# Rhychee-FL: Robust and Efficient Hyperdimensional Federated Learning with Homomorphic Encryption

Yujin Nam*, Abhishek Moitra†¶, Yeshwanth Venkatesha†¶, Xiaofan Yu*¶, Gabrielle De Micheli*,
Xuan Wang*, Minxuan Zhou‡, Augusto Vega§, Priyadarshini Panda†, Tajana Rosing*
**CSE, UC San Diego* †*ECE, Yale University* ‡*CS, Illinois Institute of Technology* § *IBM*
¶ Contributed Equally
{yujinnam, x1yu gdemicheli, xuw009, tajana}@ucsd.edu
{abhishek.moitra, yeshwanth.venkatesha, priya.panda }@yale.edu
mzhou26@iit.edu, ajvega@us.ibm.com

*Abstract*—**Federated learning (FL) is a widely-used collaborative learning approach where clients train models locally without sharing their data with servers. However, privacy concerns remain since clients still upload locally trained models, which could reveal sensitive information. Fully homomorphic encryption (FHE) addresses this issue by enabling clients to share encrypted models and the server to aggregate them without decryption. While FHE resolves the privacy concerns, the encrypted data introduces larger communication and computational complexity. Moreover, ciphertexts are vulnerable to channel noise, where a single bit error can disrupt model convergence. To overcome these limitations, we introduce Rhychee-FL, the first lightweight and noise-resilient FHE-enabled FL framework based on Hyperdimensional Computing (HDC), a low-overhead training method. Rhychee-FL leverages HDC's small model size and noise resilience to reduce communication overhead and enhance model robustness without sacrificing accuracy or privacy. Additionally, we thoroughly investigate the parameter space of Rhychee-FL and propose an optimized system in terms of computation and communication costs. Finally, we show that our global model can successfully converge without being impacted by channel noise. Rhychee-FL achieves comparable final accuracy to CNN, while reaching $90\%$ accuracy in $6\times$ fewer rounds and with $2.2\times$ greater communication efficiency. Our framework shows at least $4.5\times$ faster client side latency compared to previous FHE-based FL works.**

## I. Introduction

Federated learning (FL) is a distributed learning approach where multiple clients collaboratively train a model without sharing their local data. FL has gained significant interest in fields such as Internet of Things (IoT), finance, healthcare, and smart cities [1]. In each global round, participants train their models locally and share them with a central server, which then aggregates the models to combine their knowledge. FL ensures data privacy by keeping raw data on local devices.

However, FL does not address all privacy concerns. Sharing local models with the server can still expose sensitive information or lead to misuse [1]. Fully homomorphic encryption (FHE), an encryption scheme that supports arbitrary computations over encrypted data, addresses this issue [2]. With FHE, clients can encrypt their local models before sharing, enabling the server to aggregate these models without decryption.

Although FHE ensures model privacy, it also introduces several challenges. First, computations and communications in FHE are significantly more expensive than in plaintext. For instance, a single multiplication in the FHE domain can take more than hundreds or thousands of operations, and an 8-bit integer can turn into a ciphertext of around 1KB. These overheads pose significant challenges for FL in resource-limited applications. Additionally, inputs in FHE are typically quantized into plaintext domains, which can degrade accuracy dramatically. The ciphertext size, operation complexity, and quantization precision depend on the chosen FHE parameter and scheme, necessitating design space exploration to manage these costs effectively. Finally, FHE-based FL is prone to communication noise. Even a single bit error in a ciphertext can result in completely incorrect decryption, which may propagate through the system and lead to erroneous outcomes at both the server and client levels. Although previous studies have explored the use of homomorphic encryption in FL [3]–[5], their methods are inefficient in terms of both computation and communication overhead and do not address communication noise.

We introduce Rhychee-FL, the first FHE-enabled FL framework that excels in lightweight training, efficient communication and robustness to communication noises. Rhychee-FL is designed based on hyperdimensional computing (HDC), which is a lightweight computing paradigm with small model sizes and simple learning procedures. HDC encodes raw data into high-dimensional, low-precision vectors. On client devices, local HDC learning is done through simple element-wise addition, producing class-specific vectors as local models. Rhychee-FL securely exchanges these encrypted class vectors with the server, where they are homomorphically aggregated. In Rhychee-FL, we systematically explore the design space of the scheme selection and parameter settings in FHE to reduce computational and communication overhead. Rhychee-FL also mitigates the precision challenges associated with FHE thanks to the noise resilience of HDC models. Finally, we perform experiments involving communication channels and error detection, making Rhychee-FL the first FL-FHE system to be tested for robustness to communication noise. Our results show that the global model converges before such errors can affect performance, ensuring reliable FL outcomes.

The key contributions of the work are as follows:

- We propose an efficient and privacy-preserving federated learning system using hyperdimensional computing and fully homomorphic encryption.
- We explore the model and FHE design space to provide

insights on parameter selection aimed at optimizing both computation and communication costs.
- We evaluate the impact of channel noise during transmission, ensuring the global model convergence.
- Adopting hyperdimensional computing, our framework shows high communication efficiency, fast convergence, and noise robustness. Through experiments, we reduced the model size by $2.2\times$ while reaching comparable final accuracy to a CNN model.
- Compared to previous FHE-based FL frameworks, our framework shows at least $4.5\times$ faster client-side latency.

## II. Background

### A. Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) is an encryption technique that allows computations to be performed directly on encrypted data. The computing party, typically a server, can carry out operations without needing to decrypt the data. Since the server never accesses the plaintext, FHE is one of the key techniques for data privacy.

The most commonly used FHE schemes are classified into two branches: SIMD style and single-value style. Both of them are based on learning with error (LWE) problems. However, they differ in data packing and supporting homomorphic operations. The SIMD style, including BGV [6] and CKKS [7], [8] schemes, encrypts a vector of integer/ real values into a ciphertext. They use RLWE ciphertexts in $\mathcal{R}_Q^2$, where $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$, $\mathcal{R} = \mathbb{Z}[X]/(X^N+1)$, $N$ is ring dimension and $Q$ is ciphertext modulus. They support element-wise arithmetic operations like addition and multiplication. The single-value style schemes, FHEW [9] or TFHE [10], can encrypt a single modulus integer as a ciphertext and use LWE ciphertexts in $\mathbb{Z}_q^{n+1}$, where $n$ is LWE dimension and $q$ is ciphertext modulus. They support simple addition and look-up-table (LUT) operation.

We next define a few key FHE operations to be used in our FL framework. Note that the inputs $x$ and $y$ can be either vectors or scalar values. If they are vectors, the operations are performed element-wise.

- $\mathit{Enc}(x) = c$ : Encryption of the input plaintext $x$.
- $\mathit{Dec}(c) = y$ : Decryption of the plaintext $c$. The secret key is needed to perform decryption.
- $\mathit{HomAdd}(\mathit{Enc}(x), \mathit{Enc}(y))) = z$ : Homomorphic addition of two ciphertexts. Conversely, $\mathit{Dec}(z) = x + y + \epsilon$, where $\epsilon$ is an error value.
- $\sum_i^{\mathit{Hom}} c_i = s$ : Homomorphic summation of input ciphertexts $c_i$. Same as evaluating $s$ by repeating $\mathit{HomAdd}(s, c_i)$ for $\forall i$.
- $\mathit{HomMul}(\mathit{Enc}(x), a) = z$ : Homomorphic multiplication with a plaintext value $a$. Conversely, $\mathit{Dec}(z) = ax + \epsilon$.

The substantial ciphertext size in FHE systems necessitates high memory bandwidth, and their performance is constrained by computationally intensive operations such as polynomial multiplication. This throughput bottleneck poses a significant obstacle to the scalability and broader adoption of FHE, including its application in federated learning frameworks.

### B. Hyperdimensional Computing

Hyperdimensional computing (HDC) is a lightweight computing framework that uses high-dimensional holographic representations of data, called *hypervectors*. We adopt the random projection [11] and RBF [12] encoding in our work due to its state-of-the-art performance in classification tasks. Suppose $D$ represents the dimension of hypervectors.

**Random Projection Encoding** Given input feature vector of size $f$, $F \in \mathcal{R}^f$, we sample base $B_i \in \mathcal{R}^f$ from $\{-1, 1\}$. Each element of the encoded hypervector $H = (h_1, h_2, \dots h_D)$ is computed as $h_i = \text{sign}(B_i \cdot F)$.

**RBF Encoding** Given input feature vector of size $f$, $F \in \mathcal{R}^f$, we sample base $B_i \in \mathcal{R}^f$ from Gaussian distribution $\mathcal{N}(0, 1)$. We use uniformly random bias $b_i \in [0, 2\pi]$ for $i = \{1, 2, \dots D\}$. Each element of the encoded hypervector $H = (h_1, h_2, \dots h_D)$ is calculated as $h_i = \cos(B_i \cdot F + b_i)$.

**HDC Training** Once the dataset is mapped to hypervectors, we perform HDC training, which involves similarity evaluation and vector addition. Let $\sigma(X, Y) \in [0, 1]$ represent the similarity metric used to measure the distance between two hypervectors, $X$ and $Y$. We train the model by updating the class hypervectors $C_l$ for $l \in \{1, 2, \dots L\}$, where $L$ is the number of input classes. Assuming $c$ is the class of $H$, and $p$ is another class whose hypervector is closest to $H$, we define $p = \arg\min_l(\sigma(C_l, H))$. The model training proceeds as follows with a learning rate $lr$:

$$\begin{aligned} C_c &\leftarrow C_c + lr \cdot (1 - \sigma(C_c, H)) \cdot H \\ C_p &\leftarrow C_p - lr \cdot (1 - \sigma(C_p, H)) \cdot H \end{aligned} \tag{1}$$

**HDC Inference** HDC classification inference uses the same similarity metric as in the training stage. For a given input query hypervector $H$, we evaluate its distance to each class hypervector and assign the label of the closest one: $p = \arg\min_l(\sigma(C_l, H))$. We use cosine similarity for $\sigma$ to measure the closeness between hypervectors.

## III. Related Work

### A. FHE and Federated Learning

PFMLP [3] proposed an FL framework with HE, using multilayer perceptron (MLP) model and the Paillier encryption scheme [13]. Their method reduced latency by 13.3% compared to a standard MLP and by up to 28% after optimizing the Paillier system. However, since Paillier is an additive homomorphic scheme, it only supports basic operations, limiting PFMLP to a simple addition-based aggregation strategy.

tMK-CKKS [5] introduced a privacy-preserving FL framework based on a multi-key variant of the CKKS scheme. By leveraging CKKS's batching technique, the framework reduced computation latency and communication size compared to prior FL frameworks built on CKKS and Paillier cryptosystems. However, despite the improvements, the framework shows relatively low accuracy, including 81.87% on the MNIST dataset. This implies that decreasing the model complexsity to improve the communication efficiency could result in significant accuracy degradation.

In constrast, Rhychee-FL leverages HDC for lightweight training and computation. We thoroughly evaluate Rhychee-FL's design space, examining various parameters such as HDC dimensions and FHE cryptographic settings to balance accuracy, communication size, and computational latency.

### B. Hyperdimensional Computing and Federated Learning

Previous works have shown that the lightweight operations and model of HDC can improve both computational and communication efficiency in FL.

FedHD [14] and FHDnn [15], [16] represent the early efforts for applying HDC in FL. Unlike conventional FL methods that transmit CNN models, FedHD and FHDnn exchange class hypervectors between clients and the server, resulting in smaller model sizes and greater tolerance to communication noise. Their results demonstrated comparable accuracy with up to 66× improvement in communication efficiency and 3× faster convergence compared to DNNs. However, the framework still suffers from the privacy concerns related to model sharing. In this work, we address privacy concerns by employing FHE, with further parameter optimization to reduce overhead.

## IV. Methodology

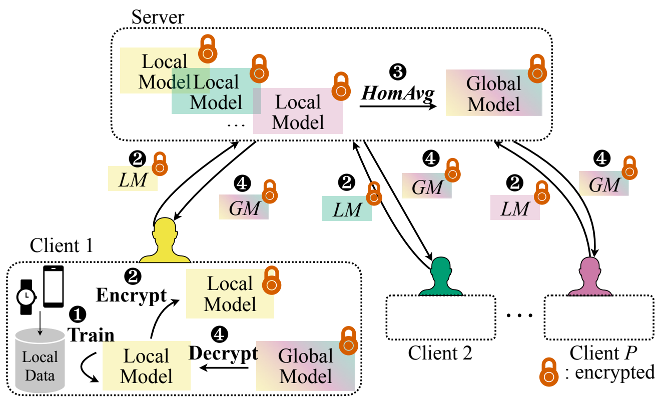


Fig. 1: Overview of Aggregation Round

In this paper, we propose Rhychee-FL, a novel privacy-preserving federated learning framework based on HDC. Rhychee-FL enhances the computational and communication efficiency by utilizing lightweight HDC learning, without compromising accuracy and privacy. In Section IV-A, we introduce the overall procedure of Rhychee-FL. In Section IV-B we explain the design space of the framework, that we explore to further optimize it. Finally, Section IV-C shows the analytical model that we used to evaluate the communication channels in our FL setting.

### A. Overview of Rhychee-FL

Before the learning process begins, a key sharing process is performed. In this step, the clients decide on an FHE parameter set and share a common secret key. The secret key is not shared with the server, preventing the server from accessing their raw models and guaranteeing model privacy. The clients also provide the server with a set of public keys, which are used for homomorphic operations.

Fig. 1 illustrates one global round of FL in Rhychee-FL, including local training, local model collection, homomorphic aggregation and global model distribution.

❶ **Local Training** In the $t$-th round of model aggregation, each $i$-th client trains their local models $LM_i^t$ using their local datasets for a few epochs. In our framework, the local model training is done by updating the class hypervectors as defined in Eq. (1).

One of the key advantages of HDC is that the model training process is simpler compared to DNN and gradient descents. As shown above, the training only requires similarity computations and vector additions. This can reduce the local training time, as well as the total training time.

❷ **Local Model Collection** Subsequently, each of the clients encrypts their local model ($Enc(LM_i^t)$) and sends it to the server. While encrypting the models with a packed FHE scheme like CKKS, we maximize the efficiency of the packing method. To elaborate more, encrypting each hypervector as one ciphertext would be a simple and straightforward approach. However, a ciphertext can often encrypt larger vectors than a single hypervector. This simple approach would waste the capacity of ciphertexts. To avoid this, we ensure that every slot of the ciphertext is fully utilized. More detailed explanation about the ciphertext size is provided in Section IV-B2.

❸ **Homomorphic Aggregation** Once the server receives local models, it homomorphically evaluates their average $HomAvg_i(Enc(LM_i^t))$. The homomorphic averaging function is defined as follows, where $P$ is the number of clients:

$$HomAvg_i(Enc(LM_i^t)) = HomMul(\overset{Hom}{\sum_i} Enc(LM_i^t), 1/P) \quad (2)$$

The $HomAvg(\cdot)$ function can be directly applied even when utilizing maximum packing, as the homomorphic operations are performed element-wise.

In this work, we adopted FedAvg [17] which is the most fundamental and widely-adpoted aggregation method in FL. Notably, Rhychee-FL can be easily expanded to use other aggregation strategies such as FedNova [18] and FedProx [19], which we leave for future investigation.

❹ **Global Model Distribution** The homomorphic aggregation results in the encryption of the global model $Enc(GM^{t+1})$ for the $(t+1)$-th round. Finally, this global model is sent back to the clients, who decrypt it. They then update their local models with the new global model: $LM_i^{t+1} \leftarrow GM^{t+1}$.

Because the clients receive the global models directly and not the average of gradient descents, they can directly use the model without further computations.

### B. Design Space Exploration

We further define and optimize the design space of Rhychee-FL. This parameter selection is crucial because it significantly impacts both performance and overhead. For instance, in the HDC domain, smaller models help reduce communication and computation overhead, but the model must still be large enough to maintain reasonable accuracy. Similarly, in the FHE domain, the choice of encryption scheme and cryptographic parameters determines ciphertext size and computational complexity. Thus, selecting an optimized set of parameters is crucial for optimizing the framework.

In the following sections, we describe the tunable parameters in both the HDC and FHE domains.

#### 1) HDC Design Space

In HDC classification, class hypervectors represent the model. Assuming we have a hypervector of dimension $D$ and a dataset with $L$ classes, the model consists of a total of $D \times L$ trainable variables, which are sent to the server for aggregation. Therefore, minimizing $D$ is a key optimization to reduce model size. However, as previous studies like ManiHD [12] have shown, smaller $D$ can lead to accuracy degradation, so this parameter must be carefully selected through evaluations.

### *2) FHE Design Space*

In FHE-based FL, clients share ciphertexts with the server, so reducing the size or number of ciphertexts directly affects communication overhead. The size of a ciphertext is determined by the choosen FHE scheme and their parameters. It is important to note that the number of ciphertexts may not be the same as the number of model variables, depending on the selected FHE scheme. For example, CKKS [7], [8], [20] is a SIMD-style scheme that uses Ring-LWE (RLWE) ciphertexts, allowing it to encrypt up to $N/2$ numbers into a single ciphertext. Therefore, the number of ciphertexts is $\lceil \frac{DL}{N/2} \rceil$.

On the other hand, FHEW and TFHE [9], [10] are single-valued schemes using LWE ciphertexts. Unlike CKKS, these schemes can encrypt only one value at a time, resulting in a total of $D \times L$ ciphertexts. Table I summarizes the communication size per round based on the model parameter $D, L$ and cryptographic parameters $N, Q, n, q$. This evaluation can be generalized to other models by replacing the $D \times L$ term with the total number of trainable parameters in those models.

| Scheme | Communication Size (bits) |
|---|---|
| CKKS | $\lceil \frac{DL}{N/2} \rceil 2N \log Q$ |
| TFHE | $DL(n+1) \log q$ |

TABLE I: Design Space and Communication Size

Scheme selection between CKKS and TFHE should be made considering their trade-offs. CKKS supports SIMD-style operation and has smaller amortized latency, making this scheme more suitable for processing numerous data at once. However, it only supports arithmetic operations – multiplication and addition. Therefore, evaluating a non-linear function would require polynomial approximations, which can lead to multiple expensive CKKS bootstrappings. On the other side, TFHE scheme only encrypts a single value, resulting in a total number of ciphertexts that can be very large depending on the application. But TFHE is known to support an arbitrary LUT without loosing integer precision. Therefore, TFHE is more suitable for high-precision computation requiring non-linear functions.

## *C. Analytical Models for Noisy Communication Channels*

Although FHE guarantees privacy in FL setting, ciphertexts can get compromised during data transmission. Even a single-bit error can cause the ciphertext to decrypt to an entirely incorrect value and not allowing the global model to converge. Therefore, stricter error detection and correction methods are necessary to ensure accurate computations. We establish analytical models to evaluate such effects and calculate the expected number of FL rounds until failure. From these models, we reveal that Rhychee-FL is more resilient to communication noise due to its smaller model size. This is the first time that communication errors are considered jointly with FHE.

To evaluate communication latency, we use a model where a transmitter sends 1400-bit TCP/IP packets to a receiver over a 3GPP Urban Microcell (UMi) channel. The system follows the 5G standard with 14 symbols and 12 subcarriers per physical resource block (PRB), using 16-quadrature amplitude modulation (QAM-16) for 4 bits per symbol. It employs 16 receiver antennas, 4 transmission antennas, and a signal-to-noise ratio (SNR) of 12 decibels (dB), based on established benchmarks [21].

For the receiver, error detection algorithms like Checksum and CRC are used to detect bit errors and initiate packet retransmission if necessary. Checksum, which sums up data blocks to verify errors, is simple but less robust [22]. On the other hand, CRC is more robust and effectively detects common errors through polynomial division of data, making it well-suited for high-reliability applications in networking and communications. The corresponding communication latency is shown in Eqn. 3, computed from the latency for transmitting a single TCP packet $L_{TCP\ Packet}$, error detection $L_{CRC/Checksum}$, and the number of packet retransmissions $N_{re}$.

$$L_{comm} = (L_{TCPPacket} + L_{CRC/Checksum}) \times N_{re} \quad (3)$$

We assume that $P_{re}$ is a probability for an undetected error to occur in CRC or checksum. The probability of at least one bit error in a packet is given by $N \times BER$, where $N$ is the number of bits in the packet and $BER$ is the bit error rate. Consequently, the overall probability of an undetected error is expressed as $P_{ue} = N \times BER \times P_{re}$.

The Poisson distribution can be used to model the number of transmissions until an undetected error occurs. The expected number of transmissions $E[T]$ is the inverse of the probability of an undetected error, $\frac{1}{P_{ue}}$.

In a federated learning setup with $P$ clients and 1 server, the total number of data packet transmissions per round includes both client-to-server and server-to-client packets. The expected number of rounds until failure, $E[R]$, is the maximum number of aggregation rounds before an undetected error occurs. This is approximated by dividing the expected number of transmissions until failure by the total number of transmissions per round. For a $P$-client system, this is $E[R] = \frac{E[T]}{2 \times P \times (\text{\# Packets Communicated})}$. Given the expected number of transmissions to failure $E[T]$ and considering the two-way communication between the server and the clients in each round, this formula provides an estimate of how many federated learning rounds can be completed without errors.

## V. Results

## *A. Experimental Setup*

We run our algorithm on a CPU environment with Apple M1 Pro and 16GB RAM using the MNIST [23] and human activity recognition (HAR) datasets [24]. We tested the framework with varying number of clients from 10 to 100. The dataset was distributed in non-iid manner, using Dirichlet distribution as in Li *et al.* [25]. We used random projection encoding for HAR dataset and RBF encdoing for MNIST dataset. We used one of the state-of-the-art federated learning frameworks by Li *et al.* [25] with FedAvg [17] to measure the baseline performance. We used their CNN model with two convolutional layers and

| | PFMLP [3] | xMK-CKKS [5] | Ours |
|---|---|---|---|
| Model | MLP | LR | HDC |
| HE Scheme | Partial HE (Paillier) | tMK-CKKS | CKKS |
| Parameters | $54,912$ | $7,850$ | $20,000$ |
| Accuracy | 0.925 | 0.819 | 0.960 |
| Enc/Dec Latency | 779.37 s | 389.2 ms | 86 ms |

TABLE II: Comparison of Previous Works and Ours (MNIST)

two fully connected layers as our baseline model, because it provides one of the best accuracy results with smaller network for the MNIST dataset. We also compared our framework with previous works on FL with homomorphic encryption, PFMLP [3] and xMK-CKKS [5].

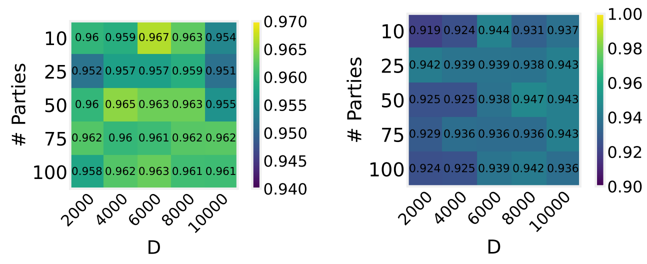
## B. Performance Comparison with SOTA

We first compare our framework's performance with state-of-the-art FL-FHE frameworks, PFMLP [3] and xMK-CKKS [5], based on MNIST workload. Although the experimental setups differ in the number of clients, it is reasonable to compare key results that are commonly applicable regardless of the client count. In Table II, we compare the number of trainable parameters, final global accuracy and encryption and decryption latency per iteration for each client. In all these aspects our framework outperformed PFMLP [3] and xMK-CKKS [5]. For example, our final global model shows $3.5\%$ and $14.1\%$ higher accuracy, while encryption and decryption is $9,000\times$ and $4.5\times$ faster each. It is also worth noting that when the baseline models increase their size to achieve comparable accuracy to ours, their latency also increases due to the larger number of parameters.

## C. System Parameters and Accuracy

We comprehensively evaluate the impact of system parameters on accuracy. These system parameters encompass both model parameters and federated learning parameters.

### 1) Model Parameter and Accuracy



(a) MNIST (b) HAR

Fig. 2: Final Global Model Accuracy

The HDC model typically utilizes a hypervector dimension $D$ in the range of $1,000$ to $10,000$. For certain datasets, a larger dimension is necessary to achieve good accuracy. Fig. 2 illustrates our HDC global model's accuracy as a function of $D$ and the number of clients for HAR and MNIST datasets, respectively. These experiments are conducted in non-encrypted data. For all values of $D$, our global model reached $\geq 95\%$ accuracy for MNIST and $\geq 92\%$ for HAR. The fact that there is no significant accuracy difference for different $D$ indicates that these datasets do not require large dimension to achieve satisfactory accuracy. In other words, hypervectors of size $D \leq 4,000$ is enough to represent the meaningful features of the datasets. Therefore, we can avoid using large dimension and enhance the communication efficiency.

| Set | Scheme | $N(n)$ | $\log Q(\log q)$ |
|---|---|---|---|
| CKKS-1 | CKKS | 32768 | 160 |
| CKKS-2 | | 16384 | 130 |
| CKKS-3 | | 8192 | 100 |
| CKKS-4 | | 8192 | 61 |
| TFHE-1 | TFHE | 534 | 10 |
| TFHE-2 | | 503 | 10 |
| TFHE-3 | | 448 | 10 |

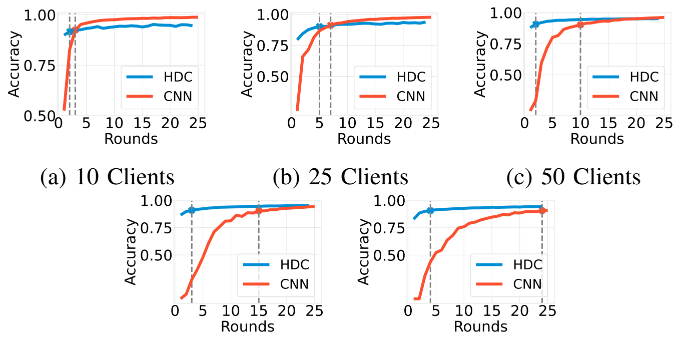
TABLE III: FHE Parameter Sets

### 2) Federated Learning Parameter and Accuracy

From Fig. 2, we gain insights about the relationship between the number of parties, which is a key federated learning parameter, and the resulting global model accuracy. For every case of our experiment, the global model showed a similar rate of convergence and final accuracy. This consistency suggests that our framework can maintain stable performance across varying client numbers. We credit such stability to HDC's robustness to noises and less sensitivity to parameter settings.

### 3) Fast Convergence

Fig. 3 compares the global model's accuracy by aggregation rounds with CNN baseline [25]. Our framework used $D = 2,000$. We have marked in the graph when the accuracy first reaches $90\%$. In all cases, our HDC model was able to reach $90\%$ accuracy within 5 rounds of communication, which is much faster than the CNN model. For example, with 100 clients, our model reached the target accuracy $6\times$ faster than when using CNN [25]. This highlights the fast convergence of HDC-based models in Rhychee-FL, suggesting its effectiveness in real-world federated learning with many clients.



(a) 10 Clients (b) 25 Clients (c) 50 Clients

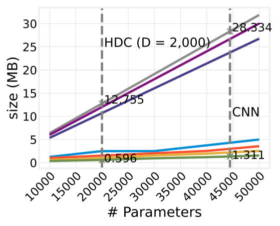
(d) 75 Clients (e) 100 Clients

Fig. 3: Global Model Accuracy by Rounds (MNIST)

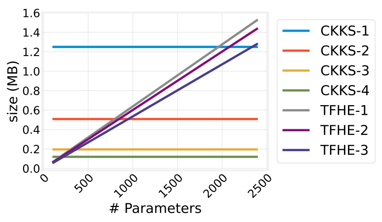
## D. FHE Settings and Communication Efficiency

We conduct a series of experiments to evaluate the impact of FHE schemes and parameter selection on communication size. We utilized the CKKS and FHEW implementations from the OpenFHE library [26]. To thoroughly evaluate communication efficiency, we selected seven distinct sets of FHE parameters: four from the CKKS scheme and three from the FHEW scheme. Table III details these parameters, all of which meet the 128-bit security standard. $N$ is ring dimension, $n$ is LWE dimension, $Q$ and $q$ are ciphertext modulus.

Fig. 4 illustrates the communication overhead for various FHE parameter sets, in relation to the model size. The overhead is evaluated following the Table III and the model defined as the number of trainable parameters in the model. We assumed



(a) # Params = (10K, 60M)



(b) # Params = (100, 3K)

Fig. 4: Model Size and Communication Overhead

maximum utilization of CKKS packing. The figure shows our HDC model with $D = 2,000$ and the CNN model [25], both trained on the MNIST dataset. The results indicate that our HDC model offers communication advantage over the baseline CNN. Our model shows at most $2.2\times$ smaller communication size than the CNN [25], using CKKS-4 parameter set. The communication size in the framework is primarily determined by the number of ciphertexts to be transmitted, which in turn is dictated by the number of trainable parameters in the model. Thus, models with fewer parameters, such as the HDC, can significantly reduce communication size.

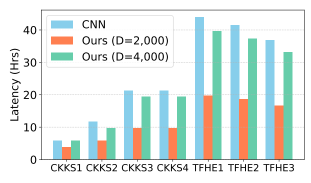
Furthermore, our observations highlight the significance of the FHE scheme and parameter selection. For example, using parameter set CKKS-4 instead of TFHE-1, our framework can achieve up to $21.4\times$ improvement in communication size. Shown in parameter sets CKKS-3 and CKKS-4, we evaluated the impact of ciphertext modulus by reducing the scaling factor parameter in FHE. The scaling factor parameter primarily determines the precision of the plaintext preserved in our application, which does not involve any FHE bootstrapping. We tested lowering the ciphertext modulus $Q$ as low as 61 bits does not degrade the global model accuracy. In the meantime, this exploration brought us to further reduce the communication size by 39%.

The results indicate that the CKKS scheme consistently achieves a significantly smaller communication size compared to the TFHE scheme. This advantage stems from CKKS's ability to efficiently pack multiple parameters into a single ciphertext, reducing the average ciphertext size per model parameter. However, the ciphertext size remains fixed, even when the number of parameters is less than the maximum capacity of a single ciphertext. As a result, the TFHE scheme shows advantages when the number of parameters is smaller, as illustrated in Fig. 4b. Additionally, TFHE supports non-linear LUT operations without losing input precision, making it ideal for systems where high-precision non-linear operations are prioritized over communication overhead.

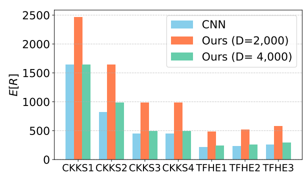
### *E. Robustness to Communication Noise*

We use the communication channel setup described in Section IV-C with 32-bit CRC and 10 clients and 1 server setting to evaluate the net transmission latency as well as the time until the first error occurs for our FL-FHE framework. We set a BER of $10^{-3}$ and a packet size of 1400 bits (typical to standard 5G protocol). With this setting, $P_{re} = \frac{1}{2^{32}} = 2.328 \times 10^{-10}$ and $E[T] = 3.039 \times 10^{9}$.

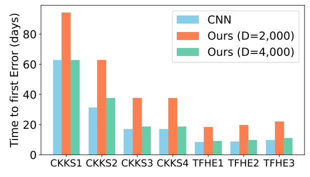
In Fig. 5a, we plot the communication latency for each round,



(a) Communication Latency



(b) # Error-Free Rounds



(c) Expected Time to First Error

Fig. 5: Communication Overhead with CRC Error Detection and Retransmissions (MNIST)

based on Eqn. 3, with CRC error detection and retransmissions. Our method with $D = 2,000$ has 54% lower latency compared to CNN baseline when using CKKS-4. In Fig. 5b and Fig. 5c, we plot the expected number of aggregation rounds and the total time until failure for the 10 client - 1 server FL scenario based on the calculations shown in Section IV-C. Compared to CNN, our HDC ($D = 2,000$) takes 2.2x more transmissions and time (37 days for HDC vs. 17 days for CNN) to run into an error with CKKS-4, thanks to the small model size of HDC. Moreover, given the fast convergence of the HDC model, as shown in Section V-C3, we can conclude that our global model can successfully converge before channel noise causes any interruptions.

## VI. Conclusion

Federated learning (FL) enables multiple clients to collaboratively train models, but it raises privacy concerns. We propose a privacy-preserving FL framework that leverages fully homomorphic encryption (FHE), allowing computations on encrypted data to protect client models. However, FHE introduces significant memory and computation overhead.

To address this challenge, we adopted a lightweight, noise-resilient hyperdimensional computing (HDC) model, known for its efficiency and robustness. Furthermore, we explored various design choices, including model and FHE parameters, to optimize the framework. Lastly, we present an analytical model of the communication channel to assess the impact of channel noise on our framework.

Our results show that the communication overhead is reduced by up to $2.2\times$ using HDC when compared against a CNN model. Compared to previous FHE-FL frameworks, ours show over $4.5\times$ faster client side operation. We also demonstrated that with a careful FHE parameter selection, we can improve the communication efficiency by $21.4\times$. Finally, our communication simulation demonstrated that our framework can converge before channel noise causes any significant error.

## Acknowledgment

This work was supported in part by PRISM and CoCoSys, centers in JUMP 2.0, an SRC program sponsored by DARPA.

## References

[1] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, "A survey on federated learning: challenges and applications," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 2, pp. 513–535, 2023.

[2] N. M. Hijazi, M. Aloqaily, M. Guizani, B. Ouni, and F. Karray, "Secure federated learning with fully homomorphic encryption for iot communications," *IEEE Internet of Things Journal*, 2023.

[3] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, p. 94, 2021.

[4] Z. Liu, S. Chen, J. Ye, J. Fan, H. Li, and X. Li, "Dhsa: efficient doubly homomorphic secure aggregation for cross-silo federated learning," *The Journal of Supercomputing*, vol. 79, no. 3, pp. 2819–2849, 2023.

[5] W. Du, M. Li, L. Wu, Y. Han, T. Zhou, and X. Yang, "A efficient and robust privacy-preserving framework for cross-device federated learning," *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 4923–4937, 2023.

[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[7] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.

[8] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 360–384.

[9] L. Ducas and D. Micciancio, "Fhew: bootstrapping homomorphic encryption in less than a second," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 617–640.

[10] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.

[11] K. Behnam, X. Hanyang, M. Justin, and R. Tajana, "tiny-hd: Ultra-efficient hyperdimensional computing engine for iot applications," in *IEEE/ACM Design Automation and Test in Europe Conference (DATE), IEEE, IEEE*, vol. 10, 2021.

[12] Z. Zou, Y. Kim, M. H. Najafi, and M. Imani, "Manihd: Efficient hyper-dimensional learning using manifold trainable encoder," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 850–855.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238.

[14] Q. Zhao, K. Lee, J. Liu, M. Huzaifa, X. Yu, and T. Rosing, "Fedhd: Federated learning with hyperdimensional computing," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 2022, pp. 791–793.

[15] R. Chandrasekaran, K. Ergun, J. Lee, D. Nanjunda, J. Kang, and T. Rosing, "Fhdnn: communication efficient and robust federated learning for aiot networks," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 37–42. [Online]. Available: https://doi.org/10.1145/3489517.3530394

[16] K. Ergun, R. Chandrasekaran, and T. Rosing, "Federated hyperdimensional computing," 2023.

[17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[18] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[19] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[20] K. Han and D. Ki, "Better bootstrapping for approximate homomorphic encryption," in *Cryptographers' Track at the RSA Conference*. Springer, 2020, pp. 364–390.

[21] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, "Millimeter wave mobile communications for 5g cellular: It will work!" *IEEE access*, vol. 1, pp. 335–349, 2013.

[22] T. C. Maxino and P. J. Koopman, "The effectiveness of checksums for embedded control networks," *IEEE Transactions on dependable and secure computing*, vol. 6, no. 1, pp. 59–72, 2009.

[23] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[24] J. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra, "Human Activity Recognition Using Smartphones," UCI Machine Learning Repository, 2012, DOI: https://doi.org/10.24432/C54S4K.

[25] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 2022, pp. 965–978.

[26] A. A. Badawi, A. Alexandru, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, C. Pascoe, Y. Polyakov, I. Quah, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, "Openfhe: Open-source fully homomorphic encryption library," Cryptology ePrint Archive, Paper 2022/915, 2022, https://eprint.iacr.org/2022/915. [Online]. Available: https://eprint.iacr.org/2022/915