

---

# DictPFL: Efficient and Private Federated Learning on Encrypted Gradients

---

Jiaqi Xue<sup>1</sup>, Mayank Kumar<sup>1</sup>, Yuzhang Shang<sup>1</sup>, Shangqian Gao<sup>2</sup>, Rui Ning<sup>3</sup>

Mengxin Zheng<sup>1</sup>, Xiaoqian Jiang<sup>4</sup>, Qian Lou<sup>1</sup>

<sup>1</sup>University of Central Florida   <sup>2</sup>Florida State University   <sup>3</sup>Old Dominion University

<sup>4</sup>University of Texas Health Science Center at Houston

{jiaqi.xue, mayank.kumar, yuzhang.shang, mengxin.zheng, qian.lou}@ucf.edu  
sgao@cs.fsu.edu, rning@odu.edu, xiaoqian.jiang@uth.tmc.edu

## Abstract

Federated Learning (FL) enables collaborative model training across institutions without sharing raw data. However, gradient sharing still risks privacy leakage, such as gradient inversion attacks. Homomorphic Encryption (HE) can secure aggregation but often incurs prohibitive computational and communication overhead. Existing HE-based FL methods sit at two extremes: encrypting all gradients for full privacy at high cost, or partially encrypting gradients to save resources while exposing vulnerabilities. We present **DictPFL**, a practical framework that achieves full gradient protection with minimal overhead. DictPFL encrypts every transmitted gradient while keeping non-transmitted parameters local, preserving privacy without heavy computation. It introduces two key modules: **Decompose-for-Partial-Encrypt (DePE)**, which decomposes model weights into a static dictionary and an updatable lookup table—only the latter is encrypted and aggregated, while the static dictionary remains local and requires neither sharing nor encryption; and **Prune-for-Minimum-Encrypt (PrME)**, which applies encryption-aware pruning to minimize encrypted parameters via consistent, history-guided masks. Experiments show that DictPFL reduces communication cost by **402–748×** and accelerates training by **28–65×** compared to fully encrypted FL, while outperforming state-of-the-art selective encryption methods by **51–155×** in overhead and **4–19×** in speed. Remarkably, DictPFL’s runtime is within **2×** of plaintext FL, demonstrating—for the first time—that HE-based private federated learning is practical for real-world deployment. The code is publicly available at <https://github.com/UCF-ML-Research/DictPFL>.

## 1 Introduction

Federated Learning (FL) [1] was introduced to enable collaborative training of a shared machine learning model among different data owners (e.g., hospitals or banks), where model gradients (or weights), rather than raw data, are shared to address privacy concerns. However, even sharing gradients poses privacy risks, as attackers could potentially exploit this information. For instance, model inversion (or gradient inversion) attacks [2, 3] have demonstrated the feasibility of reconstructing a client’s original training data from the gradients shared by clients. In such scenarios, the server or users with access to the server can act as potential attackers.

To protect the privacy of clients’ gradients during aggregation and enable private FL, various privacy-preserving primitives such as Differential Privacy (DP) [4, 5], Secure Multiparty Computation (MPC) [6, 7], and Homomorphic Encryption (HE) [8–20] have been utilized. Among these methods, HE is especially appealing in cross-silo settings [21], as it provides non-interactive privacy

protection without the accuracy-privacy trade-off associated with DP [22–24] and without requiring the assumption of trusted servers, as in MPC [25–28]. In HE-based privacy-preserving federated learning [21, 29–31], locally updated gradients are encrypted by clients before being shared with the server, allowing the server to perform homomorphic aggregation directly on ciphertexts. Despite its security benefits, HE introduces significant overhead: ciphertext expansion increases communication costs by 1 to 3 orders of magnitude, while encryption, decryption, and homomorphic aggregation impose high computational costs [21, 31].

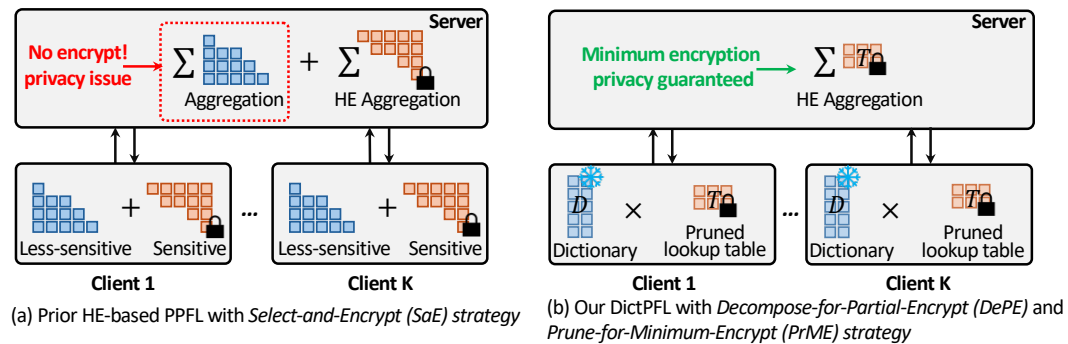


Figure 1: (a) Prior HE-based FL [31] encrypts only deemed sensitive gradients. The less-sensitive weights are shared in plaintext, which may lead to privacy concerns. (b) In contrast, our DictPFL minimizes encryption while ensuring privacy guarantees through Decompose-for-Partial-Encrypt (DePE) and Prune-for-Minimum-Encrypt (PrME). DePE decomposes gradients into a frozen dictionary and a trainable lookup table, with only the lookup table being encrypted and shared for aggregation. PrME further prunes the lookup table parameters on the client side to reduce encryption costs.

Prior efforts to improve the efficiency of HE-based FL often compromise privacy. The state-of-the-art method, FedML-HE [31], as illustrated in Figure 1 (a), adopts a *Select-and-Encrypt (SaE)* strategy: clients pre-compute sensitivity scores for model parameters and encrypt only the gradients of the most sensitive subset (e.g., the top 10%), while transmitting the remaining parameters in plaintext. However, these unencrypted gradients still expose private information. As shown in Figure 7 (a), when 30% of gradients remain unencrypted under FedML-HE, gradient inversion attacks can reconstruct images with up to 23% similarity to the originals. Moreover, the pre-computed sensitivity scores fail to capture dynamic sensitivity shifts during training, as parameter updates continually alter their privacy relevance. Consequently, encrypting all transmitted gradients remains essential to eliminate leakage. Although the SaE strategy achieves lower communication overhead and faster training than fully encrypted methods, it inevitably exposes privacy risks due to the shared plaintext gradients.

To address this challenge, we propose DictPFL, as shown in Figure 1 (b), which ensures that all shared parameters are fully encrypted to guarantee privacy while minimizing the number of shared parameters through two modules: *Decompose-for-Partial-Encrypt (DePE)* and *Prune-for-Minimum-Encrypt (PrME)*. *DePE* decomposes the model weights to be trained into a globally consistent dictionary, which is identical across all clients, and a lookup table, which each client trains independently. Only the encrypted gradients of the lookup table are shared with the server for aggregation, while the globally consistent dictionary remains frozen and is *never* shared. Building on *DePE*, *PrME* further reduces encrypted lookup tables through consistent pruning across clients. Unlike plaintext-level pruning in FL [32–34], where clients perform pruning locally and the server aligns the retained gradients before aggregation, HE-based FL presents unique challenges: retained gradients are batch-encrypted into ciphertexts in a SIMD format, preventing the server from aligning them without decryption. *PrME* addresses this by pruning based on shared global gradient history, ensuring consistent indices. Additionally, dynamic probabilities are assigned to the pruned parameters, allowing for their potential reintroduction in future rounds and mitigating the negative effects of premature pruning. Since the pruned lookup tables are significantly smaller than the full model weights, and all transmissions are encrypted, this approach substantially reduces the number of ciphertexts without compromising privacy.

Extensive experiments demonstrate that **DictPFL** delivers substantial performance gains over the state-of-the-art FedML-HE [31] across diverse tasks, including (i) image recognition, (ii) text classification,

and (iii) text generation. Compared with fully encrypted frameworks [35], DictPFL outperforms the selectively encrypted FedML-HE [31] by lowering communication overhead by **51–155×** and speeding up training by **4–19×**. Remarkably, DictPFL introduces less than a **2×** training-time increase even relative to plaintext FL, demonstrating that homomorphic encryption—commonly considered prohibitively expensive—can in fact be practical for federated learning at scale.

## 2 Background and Motivation

### 2.1 Privacy-preserving Federated Learning

Federated Learning (FL) enables collaborative training among distributed clients without directly sharing their datasets. In this framework, clients train their models locally and send gradients (or model updates) to a central server, which aggregates them using algorithms such as FedAvg [36] and FedSGD [1]. However, the direct exposure of local gradients to the server poses severe privacy risks [37]. For instance, with access to a client’s local gradients, the server can perform model inversion attacks [2, 3, 38] to reconstruct the client’s dataset.

Several methods have been proposed to protect the gradients transmitted between clients and the server. One strategy employs Differential Privacy (DP) [22–24] by injecting noise into the gradients before sharing them. Although DP imposes minimal computational overhead, it inevitably degrades model performance due to the added noise. Secure Multi-Party Computation (MPC) [25, 28] ensures that the server can access only aggregated gradients rather than individual ones. However, the aggregated gradients remain exposed to the server, and the reliance on multiple non-colluding servers makes MPC unsuitable for single-server settings.

Another approach leverages Homomorphic Encryption (HE) [21, 31] to encrypt gradients on the client side, enabling the server to aggregate them without decryption. HE provides end-to-end protection by securing gradient transmission, aggregation, and server storage. This protection addresses multiple security threats, including adversaries in network communications, multi-tenant vulnerabilities during computation on servers, and insider attacks on stored data. While platforms such as IBM FL [39] and Nvidia FLARE [35] have explored integrating HE into FL, they fail to address its significant overhead. As shown in Figure 2, HE-related operations dominate training time, and ciphertext expansion substantially increases communication costs. Reducing HE’s computational and communication overhead is key to realizing its practical benefits in FL.

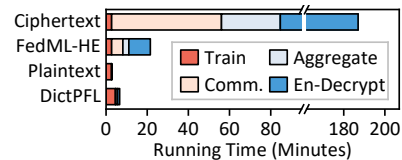


Figure 2: Time breakdown for training a ViT on GTSRB.

### 2.2 Efficient HE-based Federated Learning

Many efforts have been made to improve the efficiency of HE-based FL. These approaches can be broadly classified into two categories: encryption-scheme optimization and algorithmic optimization.

Quantization [21, 40–43] and packing [21, 44, 45] are widely studied techniques within the realm of encryption-scheme optimization for HE-based FL. Quantization reduces communication costs by converting high-precision gradients into low-precision values, whereas packing (also referred to as batching) consolidates multiple local gradients into a single plaintext, significantly reducing the number of plaintexts that need to be encrypted and transmitted.

Algorithmic optimization involves tailoring efficient strategies based on the characteristics of the machine learning model, and our DictPFL falls into this category. The state-of-the-art work, FedML-HE [31] proposes to selectively encrypt the gradients based on privacy-sensitive scores, i.e., *Select-and-Encrypt (SaE)*, as shown in Figure 1 (a). However, it suffers from several critical limitations. First, privacy-sensitive scores are computed once before training and remain static throughout the training process. This static approach fails to account for how weight sensitivity changes during training, because weights classified as non-sensitive on the initialized model may later become critical for privacy protection. Most critically, it cannot ensure complete privacy protection. Since only the gradients of selected parameters are encrypted, the remaining gradients are transmitted in plaintext, leading to inevitable information leakage and making it impossible to guarantee privacy protection regardless of which gradients are selected for encryption. Additionally, as illustrated in Figure 2, although FedML-HE substantially reduces the communication overhead and HE operations

(including aggregation, encryption, and decryption) by a factor of ten when only the top 10% of sensitive parameters are encrypted, these overheads induced by ciphertexts are still primary bottlenecks in the training process. In contrast, our DictPFL effectively reduces HE-related overhead and achieves efficiency comparable to non-private plaintext FL.

## 2.3 Motivation

As illustrated in Figure 2, communication and computation overheads caused by ciphertexts become the main bottleneck in HE-based FL. Although the state-of-the-art FedML-HE [31] attempts to improve efficiency by selectively omitting encryption for partial parameters, it not only compromises privacy but also continues to struggle with significant HE-induced communication and computation overheads. To achieve higher efficiency without sacrificing privacy, we focus on reducing the total number of trainable parameters. Guided by this principle, we propose DictPFL, which employs two strategies: *Decompose-for-Partial-Encrypt (DePE)* (Sec. 4.1) to decompose gradients and *Prune-for-Minimum-Encrypt (PrME)* (Sec. 4.2) to prune the gradients of parameters with minimal updates.

## 3 Preliminaries

### 3.1 System Overview

Same with FedML-HE [31], the workflow of HE-based privacy-preserving federated learning begins with clients using a trusted key authority to generate a public-secret HE key pair. During each training iteration: (1) clients compute local gradients; (2) these gradients are encrypted with the public key and transmitted to the server; (3) the server aggregates the encrypted gradients; and (4) the aggregated ciphertext is broadcast back to the clients, who decrypt it using their secret keys and update their local models with the decrypted result.

### 3.2 Threat Model

We consider a semi-honest adversary  $\mathcal{A}$  that may corrupt the server, which is the same as the setting of FedML-HE [31]. While  $\mathcal{A}$  follows the protocol, it attempts to infer private information from benign participants. Security guarantees ensure  $\mathcal{A}$  learns no information from the data of clients.

## 4 DictPFL

DictPFL consists of two modules: *Decompose-for-Partial-Encrypt (DePE)* and *Prune-for-Minimum-Encrypt (PrME)*. *DePE* decomposes model weights into a fixed global dictionary  $D$  and a trainable lookup table  $T$ . Only the encrypted gradients of  $T$  are transmitted and aggregated, whereas  $D$  remains identical across clients and never leaves local devices. *PrME* further reduces encryption cost by pruning parameters with persistently small gradients, using shared historical statistics to ensure consistent pruning across clients. Together, these two modules ensure that all transmitted gradients are encrypted and all unencrypted ones remain strictly local, while significantly reducing the number of ciphertexts exchanged between clients and the server to achieve high efficiency without compromising privacy.

### 4.1 Decompose-for-Partial-Encrypt (DePE)

Model weight decomposition, representing a weight matrix  $W$  as a linear combination of vectors from a compact dictionary  $D$  and a sparse lookup table  $T$ , is a proven strategy for parameter reduction in inference [46–48]. The key insight lies in reducing the inherent redundancy in weight parameters: correlated parameters can be represented as sparse linear combinations of a dictionary of vectors. We adapt this principle to HE-based FL, where reducing the dimensionality of trainable parameters directly minimizes the number of ciphertexts.

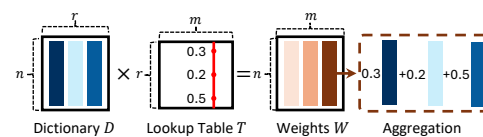


Figure 3: Representing the weight matrix  $W$  with dictionary  $D$  and lookup table  $T$ .

**Constructing  $W$  with  $D$  and  $T$ .** Figure 3 demonstrates the construction of the weight matrix  $W \in \mathbb{R}^{n \times m}$  using a dictionary  $D \in \mathbb{R}^{n \times r}$  and a lookup table  $T \in \mathbb{R}^{r \times m}$ . Each column vector  $W[:, i]$  of  $W$  is derived through a linear combination of the  $r$  vectors in  $D$ , weighted by the corresponding scalars in the  $i$ -th column of  $T$ , denoted  $T[:, i]$ . This process is formally expressed by:

$$W[:, i] = \sum_{k=0}^r D[:, k] \cdot T[k][i] \quad (1)$$

Take Figure 3 as an example. Given  $r = 3$  and the  $i$ -th column of  $T$  as  $[0.3, 0.2, 0.5]$ , the corresponding  $i$ -th column of weights in  $W$  can be represented as  $W[:, i] = 0.3 \cdot D[:, 0] + 0.2 \cdot D[:, 1] + 0.5 \cdot D[:, 2]$ . By reducing  $r$ , the dictionary size, we effectively decrease the number of trainable parameters and thus reduce the communication overhead associated with ciphertexts.

**Factorization of Dictionary and Lookup Tables.** To ensure that the dictionary  $D$  contains critical and generalizable weight vectors while reducing parameter redundancy, we employ a truncated SVD factorization to decompose the weights to be trained, i.e.,  $W_0$ , which has dimensions  $n \times m$ , into a smaller dictionary  $D$  and a lookup table  $T'$ . Specifically,  $W_0$  is approximated as  $U_r \Sigma_r V_r^\top$ , where  $U_r$ ,  $\Sigma_r$ , and  $V_r^\top$  correspond to the top- $r$  singular values and vectors, thus reducing the dimensionality to  $n \times r$  for  $D$  and  $r \times m$  for  $T'$ ,

$$W_0 \approx U_r \Sigma_r V_r^\top \quad (2)$$

$$D, T' = \text{SVD}(W_0, r) = U_r \Sigma_r, V_r^\top \quad (3)$$

DePE initializes  $D$  as  $U_r \Sigma_r$  and  $T'$  as  $V_r^\top$  according to Equation 3. However, directly freezing  $D$  and training  $T'$  can lead to suboptimal performance due to the information loss inherent in SVD truncation, particularly when  $r$  is much smaller than  $m$  or  $n$ . To counteract this, we retain the original weight  $W_0$  and initialize  $T$  by zeroing out  $T'$ . This strategy allows for the construction of  $W$  as  $W_0 + D \cdot T$ , with  $D$  remaining static and shared among all clients, while  $T$  is updated locally and aggregated on the server. By selecting a smaller  $r$ , we significantly reduce the communication overhead for encrypted parameters, as encryption is only required for the  $r \times m$  entries in  $T$ .

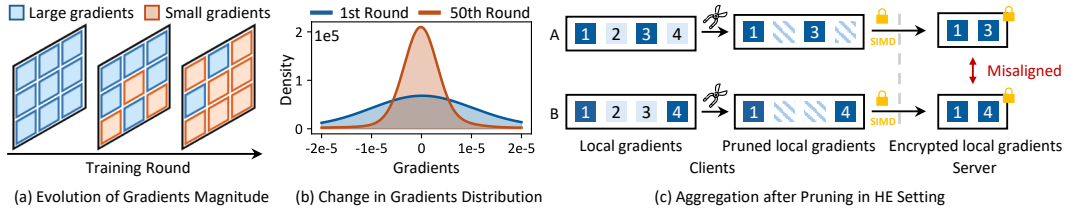


Figure 4: (a) As training progresses, parameters that initially have large gradients gradually transition to having smaller ones. (b) Concurrently, the number of parameters with substantial gradients decreases significantly. (c) An example of failed aggregation caused by different pruning locations on clients A and B.

## 4.2 Prune-for-Minimum-Encrypt (PrME)

As DePE training progresses, the number of parameters with large gradients gradually declines, as shown in Figure 4 (a). By the 50-th training round, only a small subset of parameters still exhibit gradients exceeding  $10^{-5}$ , as shown in Figure 4 (b). Encrypting and transmitting all gradients to the server for aggregation, including those of parameters that no longer change significantly, introduces unnecessary redundancy. By enabling clients to upload only substantial gradients, communication overhead can be dramatically reduced.

Existing gradients pruning methods in plaintext federated learning involve clients independently pruning their smallest local gradients before transmission to the server for aggregation. Since clients possess different local gradients, they may prune parameters at different positions, necessitating the sharing of pruning indices with the server to ensure proper aggregation. However, implementing such methods to HE-based federated learning presents two fundamental challenges. First, indices must be encrypted to protect privacy, while encrypted indices force the server to perform non-linear operations (e.g., comparing encrypted indices to match) alongside linear operations (e.g., aggregation), a hybrid workflow that incurs prohibitive computational overhead [49, 50]. Second, the SIMD batching

mechanism, which packs multiple plaintext gradients into several slots of a single ciphertext, renders index-specific operations infeasible. Since HE aggregation occurs slot-wise, gradients occupying the same slot across clients are combined automatically, regardless of their indices.

Figure 4 (c) illustrates the above challenges of pruned HE aggregation. Consider a scenario where client A encrypts and uploads gradients from positions 1 and 3, while client B encrypts and uploads gradients from positions 1 and 4. The server cannot perform correct aggregation because the ciphertext slots are misaligned, and the encryption prevents any coordination or realignment of the gradients. To ensure consistent gradient pruning across clients, they require an identical metric for determining which gradients to prune. The optimal approach would involve clients pruning their local gradients based on current round global gradients. However, clients cannot access the current round global gradients until after sharing their complete local gradients with the server for aggregation. This creates a dilemma: clients cannot prune independently as it leads to inconsistencies, nor can they rely on global gradients to coordinate pruning.

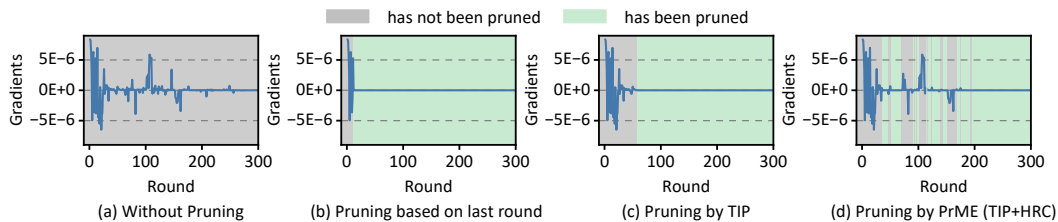


Figure 5: Evolution of a parameter's global gradients under different pruning strategies. Green background indicates the parameter is pruned (excluded from aggregation), while gray background indicates the opposite. Larger green areas reflect more overhead reduction. Closer alignment of gradient trends with the baseline (a) signifies preserved convergence performance.

**Temporal Inactivity Pruning (TIP).** To resolve this dilemma, clients require a shared pruning metric that is independent of the current round's global gradients. A straightforward solution is to base pruning decisions on the last round's global gradients, which are identical across clients and accessible before aggregation. Specifically, clients prune local gradients corresponding to parameters with the smallest  $s\%$  magnitudes from the prior global gradients. However, parameters showing minimal activity in one round may experience significant updates in subsequent rounds, leading to unintended removal if pruning decisions rely exclusively on last round gradients. For instance, as illustrated in Figure 5 (b), the parameter with a small gradient magnitude in an earlier round may be pruned, despite its gradient resurgence in later rounds, as indicated in Figure 5 (a).

To mitigate the influence of transient fluctuations and retain critical gradients, we introduce a temporal windowing strategy that leverages information from the previous  $\tau$  consecutive rounds. Clients identify parameters whose gradients fall within the smallest  $s\%$  across all  $\tau$  rounds (referred to as pruning patience). Formally, the pruning mask for parameter  $w_i$  at round  $t$  is defined as:

$$M_{i,t} = \begin{cases} 0 & \text{if } \sum_{k=1}^{\tau} \mathbf{1}(|\delta w_{i,t-k}| < \theta_{s,t-k}) = \tau \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

Here,  $M_{i,t} = 0$  indicates that the local gradient of  $w_i$  is pruned, while  $M_{i,t} = 1$  retains its local gradient for aggregation. The  $\delta w_{i,t-k}$  denotes the global gradient of parameter  $w_i$  at round  $t - k$ , and  $\mathbf{1}$  is the indicator function. The threshold  $\theta_{s,t-k}$  dynamically adapts as the  $(100-s)$ -th percentile of  $|\delta w_{i,t-k}|$ . As shown in Figure 5 (c), the pruning is postponed to a later round when gradients exhibit more stable behavior, thereby preserving gradients that regain significance after initially being considered for pruning.

**Holistic Reactivation Correction (HRC).** Although TIP reduces communication overhead while preventing premature pruning by chance, it still has an inherent limitation: once a parameter is pruned, its local gradients no longer participate in aggregation. Consequently, its global gradient magnitudes remain zero in subsequent rounds, effectively excluding it permanently. This irreversible pruning can hinder training convergence, as parameters with substantial gradients in later rounds may no longer be updated. For example, in Figure 5 (a), the example parameter may have significant gradient magnitudes even after the 100-th round.

To mitigate the performance loss caused by irreversible pruning, we propose a dynamic reactivation scheme, Holistic Reactivation Correction (HRC). Instead of permanently excluding pruned parameters, HRC assigns each pruned parameter  $w_i$  a reactivation probability  $p_i$ , which is dynamically adjusted based on its aggregated global gradients  $\delta w_{i,t}$  after reactivation:

$$p_i[t+1] = \begin{cases} p_i[t] \times \beta & \text{if } |\delta w_{i,t}| < \theta_{s,t} \\ \min(p_i[t]/\beta, 1) & \text{otherwise} \end{cases} \quad (5)$$

Here,  $\beta$  is a decay factor less than 1. When a pruned parameter is reactivated, the client uploads its *accumulated* local gradients since the pruning round for aggregation and gets the current round's global gradients  $\delta w_{i,t}$ . This approach preserves small gradients that, while individually minor, can meaningfully accumulate over time, rather than discarding these gradients, maintaining them locally for future aggregation helps convergence. If  $|\delta w_{i,t}| < \theta_{s,t}$ , indicating that the parameter's cumulative global gradients remain small even after reactivation, the reactivation probability  $p_i$  decreases, discouraging further reactivation. Conversely, if  $|\delta w_{i,t}| \geq \theta_{s,t}$ ,  $p_i$  increases, encouraging the update of this parameter to rejoin aggregation. This adaptive mechanism mitigates information loss from premature pruning by flexibly adjusting the likelihood of reactivation. Although HRC introduces some uncertainty, consistency across clients can be easily maintained by preserving a shared random seed for the pruning mask. Notably, our PrME does not need to share pruning mask to server, eliminating the risk of attacks via plaintext mask, e.g., inferring sensitive patterns from pruned parameter locations.

## 5 Experimental Methodology

**Datasets.** We conduct experiments on three image classification tasks: CIFAR-10 [51], GTSRB [52], and Diabetic Retinopathy [53], as well as AG's News [54] for sentence classification and MetaMathQA [55] for text generation. The experiments are performed under varying levels of data heterogeneity and different numbers of clients. We generate homogeneous data splits by randomly assigning training examples to individual clients without replacement. For heterogeneous settings, we simulate data heterogeneity by sampling the label ratios from a Dirichlet distribution with a symmetric parameter, following the [56]. In both settings, each client holds the same number of samples, following [57].

**Models.** We perform DictPFL on multiple prevalent transformer-based models including, ViT [58] designed for image recognition, BERT [59], and TinyLlama [60] for natural language processing.

**Baselines.** We compare DictPFL with three baselines: FedHE-Full [35], which trains the entire model and encrypts all gradients; FedHE-Top2, fine-tuning only the last two layers; and FedHE-ML [31], which encrypts a subset of gradients (10% unless specified otherwise) while leaving the rest in plaintext.

**Evaluation Metrics.** We assess the efficacy of our proposed DictPFL by comparing its communication overhead, training time, and model accuracy against existing methods. For privacy evaluation, we compare DictPFL with FedML-HE [31] in terms of potential privacy leakage. We utilize recovered image similarity scores derived from  $1 - \text{LPIPS}$ , where the Learned Perceptual Image Patch Similarity (LPIPS) [61] measures discrepancies between reconstructed and original images. Therefore, higher scores indicate greater similarity and consequently, higher privacy risks.

**Hyperparameters.** Unless otherwise specified, we set the dictionary size  $r$  to 4, the pruning ratio  $s\%$  to 70%, the pruning patience  $\tau$  to 3, and the reactivation probability scaler  $\beta$  to 0.2. Detailed analyses of these hyperparameters are provided in Section 6.2.

**HE Implementation.** We adopt the CKKS homomorphic encryption scheme with bootstrapping [62–64], implemented via OpenFHE [65]. The scheme is configured for 128-bit security following the Homomorphic Encryption Standard [66], with a cyclotomic ring dimension of  $N = 2^{16}$ , ciphertext modulus of 1555 bits, and multiplicative depth  $L = 12$ . Each ciphertext contains  $N/2 = 32,768$  slots, enabling parallelized SIMD operations [67]. Data encoding follows the approach in [68]. All experiments were conducted on an AMD Ryzen Threadripper PRO 3955WX processor (2.2 GHz) with 125 GB of memory.

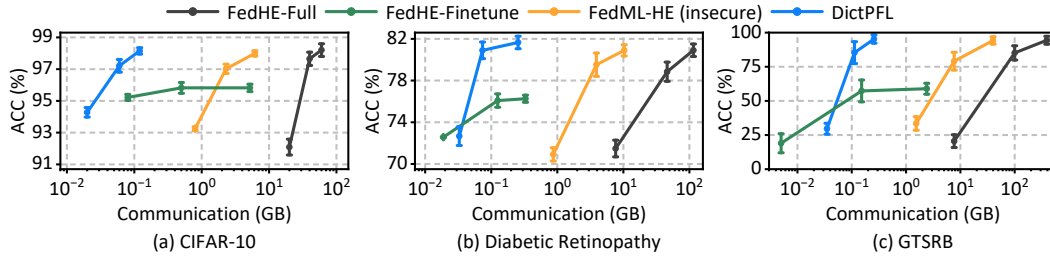


Figure 6: Efficiency comparison of different federated learning frameworks, in terms of accuracy versus communication overhead on three datasets using the ViT model. Higher efficiency is indicated by higher accuracy for the same communication or achieving the same accuracy with less communication, as shown by lines closer to the upper left corner. Communication is quantified by the total amount of data exchanged, including both plaintexts and ciphertexts, training iterations.

## 6 Results

### 6.1 Main Results

**Comparison with Existing Works.** To demonstrate DictPFL’s effectiveness, we compare it with other HE-based FL frameworks on the CIFAR-10, Diabetic Retinopathy, and GTSRB datasets using the ViT-16 model within a 3-client homogeneous setting. All experiments are conducted on the same pre-trained model to ensure a fair comparison. Figure 6 provides an overall comparison. Notably, DictPFL significantly and consistently reduces communication overhead compared to the baselines without sacrificing accuracy. Specifically, FedHE-Full has the highest communication. FedHE-Top2, which fine-tunes only the last two layers, shows reduced overhead but underperforms, because freezing most layers limits learning capacity, particularly on datasets that diverge from those used in pre-training. For instance, it achieves only 58.9% accuracy on GTSRB versus DictPFL’s 95.27%.

DictPFL achieves a 98.3% average reduction in communication overhead compared to the state-of-the-art FedML-HE (encrypting 10%), while maintaining the same level of accuracy. Although FedML-HE also reduces communication costs, it does so at the expense of privacy by exposing part of the gradients in plaintext. DictPFL, on the other hand, fully preserves privacy. This is further demonstrated in Figure 7 (a), which highlights the vulnerability of FedML-HE to state-of-the-art gradient inversion attacks [69]. Notably, DictPFL can prevent such privacy leakage for any data type, not only for vision tasks.

In addition to ViT, we evaluate several other models, as shown in Figure 7 (b). The results show that DictPFL consistently outperforms the baselines across models of different scales. Compared with the fully encrypted baseline FedHE-Full, DictPFL reduces communication by 402 to 748 times and accelerates training by 28 to 65 times. It also outperforms the selectively encrypted baseline FedML-HE by reducing overhead by 51 to 155 times and speeding up training by 4 to 19 times.

**Breakdown Analysis.** In Figure 8, we break down the training time for various HE-based FL frameworks under both LAN and WAN settings. In FedHE-Full, where all gradients are encrypted, communication and ciphertext-related operations (encryption, decryption, and aggregation) dominate the training time. FedHE-Top2 reduces communication and ciphertext-related operations by fine-tuning the last two layers, but this comes at the cost of reduced accuracy, achieving only 58.9%. On the contrary, our proposed DePE and PrME techniques significantly reduce the number of ciphertexts, resulting in a total training time that is 1 to 2 orders of magnitude lower than that of other baselines while maintaining a comparable level of accuracy.

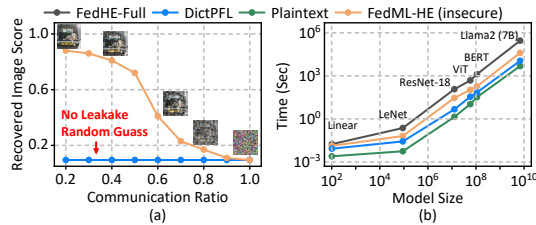


Figure 7: (a) Gradient inversion attacks against FedML-HE and DictPFL. The communication ratio is the communication overhead relative to encrypting the full-size model gradients in FedHE-Full. (b) Communication overhead of DictPFL and the baselines on models of different sizes.

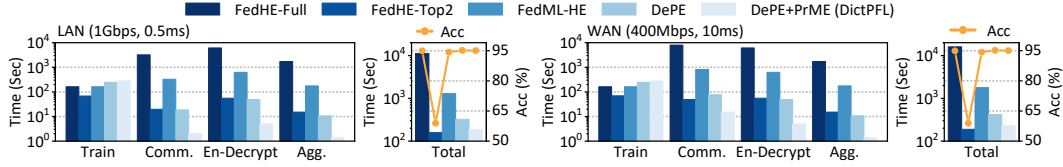


Figure 8: Training time breakdown of ViT on GTSRB under LAN and WAN settings.

## 6.2 Ablation Study

In this section, we explore the design space of DictPFL and study the impact of various settings on its performance. Unless otherwise specified, all experiments are conducted using the Diabetic Retinopathy dataset within a 3-client homogeneous setting within 10 rounds, and follows the default hyperparameter settings detailed in Section 5.

**Hyperparameters of DePE.** The dictionary size is a crucial hyperparameter in our DePE. A larger dictionary captures more comprehensive representations of gradients, enhancing accuracy but increasing overhead. As shown in Table 1, even a small dictionary with  $r = 4$  achieves commendable training performance, e.g., an accuracy of 81.99%, close to the 82.74% achieved by FedHE-Full. This efficacy stems from the dictionary’s ability to retain essential information corresponding to the largest singular values.

Table 1: Ablation on dictionary size  $r$ .

| $r$ | Accuracy (%) $\uparrow$ | Comm. (GB) $\downarrow$ | Time (min) $\downarrow$ |
|-----|-------------------------|-------------------------|-------------------------|
| 2   | 74.26 $\pm$ 0.5         | 0.046                   | 6.11 $\pm$ 0.1          |
| 4   | 81.99 $\pm$ 0.4         | 0.088                   | 6.23 $\pm$ 0.1          |
| 8   | 82.67 $\pm$ 0.2         | 0.160                   | 6.42 $\pm$ 0.2          |
| 16  | 82.71 $\pm$ 0.2         | 0.332                   | 7.27 $\pm$ 0.1          |

Table 2: Ablation on pruning patience  $\tau$ .

| $\tau$ | Accuracy (%) $\uparrow$ | Comm. (GB) $\downarrow$ | Time (min) $\downarrow$ |
|--------|-------------------------|-------------------------|-------------------------|
| 1      | 80.55 $\pm$ 0.6         | 0.001                   | 6.26 $\pm$ 0.1          |
| 3      | 82.29 $\pm$ 0.3         | 0.003                   | 6.36 $\pm$ 0.1          |
| 5      | 82.67 $\pm$ 0.2         | 0.160                   | 6.42 $\pm$ 0.2          |
| 10     | 82.77 $\pm$ 0.3         | 0.474                   | 6.92 $\pm$ 0.1          |

**Hyperparameters of PrME.** We explore the impact of pruning ratio  $s\%$  and pruning patience  $\tau$  in PrME. A higher  $s\%$  results in more minor gradients being pruned, whereas a lower value preserves them. As shown in Figure 9, without PrME (prune 0%), training converges rapidly within 10 rounds, but each round incurs the highest communication cost. Pruning 70% drastically reduces communication overhead but significantly affects accuracy. By contrast, pruning 20% preserves accuracy but results in far less communication reduction compared to the 70% pruning scenario. Notably, with our HRC reactivation scheme, prematurely pruned gradients in earlier rounds can be selectively reintroduced in later rounds. This enables the model to achieve accuracy similar to the 20% pruning scenario while achieving the communication efficiency of the 70% pruning ratio.

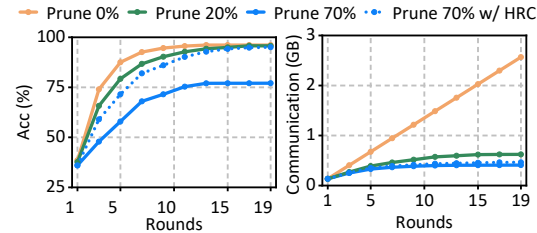


Figure 9: Ablation on the pruning ratio.

Table 2 studies different pruning patience  $\tau$ . Higher  $\tau$  values delay the pruning of gradients, reducing accuracy degradation but limiting communication reduction. Notably, setting  $\tau = 3$  already results in a small accuracy loss. This resilience can be attributed to our HRC, which mitigates the impact on accuracy by reintroducing pruned gradients, effectively correcting errors over time.

Table 4 in Appendix A.2 showcases that our PrME works well under various reactivation probability scalars  $\beta$ . For different numbers of clients and heterogeneous levels, we show the results in Appendix A.3 and A.4, which show that DictPFL performs well across different client scales and heterogeneous settings.

## 6.3 Other Experiments

The results for text tasks and large language models, including classification and generation tasks are in Appendix A.5. DictPFL outperforms all the baselines on language tasks. In Appendix A.7, we compare DictPFL with other non-HE based FL.

## 7 Conclusion

In this work, we present DictPFL, a novel framework for efficient HE-based FL. To address the prohibitive ciphertext-related overhead and eliminate information leakage, we propose *Decompose-for-Partial-Encrypt (DePE)*, which decomposes model weights into a static dictionary and a trainable lookup table. Only the small lookup table is encrypted and shared for aggregation, while the dictionary is never transmitted. To further improve communication efficiency, we propose *Prune-for-Minimum-Encrypt (PrME)*, which prunes gradients based on their long-term activity to minimize redundant ciphertext operations. Compared with the fully encrypted baseline, DictPFL accelerates training by up to  $65\times$  and outperforms the selectively encrypted FedML-HE by up to  $19\times$  while maintaining accuracy and fully eliminating privacy risks from partial plaintext gradient transmission, achieving a runtime only  $2\times$  that of plaintext FL.

## 8 Discussion

**Broader Impact.** The paper introduces DictFPL, a method designed to reduce the computational and communication overheads associated with protecting federated learning shared weights using homomorphic encryption. This approach enhances privacy protections without compromising accuracy, making it a more feasible solution for large-scale, real-world applications. By ensuring that sensitive weights remains private, DictFPL can accelerate the adoption of federated learning across industries such as healthcare, finance, and beyond, while fostering trust in AI systems and promoting global data privacy.

**Limitations.** Future work could explore broader scenarios, such as cross-device FL with constrained client resources or non-transformer model families. Moreover, since DictPFL employs a **fixed** shared dictionary, extending it to a **dynamic** dictionary design could enhance adaptability in highly heterogeneous client environments and improve model personalization.

## 9 Acknowledgement

This work was supported in part by NSF CSR-2413232. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of grant agencies or their contractors.

## References

- [1] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.
- [2] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.
- [3] Shanghao Shi, Ning Wang, Yang Xiao, Chaoyu Zhang, Yi Shi, Y Thomas Hou, and Wenjing Lou. Scale-mia: A scalable model inversion attack against secure federated learning via latent space reconstruction. *arXiv preprint arXiv:2311.05808*, 2023.
- [4] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.
- [5] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.
- [6] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure multiparty computation and secret sharing*. Cambridge University Press, 2015.
- [7] Yehuda Lindell. Secure multiparty computation. *Communications of the ACM*, 64(1):86–96, 2020.

- [8] Qian Lou and Lei Jiang. She: A fast and accurate deep neural network for encrypted data. *Advances in neural information processing systems*, 32, 2019.
- [9] Qian Lou, Bo Feng, Geoffrey Charles Fox, and Lei Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. *Advances in neural information processing systems*, 33:9193–9202, 2020.
- [10] Qian Lou, Wen-jie Lu, Cheng Hong, and Lei Jiang. Falcon: Fast spectral inference on encrypted data. *Advances in Neural Information Processing Systems*, 33:2364–2374, 2020.
- [11] Yancheng Zhang, Jiaqi Xue, Mengxin Zheng, Mimi Xie, Mingzhe Zhang, Lei Jiang, and Qian Lou. Cipherprune: Efficient and scalable private transformer inference. *arXiv preprint arXiv:2502.16782*, 2025.
- [12] Qian Lou and Lei Jiang. Hemet: A homomorphic-encryption-friendly privacy-preserving mobile neural network architecture. In *International conference on machine learning*, pages 7102–7110. PMLR, 2021.
- [13] Qian Lou, Yilin Shen, Hongxia Jin, and Lei Jiang. Safenet: A secure, accurate and fast neural network inference. In *International Conference on Learning Representations*, 2020.
- [14] Jiaqi Xue, Yancheng Zhang, Yanshan Wang, Xueqiang Wang, Hao Zheng, and Qian Lou. Cryptotrain: Fast secure training on encrypted dataset. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, pages 97–104, 2023.
- [15] Yancheng Zhang, Mengxin Zheng, Yuzhang Shang, Xun Chen, and Qian Lou. Heprune: Fast private training of deep neural networks with encrypted data pruning. *Advances in Neural Information Processing Systems*, 37:51063–51084, 2024.
- [16] Muhammad Husni Santrijaji, Jiaqi Xue, Yancheng Zhang, Qian Lou, and Yan Solihin. Datasael: Ensuring the verifiability of private computation on encrypted data. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 2378–2394. IEEE, 2025.
- [17] Mengxin Zheng, Qian Lou, and Lei Jiang. Primer: Fast private transformer inference on encrypted data. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.
- [18] Jiaqi Xue, Lei Xu, Lin Chen, Weidong Shi, Kaidi Xu, and Qian Lou. Audit and improve robustness of private neural networks on encrypted data. *arXiv preprint arXiv:2209.09996*, 2022.
- [19] Ardhi Wiratama Baskara Yudha, Jiaqi Xue, Qian Lou, Huiyang Zhou, and Yan Solihin. Boostcom: Towards efficient universal fully homomorphic encryption by boosting the word-wise comparisons. In *Proceedings of the 2024 International Conference on Parallel Architectures and Compilation Techniques*, pages 121–132, 2024.
- [20] Mayank Kumar, Jiaqi Xue, Mengxin Zheng, and Qian Lou. Tfhe-coder: Evaluating llm-agentic fully homomorphic encryption code generation. *arXiv preprint arXiv:2503.12217*, 2025.
- [21] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 493–506, 2020.
- [22] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pages 1–11, 2019.
- [23] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the third ACM international workshop on edge systems, analytics and networking*, pages 61–66, 2020.
- [24] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations*.

- [25] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.
- [26] Jinhyun So, Chaoyang He, Chien-Sheng Yang, Songze Li, Qian Yu, Ramy E Ali, Basak Guler, and Salman Avestimehr. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems*, 4:694–720, 2022.
- [27] Marcel Keller and Ke Sun. Secure quantized training for deep learning. In *International Conference on Machine Learning*, pages 10912–10938. PMLR, 2022.
- [28] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 56–62. IEEE, 2021.
- [29] Haokun Fang and Quan Qian. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, 13(4):94, 2021.
- [30] Zhifeng Jiang, Wei Wang, and Yang Liu. Flashe: Additively symmetric homomorphic encryption for cross-silo federated learning. *arXiv preprint arXiv:2109.00675*, 2021.
- [31] Weizhao Jin, Yuhang Yao, Shanshan Han, Carlee Joe-Wong, Srivatsan Ravi, Salman Avestimehr, and Chaoyang He. Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system. *arXiv preprint arXiv:2303.10837*, 2023.
- [32] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.
- [33] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 42–55, 2021.
- [34] Sameer Bibikar, Haris Vikalo, Zhangyang Wang, and Xiaohan Chen. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6080–6088, 2022.
- [35] Holger R Roth, Yan Cheng, Yuhong Wen, Isaac Yang, Ziyue Xu, Yuan-Ting Hsieh, Kristopher Kersten, Ahmed Harouni, Can Zhao, Kevin Lu, et al. Nvidia flare: Federated learning from simulation to real-world. *arXiv preprint arXiv:2210.13291*, 2022.
- [36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguerre y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [37] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.
- [38] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 603–618, 2017.
- [39] IBM. Ibmfl crypto. GitHub repository, 2022. Accessed: 2023-01-25.
- [40] Wuxing Xu, Hao Fan, Kaixin Li, and Kai Yang. Efficient batch homomorphic encryption for vertically federated xgboost. *arXiv preprint arXiv:2112.04261*, 2021.
- [41] Junhao Han and Li Yan. Adaptive batch homomorphic encryption for joint federated learning in cross-device scenarios. *IEEE Internet of Things Journal*, 2023.
- [42] Changsheng Zhao, Ting Hua, Yilin Shen, Qian Lou, and Hongxia Jin. Automatic mixed-precision quantization search of bert. *arXiv preprint arXiv:2112.14938*, 2021.

- [43] Qian Lou, Feng Guo, Lantao Liu, Minje Kim, and Lei Jiang. Autoq: Automated kernel-wise neural network quantization. *arXiv preprint arXiv:1902.05690*, 2019.
- [44] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security*, 13(5):1333–1345, 2017.
- [45] Changchang Liu, Supriyo Chakraborty, and Dinesh Verma. Secure model fusion for distributed learning using partial homomorphic encryption. *Policy-Based Autonomic Data Governance*, pages 154–179, 2019.
- [46] Qian Lou, Ting Hua, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. Dictformer: Tiny transformer with shared dictionary. In *International Conference on Learning Representations*, 2022.
- [47] Qian Lou, Yen-Chang Hsu, Burak Uzcent, Ting Hua, Yilin Shen, and Hongxia Jin. Lite-mdetr: A lightweight multi-modal detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12206–12215, 2022.
- [48] Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. *arXiv preprint arXiv:2207.00112*, 2022.
- [49] Yancheng Zhang, Xun Chen, and Qian Lou. Hebridge: Connecting arithmetic and logic operations in fv-style he schemes. In *Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 23–35, 2024.
- [50] Mansour Al Ghanim, Jiaqi Xue, Rochana Prih Hastuti, Mengxin Zheng, Yan Solihin, and Qian Lou. Uncovering the hidden threat of text watermarking from users with cross-lingual knowledge. *arXiv preprint arXiv:2502.16699*, 2025.
- [51] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.
- [52] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288.
- [53] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *jama*, 316(22):2402–2410.
- [54] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.
- [55] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.
- [56] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.
- [57] Geeho Kim, Jinkyu Kim, and Bohyung Han. Communication-efficient federated learning with accelerated client gradient. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12385–12394, 2024.
- [58] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [59] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.
- [60] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024.

- [61] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in neural information processing systems*, 34:7232–7241, 2021.
- [62] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I* 23, pages 409–437. Springer, 2017.
- [63] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full rms variant of approximate homomorphic encryption. In *Selected Areas in Cryptography–SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers* 25, pages 347–368. Springer, 2019.
- [64] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part I* 37, pages 360–384. Springer, 2018.
- [65] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915, 2022. <https://eprint.iacr.org/2022/915>.
- [66] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. *Protecting privacy through homomorphic encryption*, pages 31–62, 2021.
- [67] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71:57–81, 2014.
- [68] Eric Crockett. A low-depth homomorphic circuit for logistic regression model training. *Cryptology ePrint Archive*, 2020.
- [69] Yuxin Wen, Jonas A Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *International Conference on Machine Learning*, pages 23668–23684. PMLR, 2022.
- [70] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [71] Tianshi Xu, Meng Li, and Runsheng Wang. Hequant: Marrying homomorphic encryption and quantization for communication-efficient private inference. *arXiv preprint arXiv:2401.15970*, 2024.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: We accurately describe the Abstract and Introduction.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: In Appendix 8

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: In Section 4

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Section 5

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

#### 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: Code is in the support material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Section 5

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: Ahe results are based on 5 individual runs. We report the mean and variance of the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

#### 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: Section 5

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

#### 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: Appendix 8 discusses the broader impact.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

#### 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: In Appendix 8.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: In Section 5, we cited the related code, datasets and models.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

### 13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

### 14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

### 15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

### 16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

## Appendix

### A More Experiments

#### A.1 Comparison without pre-trained weights

As shown in Table 3, even without using pre-trained weights, DictPFL achieves the highest accuracy among all methods, reaching 95.06%, compared to 94.17% for FedHE-FULL and 94.99% for FedML-HE. More importantly, DictPFL offers substantial efficiency gains: the total communication cost is reduced to 0.51 GB, while FedHE-FULL and FedML-HE require 720.72 GB and 73.62 GB, respectively. In terms of training time, DictPFL completes in just 11.8 minutes, far less than 294.6 minutes for FedHE-FULL and 56.7 minutes for FedML-HE.

Table 3: Comparison with baselines on without pre-trained weights.

|                | Acc. (%) $\uparrow$ | Comm. (GB) $\downarrow$ | Time (min) $\downarrow$ |
|----------------|---------------------|-------------------------|-------------------------|
| FedHE-FULL     | 94.17               | 720.72                  | 294.6                   |
| FedML-HE       | 94.99               | 73.62                   | 56.7                    |
| DictPFL (ours) | 95.06               | 0.51                    | 11.8                    |

#### A.2 Different reactivation probability scale $\beta$

Table 4 studies different reactivation probability scalars  $\beta$ . The result showcase the our PrME works well under different  $\beta$ .

Table 4: Ablation study on  $\beta$  under  $s\% = 70\%$  and  $\tau = 3$ .

| $\beta$ | Accuracy (%) $\uparrow$ | Comm. (GB) $\downarrow$ | Time (min) $\downarrow$ |
|---------|-------------------------|-------------------------|-------------------------|
| 0.2     | 82.29 $\pm$ 0.3         | 0.003                   | 6.36 $\pm$ 0.1          |
| 0.5     | 82.37 $\pm$ 0.3         | 0.007                   | 6.36 $\pm$ 0.1          |
| 0.8     | 82.55 $\pm$ 0.2         | 0.031                   | 6.39 $\pm$ 0.2          |

#### A.3 Different Number of Clients

We assess the performance of DictPFL in environments with varying numbers of clients. The findings, presented in Table 5, demonstrate that DictPFL performs effectively and consistently across settings with different client counts.

Table 5: The results of DictPFL under client numbers.

| Clients | Accuracy (%) $\uparrow$ | Comm. (GB) $\downarrow$ | Time (min) $\downarrow$ |
|---------|-------------------------|-------------------------|-------------------------|
| 3       | 82.67                   | 0.160                   | 6.42                    |
| 5       | 82.64                   | 0.092                   | 3.70                    |
| 10      | 81.94                   | 0.046                   | 1.85                    |
| 20      | 81.82                   | 0.041                   | 0.93                    |
| 50      | 80.42                   | 0.041                   | 0.75                    |
| 200     | 80.56                   | 0.041                   | 1.96                    |

#### A.4 Different Heterogeneous Level

Unsurprisingly, DictPFL performs better in homogeneous settings than in heterogeneous settings. As the table 6 shows, we evaluated DictPFL in various heterogeneous settings under different Dirichlet distributions from 0.3 to 0.9 and compared it with a homogeneous setting. The results indicate that DictPFL's performance remains stable across different heterogeneous dataset splits. Specifically, a smaller  $\alpha$  (more heterogeneous) requires more communication size and training time to achieve comparable accuracy to a larger  $\alpha$  (less heterogeneous).

Table 6: The results under different heterogeneous settings.

| $\alpha$ | Accuracy (%) $\uparrow$ | Comm. (GB) $\downarrow$ | Time (min) $\downarrow$ |
|----------|-------------------------|-------------------------|-------------------------|
| 0.3      | 79.62 $\pm$ 0.4         | 0.103                   | 6.22 $\pm$ 0.2          |
| 0.6      | 80.28 $\pm$ 0.2         | 0.145                   | 6.44 $\pm$ 0.1          |
| 0.9      | 82.06 $\pm$ 0.3         | 0.151                   | 6.45 $\pm$ 0.2          |
| $\infty$ | 82.67 $\pm$ 0.2         | 0.160                   | 6.42 $\pm$ 0.2          |

### A.5 Performance on NLP tasks.

Table 7 shows that DictPFL significantly improves efficiency in both sentence classification and generation (instruction tuning) tasks. For the generation task, we train on the MetaMathQA [55] dataset and evaluate on GSM8K [70], focusing on mathematical reasoning. These gains are especially pronounced in larger models, where DictPFL reduces training time by 99.4% percent for TinyLlama and 96.1% percent for BERT. This improvement stems from the high cost of ciphertext operations in larger models, making DictPFL’s optimizations more impactful.

Table 7: Comparison with baselines on TinyLlama and BERT.

|                          | Methods        | Acc. (%) $\uparrow$ | Comm. $\downarrow$ | Time $\downarrow$ |
|--------------------------|----------------|---------------------|--------------------|-------------------|
| TinyLlama-<br>MetaMathQA | FedHE-Full     | 45.86               | 30.0 TB            | 214.2 h           |
|                          | FedHE-FT       | 6.92                | 2.4 TB             | 17.9 h            |
|                          | FedML-HE       | 45.86               | 3.0 TB             | 22.6 h            |
|                          | DictPFL (ours) | 45.93               | 0.3 TB             | 1.3 h             |
| BERT-<br>AgNews          | FedHE-Full     | 91.38               | 137.2 GB           | 342.6 m           |
|                          | FedHE-FT       | 90.05               | 17.5 GB            | 47.9 m            |
|                          | FedML-HE       | 91.38               | 13.7 GB            | 32.8 m            |
|                          | DictPFL (ours) | 91.24               | 4.8 GB             | 13.4 m            |

### A.6 Comparision with Non-HE based FL

We compare DictPFL with Secure Aggregation [25] by training a ViT model on the Diabetic Retinopathy dataset under a 3-client LAN setting (1 Gbps, 0.5 ms). Secure Aggregation increases training time from 257.6 s to 363.2 s, while DictPFL achieves the same 82.7% accuracy in 385.2 s. This demonstrates that HE-based FL with DictPFL is practically efficient, with much lower overhead than commonly assumed.

### A.7 Combination with Existing Quantization Techniques

Algorithmic optimization directly reduces gradient redundancy without sacrificing accuracy, whereas quantization and packing often lead to accuracy degradation. Moreover, their improvements are limited and easily saturate [71]. More importantly, these optimization approaches are orthogonal and can be combined—by first reducing gradients through algorithmic optimization and then applying quantization or packing, overall communication cost can be further minimized. Here we perform experiments (3-client ViT on CIFAR-10, to compare DictPFL (algorithmic optimization) and AdaptiveBatchHE [41] (packing optimization), as shown in the table below. It reveals that DictPFL outperforms AdaptiveBatchHE in both efficiency and accuracy and combining them will further reduce communication overhead.

Table 8: Comparison of accuracy and communication cost among HE-based frameworks.

|                           | Accuracy (%) $\uparrow$ | Communication (GB) $\downarrow$ |
|---------------------------|-------------------------|---------------------------------|
| FedHE-FULL                | 98.2                    | 60.14                           |
| AdaptiveBatchHE           | 96.7                    | 5.41                            |
| DictPFL (ours)            | 98.2                    | 0.43                            |
| DictPFL + AdaptiveBatchHE | 96.6                    | 0.0872                          |

## A.8 Client Personality Preservation

DictPFL preserves the personality of each client based on our HRC mechanism. Specifically, while pruning is guided by the magnitude of global gradients, the HRC mechanism allows each client to upload accumulated local gradients for parameters that are reactivated, even if they were previously pruned due to low global gradient magnitudes. This ensures that important client-specific significant gradients are not lost: whenever such parameters are reactivated again, clients contribute their accumulated local gradients.

In Table 9, we evaluate DictPFL under various degrees of data heterogeneity by adjusting the Dirichlet factor  $\alpha$ . While greater heterogeneity (lower  $\alpha$ ) increases training time and communication overhead, DictPFL consistently maintains strong performance.

To further demonstrate the effect of HRC’s accumulative gradient sharing mechanism, we compare “accumulative gradient sharing” versus “non-accumulative sharing,” measuring the resulting accuracy under comparable training time. Our results (3-client ViT on Diabetic Retinopathy,  $r = 4$ ,  $s = 0.7$ ,  $\tau = 3$ ,  $\beta = 0.2$ ) show that omitting accumulated gradients notably reduces accuracy, particularly in more heterogeneous settings—because discarding small but meaningful gradients impairs learning for clients with diverse data. Overall, these results highlight that DictPFL achieves robust performance across a wide range of data distributions.

Table 9: Effect of accumulative gradient sharing under different Dirichlet factors  $\alpha$ .

| $\alpha$ | Accumulative gradient sharing (%) | Non-accumulative sharing (%) |
|----------|-----------------------------------|------------------------------|
| 0.3      | 79.62                             | 74.26                        |
| 0.6      | 80.28                             | 76.13                        |
| 0.9      | 82.06                             | 80.45                        |

## B Analysis on FedML-HE [31]

FedML-HE trades security for efficiency, and this trade-off persists regardless of whether sensitivity is dynamically recalculated. While dynamic recalculation can enhance security, it incurs substantial computational overhead to achieve an empirical 0% attack success rate. Because recalculating sensitivity scores requires each client to perform a forward pass on the training dataset and share encrypted sensitivity values for secure aggregation, it introduces overhead comparable to the original training round and HE aggregation step.

Our experiments (3-client ViT on CIFAR-10, encrypt 10%) with varying recalculation frequencies (i.e., recalculating every  $K$  rounds) show that more frequent updates do improve privacy, but at the cost of significantly reduced efficiency. Even under these settings, FedML-HE still cannot achieve the strong privacy guarantees or efficiency of DictPFL.

Table 10: Effect of dynamic sensitivity recalculation in FedML-HE.

| Method          | Accuracy (%) $\uparrow$ | Communication (GB) $\downarrow$ | Attack Success Rate (LPIPS) $\downarrow$ |
|-----------------|-------------------------|---------------------------------|------------------------------------------|
| FedHE-Full      | 98.17                   | 60.14                           | 0.00                                     |
| FedML-HE (K=1)  | 98.16                   | 61.07                           | 0.00                                     |
| FedML-HE (K=2)  | 98.16                   | 54.28                           | 0.092                                    |
| FedML-HE (K=5)  | 98.16                   | 30.8                            | 0.309                                    |
| FedML-HE (K=10) | 98.16                   | 14.2                            | 0.788                                    |
| DictPFL (ours)  | 98.15                   | 0.43                            | 0.00                                     |