-BFV is shown in Protocol 3. The security of the protocol is quite similar to DoubleR-CKKS. A difference is that BFV uses different polynomial rings for plaintext $\mathcal{R}_t$ and ciphertext $\mathcal{R}_q$. Therefore, selecting randomness r is more challenging compared to DoubleR-CKKS. For security, r needs to be big enough to mask the message m, but if r is too big, the homomorphic addition between the message m and randomness r (line 2) may result in a polynomial that exceeds plaintext space $\mathcal{R}_t$ which will yield an incorrect decryption result. Our strategy for choosing r is as follows: the server holds the pre-trained model, the training data, and the scaling factor Δ_L, which can know the range of m. Then, r is randomly chosen in the space $\mathcal{R}_t^*$ with t^* bigger than the maximum of m. This selection guarantees that r can mask the message m. Finally, the server chooses t, which is bigger than the maximum of $r + m$, which guarantees the protocol's correctness. Besides, as mentioned above, the rounding in our rescale algorithm loses some precision in the message, which may cause accuracy loss in SNNI. Parameter selection is the most complicated part of DoubleR-BFV. In Sect. 6, we show that we boost the inference time with appropriate parameters' selection without degrading the neural network's accuracy. The security of DoubleR-BFV is shown in Lemma 2, which is straightforward based on the security of Double-CKKS. Our protocol can naturally extend to other integer-HE schemes, e.g., BGV scheme [11].

Protocol 3. DoubleR-BFV protocol

Input A: Secret key sk, public key pk, scaling factor Δ_0, Δ_L
Input B: Public key pk, ciphertext $[m, \Delta_L, e_L]$, scaling factor Δ_0, Δ_L
Output B: ciphertext $[m', \Delta_0, e_0]$ where $m' \approx m$, $e_0 < e_L$, and $\Delta_0 < \Delta_L$

Party B runs following steps:
Uniformly picks $r \leftarrow \mathcal{R}_{t^*}$ ▷ r are considered message with scaling factor Δ_L
Computes $[mask, \Delta_L, e_L] \leftarrow [m, \Delta_L, e_L] \oplus (r, \Delta_L)$
B sends to A $[mask, \Delta_L, e_L]$
Party A runs following steps:
Computes $(mask, \Delta_L) \leftarrow Decrypt\left([mask, \Delta_L, e_L], sk\right)$
Computes $(mask', \Delta_0) \leftarrow PRescale\left((mask, \Delta_L), \Delta_0\right)$
Computes $[mask', \Delta_0, e_0] \leftarrow Encrypt\left((mask', \Delta_0), pk\right)$
A sends to B $[mask', \Delta_0, e_0]$
Party B runs following steps:
Computes $(r', \Delta_0) \leftarrow \text{PRescale}((r, \Delta_L), \Delta_0)$
Computes $[m', \Delta_0, e_0]_A \leftarrow [mask', \Delta_0, e_0] \ominus (r', \Delta_0)$

Lemma 2. *Protocol 3 is secure in the semi-honest model.*

5 HeSUN: Homomorphic Encryption for Secure Unbounded Neural Network Inference Framework

This section presents our framework, HeSUN, which leverages our DoubleR protocol for SNNI. Thanks to DoubleR, we achieve an unbounded, secure computation framework. We first set up the SNNI problem we tackle in this paper. We consider the typical scenario of cloud-driven prediction services. In this context, a server possesses a neural network model, and clients provide their data to receive the corresponding predictions. The neural network inference is defined as:

$$z := f_L\left(W_L, f_{L-1}\left(...f_1\left(W_1, X, B_1\right)\right), B_L\right)$$

where f_L is the function evaluated in the L-th layer, and W_L and B_L are the weight and bias parameters used in that layer. The problem we tackle is SNNI: At the end of the process, the server learns nothing about X, and the client learns nothing about W_i, b_i and f_i with $i = \overline{1, n}$ except z. Briefly summarized, our framework, HeSUN, refreshes the intermediate layer values regularly without revealing these values. The details of HeSUN are shown in Protocol 4, and its security follows Lemma 3. We prove the security using modular sequential composition introduced in [13]. The proof details are in Appendix B.

Protocol 4. HeSUN framework

Input Client: Data X, secrete key sk and public key pk
Input Server: $f = f_1 \circ f_2 ... \circ f_n$, public key pk, trained weights $W_1, W_2, ...W_n$
Output Client: $f(W, X)$

1: Client encrypts data X and send $[X]$ to server
2: Server computes $[h_1] = f_1([W_1], [X], B_1)$
3: **for** $i = 1$ to $n - 1$ **do**
4: $\quad [h_i'] \leftarrow \text{DoubleR}([h_i])$ ▷ Refresh the intermediate layer values
5: $\quad$ Server computes $[h_{i+1}] = f_{i+1}([W_{i+1}], [h_i'], B_{i+1})$
6: Server sends $[h_n]$ to the Client
7: Client decrypts $[h_n]$ and obtains z

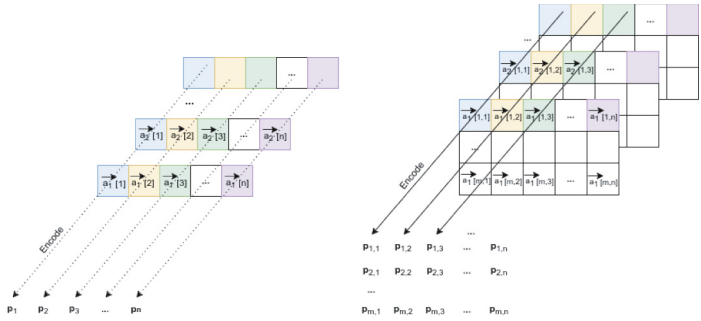
Lemma 3. *The HeSUN framework is secure in the semi-honest model.*

The above section presents a simple version of the HeSUN framework where the DoubleR protocols are called after every layer. In practice, HeSUN's variants can be created where DoubleR is called at different times, which may help reduce communication costs. We analyze the performance of HeSUN's variants in Sect. 6. The security proof of these variants is straightforward, according to Lemma 3.

The following sections provide details on the building blocks of the HeSUN framework.

5.1 HE-SIMD Packing

HeSUN uses BFV and CKKS schemes, which support SIMD [40], which we call HE-SIMD. In simple words, a vector can be encoded as a single plaintext and hence encrypted as a single ciphertext, and operations on this ciphertext are equivalent to the same HE-SIMD operation on the values in the vector individually. This packing method is the key component for processing multiple records simultaneously, enhancing neural network inference throughput. HeSUN utilizes HE-SIMD packing across the mini-batch dimension, as in [21]. In particular, given a 4D tensor with shape (N, C, H, W) format (batch size, channels, height, width), which typically requires $N \times C \times H \times W$ ciphertexts, HeSUN uses HE-SIMD packing to store the tensor as a 3D tensor of $C \times H \times W$ ciphertexts, with each ciphertext packing N values. In Fig. 1, considering single channel data, we show how HeSUN packs values of vectors/matrices, where values of the j-th pixel of vectors, i.e., $p_i[j]$, where $i = \overline{1, N}$, are packed into one plaintext p_j. For the degree N of the polynomial modulus $\left(X^N + 1\right)$, in the BFV scheme, we can pack N integer values into one ciphertext, while in the CKKS scheme, we can pack $N/2$ real values [15,19].



(a) 1D data sample (b) 2D data sample

Fig. 1. HE-SIMD packing.

5.2 Neural Network Layers

Convolution Layer. A convolutional layer consists of filters that act on input values. In the first layer, this input is a raw image. Convolutional layers aim to extract features from the given image. Every filter is an $n \times n$ square that moves with a certain stride. By convolving the image pixels, we compute the dot product between filter values and the values adjacent to a particular pixel. Since this procedure solely involves addition and multiplication, we can use the same computation over the encrypted data.

Approximate Activation Function. After a convolutional layer, an activation function is applied. This function is non-linear and operates on individual numbers by applying a consistent mathematical transformation. In practice, there's a range of activation functions one might come across, such as ReLU ($ReLU\,(x) = max\,(0, x)$), Sigmoid ($\sigma = \frac{1}{1+e^{-x}}$), and Tanh($2\sigma\,(2x) - 1$). Computing these functions over encrypted values is challenging, and we need alternative functions that rely solely on addition and multiplication. In this paper, following the approach of [21], we use the square function, represented as $sqrt(z) := z^2$, as our activation function.

Linear Layer. The linear layer can be considered matrix multiplication between (encrypted) vectors with weights matrix and addition with bias vector. Like the convolution layer, this layer can be directly computed over encrypted data.

5.3 Ciphertext Refreshing

HeSUN uses DoubleR-CKKS and DoubleR-BFV to refresh the intermediate layers' values. As mentioned in Sect. 4, the DoubleR-BFV protocol introduces some errors in the ciphertext refreshing process. Therefore, it requires careful parameter selection to minimize the accuracy loss in SNNI. The details of parameter

selection are presented in Sect. 6. The experiments show that HeSUN efficiently improves inference time without degrading the neural network accuracy.

6 Experiments

We first test our DoubleR protocol (Sect. 6.1) and then evaluate our HeSUN framework (Sect. 6.2). We run experiments on a PC with a single Intel i9-13900K running at 5.80 GHz and 64 GB of RAM, running Ubuntu 22.04. We used OpenFHE [2] for implementation, which supports both BFV and CKKS schemes. All parameters in the below sections are chosen to comply with HE standards recommendation [4], which satisfies 128 bits of security. Overall, the results presented below show that:

- Our DoubleR protocol is super fast compared to the current bootstrapping procedure.
- Considering computation, all HeSUN variants outperform the current HE-based approach for SNNI without degrading the accuracy of the neural network.
- Considering communication, by appropriate parameter selection, HeSUN is more efficient than the current non-interactive HE-based approaches.

6.1 DoubleR Experimental Results

In the HE scheme, our refreshing protocol is equivalent to the bootstrapping procedure, which both aim to reduce the noise in ciphertexts. We compare DoubleR-CKKS with the bootstrapping technique for CKKS [3], implemented in OpenFHE. We report the result for refreshing noise for 1 ciphertext. To ensure the reliability of the results, we repeat the experiment 1000 times and take the average result. The comparison results are shown in Table 2. The depth is the number of expected homomorphic multiplications performed before noise refreshing, N is the degree of the polynomial used in the CKKS scheme, and the level is the multiplicative level required for noise refreshing. Bootstrapping homomorphically computes the decryption equation in the encrypted domain. The procedure consumes an amount of homomorphic multiplication, so the level of the CKKS scheme need to be big enough for this procedure. The detail of the bootstrapping procedure is out of the scope of this work. Please refer to [3] for details. While DoubleR (Protocol 1) only requires simple computation, i.e., encryption, decryption, addition, and subtraction, we do not need to increase the level of CKKS. The larger level of CKKS leads to a bigger value of N to meet the security requirement. With simpler operators and smaller parameters, DoubleR is much faster than bootstrapping. For example, with a depth of 1, DoubleR is nearly 300 times faster than bootstrapping. For the depth of 4, with the same N, DoubleR is nearly 225 times faster than bootstrapping. The bootstrapping implementation for BFV is presented in [14]. Unfortunately, the implementation was never publicly released, so we can not produce a comparison between DoubleR-BFV and the bootstrapping of the BFV scheme. In Sect. 6.2, we show that DoubleR-BFV is very efficient and even faster than Double-CKKS.

Table 2. Comparison between Double-CKKS protocol vs Bootstrapping.

Depth	Runtime (seconds)		N		Level	
	DoubleR-CKKS	Bootstrapping	DoubleR-CKKS	Bootstrapping	DoubleR-CKKS	Bootstrapping
1	0.04	11.85	16384	65536	1	22
2	0.05	12.71	16384	65536	2	23
4	0.07	14.69	16384	65536	4	25
8	0.19	17.39	32768	65536	8	29
16	0.10	22.51	65536	65536	16	37

6.2 HeSUN Experimental Results

As shown in Table 1, while HeSUN satisfies the security and unbounded criteria, efficiency is the remaining consideration in HeSUN. In particular, HeSUN requires communication between the client and the server. This section aims to compare HeSUN with the LHE-based approach, which is a non-interactive one. The idea of LHE-based frameworks is based on the seminal paper CryptoNets [21], which chose an LHE scheme where the depth is specified in advance based on the depth of the neural network. We call the LHE-based approach is CryptoNets. We name our approach in the format HeSUN-[scheme]-[refresh frequency]; for example, HeSUN-CKKS-1 means the HeSUN framework is based on the CKKS scheme, and the ciphertext is refreshed after every single layer. We evaluated the performance of HeSUN on the MNIST dataset [28], which has a standard split containing 28×28 grayscale images of Arabic numerals 0 to 9 of 50,000 training images and 10,000 test images. MNIST is the standard benchmark for homomorphic inference tasks [8,9,21,24].

To provide a fair comparison, we ran experiments on the same neural network with the same machine configuration. The neural network consists of convolutional layers, activation functions, and fully-connected layers. Our objective was to take a basic neural network and evaluate the performance of HeSUN. We achieved an accuracy of 98.95%, matching that reported in [21]. The details of the network architecture is presented below.

1. Convolution layer: The input image is 28, this layer employs 5 kernels, each 3×3 in size, with a stride of 2 and no padding. The output is a $5 \times 13 \times 13$ tensor.
2. Square layer: Every value from the input node is squared in this layer.
3. Linear layer: A fully connected layer connects the 845 incoming nodes to 100 outgoing nodes. This can be seen as the multiplication involving a 845×100 matrix.
4. Square layer: Every value from the input node is squared in this layer.
5. fully connected layer connects the 100 incoming nodes to 10 outgoing nodes. This can be seen as the multiplication involving a 100×10 matrix.
6. Sigmoid Activation Function: This layer applies the sigmoid function to each of the 10 incoming values.

The last sigmoid activation function can be removed in the inference phase without interfering with prediction accuracy. This is because the prediction of

the neural network is given by the index of the maximum value of its output vector, and since the sigmoid function is monotone increasing, whether or not we apply it will not affect the prediction. The following sections present the inference results of HeSUN-BFV and HeSUN-CKKS.

Parameters. *HeSUN-BFV.* The main parameters defining the BFV scheme are the degree N of the polynomial modulus $\left(X^N + 1\right)$, the plaintext modulus t, and the coefficient modulus q. In HeSUN, we chose $N = 8192$, which allows packing 8192 integers into a plaintext. Parameter t must be large enough so that during the inference process, no number wraps around t. CryptoNets [21] uses $t = 40961 \times 65537 \times 114689 \times 147457 \times 188417$ (the factors are combined using the Chinese Reminder Theorem), which corresponds to 5 separate BFV schemes (each scheme has plaintext modulus corresponds to a factor). The selection of t guarantees that t is greater than 2^{80}, which is large enough for precision. The reason for this big value of t is that the scaling factor increases rapidly during inference (details in Table 3). It is worth noting that these t values do not comply with HE standards recommendation [4]. It is understandable because CryptoNets was proposed before the HE standard was released. Hence, we select the other 5 values of t, which comply with HE standards and are big enough for the necessary precision. These t values can be seen in Appendix, Table 4. Besides, for a fair comparison, we implement a new version of CryptoNets called CryptoNet*, which only needs 2 values of t to satisfy the precision and security requirements.

In the HeSUN-BFV, the output of intermediate layers is refreshed using the DoubleR-BFV protocol to reduce the scaling factor in ciphertexts. Therefore, our approach keeps the scaling factor small during inference time while the scaling factor increases tremendously in CryptoNets (details in Table 3). Based on the bound of scaling factor growth, we can choose a smaller t, which is still large enough for the precision needed. In particular, in HeSUN-BFV-1 and HeSUN-BFV-2 the values of t were 10027009, 1286504449 respectively. HeSUN-3 and HeSUN-4 are not realizable because the scaling factor growth requires a very big t, which exceeds the capacity of current LHE schemes. With scaling factor $\Delta = 10^2$ and these t values, HeSUN-BFV-1, HeSUN-BFV-2 achieve the accuracy of 99.27%, 99.33% which is comparable with the plain neural network (99.95%). Based on the value of plaintext modulus t and the expected multiplication level, q must be chosen to meet the security standards [4]. The details of our choice of parameters can be seen in Table 4.

HeSUN-CKKS. For a fair comparison with the BFV scheme, in the CKKS scheme, we chose N = 16384, which allows packing 8192 values (the same as BFV) into plaintext. Based on expected multiplication depth, q was selected to satisfy 128-bit security. Like HeSUN-BFV, HeSUN-CKKS uses smaller q than CryptoNets. The parameter details are Table 5. With the parameters selection, HeSUN-CKKS achieves the same accuracy as the plain neural network, i.e., 99.95%.

Figure 2 shows the experimental results of the HeSUN framework. Analysis of the communication and computation costs are provided below.

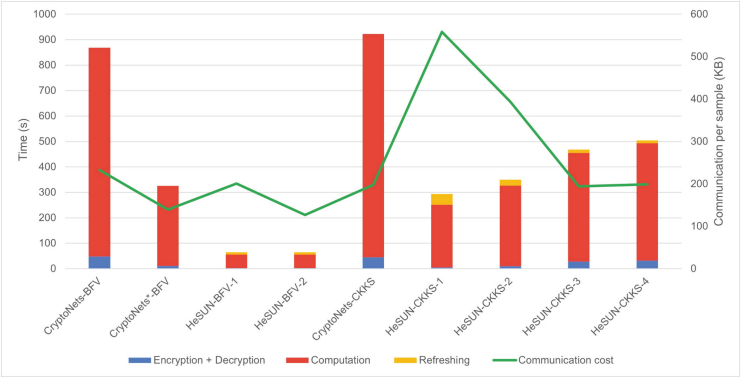


Fig. 2. MNIST inference performance comparisons.

Communication Cost. By communication cost, we mean the size of the message exchanged between the client and the server during the inference process, from when the client sends encrypted images to the server to when the server sends encrypted results to the client. Our refreshing ciphertext protocols, DoubleR-CKKS and DoubleR-BFV, require two communication rounds; each round sends one ciphertext. We can compute communication cost based on the size of polynomial ring N and the size of ciphertext modulus q. Even though HeSUN requires more communication rounds, it achieves comparable communication costs to CryptoNets, a non-interactive one. The reason for such a result is that HeSUN reduces the size of the parameters required. Overall, the communication cost of BFV-based frameworks is smaller than CKKS-based frameworks. The reason is that to pack 8192 numbers, BFV needs $N = 8192$ while CKKS needs $N = 16384$.

BFV-Based Frameworks. HeSUN-BFV-2 has the smallest communication cost, which is even smaller than non-interactive approaches, i.e., CryptoNets and CryptoNets*. There are two reasons for this result. Firstly, HeSUN-BFV-2 allows the use of smaller parameters, i.e., t and q. For the same t, CryptoNets-BFV needs a multiplication level of 5 while HeSUN-2 needs only 1; this reduces q's size. Secondly, HeSUN-BFV refreshes the scaling factor that causes the reduction on the bound of t. For example, HeSUN-BFV-1 uses only one BFV scheme with $t = 10027009$ while CryptoNets and CryptoNets* need to use 5 and 2 BFV separate schemes, respectively, to guarantee precision. Detail of the communication cost for each layer can be seen in Table 6.

CKKS-Based Frameworks. Compared to CryptoNets, we achieved a smaller communication cost with HeSUN-CKKS-3 and a comparable communication cost

with HESUN-CKKS-4. While the communication cost of HeSUN-CKKS-1 and HESUN-CKKS-2 is significantly higher than others. The reason is that HeSUN-CKKS-1 and HESUN-CKKS-2 run the DoubleR protocol for more time. Furthermore, the refresh protocol is run after the convolution layer where the output size is big, i.e., 845 dimensions. Detail of the communication cost for each layer can be seen in Table 7.

Computation Cost. We assume messages arrive at the server(client) immediately after the client (server) sends them. In practice, we need to consider the delay in message transmission. We measure computation cost by total running time at the client and server. Overall, all HeSUN variants outperform CryptoNets. The DoubleR protocol is relatively fast compared to the computation of other layers. HeSUN-BFV-1 and HeSUN-BFV-2 use almost the same HE parameter (N and q), so we only report the computation cost of HeSUN-BFV-2 (the one with bigger t). HeSUN-BFV-1,2 is the fastest framework, which is 14× faster than CryptoNets-BFV and 5× faster than CryptoNets*-BFV. Details of computation cost for each layer can be seen in Table 6 and 7.

Throughput. It is worth noting that CryptoNet-BFV, CryptoNet-BFV*, and HeSUN-BFV-3 use a bigger polynomial ring compared to others to ensure the security requirement. This big parameter slows down the computation. However, this allows us to pack more values in a single plaintext. The same situation occurs in the CKKS-based frameworks. The details of parameters are shown in Table 4 and 5. Therefore, for a fair comparison, we compare the throughput of frameworks. The results are shown in Table 8. The results show that all HeSUN variants get better throuhgput compared to CryptoNets.

Strategy for Ciphertext Refreshing. From the experimental results, we can see the trade-off in HeSUN variants. In general, performing ciphertext refreshing more frequently allows us to use smaller parameters for the HE scheme, which speeds up the computation time, but we pay more for communication costs. The communication cost is also affected by the size of the layer we want to refresh. One strategy to reduce communication costs is to refresh ciphertext at layers with a small number of neurons.

6.3 Towards Unbounded SNNI

We adopt a scale-invariant approach to the SNNI problem where the ciphertexts are refreshed regularly during inference. Therefore, our framework is scalable to deeper neural network architecture and other datasets, but we left it for future research. After CryptoNets, much research focuses on optimization, such as efficient approximation method for activation function [24,25], efficient data encoding method [12,26], and hardware acceleration [37,42]. However, the works are based on LHE, so they cannot infer deep neural networks. In the future, these

improvements can be combined with the HeSUN framework to boost performance and make the inference of unbounded SNNI feasible. In [29], the authors first implemented the standard ResNet-20 model with CKKS scheme[1]. The work used bootstrapping, which caused the computation overhead and relied on circular security. As reported in the paper, the bootstrapping procedure time accounts for 31.55% of total inference time. Our DoubleR protocol can replace the bootstrapping phase to boost the computation performance. As shown in Sect. 6.1, Double-CKKS is $\approx \times 300$ faster than the CKKS bootstrapping procedure. Therefore, theoretically, we can reduce the inference time for ResNet-20 from t (s) to $t - 0.3155t + \frac{0.3155.t}{300} = 0.68555.t$ (s). Extending to unbounded SNNI with the HeSUN-BFV variant needs to consider the precision loss by scaling factor refreshing algorithm to ensure not degrade the model's accuracy.

6.4 Model Extraction Attack

In the context of model extraction attacks, a client with sufficient inference queries can potentially deduce confidential details. This might be parameters of the model (model extraction attack [41]), training data sample with given labels (model inversion attack [1,6]), or presence of a sample in training data (membership inference attack [39]). Current HE-based SNNI frameworks safeguard client data from servers and model weights from clients. Nonetheless, these frameworks are not robust against model extraction attacks because adversaries retain black-box access to inferences. We perceive model extraction attacks as an orthogonal issue to SNNI using cryptographic primitives. Indeed, all the frameworks in Table 1 remain susceptible to such attacks.

7 Conclusion

In this work, we propose a novel double ciphertext refreshing protocol, DoubleR, which reduces noise and scaling factor in the ciphertext and allows for evaluating circuits of arbitrary depth on encrypted data in the context of SNNI. The protocol is proven to be secure in the semi-honest model. The experimental results show that DoubleR outperforms bootstrapping by $300\times$ in running time. Based on DoubleR, we build HeSUN, a versatile framework for SNNI, which significantly accelerates the inference time with comparable communication costs. Our work is a step toward unbounded SNNI.

A Preliminaries for Proofs

A.1 Semantic Security of HE Schemes

Our protocols are proved to be secure based on the semantic security [22] of the homomorphic encryption schemes. In essence, a HE scheme is semantically

[1] The implementation was not publicly released.

secure when no adversary can guess (with better than a 1/2 probability) whether a given ciphertext corresponds to the encryption of one of two different messages. For this, encryption must be randomized, ensuring that distinct encryptions of an identical message are indistinguishable. BFV and CKKS are built upon conventional, thoroughly researched computational assumptions, Ring Learning With Error (RLWE) assumption [34]. Notably, RLWE is considered a hard problem for even quantum computers.

A.2 Secure Two-Party Computation Framework for Semi-honest Adversaries

Let $f = (f_A, f_B)$ be a polynomial function. We consider the following context: two parties, A and B, run a protocol Π to jointly compute the function $f(a, b)$ where a and b are the input of A and B, respectively. The view of A during an execution of Π on (a, b), denoted $V_A(a, b) = \left(a, r^A, m_1^A, ...m_t^A\right)$, where r represents A's random tape, and m_i^A represents the i-th message A has received. The view of B is defined similarly.

In this work, we mainly consider deterministic functions f. According to Definition 7.2.1 in [22], we have the following security definition.

Definition 1 (privacy in semi-honest model). *The two parties protocol Π securely computes a deterministic function f if there exists two probabilistic polynomial time algorithms S_A and S_B such that for every possible input a, b of f,*

$$\{S_A(a, f_A(a, b))\} \equiv_c \{V_A(a, b)\} \tag{1}$$

and

$$\{S_B(b, f_B(a, b))\} \equiv_c \{V_B(a, b)\} \tag{2}$$

where $\equiv_c$ denotes computational indistinguishability against probabilistic polynomial time adversaries with negligible advantage.

From the Definition 1, it is clear that a party's view on each possible pair of inputs can be efficiently simulated using just its input and output. This means that everything the party observes from the full transcript of a legitimate execution is essentially inferred from its output (coupled with its corresponding input). In other words, any knowledge the party acquires from the protocol execution is fundamentally implied in the output itself, leaving the party with only the output as new knowledge. In short, for the security proof of a protocol Π, we must demonstrate the existence of simulators S_A and S_B that conform to Eqs. (1) and (2).

A.3 Modular Sequential Composition

Our framework, HeSUN (Sect. 5), is created by composing sequential secure protocols. To prove the security of our framework, we invoke modular sequential

composition, as introduced in [13]. The idea is that the parties implement a protocol Π while making calls to an ideal functionality f within Π. For instance, A and B privately compute f by sending their inputs to a trusted third party and obtaining the result. If we can prove that Π is secure in the semi-honest model and if we have a protocol ρ that privately computes f in the same model, then the abstract calls to f can be replaced by the execution of ρ within Π; the new protocol denoted Π^{ρ} is subsequently proven to be secure in the honest-but-curious model.

We refer to a *hybrid model with ideal access to* $f_1, ..., f_m$, or *the* $(f_1, ..., f_m)$-*hybrid model*, as an augmentation of the semi-honest model with an incorruptible trusted party T, responsible for assessing functionalities $f_1, ..., fm$. The involved parties execute a protocol Π that involves making calls to T to evaluate one of the functionalities $f_1, ..., f_m$. Every participating party transmits its input in each call and waits until T responds. It is crucial to underline that parties are permitted to exchange information only after they have received the response from T (we focus exclusively on sequential composition). It is worth noting that the ideal calls to T can be invoked several times, even for the same function, but each call operates independently; T does not retain any state or memory between two calls.

Consider Π as a two-party protocol in the $(f_1, ..., f_m)$-hybrid model. Let define $\rho_1, ..., \rho_m$ as real protocols, which are protocols in the semi-honest model, and compute $f_1, ..., f_m$, respectively. Now, let's define $\Pi^{\rho_1,...,\rho_m}$ as follows. Whenever Π makes ideal calls to T for f_i, those calls are substituted with the actual execution of ρ_i. For example, when party P_j needs to compute f_i with input x_j, P_j temporarily suspends its actions, starts to run ρ_i with other parties, obtains the outcome β_j once ρ_i finished, and continues as if β_j was received directly from T.

Theorem 1. *(Theorem 5) restated as in [31] (Theorem 3) - Let* $f_1, ..., f_m$ *be two-party probabilistic polynomial time functionalities and* $\rho_1, ..., \rho_m$ *protocols that compute respectively* $f_1, ..., f_m$ *in the presence of semi-honest adversaries.*

Let g *be a two-party probabilistic polynomial time functionality and* Π *a protocol that securely computes* g *in the* $(f_1, ..., f_m)$*-hybrid model in the presence of semi-honest adversaries.*

Then $\Pi^{\rho_1,...,\rho_m}$ *securely computes* g *in the presence of semi-honest adversaries.*

B Proofs

Proof (of Lemma 1). We prove the security of our protocol follows the Definition 1. For simplification, we remove the noise term and scaling factor term in ciphertexts.

A's view is $V_A = (sk, pk; coins; [mask])$ where $coins$ is the random coins used for the encryption of $[mask]$. Given (sk, pk) the simulator S_A run as follows:

1. Uniformly picks $\widetilde{mask} \leftarrow \mathcal{R}_q$

2. Let coins be random $\widetilde{coins}$ for one HE encryption.
3. Output $\left(sk, pk; \widetilde{coins}; \left[\widetilde{mask}\right]\right)$

The random tapes $coins$ and $\widetilde{coins}$ are generated in the exact same manner and independently from any parameter, so

$$\left(sk, pk; \widetilde{coins}; \left[\widetilde{mask}\right]\right) = \left(sk, pk; coins; \left[\widetilde{mask}\right]\right)$$

Recall that we have $mask = m + r \in \mathcal{R}_q$ (at line 2) where $m \in \mathcal{R}_q$ and $r \leftarrow \mathcal{R}_q$. Therefore, the distribution of $\widetilde{mask}$ is statistically indistinguishable from the distribution of $mask$ so we directly have $\left(sk, \left[\widetilde{mask}\right]\right) \equiv_s (sk, [mask])$. We have:

$$\left(sk, pk; \widetilde{coins}; \left[\widetilde{mask}\right]\right) \equiv_s (sk, pk; coins; [mask])$$

and

$$S_A\left(sk, pk\right) \equiv_c V_A\left(sk, pk, [m]\right) \tag{3}$$

B's view is $V_B = \left([m], pk; r; [mask']\right)$. Given $\left([m], pk\right)$, we build simulator S_B:

1. Uniformly picks $\widetilde{r}, \widetilde{mask'} \leftarrow \mathcal{R}_q$
2. Output $\left([m], pk; \widetilde{coins}, \widetilde{r}; \left[\widetilde{mask'}\right]\right)$

$\widetilde{r}$ and r are indistinguishable because the randomness are uniformly taken from the same space $\mathcal{R}_q$, so

$$\left([m], pk; \widetilde{r}; \left[\widetilde{mask'}\right]\right) = \left([m], pk; r; \left[\widetilde{mask'}\right]\right)$$

By semantic security of the HE scheme,

$$\left([m], pk; \widetilde{r}; \left[\widetilde{mask'}\right]\right) = \left([m], pk; r; [mask']\right)$$

and

$$S_B\left([m], pk\right) \equiv_c V_B\left(sk, pk, [m]\right) \tag{4}$$

The security is obtained using Eqs. 3 and 4 and the Definition 1.

Proof (of Lemma 3). We suppose the refreshed ciphertexts $[h'_i]$ is ideally computed (using calls to a trusted party in the hybrid model). We show that the HeSUN protocol is secure in this model and conclude using the sequential modular composition theorem (Theorem 1).

A's view is $V_A = (sk, pk, X; [h_n])$. Given $(sk, pk, X; z)$ we simply build the simulator S_A by computing $[z]$ the encryption of z and output $(sk, pk, X, [z])$. Because the ciphertext encrypts the same message, the following distributions are the same:

$$\mathrm{S}_A\left(sk, pk, X, z\right) = V_A\left(sk, pk, X, f_1, f_2, ..., f_n, W_1, W_2, ..., W_n, [h_n]\right)$$

B's view is $V_B = (pk, f_1, f_2, ..., f_n, W_1, W_2, ..., W_n; [h_0], [h_1], ..., [h_n])$. Given $(pk, f_1, f_2, ..., f_n, W_1, W_2, ..., W_n)$ we build a simulator S_B as below:

1. Picks $\widetilde{h_0}, \widetilde{h_0}, .., \widetilde{h_n} \leftarrow \mathcal{R}_q$
2. Output $\left(pk, f_1, f_2, ..., f_n, W_1, W_2, ..., W_n; \left[\widetilde{h_0}\right], \left[\widetilde{h_1}\right], ..., \left[\widetilde{h_n}\right]\right)$

By semantic security of the HE scheme, we have

$$S_B\left(pk_A, f_1, f_2, ..., f_n, W_1, W_2, ..., W_n\right) \equiv_c$$
$$V_B\left(sk_A, pk_A, X, f_1, f_2, ..., f_n, W_1, W_2, ..., W_n, [h_n]\right)$$

We conclude the proof of security using modular sequential composition. We replace the ideal calls for computing the refreshed ciphertexts $[h_i]$ by the provable secure DoubleR protocol and invoke Theorem 1 to prove security in the semi-honest model.

C HeSUN for SNNI

C.1 Scaling Factor Growth

Table 3 shows the scaling factor growth after layers in the neural network.

Table 3. Scaling factor growth.

Layer	CryptoNets	HeSUN-BFV-1	HeSUN-BFV-2	HeSUN-BFV-3	HeSUN-BFV-4
Convolution	Δ^2	Δ^2	Δ^2	Δ^2	Δ^2
Square 1	Δ^4	Δ^2	Δ^4	Δ^4	Δ^4
Linear 1	Δ^5	Δ^2	Δ^2	Δ^5	Δ^5
Square 2	Δ^{10}	Δ^2	Δ^4	Δ^2	Δ^{10}
Linear 2	Δ^{11}	Δ^2	Δ^2	Δ^3	Δ^2

C.2 Parameters Selection

Table 4 and Table 5 show the parameters for the frameworks that satisfy 128 bits security.

Table 4. Parameters for BFV-based frameworks.

	Batchsize	Security level	Multiplicative depth	Plaintext modulus	N	q (bit length)	Total q size
CryptoNets	8192	128	5	65537, 114689, 147457, 163841, 557057	16384	240	1200
CryptoNets*	8192	128	5	1099511922689, 1099512004609	16384	360	720
HeSUN-2	8192	128	2	1286504449	8192	180	180
HeSUN-1	8192	128	1	10027009	8192	180	180

C.3 Performance Details

Table 6 and Table 7 show the performance of frameworks on MNIST dataset. We detail the results of each step in the inference process.

Table 5. Parameters for CKKS-based frameworks.

	Batchsize	Security level	Multiplicative depth	N	q (bit length)
CryptoNets	8192	128	5	32768	510
HeSUN4	8192	128	4	16384	410
HeSUN3	8192	128	3	16384	400
HeSUN2	8192	128	2	16384	300
HeSUN1	8192	128	1	16384	250

Table 6. MNIST inference performance comparisons of BFV-based framework.

Layer	(exchanged) message size (per instance) (KB)								Times		
	CryptoNets		CryptoNets*		HeSUN-BFV-1		HeSUN-BFV-2		CryptoNets	CryptoNets*	HeSUN-BFV-1,2
	C2S	S2C	C2S	S2C	C2S	S2C	C2S	S2C			
Encoding+Encryption	–	–	–	–	–	–	–	–	47.05	10.51	2.17
The client sends data to the Server	229.69	–	137.81	–	34.45	–	34.45	–	–	–	–
Convolution	–	–	–	–	–	–	–	–	60.18	20.85	5.01
Refresh 1	–	–	–	–	37.13	37.13	–	–	–	–	3.93
Square 1	–	–	–	–	–	–	–	–	46.01	13.04	3.25
Refresh 2	–	–	–	–	37.13	37.13	37.13	37.13	–	–	3.52
Linear 1	–	–	–	–	–	–	–	–	699.12	268.77	44.03
Refresh 3	–	–	–	–	4.39	4.39	4.39	4.39	–	–	0.68
Square 2	–	–	–	–	–	–	–	–	6.36	2.67	0.39
Refresh 4	–	–	–	–	4.39	4.39	4.39	4.39	–	–	0.51
Linear 2	–	–	–	–	–	–	–	–	9.02	9.88	0.69
The server sends data to the Client	–	2.93	–	1.76	–	0.44	–	0.44	–	–	–
Decryption+Decoding	–	–	–	–	–	–	–	–	0.08	0.02	0.01
Total	229.69	2.93	137.81	1.76	117.49	83.48	80.36	46.35	867.82	325.74	64.19
	232.62		139.57		200.97		126.71				

Table 7. MNIST inference performance comparisons of CKKS-based framework.

Layer	(exchanged) message size (per instance) (KB)										Times				
	CryptoNets-CKKS		HeSUN-CKKS-1		HeSUN-CKKS-2		HeSUN-CKKS-3		HeSUN-CKKS-4		CryptoNets	HeSUN-CKKS-1	HeSUN-CKKS-2	HeSUN-CKKS-3	HeSUN-CKKS-4
	C2S	S2C	C2S	S2C	C2S	S2C	C2S	S2C	C2S	S2C					
Encoding + Encryption	–	–	–	–	–	–	–	–	–	–	44.22	4.65	8.86	27.21	31.22
The Client sends data to the Server	195.23	–	95.7	–	114.84	–	153.13	–	156.95	–	–	–	–	–	–
Convolution	–	–	–	–	–	–	–	–	–	–	71.84	13.51	25.67	41.48	49.79
Refresh 1	–	–	103.15	103.15	–	–	–	–	–	–	–	15.97	–	–	–
Square 1	–	–	–	–	–	–	–	–	–	–	42.06	6.02	9.07	12.45	15.99
Refresh 2	–	–	103.15	103.15	123.78	123.78	–	–	–	–	–	15.28	19.72	–	
Linear 1	–	–	–	–	–	–	–	–	–	–	815.41	180.47	251.342	349.94	371.46
Refresh 3	–	–	12.21	12.21	–	–	19.53	19.53	–	–	–	5.03	–	12.53	–
Square 2	–	–	–	–	–	–	–	–	–	–	10.83	1.39	3.94	4.97	5.27
Refresh 4	–	–	12.21	12.21	14.65	14.65	–	–	20.02	20.02	–	5.86	3.16	–	11.14
Linear 2	–	–	–	–	–	–	–	–	–	–	10.77	2.73	5.01	6.61	7.57
The Server sends data to the Client	–	2.49	–	1.22	–	1.46	–	1.95	–	2	–	–	–	–	–
Decryption + Decoding	–	–	–	–	–	–	–	–	–	–	0.57	0.04	0.15	0.21	0.23
Total	195.23	2.49	326.42	231.94	253.27	139.89	172.66	21.48	176.97	22.02	995.7	250.95	326.922	455.4	492.67
	197.72		558.36		393.16		194.14		198.99						

Table 8. MNIST inference throughput comparison (images per second).

	CryptoNets-BFV	CryptoNets*-BFV	HeSUN-BFV-1	HeSUN-BFV-2	CryptoNets-CKKS	HeSUN-CKKS-1	HeSUN-CKKS-2	HeSUN-CKKS-3	HeSUN-CKKS-4
Encoding + Encryption	348	1,558	3,775	3,775	741	3,523	1,849	602	524
Computation	19	51	132	132	34	56	48	37	34
Decryption + Decoding	204,800	819,200	819,200	819,200	57,487	409,600	109,226	78,019	71,234

References

1. Aïvodji, U., Gambs, S., Ther, T.: Gamin: an adversarial approach to black-box model inversion. arXiv preprint arXiv:1909.11835 (2019)
2. Al Badawi, A., et al.: OpenFHE: open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 53–63 (2022)
3. Al Badawi, A., Polyakov, Y.: Demystifying bootstrapping in fully homomorphic encryption. Cryptology ePrint Archive (2023)
4. Albrecht, M., et al.: Homomorphic encryption standard. In: Lauter, K., Dai, W., Laine, K. (eds.) Protecting Privacy through Homomorphic Encryption, pp. 31–62. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77287-1_2
5. Archer, D.W., et al.: RAMPARTS: a programmer-friendly system for building homomorphic encryption applications. In: Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 57–68 (2019)
6. Bekman, T., Abolfathi, M., Jafarian, H., Biswas, A., Banaei-Kashani, F., Das, K.: Practical black box model inversion attacks against neural nets. In: Kamp, M., et al. (eds.) ECML PKDD 2021. CCIS, vol. 1525, pp. 39–54. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-93733-1_3
7. Boemer, F., Cammarota, R., Demmler, D., Schneider, T., Yalame, H.: MP2ML: a mixed-protocol machine learning framework for private inference. In: Proceedings of the 15th International Conference on Availability, Reliability and Security, pp. 1–10 (2020)
8. Boemer, F., Costache, A., Cammarota, R., Wierzynski, C.: nGraph-HE2: a high-throughput framework for neural network inference on encrypted data. In: Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 45–56 (2019)
9. Boemer, F., Lao, Y., Cammarota, R., Wierzynski, C.: nGraph-HE: a graph compiler for deep learning on homomorphically encrypted data. In: Proceedings of the 16th ACM International Conference on Computing Frontiers, pp. 3–13 (2019)
10. Bourse, F., Minelli, M., Minihold, M., Paillier, P.: Fast homomorphic evaluation of deep discretized neural networks. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 483–512. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_17
11. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory (TOCT) **6**(3), 1–36 (2014)
12. Brutzkus, A., Gilad-Bachrach, R., Elisha, O.: Low latency privacy preserving inference. In: International Conference on Machine Learning, pp. 812–821. PMLR (2019)
13. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**, 143–202 (2000)
14. Chen, H., Han, K.: Homomorphic lower digits removal and improved FHE bootstrapping. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 315–337. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_12
15. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15

16. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**(1), 34–91 (2020)
17. Clet, P.-E., Stan, O., Zuber, M.: BFV, CKKS, TFHE: which one is the best for a secure neural network evaluation in the cloud? In: Zhou, J., et al. (eds.) ACNS 2021. LNCS, vol. 12809, pp. 279–300. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81645-2_16
18. Dathathri, R., et al.: CHET: an optimizing compiler for fully-homomorphic neural-network inferencing. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 142–156 (2019)
19. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
20. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, pp. 169–178 (2009)
21. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: CryptoNets: applying neural networks to encrypted data with high throughput and accuracy. In: International Conference on Machine Learning, pp. 201–210. PMLR (2016)
22. Goldreich, O.: Foundations of Cryptography: Volume 2, Basic Applications. Cambridge University Press, Cambridge (2009)
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
24. Hesamifard, E., Takabi, H., Ghasemi, M.: CryptoDL: deep neural networks over encrypted data. arXiv preprint arXiv:1711.05189 (2017)
25. Ishiyama, T., Suzuki, T., Yamana, H.: Highly accurate CNN inference using approximate activation functions over homomorphic encryption. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 3989–3995. IEEE (2020)
26. Jiang, X., Kim, M., Lauter, K., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1209–1222 (2018)
27. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: GAZELLE: a low latency framework for secure neural network inference. In: 27th USENIX Security Symposium (USENIX Security 2018), pp. 1651–1669 (2018)
28. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
29. Lee, J.W., et al.: Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. IEEE Access **10**, 30039–30054 (2022)
30. Lehmkuhl, R., Mishra, P., Srinivasan, A., Popa, R.A.: Muse: secure inference resilient to malicious clients. In: USENIX Security Symposium, pp. 2201–2218 (2021)
31. Lindell, Y.: Secure multiparty computation for privacy preserving data mining. In: Encyclopedia of Data Warehousing and Mining, pp. 1005–1009. IGI Global (2005)
32. Liu, J., Juuti, M., Lu, Y., Asokan, N.: Oblivious neural network predictions via minionn transformations. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 619–631 (2017)
33. Lou, Q., Jiang, L.: SHE: a fast and accurate deep neural network for encrypted data. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
34. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM (JACM) **60**(6), 1–35 (2013)

35. Mohassel, P., Zhang, Y.: SecureML: a system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 19–38. IEEE (2017)
36. Podschwadt, R., Takabi, D., Hu, P., Rafiei, M.H., Cai, Z.: A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption. IEEE Access **10**, 117477–117500 (2022)
37. Reagen, B., et al.: Cheetah: optimizing and accelerating homomorphic encryption for private inference. In: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 26–39. IEEE (2021)
38. Riazi, M.S., Samragh, M., Chen, H., Laine, K., Lauter, K.E., Koushanfar, F.: XONN: XNOR-based oblivious deep neural network inference. In: USENIX Security Symposium, pp. 1501–1518 (2019)
39. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 3–18. IEEE (2017)
40. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Crypt. **71**, 57–81 (2014)
41. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction APIs. In: USENIX Security Symposium, vol. 16, pp. 601–618 (2016)
42. Yang, Y., Kuppannagari, S.R., Kannan, R., Prasanna, V.K.: FPGA accelerator for homomorphic encrypted sparse convolutional neural network inference. In: 2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 1–9. IEEE (2022)
43. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)