

Efficient Federated Learning with Low-Rank Updates under Homomorphic Encryption

Mohamed Aboelenien Ahmed*, Mohamed Alsharkawy*, Hassan Nassar*, Heba Khdr*
Jeferson Gonzalez-Gomez†, Jörg Henkel*

* Karlsruhe Institute of Technology (KIT), Germany

†Costa Rica Institute of Technology (TEC), Costa Rica

*{mohamed.ahmed3, mohamed.alsharkawy, hassan.nassar, heba.khdr, henkel}@kit.edu, †jgonzalez@itcr.ac.cr

Abstract—Federated Learning has been widely adopted for its ability to collaboratively train models without exposing raw data. However, the server-side aggregation process may still leak sensitive information about client data. Homomorphic Encryption enables privacy-preserving aggregation, but it introduces substantial communication overhead for clients and high computational costs for the server. To address these challenges, we propose *HEAL-FL*, a federated learning framework that is based on low-rank shared basis vectors across clients. Instead of transmitting full encrypted model updates, clients send only encrypted low-rank coefficients, thereby reducing both communication costs and server-side aggregation overhead. Furthermore, *HEAL-FL* incorporates a communication-efficient basis update scheme that relies exclusively on homomorphic addition at the server. Our evaluation across various homomorphic encryption schemes shows that *HEAL-FL* reduces client communication and server aggregation costs, leading to improved efficiency of Federated Learning systems. Notably, these savings translate into up to a significant reduction of 38.6% in total training time compared to conventional homomorphic FedAvg with full model parameter transmission, demonstrating the practical benefits of our approach.

Index Terms—Federated Learning, Homomorphic Encryption

I. INTRODUCTION

The remarkable success of deep learning is largely attributed to the availability of large, diverse, and high-quality datasets. However, in many domains, the data is sensitive and distributed across devices or organizations, making direct centralization infeasible due to privacy concerns. Federated Learning (FL) has emerged as a promising paradigm to address this issue by enabling collaborative model training without sharing raw data [1]. FL has been successfully applied in different domains such as healthcare [2], [3] and finance [4], where data privacy is critical. FL relies on two main steps: local computation, where each device trains the model on its local dataset, and model aggregation, where clients transmit their updated models to a central server for aggregation.

Although FL keeps each device’s data local to preserve privacy, recent attacks have shown that client sensitive information can still be leaked during aggregation when model updates are sent to the server [5]–[7]. Even parameter-efficient methods that share only a small subset of model parameters, such as

Low-Rank Adaptation (LoRA) [8]–[10], remain vulnerable to such data leakage [7]. To mitigate these privacy risks, Homomorphic Encryption (HE) has been integrated with FL, enabling the server to aggregate client updates in the *ciphertext* domain without revealing them in *plaintext* [11], [12]. While HE offers strong privacy guarantees, its practical deployment in large-scale FL systems remains challenging, as ciphertexts can be up to $128\times$ larger than their plaintext counterparts [13]. For example, uploading an encrypted ResNet-18 model using Paillier [13] takes nearly 45 minutes at an upload bandwidth of 2 MB/s, compared to only 22 seconds for the same model in plaintext. As a result, the primary FL system bottleneck is shifted from local model training to the ciphertext transmission as well as the encryption, decryption, and aggregation operations required by homomorphic encryption.

Recent research has explored several strategies to enhance HE-based FL frameworks and make them more practical. For instance, authors in [14] employ efficient HE schemes such as CKKS [15] that enable faster linear algebra encrypted operations compared to other schemes. To further accelerate HE schemes, some works have proposed specialized hardware accelerators to speed up homomorphic computations [16]–[19], while others focus on optimizing encryption and decryption on client devices [20], [21], thereby reducing homomorphic operation latency and making aggregation more practical. To address the communication bottleneck, BatchCrypt [22] uses gradient clipping and quantization to reduce client communication and server computation when using Paillier encryption. Similarly, Pack [23] reduces CKKS ciphertext size to achieve a more compact encryption.

Although recent work has reduced computation and communication time, these approaches remain bounded by the number of model parameters that must be transmitted and encrypted. Even relatively small models, such as ResNet-18, require each client to perform millions of homomorphic operations per FL round, generating a large volume of ciphertexts and introducing substantial communication overhead. On the server side, aggregation costs scale linearly with the number of clients, making direct homomorphic aggregation prohibitively expensive [1], [11], [22]. Consequently, the sheer volume of encrypted operations limits the effectiveness of state-of-the-art solutions, making direct homomorphic aggregation prohibitively expensive even with existing optimizations.

This work was partially funded by the German Federal Ministry of Research Technology and Space (BMFTR) through grant 01IS23066 as part of the Software Campus Projects “HE-Trust” and “VERITAS”.

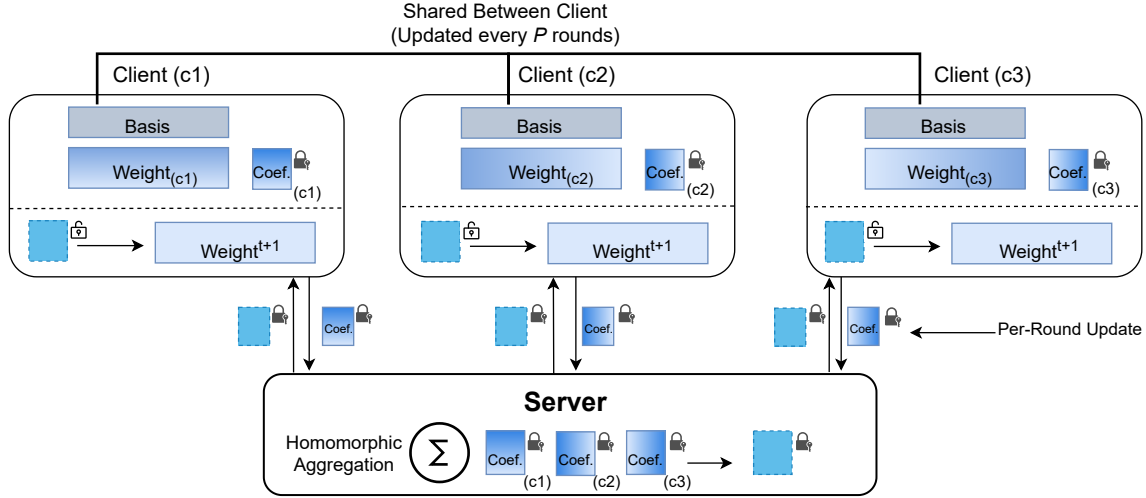


Fig. 1. Overview of the proposed framework. After local training, each client computes its coefficient matrices, encrypts them, and transmits the encrypted updates to the server. The server performs homomorphic aggregation to obtain the global coefficients, which are then sent back to the clients. Each client decrypts the aggregated coefficients and reconstructs the model weights for the next round.

To overcome this limitation, we propose a novel FL framework with HE based on low-rank updates, enabling efficient privacy-preserving federated learning as illustrated in Figure 1. In our approach, only a smaller set of parameters (*coefficients*) is transmitted to the server for homomorphic operations, while the remaining parameters (weights and *shared basis between clients*) remain locally on the client. This reduction is achieved using singular value decomposition (SVD). This design simultaneously addresses the three main bottlenecks of HE-based FL: it reduces communication overhead between clients and the server, decreases encryption and decryption time on the client side, and lowers server-side computation. A main challenge we address is how to efficiently maintain the low-rank shared parameters to remain representative of the full model without compromising privacy or performance of the model. To address this, we utilize a distributed subspace iteration method [24] from computational linear algebra, combined with the iterative nature of FL, to update these parameters whenever necessary. We evaluated our framework using ResNet-18 and ResNet-34 models and different HE schemes to show that the framework is not limited to a specific scheme. In summary, the major contributions of this work are:

- We introduce Homomorphically Encrypted Aggregation of Low-rank updates in Federated Learning (HEAL-FL), the first privacy preserving federated learning framework that enables training models from scratch while employing compressed model updates, substantially reducing encryption, decryption, and communication costs.
- We propose updating shared basis between clients in a privacy-preserving manner with minimal overhead by replacing costly homomorphic SVD operations on the server with simple homomorphic additions.
- We demonstrate the efficiency of our protocol through extensive evaluation across three homomorphic encryption schemes utilizing real server and edge devices.

II. BACKGROUND

As many AI systems currently deal with sensitive data, privacy-preserving solutions are required to ensure that no data violations occur [25], [26]. This is very relevant at the training phase where large amounts of data from different sources have to be gathered to train the model. We need as many data sources as possible while ensuring at the same time that the data privacy remains intact. In this work, we use two concepts, Homomorphic Encryption [15] and Federated Learning [1], [27], to ensure data privacy. Additionally, we incorporate low-rank decomposition to design an efficient FL system under homomorphic encryption.

Homomorphic encryption allows computation on encrypted data without decryption. It is based on the concept of homomorphism from mathematics [28]. If we have two groups (G, \cdot) and (H, \times) , a function $h : G \rightarrow H$ is homomorphic, if and only if $h(u \cdot v) = h(u) \times h(v)$.

Let $(P, \diamond, C, \circ, e, d)$ be our homomorphic encryption scheme. P is the plaintext group with operation \diamond , and C is the ciphertext group with operation \circ . Functions e and d are the encryption and decryption algorithms. For plaintexts $a \in P$ and $b \in P$, the scheme satisfies: $e(a) \circ e(b) = e(a \diamond b)$.

Federated learning enables devices to collaboratively train models without sharing local data [29]. It involves multiple clients and a coordinating server. Each client c has local data \mathcal{D}_c . The process is iterative, with multiple rounds. In each round, the server sends model parameters (θ) to clients, which then train their local models and upload updated parameters back. The server aggregates these to form a new model, initiating a new round. This repeats until the model converges [30], [31]. This distributed setting introduces security vulnerabilities. Client-side, poisoning attacks [32] involve malicious clients altering data labels or model updates to corrupt the global model and create vulnerabilities or backdoors. Countermeasures like model-based analysis and robust

aggregation are suggested to mitigate these attacks [33], [34]. HE is explored for privacy-preserving FL on the server side, particularly against servers inferring client information.

Low-rank refers to representing a matrix using fewer independent components than its full size, capturing only its most important information while ignoring smaller, less significant variations. This reduces the complexity of the matrix and allows for more efficient storage. Singular Value Decomposition (SVD) is a widely used technique to achieve this by breaking a matrix into its main components: the left and right singular vectors, and a diagonal matrix of singular values sorted in decreasing order. The singular vectors can be viewed as basis vectors that span the dominant subspace. By retaining the top r singular values and corresponding singular vectors, a matrix can be compressed, preserving most of the original structure.

III. RELATED WORK

Recent works have explored communication reduction techniques for homomorphic encryption in federated learning. BatchCrypt [22], built on Paillier encryption, proposes a batching scheme that uses gradient clipping and quantization to reduce client communication and encryption costs as well as server computation overhead. However, BatchCrypt considers the step-wise federated learning system (FedSGD), in which gradients are transmitted to the server after each training step. This setting requires more communication than the more common FedAvg setting, where multiple local updates are performed before transmitting the model, rather than the gradients.

Pack [23] introduces a federated learning system, built on the CKKS scheme [15], that reduces communication overhead on clients by using a smaller ciphertext modulus for compact encryption of the uploaded data. However, compressing the ciphertext introduces significant errors that can lead to divergence during training. To mitigate this, the authors train an additional linear regression model for error correction and filter updates on weights with large absolute differences between the aggregated and local weights on the client side. Moreover, to mitigate bias towards client-specific models, periodic update rounds with a larger ciphertext modulus are still required.

Other approaches [35], [36] propose to selectively encrypt only the sensitive parts of the parameters to be aggregated using homomorphic encryption, while leaving less sensitive parameters to be aggregated in plaintext. Although this strategy improves communication efficiency, it does not fully guarantee the privacy of the clients.

Low-rank adaptation and weight decomposition have been considered in the context of model finetuning [9], [10], [30]. Specifically, only a few parameters are updated and sent to the server for aggregation, and the rest of the model is not updated throughout the whole training since they are already pretrained. This differs from our work since we consider a more general case of training from scratch, where all the model parameters should be trained and updated. There are also other works that consider using SVD for communication reduction

[37], [38] in FL for training from scratch by sending either low-rank weights or representation coefficients for gradients that are smaller than the actual model weights. Those approaches do not account for homomorphic encryption and are not readily applicable to every HE scheme. Moreover, they can incur a large computational cost on the server, as performing SVD under homomorphic encryption is highly expensive.

We note that our use of SVD should be distinguished from Federated SVD works [39]–[42], which aim to distribute the computation of SVD across clients’ data. By contrast, we employ SVD for low-rank decomposition to efficiently train neural networks in a federated manner.

In contrast to prior works, we introduce the first FL framework with HE that uses low-rank updates to improve efficiency and supports training models from scratch.

IV. METHODOLOGY

This section presents the proposed framework for efficient and privacy-preserving federated learning with homomorphic encryption and low-rank updates. Our goal is to reduce three main bottlenecks: (i) communication between the edge device and the server, (ii) the computation on the server side, and (iii) encryption and decryption on the edge device.

A. Problem Setup

We consider an FL system with a set of clients \mathcal{C} and a single server. The server is responsible for performing homomorphic aggregation of the client models. Each client $c \in \mathcal{C}$ possesses a local dataset $D_{(c)}$, which is kept private and not shared with other clients or the server. The overall objective of the FL process is to collaboratively learn a global model θ that minimizes the average loss across all client datasets, where θ consists of a set of weights and biases, collectively denoted as

$$\theta = \{W^{(l)}, b^{(l)}\}_{l=1}^L \quad (1)$$

where $W^{(l)}$ and $b^{(l)}$ represent the weight matrix and bias vector of the l th layer, respectively, and L is the total number of layers in the network.

On each communication round t , each client c trains θ^t on $D_{(c)}$, encrypts, and uploads $\theta_{(c)}^t$ to the server. The server then conducts the aggregation process over encrypted data from clients to obtain θ^{t+1} and broadcasts to all clients the encrypted new model to be decrypted at each client to start a new round.

Under HE, the cost of encryption, decryption, communication, and server-side aggregation becomes the dominant factor in FL training. Our objective is to improve the overall efficiency of the federated learning system while still enabling training of the full set of model parameters.

B. Low-rank Updates

To address these challenges, we represent each $W^{(l)}$ using reduced SVD. Specifically, each $W^{(l)} \in \mathbb{R}^{m^{(l)} \times n^{(l)}}$ can be expressed as:

$$W^{(l)} = U\Sigma V^\top \quad (2)$$

Given a rectangular weight matrix where $m < n$ and a chosen rank $r \leq m$, where we set r to $0.5m$, we utilize the right singular vectors $V^{\top(l)} \in \mathbb{R}^{n^{(l)} \times r^{(l)}}$ for each layer as shared basis across clients. For simplicity, we write $W := \{W^{(1)}, \dots, W^{(L)}\}$ to refer to the collection of weights across all layers, and use W in the remainder of this paper. Each client then sends only coefficients $W_{(c)}V^{\top} \in \mathbb{R}^{m \times r}$ per round. With this representation, after local training, the clients transmit their updated coefficients to the server, which aggregates them across all participants in an additive manner. This aggregation produces a global set of coefficients that represents the average contribution of all clients given the shared basis. This yields significant benefits: it drastically reduces the amount of data that must be encrypted, uploaded, and decrypted on the client side, thereby lowering latency and energy consumption. Moreover, because the server processes only the compact coefficient matrices instead of the full model weights, its aggregation computation is substantially lighter, resulting in faster global updates and improved scalability when the number of clients grows.

A potential drawback of keeping the basis fixed throughout training may lead to a loss of representation ability, since the most informative directions in the weight space can shift as the model evolves over rounds. In particular, the low-rank approximation may no longer capture the dominant subspace of the weight matrices, resulting in slower convergence and even degraded model accuracy. To mitigate this, we update the shared basis at the server periodically, less frequently than the coefficients. As mentioned before, however, transmitting the complete weight matrices from the client for aggregation or performing SVD homomorphically on the server incurs significant both communication and computation overhead.

Therefore, we propose to use a distributed subspace iteration strategy. Each client applies its local covariance to the current global basis, producing the sketch $Y_{(c)}$

$$Y_{(c)} = W_{(c)}^{\top}(W_{(c)}V) \in \mathbb{R}^{n \times r} \quad (3)$$

and transmits only $n \times r$ matrix to the server. The server homomorphically aggregates those sketches, sends them back to the clients, and clients subsequently perform a lightweight QR decomposition, producing the new basis. Note that in the standard (and distributed) power iteration, it is an iterative algorithm to obtain a new basis vector, where the process starts with a random initial vector and is iteratively repeated p steps till obtaining a new basis. In contrast, we utilize the basis from the previous federated round that provide a high-quality initialization (compared to the random one); consequently, requiring only a single iteration to obtain the updated basis for the next round.

C. HEAL-FL Flow

The training procedure in our proposed HEAL-FL framework consists of three phases: a warm-up stage, coefficient update rounds, and basis update rounds. To enable this design, the model parameters are partitioned into two groups. The first group, which dominates the overall model size, comprises the

Algorithm 1 HEAL-FL Server

Require: Clients \mathcal{C} , Model parameters θ , Rounds T

- 1: Send θ to \mathcal{C} to start warmup round
 - 2: **for** t in $1, \dots, T$ **do**
 - 3: **if** Update basis **then**
 - 4: Homomorphic addition for $\bar{Y} \leftarrow \sum_{c \in \mathcal{C}} Y_{(c)}$
 - 5: Send \bar{Y} to clients
 - 6: **if** Coefficient update **then**
 - 7: Homomorphic addition for $\bar{C} \leftarrow \sum_{c \in \mathcal{C}} \text{Coef}_{(c)}$
 - 8: Homomorphic addition for $\bar{\theta}^{nd} \leftarrow \sum_{c \in \mathcal{C}} \theta_{(c)}^{nd}$
 - 9: Send \bar{C} and $\bar{\theta}^{nd}$ to clients
-

weight matrices that use low-rank decomposition. The second group contains parameters that are not decomposed (denoted as θ^{nd}). This includes one-dimensional vectors such as biases and normalization layers, as well as the last layer. In the following, we describe each round type in detail:

1) *Warm-up Round:* The server initializes a model with parameters θ and sends it to the clients. Each client trains for $k_{warm-up}$ local epochs then performs SVD over its local weights (of the first group) to obtain new basis for each weight per layer with rank r . HEAL-FL then uses the discussed homomorphic basis update method to obtain the shared basis that represent all the clients, while preserving privacy. Each client then computes the coefficients given the new shared basis and local weights and encrypts them along with the non-decomposed (second) group parameters. Each client sends the encrypted data to the server, receives the averaged results, and decrypts them. For the first group of parameters, a new set of weights is constructed given the shared basis and averaged coefficients, and the second group directly uses the averaged results from the server. This warmup round ensures that training starts with representative basis vectors. It is important to note that the server only performs homomorphic additions, while the final division to compute the average is carried out on the client side in plaintext.

2) *Normal Updates Rounds:* Each client begins the round by training θ^t on its own local dataset, resulting in $\theta_{(c)}^t$. A client subsequently derives the coefficients by executing a dot product between the trained weights and the shared basis. Then the client encrypts and sends those coefficients along with the second group parameters to the server. After receiving the encrypted parameters, the server then performs homomorphic addition to aggregate the encrypted data and sends them back to the clients. The client then decrypts the coefficients and divides by the number of clients to construct the weights for the next round utilizing the shared basis and second group parameters to obtain the model for the next round. The detailed process is outlined in Algorithm 2.

3) *Periodic Updates Rounds:* Each client computes $Y_{(c)}$, encrypts it, and sends the encrypted result to the server. The client then obtains the new shared basis vectors as discussed in the distributed power method given the averaged \bar{Y} . The clients then perform the coefficients update round (without the training part) to obtain the model for the next round.

Algorithm 2 outlines the client-side process in each round of

Algorithm 2 HEAL-FL Client Round

Require: Dataset $D_{(c)}$, Encrypted Coefficients \bar{C} , Encrypted non-decomposed parameters $\bar{\theta}^{nd}$, Shared Basis V^\top

- 1: Decrypt \bar{C} and divide by number of clients
 - 2: $W^t \leftarrow \bar{C}V^\top$
 - 3: Fill θ^t from W^t and $\bar{\theta}^{nd}$
 - 4: $\theta_{(c)}^t \leftarrow$ Train θ^t on $D_{(c)}$ for k steps
 - 5: **if** Update basis **then**
 - 6: $Y_{(c)} = W_{(c)}^\top(W_{(c)}V)$
 - 7: Encrypt $Y_{(c)}$
 - 8: Send to server and wait to receive \bar{Y}
 - 9: Decrypt \bar{Y} and divide by number of clients
 - 10: $V^\top \leftarrow QR(\bar{Y})$
 - 11: $\text{Coef}_{(c)} \leftarrow W_{(c)}^t V^\top$
 - 12: Encrypt $\text{Coef}_{(c)}$ and $\theta_{(c)}^{nd}$ and send to server
-

the proposed approach, covering both the standard coefficient update flow and the case where basis are updated (lines 5–11). We also provide in Algorithm 1, the training process from the server-side perspective. The discussed flow assumes full client participation in each round of FL. For partial participation, where only a subset of clients is selected per round, a simple modification is required. When a client is selected after skipping one or more rounds, it may hold an outdated version of the shared basis. In this case, along with the current global coefficients and second group parameters, the server additionally transmits the latest shared sketch. The client then encrypts and updates its basis and proceeds with the standard training flow described in Algorithm 2.

V. EVALUATION

In our evaluation, we consider an FL setup with full client participation. The system consists of 10 clients, where each client performs one local epoch on its training data before sending updates to the server. We evaluate on the independent and identically distributed (iid) scenario, where data is evenly distributed across clients, and the non-iid scenario, generated via a Dirichlet distribution with $\alpha = 0.3$. Experiments are conducted on CIFAR-10 and CIFAR-100 [43] using ResNet18 and ResNet34 [44], respectively. We use a learning rate of 0.01 for all approaches. We use 12th Gen Intel(R) Core(TM) i9-12900 for the server and Jetson Orin Nano for the clients. We assume 2MB/s for the uplink rate. Furthermore, we assume the server broadcasting time to be negligible as a common assumption in FL [1], [45]. We note that even in the absence of the broadcasting assumption, our method has further potential to achieve superior results.

For HE, we adopt the CKKS encryption scheme with polynomial degree 8192, cipher modulus of 200, and scaling factor of 2^{40} . To demonstrate the generality of our framework and show that it is not limited to a specific HE scheme, we further extend our evaluation on Paillier and Brakerski–Fan–Vercauteren (BFV). For CKKS and BFV, we use TenSeal [46]. For Paillier, we used Intel cryptopaillier [47] and Python Paillier on the clients with key size of 2048.

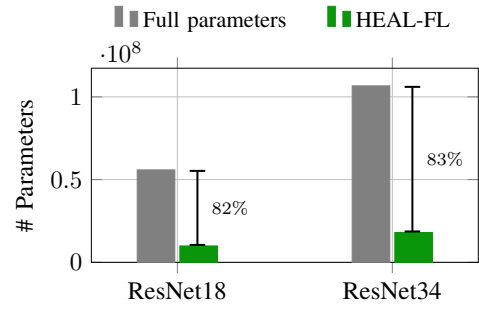


Fig. 2. Effective number of parameters to be encrypted, transmitted, and decrypted by a single client over 5 rounds.

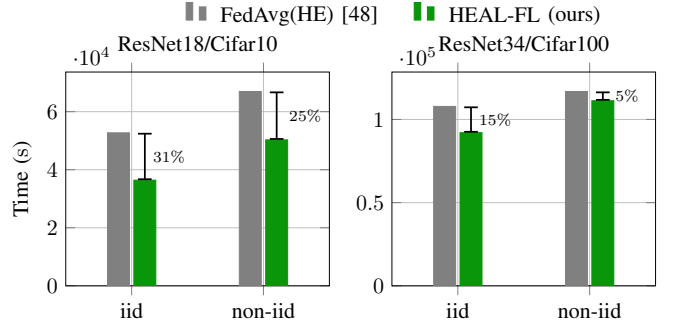


Fig. 3. Total time comparison between our proposed approach HEAL-FL and FedAvg under CKKS homomorphic scheme.

For training time, we account for both client and server-side operations. The client-side time includes the cost of local training, encryption of the coefficient set (and the shared basis matrices which are transmitted every 5 rounds), uploading the encrypted payload to the server, and decrypting the aggregated ciphertexts received from the server. Since we assume that all clients run in parallel, the time per round is approximated by the time taken for a single representative client plus the aggregation time at the server. The total training time is then obtained over all FL rounds required until convergence.

A. Encrypted and Transmitted Parameters

To quantify the efficiency gains of our proposed HEAL-FL, we first analyze the effective number of encrypted parameters exchanged between a client and the server. Figure 2 shows the effective count of parameters that require encryption, communication, and decryption over 5 rounds for ResNet18 and ResNet34. This count includes both the regular update rounds and the periodic basis update used in our approach and is compared against the cumulative size of transmitting the full model in every round. Using our method, the number of parameters exchanged is reduced by 82.3% and 83.1% compared to transmitting the full model parameters for ResNet18 and ResNet34, respectively.

B. Comparison Against State of the Art

In this section, we compare our approach with representative state-of-the-art of FL with HE methods. BatchCrypt [22] is designed for the federated SGD (FedSGD) setting, where their compression strategy is applied to per-step gradients. This differs fundamentally from the currently adopted FedAvg

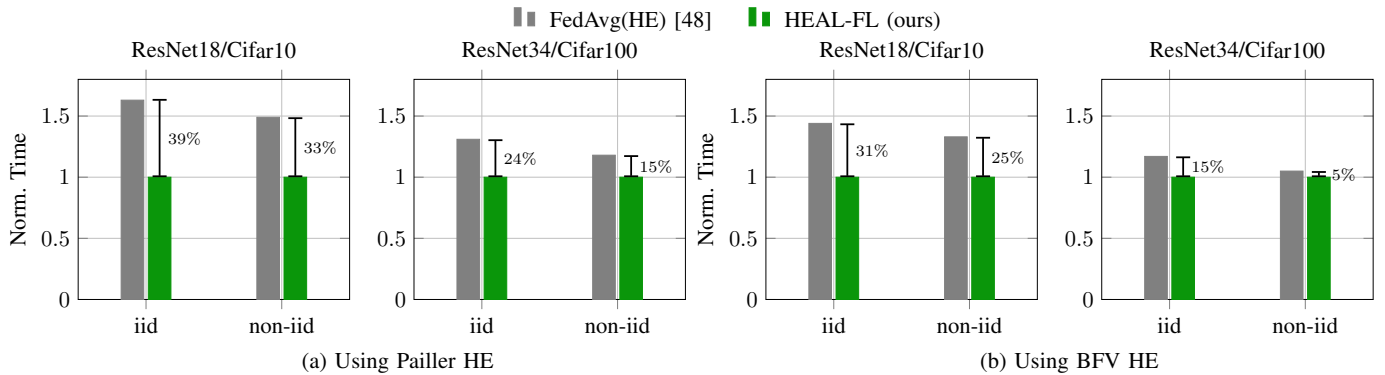


Fig. 4. Normalized time comparison between our proposed approach HEAL-FL and FedAvg under Paillier and BFV homomorphic encryption schemes. Our approach consistently achieves lower training time across schemes, model and datasets.

setting, where clients perform multiple local training steps before uploading the model parameters to the server. For example, using the current evaluation setup with data evenly partitioned across clients and a local batch size of 64, adopting the FedSGD setting would require 78 rounds of gradients to a single FedAvg-equivalent round, which makes the FedSGD setting communication inefficient since the gradients have the same size as the model.

Pack [23] adopts the FedAvg setting and is specifically tailored to compress ciphertexts under the CKKS HE scheme. To avoid model divergence caused by their compression technique, their method introduces several additional components (with extra overhead), including training error-correction models on clients, filtering weights with large errors for update, and periodically transmitting the full uncompressed ciphertext. However, since the paper does not provide sufficient details on how these error-correction models are trained (e.g., model architecture, hyperparameters, or training procedure) and different FL evaluation setup, we compare on round-level metrics. In regular rounds, Pack reduces client-side encryption time by 16% and upload cost by 63% compared to sending uncompressed ciphertexts, while the aggregation time remains unchanged. In contrast, our approach yields larger improvements across all these aspects by directly reducing the number of parameters exchanged and aggregated in each round by approximately 83% on average, accounting for both coefficient and less frequent basis update rounds. Other works that focus on accelerating aggregation and encryption through hardware optimizations [16], [20], [21] are orthogonal to ours.

Finally, we provide a detailed comparison of our approach against state-of-the-art FedAvg with HE [48]. As a first step, we present results using the CKKS scheme, as illustrated in Figure 3. Our proposed HEAL-FL typically requires a higher number of FL rounds to converge due to the additional compression step. However, the total training time is reduced because the savings in encryption, decryption, upload, and server-side aggregation time per round outweigh the increase in the number of rounds. Our approach yields up to 30.9% reduction in training time with an average of 18.8% across all evaluated models and data distribution settings.

To demonstrate the generality of our protocol, we further

extend our evaluation to additional HE schemes, starting with the BFV scheme. As shown in Figure 4(b), the results follow a similar trend to CKKS, exhibiting comparable reductions of up to 30.5% in total training time. With Paillier, our method achieves the largest gains, up to 38.6% faster than FedAvg as shown in Figure 4(a). These improvements arise because the communication and encryption-time savings become even more dominant under Paillier.

Finally, we compare the maximum accuracy achieved by each method, as reported in Table I. Despite utilizing low-rank updates and transmitting only compact coefficient-basis representations, our method does not incur any accuracy degradation compared to fully encrypted FedAvg.

In summary, our method shows reductions in communication and homomorphic operations within FL, while maintaining the quality of the model.

TABLE I
ACCURACY COMPARISON.

Method	ResNet18/CIFAR10		ResNet34/CIFAR100	
	iid	non-iid	iid	non-iid
FedAvg(HE)	91.6%	90.2%	71.0%	70.2%
HEAL-FL	90.7%	88.8%	70.7%	68.3%

VI. CONCLUSION

In this work, we consider the integration of homomorphic encryption in federated learning, where the primary system bottlenecks shift from model training to the costs of encryption, ciphertext transmission, and server-side aggregation. To address these challenges, we propose HEAL-FL, a framework that decomposes model updates into a shared basis and compact coefficient representations. By transmitting only the low-dimensional coefficients in most rounds, our approach reduces the number of encrypted parameters, thereby lowering the costs of encryption, decryption, communication, and aggregation. Our evaluation reveals that our framework achieves nearly 38.6% reduction in the overall training time compared to FedAvg on the Paillier encryption scheme. Overall, our work provides a practical and general approach to enabling scalable, privacy-preserving federated learning under homomorphic encryption.

REFERENCES

- [1] K. Pfeiffer *et al.*, “Federated learning for computationally constrained heterogeneous devices: A survey”, *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–27, 2023.
- [2] I. Dayan *et al.*, “Federated learning for predicting clinical outcomes in patients with covid-19”, *Nature medicine*, vol. 27, no. 10, pp. 1735–1743, 2021.
- [3] X. Xu *et al.*, “Privacy-preserving federated depression detection from multisource mobile health data”, *IEEE transactions on industrial informatics*, vol. 18, no. 7, pp. 4788–4797, 2021.
- [4] Y. Li *et al.*, “A blockchain-based decentralized federated learning framework with committee consensus”, *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.
- [5] L. Melis *et al.*, “Exploiting unintended feature leakage in collaborative learning”, in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 691–706.
- [6] Z. Wang *et al.*, “Beyond inferring class representatives: User-level privacy leakage from federated learning”, in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 2512–2520.
- [7] I. Petrov *et al.*, “Dager: Exact gradient inversion for large language models”, *Advances in Neural Information Processing Systems*, vol. 37, pp. 87 801–87 830, 2024.
- [8] E. J. Hu *et al.*, “Lora: Low-rank adaptation of large language models.” *ICLR*, vol. 1, no. 2, p. 3, 2022.
- [9] S. Babakniya *et al.*, “Slora: Federated parameter efficient fine-tuning of language models”, *arXiv preprint arXiv:2308.06522*, 2023.
- [10] L. Yi *et al.*, “pFedLora: Model-heterogeneous personalized federated learning with lora tuning”, *arXiv preprint arXiv:2310.13283*, 2023.
- [11] H. Fang *et al.*, “Privacy preserving machine learning with homomorphic encryption and federated learning”, *Future Internet*, vol. 13, no. 4, p. 94, 2021.
- [12] H. Nassar *et al.*, “Turbo-fhe: Accelerating fully homomorphic encryption with fpga and hbm integration”, *IEEE Design & Test*, 2025.
- [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] F. Qiu *et al.*, “Privacy preserving federated learning using ckks homomorphic encryption”, in *International conference on wireless algorithms, systems, and applications*. Springer, 2022, pp. 427–440.
- [15] J. H. Cheon *et al.*, “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.
- [16] Z. Wang *et al.*, “Pipefl: Hardware/software co-design of an fpga accelerator for federated learning”, *IEEE Access*, vol. 10, pp. 98 649–98 661, 2022.
- [17] Z. Chen *et al.*, “Safe: A scalable homomorphic encryption accelerator for vertical federated learning”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.
- [18] W. Liu *et al.*, “Efficient fast additive homomorphic encryption cryptoprocessor for privacy-preserving federated learning aggregation”, in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2024, pp. 1–6.
- [19] H. Nassar *et al.*, “Hbmorphic: Fhe acceleration via hbm-enabled recursive karatsuba multiplier on fpga”, in *2024 IEEE 32nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2024, pp. 217–217.
- [20] J. Lee *et al.*, “Configurable encryption and decryption architectures for ckks-based homomorphic encryption”, *Sensors*, vol. 23, no. 17, p. 7389, 2023.
- [21] K. Zhang *et al.*, “Hardware acceleration for fully homomorphic encryption scheme switching from ckks to fhe”, in *2024 58th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2024, pp. 1792–1796.
- [22] C. Zhang *et al.*, “{BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning”, in *2020 USENIX annual technical conference (USENIX ATC 20)*, 2020, pp. 493–506.
- [23] Z. Zuo *et al.*, “Pack: Towards communication-efficient homomorphic encryption in federated learning”, in *Proceedings of the 2024 ACM Symposium on Cloud Computing*, 2024, pp. 470–486.
- [24] X. Guo *et al.*, “Fedpower: privacy-preserving distributed eigenspace estimation”, *Machine Learning*, vol. 113, no. 11, pp. 8427–8458, 2024.
- [25] R. Hagag *et al.*, “Hardware-accelerated mode-switching polymorphic encryption for privacy preserving machine learning”, in *2025 14th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. IEEE, 2025, pp. 1–5.
- [26] M. A. Ahmed *et al.*, “Accelerated training on low-power edge devices”, *Transactions on Machine Learning Research*, 2025.
- [27] B. McMahan *et al.*, “Communication-efficient learning of deep networks from decentralized data”, in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.
- [28] S. Bothe *et al.*, “Client-server framework for fpga acceleration of fan-vercauteren-based homomorphic encryption”, in *2024 International Conference on Microelectronics (ICM)*. IEEE, 2024, pp. 1–5.
- [29] K. Pfeiffer *et al.*, “Aggregating capacity in fl through successive layer training for computationally-constrained devices”, in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS ’23. Red Hook, NY, USA: Curran Associates Inc., 2023.
- [30] Y. Yan *et al.*, “Federa: Efficient fine-tuning of language models in federated learning leveraging weight decomposition”, *arXiv preprint arXiv:2404.18848*, 2024.
- [31] K. Pfeiffer *et al.*, “Energy-aware heterogeneous federated learning via approximate dnn accelerators”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 6, pp. 2054–2066, 2025.
- [32] G. Xia *et al.*, “Poisoning attacks in federated learning: A survey”, *IEEE Access*, vol. 11, pp. 10 708–10 722, 2023.
- [33] X. Li *et al.*, “Lomar: A local defense against poisoning attack on federated learning”, *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, pp. 437–450, 2021.
- [34] S. Awan *et al.*, “Contra: Defending against poisoning attacks in federated learning”, in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer, 2021, pp. 455–475.
- [35] W. Jin *et al.*, “Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system”, *arXiv preprint arXiv:2303.10837*, 2023.
- [36] R.-Y. Huang *et al.*, “Toward efficient homomorphic encryption-based federated learning: A magnitude-sensitivity approach”, in *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 2024, pp. 7810–7821.
- [37] M. Luo *et al.*, “Fedlrs: A communication-efficient federated learning framework with low-rank and sparse decomposition”, in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2023, pp. 2091–2099.
- [38] H. Wang *et al.*, “Svdfed: Enabling communication-efficient federated learning via singular-value-decomposition”, in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.
- [39] D. Chai *et al.*, “Efficient decentralized federated singular vector decomposition”, in *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, 2024, pp. 1029–1047.
- [40] D. Froelicher *et al.*, “Scalable and privacy-preserving federated principal component analysis”, in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1908–1925.
- [41] A. Grammenos *et al.*, “Federated principal component analysis”, *Advances in neural information processing systems*, vol. 33, pp. 6453–6464, 2020.
- [42] D. Chai *et al.*, “Practical lossless federated singular vector decomposition over billion-scale data”, in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022, pp. 46–55.
- [43] A. Krizhevsky *et al.*, “Learning multiple layers of features from tiny images”, 2009.
- [44] K. He *et al.*, “Deep residual learning for image recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [45] Z. Yang *et al.*, “Energy efficient federated learning over wireless communication networks”, *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2020.
- [46] A. Benaissa *et al.*, “Tenseal: A library for encrypted tensor operations using homomorphic encryption”, 2021.
- [47] Intel Corporation, “Intel paillier cryptosystem library”. [Online]. Available: https://github.com/intel/pailliercryptolib_python
- [48] J. Park *et al.*, “Privacy-preserving federated learning using homomorphic encryption”, *Applied Sciences*, vol. 12, no. 2, p. 734, 2022.