Pack: Towards Communication-Efficient Homomorphic Encryption in Federated Learning

Zeyuan Zuo
University of Hong Kong
Hong Kong, China
zeyuanz@connect.hku.hk

Ningxin Su
University of Toronto
Toronto, Canada
ningxin.su@mail.utoronto.ca

Baochun Li
University of Toronto
Toronto, Canada
bli@ece.toronto.edu

Teng Zhang
University of Hong Kong
Hong Kong, China
tgzhang@hku.hk

ABSTRACT

Federated learning allows multiple clients to collaboratively train a shared model without sharing local private data. It is regarded as privacy-preserving since only model updates are communicated. Unfortunately, it has been shown in the recent literature that, model updates transmitted by participating clients can be used by a malicious server in gradient leakage attacks to obtain private training data. To prevent such potential leakage from occurring, it has widely been acknowledged that homomorphic encryption can be used to encrypt these model updates before sending them to the server, which performs computations directly on encrypted data. Although homomorphic encryption has a strong guarantee on privacy, its practical use increases communication overhead by around 17×, even with its most efficient implementation, called CKKS. In this paper, we present Pack, a novel communication-efficient mechanism over CKKS, designed specifically to reduce the communication overhead by a substantial margin. In addition, we propose new error correction and weight filtering mechanisms in Pack to improve the accuracy of the trained model. Compared to vanilla CKKS, Pack reduces the communication overhead by 3.1×, while increasing the accuracy by 5.5% and 2.5% under the i.i.d. and non-i.i.d. settings.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

Federated Learning, Homomorphic Encryption, Communication Efficiency

ACM Reference Format:
Zeyuan Zuo, Ningxin Su, Baochun Li, and Teng Zhang. 2024. Pack: Towards Communication-Efficient Homomorphic Encryption in Federated Learning. In *ACM Symposium on Cloud Computing (SoCC '24), November 20–22, 2024, Redmond, WA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3698038.3698557

SoCC '24, November 20–22, 2024, Redmond, WA, USA

ACM ISBN 979-8-4007-1286-9/24/11.
https://doi.org/10.1145/3698038.3698557

1 INTRODUCTION

With federated learning, multiple clients can collaboratively train a shared global model without sharing their privately held data [24]. Unfortunately, recent work in the literature indicates that, with a malicious server, model gradients and parameters can be used to obtain private data that should be protected [33, 37, 43, 47]. To prevent such data leakage, various privacy-preserving mechanisms have been proposed [1, 21, 44], including differential privacy, secure multi-party computation, and homomorphic encryption.

Among these mechanisms, *homomorphic encryption* is considered the most attractive due to its simplicity and strong privacy guarantees. It allows certain operations directly on ciphertext, such as addition and multiplication. If model updates are first encrypted using homomorphic encryption, the server can aggregate these updates from clients without revealing them. Compared to differential privacy, homomorphic encryption does not suffer from significant accuracy loss after the global model converges. Moreover, homomorphic encryption is more efficient and robust than secure multi-party computation when clients leave the training session, a scenario common in federated learning [10, 28].

However, one significant drawback of using homomorphic encryption is *communication overhead*, which has been largely overlooked in the literature. This drawback makes

it challenging to apply such techniques in real-world applications. Homomorphic encryption increases the size of data transferred to the server substantially. For example, in our experiments with a dominant and popular homomorphic encryption scheme, CKKS, the size of ciphertext after encryption can increase by up to 17.3×. With limited bandwidth, this overhead will significantly slow down the progress of the training process.

In conventional federated learning without homomorphic encryption, existing research on reducing communication overhead mainly focused on quantizing or pruning model weights. The key idea is to use fewer bits to represent each model parameter or simplify the computational graph in the model, potentially at the cost of a small amount of accuracy loss. Without using homomorphic encryption, plaintext is transmitted from clients to the server during each communication round, and the communication overhead can be reduced effectively [3, 5, 9, 14, 40].

With homomorphic encryption, however, existing techniques on the plaintext containing model weights will no longer be effective. Let us consider homomorphic encryption using learning with error (LWE) as an example. The plaintext space is a cyclotomic polynomial ring [23], which means that one must encode values into the plaintext space before encryption. This task is non-trivial for low-bit representations, requiring not only an embedding scheme from values to the plaintext space but also using a small number of bits for both plaintext and ciphertext while maintaining all properties of homomorphic encryption such as security and homomorphism.

In this paper, we propose Pack, a new mechanism that addresses the challenge of reducing communication overhead with homomorphic encryption from a decidedly different perspective. Rather than designing techniques to compress plaintext values, we choose to compress the ciphertext directly. Our solution is designed over CKKS, which is the most efficient and pre-dominant homomorphic encryption technique in practice. CKKS supports approximate computation on addition and multiplication [11]; it preserves the precision on approximate computation and only requires a linear size of the ciphertext with respect the depth of the circuit. We propose to reduce the ciphertext modulus, a property that is strongly related to the size of ciphertext. At a high level, our technique achieves the same tradeoff between communication overhead and the precision of computation, which is along the same lines as existing work on the plaintext without using homomorphic encryption.

On the flip side, since Pack compresses the ciphertext directly, a potential weakness is that errors can be significant when the ciphertext modulus is reduced. In the worst case, reducing the ciphertext modulus may lead to potential divergence in the model training process, which can be more severe than the accuracy loss caused by compressing the plaintext model weights. In order to mitigate such errors, as a highlight of our original contributions in this paper, Pack incorporates a new error correction mechanism to effectively reduce such errors by several orders of magnitude. In some cases, even with our error correction mechanism, our experiments show that the model training process may still fail to converge. We further propose a complementary weight filtering mechanism in Pack to limit how much model weights can change in each communication round, motivated by the underlying principle that model weights more likely to be updated with larger errors should be frozen during training.

The original contributions of this paper are as follows. We propose Pack, a new mechanism aimed at significantly reducing communication overhead when using homomorphic encryption in federated learning. By designing and implementing three core mechanisms — reducing the ciphertext modulus, error correction, and weight filtering — Pack is able to achieve equivalent (or superior) convergence behavior during model training while substantially decreasing the size of the ciphertext. We present a comprehensive set of experimental results utilizing Pack's implementation within *Plato*, an open-source federated learning research framework we developed from scratch over three years to enhance reproducibility.

In our experiments, we quantitatively evaluated communication overhead, converged model accuracy, and computational efficiency across a variety of federated learning algorithms with and without homomorphic encryption, as well as across several benchmark models and datasets. Our results show convincing evidence that Pack can reduce communication overhead by up to 3.1×. A surprising discovery is that while reducing communication overhead by such a significant margin, Pack is also able to *improve* the converged model accuracy by up to 5.5% and 2.5% under i.i.d. and non-i.i.d. settings, compared to vanilla CKKS with the same number of communication rounds. With these substantial savings in communication overhead while maintaining — or even marginally improving — the effectiveness of the training process, we believe that the use of homomorphic encryption may soon become a reality in practice for privacy-sensitive applications in healthcare and financial industries.

2 PRELIMINARIES

Unlike conventional encryption methods, homomorphic encryption permits operations to be performed on the ciphertext directly, producing results identical to those computed on the plaintext. With fully homomorphic encryption, both addition and multiplication operations are supported:

$$Decrypt(Ops(Encrypt(a), Encrypt(b))) = Ops(a, b)$$

In 2009, Gentry proposed the first fully homomorphic encryption scheme [13]. Among various homomorphic encryption schemes, fully homomorphic encryption schemes are particularly attractive because they permit *arbitrary* operations to be conducted on the ciphertext. For instance, a fully homomorphic encryption scheme can be used to implement an aggregation mechanism in the context of federated learning. However, Gentry's work concentrated on theoretical aspects and did not provide an efficient implementation. A major drawback of Gentry's approach is the large size of the public key and the considerable computational costs involved in its generation [2].

In 2017, Cheon *et al.* designed a novel fully homomorphic encryption scheme based on the *ring learning with error* (RLWE) problem, known as CKKS [11]. CKKS is recognized as a post-quantum encryption scheme, which is considered relatively secure against attacks from quantum computers. It offers an efficient mechanism for performing approximate computations on addition and multiplication. CKKS initially encodes raw complex values into a plaintext space represented by a cyclotomic ring. This encoding step includes a scaling factor to enhance precision. Subsequently, the plaintext is encrypted into the ciphertext space with a parameter Q as the modulo of the coefficients in the ring polynomial. The decryption and decoding operations reverse this process, recovering complex values from the ciphertext. Notably, due to rounding operations, the encoding and encryption steps discard the least significant bits, leading to an approximate computation. Consequently, CKKS is naturally compatible with floating-point computations, which are commonly used in real-world applications.

CKKS incorporates a new batching technique in RLWE-based constructions. By leveraging a complex canonical embedding, CKKS establishes a mapping between complex vectors and a cyclotomic ring, minimizing rounding errors. This batching technique significantly enhances the parallelism of CKKS, supporting SIMD operations [34]. Despite its computational efficiency, CKKS still experiences a substantial increase in the size of the ciphertext, to be communicated to the server in federated learning. Our experiments show that the communication cost increases by up to $\sim$ 17.3 times after encryption. Such an increase in data size poses a significant challenge in federated learning, where communication overhead is a critical bottleneck.

In addition to CKKS, previous papers have employed Paillier [27], an additive homomorphic encryption scheme, for the implementation of FedAvg. The approach multiplies the model weights and the aggregation coefficients on the client side before transmitting the encrypted weights to the server for aggregation. Paillier was selected due to its perceived efficiency compared to CKKS, as it only supports addition, whereas CKKS facilitates both the addition and the multiplication. However, we argue that it is not true for the following reasons.

First, both encryption and decryption steps in Paillier involve multiple modulo and exponentiation operations with large numbers, which are usually longer than 512 bits. These operations are extremely expensive to compute, which impairs its practical efficiency [46]. In contrast, CKKS only involves modulo and exponentiation operations that are usually less than 64 bits, which is far more efficient. What's more, the modulus and exponent of CKKS are normally required to be a power of 2, which further increases efficiency [11].

Second, Paillier supports only a limited number of independent operations, restricting the potential for parallel processing. In comparison, CKKS operations can be extensively vectorized and parallelized, because CKKS involves matrix operations, which can be efficiently implemented using techniques such as SIMD.

Third, the ciphertext size encrypted by Paillier is approximately equivalent to the key size, which is a minimum of 2048 bits for security purposes. Consequently, a single 32-bit number expands to 2048 bits upon encryption, resulting in a size increase of at least 64 times [46]. Conversely, CKKS can natively encode a batch of numbers into a single ciphertext, resulting in a size increase of only 17 times. These factors collectively demonstrate that CKKS is more efficient than Paillier.

In fact, BatchCrypt was designed to solve these problems of Paillier by using complex quantization and clipping techniques. To empirically evaluate the performance, we compared BatchCrypt with CKKS and Pack in our experiments. Results demonstrate that CKKS is more efficient than BatchCrypt, which is consistent with our analysis.

Additionally, Paillier can only be used to implement the most basic federated learning algorithm, FedAvg, but CKKS could support a wider range of federated learning algorithms which require multiplications on the server side, such as FedAdam, FedYogi and FedAU [30, 31]. It implies that CKKS can be generalized to other federated learning algorithms, while Paillier cannot.

3 PACK: DESIGN AND ALGORITHMS

Due to the large size of the ciphertext when homomorphic encryption is applied, we need to substantially reduce the size of the ciphertext to use it in practice in the context of federated learning. The design objective of Pack is to be as efficient as possible when clients communicate with the server in federated learning, with homomorphic encryption applied.

Fig. 1 describes the workflow in Pack. In each communication round, a subset of clients is selected by the server.

These selected clients receive the aggregated global model from the previous round in ciphertext, and then decrypt it to obtain the plaintext model weights. After decryption, these clients apply two new mechanisms in Pack: a new error correction mechanism to reduce errors, followed by a new weight filtering mechanism to filter out noisy weights by rejecting updates when there is a large difference between the aggregated results and the local model weights. After these mechanisms have been applied, the clients will perform several epochs of local training, encrypt the model weights with *compact encryption* module, and upload the ciphertext to the server for the next round of aggregation.

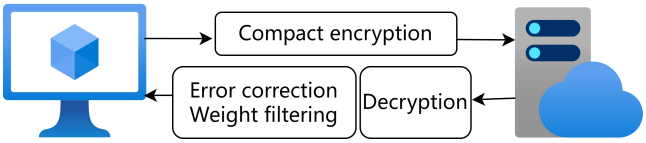

Figure 1: The workflow of Pack.

The key insight in Pack's design is to use a smaller ciphertext modulus directly, which effectively reduces the size of the ciphertext. The flip side of reducing such communication overhead, however, is that a smaller modulus introduces larger errors. These errors have negative effects on the local training process, leading to potential model divergence. In our new error correction mechanism, we train a linear regression model to reduce errors effectively, typically by several orders of magnitude. To mitigate the adverse effects of these errors and avoid potential divergence in the model training process, our new weight filtering mechanism in Pack identifies and freezes weights that are more likely to contain large errors in the aggregated results. Finally, we propose a periodic precise aggregation strategy, which ensures that Pack achieves the same convergence rate as Federated Averaging (FedAvg) with a theoretical guarantee.

3.1 Compact Encryption

The size of the ciphertext in CKKS is determined by the ciphertext modulus Q and the polynomial degree N in the ciphertext polynomial ring $Z_Q[X]/(x^N+1)$. While reducing N can indeed decrease the size of a single ciphertext, it does not effectively reduce the overall size of the ciphertexts. This is because a smaller N also reduces the number of elements that can be packed in a single ciphertext. For instance, if we halve the size of a single ciphertext, the number of required ciphertexts will double, resulting in the same overall size of the ciphertexts. Therefore, reducing N does not lead to a reduction in the total size. Instead, we achieve efficient reduction of ciphertext size by employing a smaller ciphertext modulus Q.

However, it is important to note that a smaller ciphertext modulus Q introduces additional precision loss due to the reduced scaling factor. In CKKS, the scaling factor plays a crucial role in determining the precision of computations. A larger scaling factor corresponds to higher precision. Unfortunately, choosing an arbitrarily large scaling factor is not possible due to following restrictions. First, the scaling factor must not cause overflow in floating-point numbers. For instance, a scaling factor of 2^{40} is not appropriate for a 32-bit float. Second, the scaling factor must not exceed Q, which severs as an upper bound for the scaling factor. It is essential to ensure that the ciphertext remains within the modulus to prevent failures in encryption and decryption operations. Therefore, a smaller ciphertext modulus implies a smaller scaling factor.

In addition to the precision loss introduced by the scaling factor, a smaller Q also increases the random noise, which is inserted by CKKS for security. In CKKS, a decryption structure of a plaintext polynomial m and a ciphertext c satisfies: $\langle c, sk\rangle = m + e \ (mod\ Q)$, where sk denotes a secret key and e denotes a small error term. To maintain a relatively small value for e, it is necessary to set Q to a larger value. Failure to do so would lead to a more significant negative impact on the plaintext m.

3.2 Error Correction

Based on the previous analysis, it is evident that a smaller ciphertext modulus results in larger errors. Our preliminary experiments have revealed that these errors can lead to local training divergence across various models and datasets. Therefore, it is crucial to mitigate these errors. We first quantify these errors by generating two ordered sets, S_1 and S_2, each containing 10000 randomly selected data points between -1 and 1. To simulate the average function, we use a simple equation $\{0.5 \times a_i + 0.5 \times b_i | \forall a \in S_1, \forall b \in S_2, 1 \leq i \leq 10000\}$ as the benchmark. The results show that the mean, standard deviation, and maximum of the errors are 0.1669, 0.0963, and 0.3527, respectively. During training, the error is randomly added to each value of the model weights, resulting in divergence.

If we have N sets $(S_1, S_2, \ldots, S_n)$, we can express the error as Eq. (1). $EncrAvg(\cdot)$ is an average function under the homomorphic encryption scheme and $Decr(\cdot)$ is the decryption function. For simplicity, we use $DE(\cdot)$ to represent $Decr(EncrAvg(\cdot))$

$$error = \frac{1}{N}\sum_{i=1}^{N} x_i - DE(\sum_{i=1}^{N} x_i) \tag{1}$$

The definition presented in Eq. (1) highlights that computing the error directly is impossible by a single client due to the inaccessibility of $\sum_{i=1}^{N} x_i$. Consequently, it is necessary to

approximate the error from the perspective of a single client. Because the client i can only access x_i and $DE(\sum_{i=1}^{N} x_i)$, the approximate function should only take these two values as input and output the approximate error, as shown in Eq. (2).

$$g(x_i, DE(\sum_{i=1}^{N} x_i)) \approx \frac{1}{N}\sum_{i=1}^{N} x_i - DE(\sum_{i=1}^{N} x_i) \tag{2}$$

The remaining challenge is to select a proper $g(\cdot)$. Without any prior knowledge, we choose $g(\cdot)$ to be a simple linear regression model considering that $g(\cdot)$ only has two input scalars. $g(\cdot)$ can be pre-trained by the synthetic data. We randomly generate an [M,N] matrix and compute the error with Eq. (1), which can be applied to the pre-training.

$$g(x_i, DE(\sum_{i=1}^{N} x_i)) = a \cdot x_i + b \cdot DE(\sum_{i=1}^{N} x_i) + c \tag{3}$$

To evaluate the performance of trained $g(\cdot)$, we compare the errors before and after the error correction on the test set.

Table 1: The mean, standard deviation and maximum value of corrected and uncorrected errors on the test set.

	Mean	Std Dev	Max
Uncorrected Errors	0.1109	0.0785	0.3348
Corrected Errors	0.0041	0.0042	0.0428

Table 1 presents the descriptive statistics of the errors. After applying the error correction function $g(\cdot)$, the mean, standard deviation and maximum of the errors are all reduced by orders of magnitude (Table 1). These results demonstrate that $g(\cdot)$ effectively reduces errors introduced by the compact encryption. Our further experimental results indicate that more complex models, such as MLP, do not yield better results than the simple linear regression model. This can be attributed to the fact that the input of $g(\cdot)$ is only two scalars, and the linear regression model is sufficient to approximate the error.

3.3 Weight Filtering

Despite the significant reduction in errors achieved through Eq. (3), our experiments reveal that the model still diverges during the local training stage. This is because the errors are randomly added to all elements of the weights, and the cumulative effect of these small noises can lead to model deviation from the optimal point and subsequent divergence. To mitigate this cumulative effect, we propose the weight filtering module. It filters out the weights that are more likely to contain large errors. Specifically, each client calculates the absolute difference between the aggregated weights and the local weights. Those weights with a difference above a pre-defined threshold α are prevented from updating. An appropriate α should be able to freeze the weights with large errors.

To provide an explanation of this method, we trained a LeNet-5 model on the FashionMNIST dataset with 20 clients for 200 rounds. We plot the average absolute differences between the aggregated results and local model weights in Fig. 2. The maximum difference observed occurs in the second communication round, with a value of 0.0033. The average difference and standard deviation are 0.0025 and 0.0023, respectively. As the communication round increases, the differences fluctuate in a small range $[0.0014, 0.0033]$ and stabilize at 0.0026 when the model converges. This phenomenon can be explained by the underlying FedAvg algorithm, which averages all local updates. The differences first fluctuate and then stabilize as the model converges.

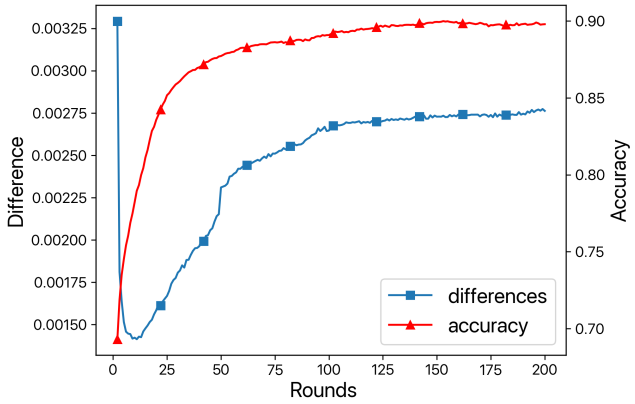

Figure 2: The average absolute differences between the aggregated results and clients' local model weights in federated learning without homomorphic encryption.

From the above explanation, we notice that the mean and standard deviation of the differences between the aggregated model and local updates are at the same magnitude as mean and standard deviation of the corrected errors in Table 1. However, the magnitude of the maximum corrected error in Table 1 is approximately ten times larger than that of the differences under FedAvg. These results indicate that the corrected errors are not negligible compared to the differences between aggregated results and the local model weights, which emphasize the importance of the weight filtering module.

One weakness of the weight filtering module is that it will discard some global information, as the clients can only update limited information received from the aggregated results due to the filtering. Consequently, the training will

bias towards the local data, leading to a decrease in accuracy, especially when the data distribution is non-i.i.d. To reduce this bias, we propose a periodic precise aggregation strategy. Specifically, every K rounds, the clients encrypt the weights with a larger modulus, which only introduces negligible errors. In this round, the clients can directly update the weights with the decrypted aggregated results without any additional processing, which can correct the bias. Theoretically, this strategy ensures that Pack achieves the same convergence rate as FedAvg.

4 THEORETICAL ANALYSIS

To summarize our work, we present Pack in detail in Algorithm 1. As Algorithm 1 shows, we follow the same client-server architecture as federated learning. In each communication round, the server returns an aggregated result and the clients produce a trained local model. Compared to FedAvg, line *11, 13, 15 — 17, 23* on the client side of Algorithm 1 are the additional modules. Line *11 and 23* represent the necessary encryption and the decryption process required by homomorphic encryption. Line 16 and 17 represent error correction and weight filtering modules. Line 13 and 15 are used to determine the size of the ciphertext modulus, because Pack uses a larger ciphertext modulus for the precise aggregation every K rounds. Due to its simplicity and good compatibility with federated learning, it is easy to be implemented as an additional plugin for different federated learning frameworks.

Algorithm 1: Pack

Server:

1 Initialize all clients C_N
2 **for** *t in 1,2,3...,T* **do**
3 Select **n** clients $C_n \subset C_N$
4 Send $\mathbf{w}_t$ to C_n
5 Received $\mathbf{w}_{t,i}$ from C_n
6 $\mathbf{w}_{t+1} \leftarrow$ `WeightedAvg` $(\{\mathbf{w}_{t,i} | i \in C_n\})$
7 **end**

Clients: i = 1,2...,N:

8 Do client initialization, define α
9 Train *LRModel* on synthetic data
10 **if** *Selected by the server* **then**
11 Receive $\mathbf{w}_t$ from the server
12 $\mathbf{w}_t \leftarrow Decrypt(\mathbf{w}_t)$
13 **if** $(t \mod K) = 0$ **then**
14 Q ← Larger-Modulus
15 **else**
16 Q ← Smaller-Modulus
17 $\mathbf{w}_t =$ `ErrorCorrection` $(\mathbf{w}_t, \mathbf{w}_{t-1}^{E+1}, LRModel)$
18 $\mathbf{w}_t =$ `WeightFiltering` $(\mathbf{w}_t, \mathbf{w}_{t-1}^{E+1}, \alpha)$
19 **end**
20 **for** *epoch τ in 0,1,2..., E-1* **do**
21 $\mathbf{w}_t^{\tau+1} \leftarrow$ train $(\mathbf{w}_t^{\tau})$
22 **end**
23 $\mathbf{w}_t^{E+1} =$ Encrypt $(\mathbf{w}_t^E, Q)$
24 Send encrypted model weights $\mathbf{w}_t^E$ to the server
25 **end**

4.1 Convergence

We provide a convergence analysis for Pack under the general non-i.i.d, and non-convex setting. We first make the following widely-used assumptions in non-convex optimization problems [9, 16, 29, 45].

Assumption 1 (L-smoothness). Each client's local loss function $f_i(\cdot)$ is L-smooth, i.e., $||\nabla f_i(x) - f_i(y)|| \leq L||x - y||$, for all $i \in [N]$.

Assumption 2 (Unbiased gradient estimator and bounded local variance). The stochastic gradient of each client is an unbiased estimator of the gradient, i.e., $\mathbb{E}[\nabla f_i(w; \xi)] = \nabla f_i(w)$. The variance of each client's stochastic gradients is bounded by σ_l^2, i.e., $\mathbb{E}||\nabla f_i(w; \xi_i) - f_i(w)||^2 \leq \sigma_l^2$, for all $i \in [N]$.

Assumption 3 (Bounded global variance). The variance of each client's gradients is bounded by σ_g^2, i.e., $\mathbb{E}||\nabla f_i(w) - f(w)||^2 \leq \sigma_g^2$, for all $i \in [N]$.

Assumption 4 (Bounded gradients). The stochastic gradient of each client is bounded by G, i.e., $\mathbb{E}||\nabla f_i(w; \xi_i)||^2 \leq G^2$, for all $i \in [N]$.

With assumptions 1-4 hold, we can prove that Pack has the same convergence rate as FedAvg under the non-convex setting, which is $O(1/\sqrt{T})$.

Theorem 1. Under Assumptions 1-4, assuming that $\eta = \sqrt{M/ET}$ and $K > 1$, for a sufficiently large T, Pack satisfies:

$$\min_{t \in [0,T-1]} \mathbb{E}||\nabla f(\mathbf{w}_t)||^2 \leq O\Big(\frac{f(\mathbf{w}_0) - f(\mathbf{w}_T)}{\sqrt{MET}}\Big) + O\Big(\frac{L^2(2K^2+1)G^2M}{T}\Big) + O\Big(\frac{\sqrt{E}L\sigma_l^2}{\sqrt{MT}}\Big) + O\Big(\frac{\sqrt{E}L\sigma_g^2}{\sqrt{MT}}\Big)$$

E is the number of local epoch; M is the number of participating clients in each round; T is the number of communication rounds; K is the interval for the precise aggregation; L, σ_l, σ_g, and G are defined in assumptions 1-4. The proof is deferred to Appendix A. Notably, the hyperparameter K only exists in the term $O(1/T)$. It implies that K does not affect the overall convergence rate since the term containing K decreases much faster than other terms containing $O(1/\sqrt{T})$.

4.2 Security

We begin our security analysis by defining an *honest-but-curious* threat adversary $\mathcal{A}$. The adversary $\mathcal{A}$ is curious on learning information about the clients' local data but it will not break the protocol. We assume that $\mathcal{A}$ can infer the clients' local data from the clients' local updates. We also assume that $\mathcal{A}$ can corrupt either the clients or the server, but it can not corrupt both at the same time, which is a common assumption used by the previous research [6, 32, 46]. We claim that Pack can protect the data privacy of the clients under the *honest-but-curious* threat model.

If $\mathcal{A}$ corrupts the server, it can not learn any information about the clients' local data because $\mathcal{A}$ can only access the encrypted local updates. The security of the encrypted data is guaranteed by the encryption method, CKKS. If $\mathcal{A}$ corrupts a set of clients C, it can access all the local data stored in C directly. For the remaining clients, they are secure. It is because $\mathcal{A}$ can not access the local data or local updates of other clients. Through the analysis, we demonstrate that Pack can protect the uncorrupted clients' local data from the adversary $\mathcal{A}$.

For the security of Pack itself, since it uses a smaller ciphertext modulus and additional error correction and weight filtering modules, these additional components may have impacts on the security of the encryption. We next analyze the influences of these modules. The security level (i.e. λ-bit security) of CKKS can be computed by Eq. (4):

$$N \geq \frac{\lambda + 110}{7.2} \log(P \cdot q_L) \tag{4}$$

It means that to achieve a λ-bit security level, the polynomial degree N should be larger than the right hand side of Eq. (4), where $P \cdot q_L$ represents the ciphertext modulus Q [11]. For a fixed polynomial degree N, the security level λ is negatively correlated with the ciphertext modulus Q. Thus a smaller ciphertext modulus can increase the security level rather than decreasing it. This conclusion is consistent with the one in homomorphic encryption standard [4]. Pack uses the same ciphertext modulus as CKKS in the precise aggregation round and a smaller ciphertext modulus in other rounds. Therefore, the security level of Pack is at least as good as CKKS.

Regarding the error correction module and the weight filtering module, these modules are executed on the client side after the decryption operations and they only perform computations on the plaintext instead of the ciphertext. Thus they do not have any impacts of the security of Pack. In addition, the pre-trained linear regression model used in the error correction module is trained on synthetic data. Our preliminary experiments indicate that the coefficients are more closely related to the number of total clients rather than the encryption and decryption keys. When the number of clients is fixed, we consistently observe similar coefficients on the different synthetic data and different keys. Consequently, we believe that these coefficients do not reveal any information about the keys. Furthermore, the linear regression models are all trained and kept locally, just as the keys, which are not transmitted to the server. Each client has its own set of coefficients, which would only be exposed in the event of an attack on the client itself.

Regarding the security of the data transmission between the clients and the server, Pack employs the HTTPS protocol to safeguard against threats such as eavesdropping and man-in-the-middle attacks.

In conclusion, the compact encryption module has the same or higher security level compared to CKKS. The error correction module and the weight filtering module do not compromise the security level of Pack as well. We can conclude that the security level of Pack is at least as good as CKKS.

4.3 Time complexity

Regarding the time complexity, the additional error correction module and the weight filtering module both have a time complexity of $O(|\mathbf{W}|)$. The error correction has a pre-trained linear regression module with two input features, the complexity is linear with respect to the size of model weights, $O(|\mathbf{W}|)$. As for the weight filtering method, it only needs to iterate the model weights once to compute the differences, which is also $O(|\mathbf{W}|)$. Adding up the time complexity of the two modules makes a total additional time complexity of $O(|\mathbf{W}|)$. Compared to local training, the computational overhead of the additional modules does not increase as the number of epochs or the size of the training set increases. To verify our analysis, we conducted experiments in Section 5.2. The results demonstrate that the wall-clock time of error correction and weight filtering modules is negligible compared to local training, encryption, and communication.

4.4 Generality

We analyze the generality of Pack through its three modules: the compact encryption module, the error correction module and the weight filtering module.

In the compact encryption module, we utilize a smaller ciphertext modulus to reduce the size of the ciphertext. This operation is clearly independent of the model, making the compact encryption module applicable to any models.

In the error correction module, we implement a linear regression model to reduce errors. This linear regression model is trained entirely on the synthetic data, which remains independent of both the local model and local datasets. Therefore, the error correction module is also generalizable to any models and datasets.

The weight filtering module freezes the updates by using the differences between the aggregated updates and the local model weights as the criteria. The Updates with larger differences indicate a higher probability of substantial errors. To filter out these updates, we defined a hyperparameter α_t. The updates with differences exceeding α_t are frozen, while those within the threshold are accepted.

Choosing α_t is critical. If it is set too high, it allows for more erroneous updates, leading to poor performance. If it is set too low, it may bias the local model towards the local data, which similarly degrades performance. We determined an optimal α_t through our preliminary experiments with the LeNet-5 model. We believe that it is generalizable to other models because the differences primarily reflect the data heterogeneity across clients rather than the model itself. Although different models have different weights, the influence of these weights will be canceled out by calculating the difference between the aggregated updates and the local model weights.

4.5 Summary

Theoretically, Pack has the same convergence rate as FedAvg in the general non-convex setting under the widely-used assumptions. Regarding the security, Pack is at least as good as CKKS since it uses a smaller or the same ciphertext modulus. The error correction and weight filtering modules have no influence on security. The security during the data transmission is guaranteed by the HTTPS protocol. Meanwhile, the computational overhead of the additional modules is $O(|\mathbf{W}|)$, which is negligible compared to the local training and encryption. In conclusion, Pack is highly compatible with federated learning. It effectively reduces the communication overhead and has only negligible additional computational overhead without sacrificing the security or convergence rate.

5 PERFORMANCE EVALUATION

In this section, we evaluated the performance of Pack, CKKS and BatchCrypt on several classic deep learning models and benchmark datasets [46]. BatchCrypt is a state-of-the-art algorithm that reduces the size of ciphertexts encrypted by Paillier scheme. BatchCrypt applies the batching technique and the gradient quantization to pack multiple gradients into a single ciphertext to reduce the size. For each batch of gradients, BatchCrypt first clipped the gradients by a threshold β to prevent them going unbounded. Then it quantized the clipped gradients and packed them into a single long ciphertext. Through its batching technique and gradient quantization, BatchCrypt reduces the size of ciphertext compared to the original Paillier scheme. We also included FedAvg to provide a baseline performance of the methods without homomorphic encryption.

The communication overhead and the accuracy are two of the most important criterion for federated learning evaluation. Therefore, we mainly focus on the comparison of these two aspects. In order to have a comprehensive comparison, we incorporate the efficiency of homomorphic encryption as an additional metric.

All experiments were implemented using the *Plato* open-source federated learning framework, which provides a real-world environment for the best possible evaluation. It uses UNIX processes to run both the clients and the server, and uses industrial-standard WebSockets to communicate. It also offers flexibility in that these processes can run on both the same physical machine, or across multiple virtual machines in the cloud.

5.1 Experimental Setup

Benchmark Datasets. In our experiments, three computer vision datasets are included: FashionMNIST, CIFAR-10, and CIFAR-100 [18, 42]. FashionMNIST contains 60,000 training images and 10,000 test images. All images are labeled with 1 of the 10 categories. Each image is in shape (28,28,1). CIFAR-10 contains 50,000 training images and 10,000 test images. All images are labeled with 1 of the 10 categories. Each image is in shape (32,32,3). CIFAR-100 contains 50,000 training images and 10,000 test images. All images are labeled with 1 of the 100 categories. Each image is in shape (32,32,3).

Data Distribution and Client Sampling. To comprehensively evaluate the performance of different methods, we conducted experiments under different data distributions and client sampling strategies. In the i.i.d. setting, the data was distributed uniformly across the clients. In the non-i.i.d. setting, the clients' local data were sampled from Dirichlet distribution with the concentration parameter equal to 1.0. As for the client sampling, we randomly sampled 20 clients from 50 clients and 40 clients from 200 clients in each round, denoted as *50-20* and *200-40*, respectively.

Model. To reveal the performance of different methods on different models, we used three classic and representative models: ResNet-18, AlexNet, LeNet-5 without any modifications in our experiments [15, 19, 20]. All models were optimized with Adam optimizer with default parameters [17]. The batch size was set to 128 for all experiments.

CKKS and Pack. The polynomial degree was set to 8192 for CKKS and Pack for a fair comparison. The ciphertext modulus was set to 200 bits for CKKS and 60 bits for Pack. The polynomial degree and the ciphertext modulus were chosen from the setting in https://github.com/microsoft/SEAL to ensure the security and high precision. The scaling factor was set to 2^{40} for CKKS and 2^{19} for Pack for a higher precision. In

the precise aggregation round, the ciphertext modulus was set to 200 bits and scaling factor was set to 2^{40} for Pack .

We set $\alpha = 0.01$ for the weight filtering module under the i.i.d. setting and $\alpha = 0.02$ under the non-i.i.d. setting. We chose the α value heuristically based on the $3 - \sigma$ rule to filter out the noisy weights. From the results in Fig. 2, we computed $\alpha = 0.0025+3\times0.0023 \approx 0.01$. For the non-i.i.d. setting, we set $\alpha = 0.02$ because we expect that the differences between the aggregated results and the local updates are larger than these under the i.i.d. setting. In addition, we use a decay rate $1/(\sqrt{\lfloor t/K \rfloor} + 1)$ for α in all conducted experiments. This is because we expect the differences between the aggregated results and the local updates to become smaller as the communication round increases.

The interval K for the periodic precise aggregations was chosen to be 20 in all conducted experiments. From our convergence analysis, we observe that a larger K results in a slower convergence rate. Intuitively, a larger K means the clients receive the precise global information less frequently, leading to a slower convergence rate. We empirically found that without this strategy, Pack can suffer from a 2% - 6% accuracy loss compared to FedAvg under the non-i.i.d setting. However, reducing K will increase the communication overhead. To find an optimal trade-off, we conducted preliminary experiments on the FashionMNIST dataset. We found that increasing K up 20 does not lead to any accuracy loss. In contrast, setting K to 40 or 50 leads to an obvious performance degradation. Therefore, we think setting K to 20 is a reasonable choice.

Hardware. Evaluating encryption, decryption, and aggregation efficiency requires consistent hardware for a fair comparison. For this experiment, we used a server with dual Intel Xeon Gold 6226R (16 Core×2) CPU with 48 GB RAM and no GPU. To compute the elapsed time of a client in one round, we used a server with a GeForce RTX 3090 GPU and an Intel Xeon Platinum 8373C CPU. For other experiments on accuracy and communication overhead, there is no specifications on the hardware.

Implementation. For a better reproducibility, both Pack and CKKS are built with the Python TenSEAL Package, which is based on Microsoft SEAL library [8, 25]. We adapted the code of BatchCrypt from https://github.com/marcoszh/BatchCrypt to track the accuracy and communication overhead.

5.2 Experimental Results

We trained LeNet-5 models on FashionMNIST and ResNet-18 models on CIFAR-10 and CIFAR-100 for 100 communication rounds under both the i.i.d. and the non-i.i.d. settings. The clients were randomly selected at each communication round and the selected clients performed local training with 1 epoch.

Ablation Study. We first conducted an ablation study to systematically assess the effectiveness of the error correction module and the weight filtering module in Pack. The results in Table 2 suggest that only when both the error correction module and the weight filtering module are employed, the model can converge. It demonstrates these two modules are necessary in Pack.

Table 2: The ablation study of the error correction module and the weight filtering module in Pack. *Failed* means the gradients exploded or the accuracy was not improved as the communication round increased.

Error correction	Weight filtering	Results
No	No	Failed
Yes	No	Failed
No	Yes	Failed
Yes	Yes	Converged

Table 3: The test set accuracy and communication overhead of different methods under the i.i.d. setting.

FashionMNIST	50-20	Size (GB)	200-40	Size (GB)
FedAvg	88.0	0.9	84.2	1.9
Pack	**89.0**	5.4	**87.3**	10.9
BatchCrypt	86.2	452.1	82.1	375.5
CKKS	88.2	16.9	82.1	33.8
CIFAR-10	50-20	Size (GB)	200-40	Size (GB)
FedAvg	82.3	166.8	65.1	333.6
Pack	**86.5**	929.0	**77.2**	1858.0
BatchCrypt	48.6	26423.3	42.1	25778.6
CKKS	82.1	2886.5	64.9	5773.0
CIFAR-100	50-20	Size (GB)	200-40	Size (GB)
FedAvg	51.0	167.5	33.4	335.0
Pack	**53.2**	933.1	**40.6**	1866.2
BatchCrypt	13.0	26604.5	9.7	28590.9
CKKS	51.3	2898.1	32.2	5796.2

Accuracy. In Table 3, the performance of Pack consistently exceeds other methods on all evaluated datasets under the i.i.d. setting. Compared to FedAvg and CKKS, Pack achieves an average increase in accuracy of 5.0% and 5.5%, respectively. It even improves the accuracy by up to 12.1% compared to FedAvg. The performances of CKKS and FedAvg are almost

the same because the errors brought by CKKS are negligible at the cost of a much larger size after encryption.

We plotted the test set accuracy during training to better understand the behaviors of different methods. In Fig. 3, Pack achieves the same convergence rate as FedAvg, which is consistent with the theoretical analysis of Pack. From the figure, we noticed that the periodic precise aggregations of Pack significantly increases the accuracy, especially on CIFAR-10 and CIFAR-100. This implies that the periodic precise aggregations can indeed reduce the bias towards the local data resulted from the weight filtering module and improve the accuracy thereby. Another interesting phenomenon is the decrease in accuracy after the first precise aggregation. This is expected because the weight filtering method restricted the updates from the aggregated results, leading to large differences across the clients' models. The aggregation of such clients' models might have a negative impact on the accuracy of this round. However, it is undeniable that the precise aggregation strategy can provide accurate global information to the clients, which is proved by the fast increase in accuracy in the following rounds and a significant increase in accuracy in the future precise aggregation rounds.

On the contrary, BatchCrypt did not show a comparable performance against FedAvg with an average decrease in the accuracy of -17.1%. BatchCrypt exhibits a slower convergence rate and a lower accuracy compared to other methods, especially on CIFAR-10 and CIFAR-100. This phenomenon can be attributed to the gradient quantization and the clipping in BatchCrypt, which are utilized to reduce the size of the ciphertext. Additional errors introduced by these operations were probably augmented regarding the more complex data and fewer participating clients. Larger errors further imply that they can have a significantly negative impact on the accuracy during the training process.

To further evaluate the performance of Pack, we conducted experiments under the non-i.i.d. setting. In this experiment, we chose FedAvg, Pack, and CKKS for comparison which are competitive on all benchmark datasets under the i.i.d. setting. Table 4 shows Pack still consistently outperforms other methods on all benchmark datasets. It achieves an average increase in accuracy of 2.3% and 2.5% compared to FedAvg and CKKS, respectively. These results demonstrate that Pack is robust to the non-i.i.d. setting without any accuracy loss. It can even achieve a marginally better performance on accuracy compared to FedAvg and CKKS.

Fig. 4 shows the accuracy plots under the non-i.i.d. setting. The general patterns are similar to the ones in the i.i.d. setting: Pack achieves the same convergence rate as FedAvg and marginally improves the accuracy. The periodic precise aggregations play an important role in improving the accuracy under the non-i.i.d. setting. From Fig. 4, the accuracy of Pack can have a decrease in accuracy from 1% to 5% without

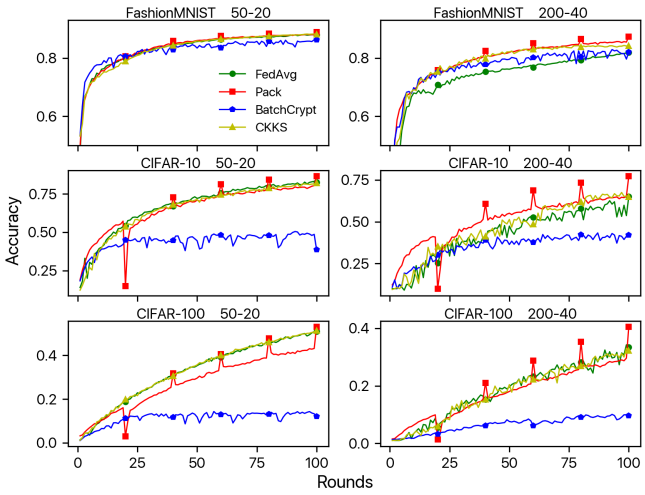

Figure 3: The plot of test set accuracy of FedAvg, Pack, BatchCrypt and CKKS under the i.i.d. setting.

Table 4: The test set accuracy and communication overhead of different methods under the non-i.i.d. setting.

FashionMNIST	50-20	Size (GB)	200-40	Size (GB)
FedAvg	84.2	0.9	77.1	1.9
Pack	**84.7**	5.4	**79.3**	10.9
CKKS	83.9	16.9	76.6	33.8
CIFAR-10	50-20	Size (GB)	200-40	Size (GB)
FedAvg	65.0	166.8	42.7	333.6
Pack	**67.8**	929.0	**50.2**	1858.0
CKKS	65.1	2886.5	43.4	5773.0
CIFAR-100	50-20	Size (GB)	200-40	Size (GB)
FedAvg	25.5	167.5	12.0	335.0
Pack	**25.8**	933.1	**12.4**	1866.2
CKKS	25.7	2898.1	10.8	5796.2

this strategy. This is consistent with our analysis that the bias towards local data resulted from the weight filtering module has a higher impact on the non-i.i.d. setting. The periodic precise aggregations can provide accurate global information to the clients, which can significantly improve the accuracy.

Communication Overhead. As for the communication overhead, FedAvg had the smallest communication overhead since it did not apply any encryption techniques. Among the evaluated homomorphic encryption-based methods, Pack achieves the smallest communication overhead. In Table 3 and Table 4, Pack achieves a reduction in communication

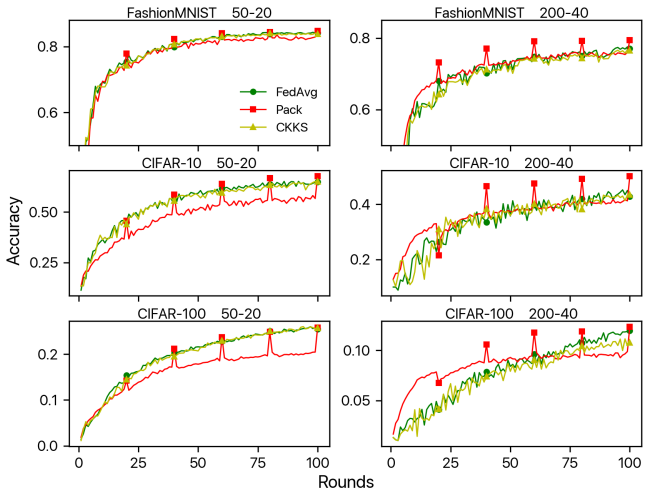

Figure 4: The plot of test set accuracy of FedAvg, Pack, and CKKS under the non-i.i.d. setting.

overhead by 3.1× and 34.1× (13.8× — 83.7×) compared to CKKS and BatchCrypt, respectively.

To better comprehend the significant performance of Pack over BatchCrypt, an analysis of the underlying principle of BatchCrypt is conducted. One major factor contributing to this behavior is that BatchCrypt communicates the model gradients between clients and the server for every batch. In contrast, FedAvg and other methods based on it communicate the model weights only once per epoch rather than every batch. This mechanism of BatchCrypt largely increases the communication overhead. Notably, the communication overhead of BatchCrypt did not increase when the number of participating clients increased from 20 to 40. This is when the number of participating clients doubled, the amount of local data reduced by a half as well, leading to roughly a half reduction of the batches per epoch. In contrast, the communication overhead of Pack and CKKS increases linearly with the number of participating clients. This is because the communication overhead of Pack and CKKS is proportional to the number of participating clients.

However, it can be argued that this comparison may not be entirely fair regarding the communication overhead, as they communicate at different frequencies. Thus, additional experiments were conducted to guarantee a fair evaluation of the communication overhead by adopting a consistent communication strategy across different methods. The communication overhead was measured by the sum of the uploading and downloading data size of all clients.

The results presented in Table 5 demonstrate the outcomes when applying the same communication strategy for CKKS, Pack, and BatchCrypt. Notably, Pack continues to outperform CKKS and BatchCrypt in this fair competition.

Table 5: The average communication overhead of 40 clients in one communication round on CKKS, Pack and BatchCrypt with the same communication strategy.

ResNet-18 (GB)	CKKS	Pack	BatchCrypt
Upload	34.02	12.63	2.90
Download	25.61	6.40	116.00
Total	59.63	19.03	118.90
Ratio	3.13	1.00	6.24
LeNet-5 (MB)	**CKKS**	**Pack**	**BatchCrypt**
Upload	204.03	75.78	31.49
Download	143.60	143.61	1529.21
Total	347.63	111.79	1290.70
Ratio	3.11	1.00	11.55
AlexNet (GB)	**CKKS**	**Pack**	**BatchCrypt**
Upload	3.81	1.33	0.33
Download	2.82	0.79	13.19
Total	6.63	2.12	13.52
Ratio	3.10	1.00	6.35

It achieves a reduction in the communication overhead by $\sim$ 3.1× compared to CKKS, and 6.2× — 11.6× compared to BatchCrypt. The $\sim$ 3.1× reduction compared to CKKS in this experiment is consistent with the previous results. The improvements of Pack compared to BatchCrypt are primarily attributed to the underlying homomorphic encryption algorithm and associated optimizations. In Table 5, it can be observed that BatchCrypt generally has a smaller encryption size compared to Pack in all experiments. However, after aggregation, the ciphertext size of BatchCrypt does not decrease, resulting in a significantly larger size compared to Pack. As the aggregated results need to be distributed to all participating clients, the communication overhead of BatchCrypt far exceeds that of Pack.

Efficiency. The computational efficiency is another important metric for the homomorphic encryption method evaluation. We used the same hardware configuration for all evaluated methods for a fair comparison. In this experiment, we measured the elapsed time for the encryption, decryption, and aggregation on CKKS, Pack, and BatchCrypt with 40 participating clients. The encryption time and decryption time were measured as the sum of the encryption and decryption time consumed by all participating clients.

From Table 6, it is evident that the computational efficiency of Pack outperforms CKKS and BatchCrypt across all settings. Although BatchCrypt is faster on aggregation, it

Table 6: The elapsed time for encryption, decryption and aggregation in CKKS, Pack, and BatchCrypt.

ResNet-18 (sec)	CKKS	Pack	BatchCrypt
Encryption	492.93	414.06	1642.96
Decryption	2.77	1.38	788.52
Aggregation	10.19	10.19	1.83
Total	505.89	425.63	2433.31
Ratio	1.18	1.00	5.72
LeNet-5 (sec)	**CKKS**	**Pack**	**BatchCrypt**
Encryption	2.862	2.336	14.177
Decryption	0.015	0.007	4.873
Aggregation	0.036	0.036	0.003
Total	2.913	2.380	19.053
Ratio	1.22	1.00	8.00
AlexNet (sec)	**CKKS**	**Pack**	**BatchCrypt**
Encryption	54.69	45.43	184.65
Decryption	0.29	0.13	94.04
Aggregation	1.13	1.13	0.13
Total	56.12	46.69	278.82
Ratio	1.20	1.00	5.97

is much slower on encryption, which is the bottleneck of the computations, thereby leading to its poor computational efficiency. A comparison between Pack and CKKS reveals that, despite no specific optimizations are made targeting computational efficiency, Pack outperforms CKKS by $\sim 1.2\times$. The better efficiency of Pack can be attributed to the utilization of a smaller ciphertext modulus. It implies fewer computations and consequently achieves involving the ciphertext. Through this experiment, we demonstrate that regarding the total time consumption, Pack is $\sim 1.2\times$ more efficient than CKKS and $\sim 6.0\times$ more efficient than BatchCrypt.

In addition to evaluating the efficiency of encryption and decryption, we measured the elapsed time of different modules on the client side. Specifically, we trained a ResNet-18 model on CIFAR-10 with 50 clients. The local epoch was set to 1 and 5, respectively.

Fig. 5 presents the elapsed time for the local training, error correction, weight filtering, encryption, and decryption modules for a single client in one round. It is evident that the encryption and local training processes are the most time-consuming components. Conversely, the error correction module and the weight filtering module together only contribute a small fraction of the elapsed time, 2.1% and 1.6% for 1 and 5 epochs, respectively. These results are consistent with our analysis of time complexity that the computational

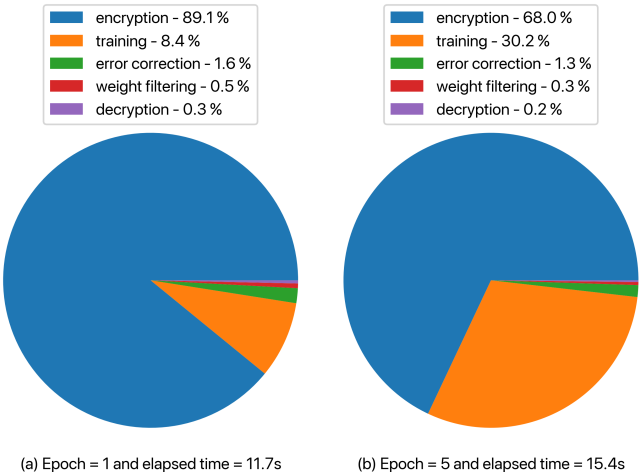

Figure 5: The breakdown of the elapsed time (second) for different modules.

overhead of the error correction and weight filtering modules is negligible.

Moreover, considering the communication overhead and even if assuming a network bandwidth of 5MB/s, which is 5 times faster than the assumption in [24], the communication overhead would dominate the time consumption in federated learning. Table 7 provides a comparison of the time consumption for a single client in one round. The communication overhead accounts for 89.3% and 86.3% of the total time for 1 and 5 epochs, respectively. These results confirm that the communication is indeed the bottleneck in federated learning.

Table 7: The consumed time (second) for communication and other operations in one round.

	Communication	Others	Total
Time (epoch=1)	97.4 (89.3%)	11.7 (10.7%)	109.1
Time (epoch=5)	97.4 (86.3%)	15.4 (13.7%)	112.8

5.3 Summary

Through solid experiments, we demonstrate that Pack can reduce the communication overhead by $3.1\times$ and $34.1\times$ compared to CKKS and BatchCrypt. In addition, since Pack utilizes a smaller ciphertext modulus, it is more computationally efficient on the homomorphic encryption operations involving the ciphertext. The experimental results demonstrate that Pack achieves a higher efficiency of the encryption, decryption and aggregation by $1.2\times$ and $6.0\times$ compared to CKKS and BatchCrypt. Pack incorporates the error correction module and the weight filtering module that are necessary to

make the training converge and their computational overhead is negligible. Moreover, Pack achieves a higher accuracy compared to the second best method CKKS by 5.5% and 2.5% under the i.i.d. and non-i.i.d. settings, respectively. Therefore, we successfully demonstrate that Pack outperforms CKKS and BatchCrypt on the communication overhead, accuracy, and computational efficiency.

6 RELATED WORK

In 2016, McMahan *et al.* formally proposed federated learning to collaboratively train a shared deep learning model across multiple devices with their private local data [24]. However, in practice, communication often becomes the major bottleneck in the pipeline due to the limited network bandwidth. The overhead is directly proportional to the number of participating clients, rounds of communication, and the size of the shared model. Existing research primarily focuses on reducing the size of the transmitted model and achieving convergence with fewer communication rounds [21, 40].

For instance, gradient compression techniques aim to reduce the number of bits used to represent individual gradient values, typically 32-bit floats. Among these techniques, SignSGD stands out as one of the most aggressive compression schemes, employing only 1 bit (the sign of the gradient) for representation [9]. In addition to gradient compression, sparsification approaches further reduce the communication overhead by setting matrix elements to zero, allowing for efficient representation in a sparse format. Other works focus on improving convergence under various conditions [26, 41].

Federated learning offers the advantage of preserving data privacy. However, the exposure of gradients and model weights has given rise to attacks targeting at extracting different types of information from the raw training data [33, 37, 43, 47]. These attacks pose a significant risk to the data. Different types of attacks can infer various information, including the reconstruction of the raw training data [22]. Although current attacks may require to know training configurations or specific prior knowledge, they are continuously evolving and should be considered a severe threat to the exposed gradients and model weights.

To prevent the risk of data leakage, several defense approaches have been proposed. Besides homomorphic encryption that has been introduced, secure multi-party computation (SMC) and differential privacy are also used to protect the privacy of the model updates. Secure multi-party computation designs a secure protocol that enables multiple parties to collectively compute a function over their inputs while keeping the inputs private. Bonawitz *et al.* introduced an novel secure aggregation protocol to compute FedAvg [10]. Subsequent research has proposed other secure aggregation protocols to improve the efficiency and reduce the communication overhead [7, 35].

While secure aggregation protocols protect the privacy of model updates with small communication overhead, they are all highly vulnerable against clients' dropout during training. Even the most robust protocols above fail to withstand the dropout of 50% of the clients. In 2023, Rathee *et al.* proposed a resilient secure aggregation protocol, known as ELSA, which can handle clients' dropout [28]. However, when facing a 50% dropout, the protocol experiences a substantial increase of over 6× in the communication overhead and more than 3× in the computational overhead. These values continue to increase as the dropout rate increases. These limitations make them hard to apply in the real-world scenarios.

Differential privacy is a privacy-preserving technique to prevent the disclosure the sensitive data [12]. In federated learning, differential privacy is achieved through the application of gradient clipping and the introduction of noise to the gradients during the training process [1, 38]. The magnitude of the added noise directly correlates with the strength of the privacy guarantee. A Larger noise implies a stronger privacy.

Despite its privacy-preserving benefits, the addition of noise to the gradients in can significantly impair a model's performance. Experiments results have demonstrated that the accuracy of a model can decrease by 5% to 20%, depending on the magnitude of the added noise [36, 38]. Moreover, the amount of noise required is positively correlated with the number of communication rounds. Consequently, as the number of communication rounds increases, more noise must be introduced to maintain the privacy [38, 39]. This indicates that a larger communication round might result in a greater degradation in performance, rather than improving it.

7 CONCLUDING REMARKS

In this paper, we introduce Pack, a novel system to reduce communication overhead in federated learning with homomorphic encryption. Pack achieves this objective by utilizing a reduced ciphertext modulus, resulting in smaller ciphertext sizes. To mitigate the precision loss from the decreased modulus, we propose two new mechanisms: error correction and weight filtering. Combined, these mechanisms significantly decrease errors and freeze updates that are likely to contain substantial errors. Additionally, we introduce the periodic precise aggregation strategy to ensure Pack converges at the same rate as FedAvg. Our experimental results show that these modules have minimal computational overhead and do not compromise the security of Pack. Compared to CKKS, Pack reduces communication overhead by 3.1× and increases average accuracy by 5.5% and 2.5% under i.i.d. and non-i.i.d. settings, respectively.

ACKNOWLEDGMENTS

This research was supported by the National Natural Science Foundation Young Scientists Fund 82303957 and the Health and Medical Research Fund (HMRF) 19200911.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 308–318. https://doi.org/10.1145/2976749.2978318

[2] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Comput. Surv.* 51, 4, Article 79 (jul 2018), 35 pages. https://doi.org/10.1145/3214303

[3] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. 2018. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc., Red Hook, NY, US.

[4] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. 2018. *Homomorphic Encryption Security Standard.* Technical Report. HomomorphicEncryption.org, Toronto, Canada.

[5] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., 300 E Ocean Blvd, Long Beach, CA 90802, United States.

[6] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE transactions on information forensics and security* 13, 5 (2017), 1333–1345.

[7] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1253–1269. https://doi.org/10.1145/3372297.3417885

[8] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. 2021. TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. arXiv:2104.03152 [cs.CR]

[9] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2018. signSGD: Compressed Optimisation for Non-Convex Problems. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm, Sweden, 560–569.

[10] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1175–1191. https://doi.org/10.1145/3133956.3133982

[11] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. 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*. Springer International Publishing, Cham, 409–437. https://doi.org/10.1007/978-3-319-70694-8_15

[12] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[13] Craig Gentry. 2009. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing* (Bethesda, MD, USA) *(STOC '09)*. Association for Computing Machinery, New York, NY, USA, 169–178. https://doi.org/10.1145/1536414.1536440

[14] P. Han, S. Wang, and K. K. Leung. 2020. Adaptive Gradient Sparsification for Efficient Federated Learning: An Online Learning Approach. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE Computer Society, Los Alamitos, CA, USA, 300–310. https://doi.org/10.1109/ICDCS47774.2020.00026

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 770–778. https://doi.org/10.1109/CVPR.2016.90

[16] Divyansh Jhunjhunwala, Pranay Sharma, Aushim Nagarkatti, and Gauri Joshi. 2022. Fedvarp: Tackling the variance due to partial client participation in federated learning. In *Uncertainty in Artificial Intelligence*. PMLR, 906–916.

[17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[18] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. *Learning multiple layers of features from tiny images*. University of Toronto.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[21] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine* 37, 3 (2020), 50–60.

[22] Lingjuan Lyu, Han Yu, Jun Zhao, and Qiang Yang. 2020. *Threats to Federated Learning*. Springer International Publishing, Cham, 3–16. https://doi.org/10.1007/978-3-030-63076-8_1

[23] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013. On Ideal Lattices and Learning with Errors over Rings. *J. ACM* 60, 6, Article 43 (nov 2013), 35 pages. https://doi.org/10.1145/2535925

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, Fort Lauderdale, FL, USA, 1273–1282.

[25] Microsoft. 2023. Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA..

[26] Hung T Nguyen, Vikash Sehwag, Seyyedali Hosseinalipour, Christopher G Brinton, Mung Chiang, and H Vincent Poor. 2020. Fast-convergent federated learning. *IEEE Journal on Selected Areas in Communications* 39, 1 (2020), 201–218.

[27] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*. Springer, 223–238.

[28] M. Rathee, C. Shen, S. Wagh, and R. Popa. 2023. ELSA: Secure Aggregation for Federated Learning with Malicious Actors. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1961–1979. https://doi.org/10.1109/SP46215.2023.10179468

[29] Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. 2020. Adaptive Federated Optimization. In *International Conference on Learning Representations*.

[30] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. 2021. Adaptive Federated Optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=LkFG3lB13U5

[31] Mingyue Ji Shiqiang Wang. 2024. A Lightweight Method for Tackling Unknown Participation Statistics in Federated Averaging. In *12th International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. https://openreview.net/forum?id=ZKEuFKfCKA

[32] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 1310–1321. https://doi.org/10.1145/2810103.2813687

[33] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 3–18. https://doi.org/10.1109/SP.2017.41

[34] N. P. Smart and F. Vercauteren. 2014. Fully Homomorphic SIMD Operations. *Des. Codes Cryptography* 71, 1 (Apr 2014), 57–81. https://doi.org/10.1007/s10623-012-9720-4

[35] Jinhyun So, Başak Güler, and A. Salman Avestimehr. 2021. Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning. *IEEE Journal on Selected Areas in Information Theory* 2, 1 (2021), 479–489. https://doi.org/10.1109/JSAIT.2021.3054610

[36] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking* (Heraklion, Greece) *(EdgeSys '20)*. Association for Computing Machinery, New York, NY, USA, 61–66. https://doi.org/10.1145/3378679.3394533

[37] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. IEEE Press, Paris, France, 2512––2520. https://doi.org/10.1109/INFOCOM.2019.8737416

[38] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security* 15 (2020), 3454–3469.

[39] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Wen Chen, Jun Wu, Meixia Tao, and H. Vincent Poor. 2023. Personalized Federated Learning With Differential Privacy and Convergence Guarantee. *IEEE Transactions on Information Forensics and Security* 18 (2023), 4488–4503. https://doi.org/10.1109/TIFS.2023.3293417

[40] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2017. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 1508–1518.

[41] Hongda Wu and Ping Wang. 2021. Fast-convergent federated learning with adaptive weighting. *IEEE Transactions on Cognitive Communications and Networking* 7, 4 (2021), 1078–1088.

[42] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:cs.LG/1708.07747 [cs.LG]

[43] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov. 2021. See through Gradients: Image Batch Recovery via GradInversion. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 16332–16341. https://doi.org/10.1109/CVPR46437.2021.01607

[44] Xuefei Yin, Yanming Zhu, and Jiankun Hu. 2021. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–36.

[45] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. 2018. Adaptive Methods for Nonconvex Optimization. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc.

[46] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. In *Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC'20)*. USENIX Association, USA, Article 33, 14 pages. https://doi.org/10.5555/3489146.3489179

[47] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., Vancouver Convention Center, Vancouver CANADA.

APPENDIX

A Convergence Proof

Notations. We define the following notations:

$$\Delta_t^i = \frac{1}{E}\sum_{j=0}^{E-1} \nabla f_i(\mathbf{w}_{t,j}^i; \xi_{t,j}^i); \quad \bar{h}_t = \frac{1}{N}\sum_{i=1}^{N} h_t^i$$

$$h_t^i = \frac{1}{E}\sum_{j=0}^{E-1} \nabla f_i(\mathbf{w}_{t,j}^i); \quad \Delta_{w,i}^t = -\eta E \Delta_t^i$$

$$w_{t+1} - w_t = \frac{1}{M}\sum_{i\in S_t} \Delta_{t,w}^i$$

We start from assumption 1,

$$\mathbb{E}[f(\mathbf{w}_{t+1}) - f(\mathbf{w}_t)] \leq \mathbb{E}[\underbrace{\langle \nabla f(\mathbf{w}_t), -\frac{E\eta}{M}\sum_{i\in S_t}\Delta_t^i\rangle}_{A}$$

$$+ \underbrace{\frac{L}{2}||\frac{E\eta}{M}\sum_{i\in S_t}\Delta_t^i||^2}_{B}]$$

By $\mathbb{E}[E/M(\sum_{i\in S_t}\Delta_t^i)] = \mathbb{E}[1/N\sum_{i=1}^N h_t^i]$ (uniform sampling clients) and $ab \le \frac{1}{2}[(a+b)^2 - a^2]$,

$$A \le \frac{E\eta}{2}\Big[\underbrace{\mathbb{E}\Big\|\nabla f(\mathbf{w}_t) - \frac{1}{N}\sum_{i=1}^N h_t^i\Big\|^2}_{A_1} - \big\|\nabla f(\mathbf{w}_t)\big\|^2\Big]$$

By applying $(a+b)^2 \le 2(a^2+b^2)$ and $(\sum_{i=1}^N a_i)^2 \le N\sum_{i=1}^N a_i^2$, we can get,

$$\begin{aligned}
B &\le LE^2\eta^2\mathbb{E}||\frac{1}{M}\sum_{i\in S_t}(\Delta_t^i - h_t^i)||^2 + L\eta^2E^2\mathbb{E}\Big\|\frac{1}{M}\sum_{i\in S_t}h_t^i\Big\|^2 \\
&\le \frac{LE^2\eta^2}{M^2}\mathbb{E}\sum_{i\in S_t}||(\Delta_t^i - h_t^i)||^2 + L\eta^2E^2\mathbb{E}\Big\|\frac{1}{M}\sum_{i\in S_t}h_t^i\Big\|^2 \\
&= \frac{LE^2\eta^2}{MN}\sum_{i=1}^N\underbrace{\mathbb{E}||(\Delta_t^i - h_t^i)||^2}_{B_1} + L\eta^2E^2\underbrace{\mathbb{E}\Big\|\frac{1}{M}\sum_{i\in S_t}h_t^i\Big\|^2}_{B_2}
\end{aligned}$$

Next, we bound term A_1 by applying L-smoothness and $(\sum_{i=1}^N a_i)^2 \le N\sum_{i=1}^N a_i^2$,

$$\begin{aligned}
A_1 &\le \frac{1}{N}\sum_{i=1}^N\mathbb{E}\Big\|\nabla f_i(\mathbf{w}_t) - h_t^i\Big\|^2 \\
&\le \frac{1}{N}\sum_{i=1}^N\mathbb{E}\Big\|\frac{1}{E}\sum_{j=0}^{E-1}(\nabla f(\mathbf{w}_t) - \nabla f_i(\mathbf{w}_{t,j}^i))\Big\|^2 \\
&\le \frac{L^2}{NE}\sum_{i=1}^N\sum_{j=0}^{E-1}\underbrace{\mathbb{E}\big\|\mathbf{w}_{t,j}^i - \mathbf{w}_t\big\|^2}_{A_2}
\end{aligned}$$

Then we bound term A_2 by assumption 2, 4 ,

$$\begin{aligned}
&\mathbb{E}\big\|\mathbf{w}_{t,j}^i - \mathbf{w}_t\big\|^2 = \\
&\mathbb{E}\Big\|w_{t_0} - \sum_{a=t_0}^{t-1}\eta\sum_{b=0}^{E-1}\nabla f_i(\mathbf{w}_{a,b}^i;\xi_{a,b}^i) - \eta\sum_{b=0}^{j-1}\nabla f_i(\mathbf{w}_{t,b}^i;\xi_{t,b}^i) \\
&- (w_{t_0} - \frac{1}{N}\sum_{a=t_0}^{t-1}\sum_{i=1}^N\eta\sum_{b=0}^{E-1}\nabla f_i(\mathbf{w}_{a,b}^i;\xi_{a,b}^i))\Big\|^2 \\
&\le 3\eta^2\mathbb{E}\Big\|\sum_{a=t_0}^{t-1}\sum_{b=0}^{E-1}\nabla f_i(\mathbf{w}_{a,b}^i;\xi_{a,b}^i)\Big\|^2 \\
&+ 3\eta^2\mathbb{E}\Big\|\sum_{b=0}^{j-1}\nabla f_i(\mathbf{w}_{t,b}^i;\xi_{t,b}^i)\Big\|^2 \\
&+ 3\eta^2\mathbb{E}\Big\|\frac{1}{N}\sum_{a=t_0}^{t-1}\sum_{i=1}^N\sum_{b=0}^{E-1}\nabla f_i(\mathbf{w}_{a,b}^i;\xi_{a,b}^i)\Big\|^2 \\
&\le 3\eta^2E^2G^2(2K^2+1)
\end{aligned}$$

Here t_0 is the last precise aggregation round. The first equation holds because $\mathbf{w}_{t,j}^i$ and $\mathbf{w}_t$ can be viewed as multiple steps of *SGD* from w_{t_0}. For the first inequality, we use $(\sum_{i=1}^N a_i)^2 \le N\sum_{i=1}^N a_i^2$ and for the last inequality, we use the fact that $t - t_0 \le K$ and $j \le E$.

Then, we can bound term A_1 and A,

$$\begin{aligned}
A_1 &\le 3\eta^2E^2L^2G^2(2K^2+1) \\
A &\le \frac{3}{2}\eta^3E^3L^2G^2(2K^2+1) - \frac{E\eta}{2}||\nabla f(\mathbf{w}_t)||^2
\end{aligned}$$

Then we start to bound term B_1 by applying $(\sum_{i=1}^N a_i)^2 \le N\sum_{i=1}^N a_i^2$,

$$\begin{aligned}
B_1 &= \mathbb{E}||\frac{1}{E}\sum_{j=0}^{E-1}(\nabla f_i(\mathbf{w}_t) - f_i(\mathbf{w}_t;\xi_t^i))||^2 \\
&\le \frac{1}{E}\sum_{j=0}^{E-1}\mathbb{E}||(\nabla f_i(\mathbf{w}_t) - f_i(\mathbf{w}_t;\xi_t^i))||^2 = \sigma_l^2
\end{aligned}$$

To bound B_2, we borrow some techniques from [16]

$$\begin{aligned}
B_2 &= \mathbb{E}||\frac{1}{M}\sum_{i\in S_t}(h_t^i - \nabla f_i(\mathbf{w}_t) + \nabla f_i(\mathbf{w}_t)) \\
&- \nabla f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)||^2 \le 3\mathbb{E}||\frac{1}{M}\sum_{i\in S_t}(h_t^i - \nabla f_i(\mathbf{w}_t)||^2 \\
&+ 3\mathbb{E}||\frac{1}{M}\sum_{i\in S_t}\nabla f_i(\mathbf{w}_t) - \nabla f(\mathbf{w}_t)||^2 + 3\mathbb{E}||f(\mathbf{w}_t)||^2 \\
&\le 3\underbrace{\frac{1}{N}\sum_{i=1}^N\mathbb{E}||h_t^i - \nabla f_i(\mathbf{w}_t)||^2}_{B_3} \\
&+ \underbrace{3\,\mathbb{E}||\frac{1}{M}\sum_{i\in S_t}\nabla f_i(\mathbf{w}_t) - \nabla f(\mathbf{w}_t)||^2 + 3\mathbb{E}||f(\mathbf{w}_t)||^2}_{B_4}
\end{aligned}$$

For B_3, we can reuse the proof to bound A_1,

$$B_3 \le 3\eta^2E^2L^2G^2(2K^2+1)$$

For B_4, we use $\mathbf{1}_t^i$ to represent $\mathbf{1}(i \in S_t))$ and δ_t^i to represent $\nabla f_i(\mathbf{w}_t) - \nabla f(\mathbf{w}_t)$. Due to the uniform client sampling

strategy, we know that $\mathbb{E}\mathbf{1}_t^i = M/N$. Then we can bound B_4,

$$B_4 = \mathbb{E}||\frac{1}{M}\sum_{i=1}^{N}\mathbf{1}_t^i\delta_t^i||^2 \le \frac{1}{M^2}\sum_{i=1}^{N}\mathbb{E}||\mathbf{1}_t^i\delta_t^i||^2$$
$$+\frac{1}{M^2}\sum_{i\neq j}^{N}\mathbb{E}\langle\mathbf{1}_t^i\delta_t^i,\mathbf{1}_t^j\delta_t^j\rangle \le \frac{1}{NM}\sum_{i=1}^{N}\mathbb{E}||\delta_t^i||^2$$
$$+\frac{M-1}{MN(N-1)}\sum_{i\neq j}^{N}\mathbb{E}\langle\delta_t^i,\delta_t^j\rangle = \frac{N-M}{NM(N-1)}\sum_{i=1}^{N}\mathbb{E}||\delta_t^i||^2$$
$$+\frac{M-1}{MN(N-1)}\sum_{i,j}^{N}\mathbb{E}\langle\delta_t^i,\delta_t^j\rangle = \frac{(N-M)\sigma_g^2}{M(N-1)}$$
$$+\frac{M-1}{MN(N-1)}\sum_{i,j}^{N}\mathbb{E}||\sum_{i=1}^{N}f_i(\mathbf{w}_t) - Nf(\mathbf{w}_t)||^2$$
$$=\frac{(N-M)\sigma_g^2}{M(N-1)} \le \frac{\sigma_g^2}{M}$$

Then, we can bound B,

$$B \le \frac{\eta^2E^2L}{M}\sigma_l^2 + 9\eta^4L^3E^3G^2(2K^2+1)$$
$$+\frac{3L\eta^2E^2\sigma_g^2}{M} + 3L\eta^2E^2\mathbb{E}||\nabla f(\mathbf{w}_t)||^2$$

Putting A and B together, we get

$$\mathbb{E}[f(\mathbf{w}_{t+1}) - f(\mathbf{w}_t)] \le$$
$$\frac{3}{2}\eta^3E^3L^2G^2(2K^2+1)(1+6\eta L)$$
$$+\frac{\eta^2E^2L\sigma_l^2}{M} + \frac{3L\eta^2E^2\sigma_g^2}{M} - \frac{E\eta}{2}(1-6\eta EL)||\nabla f(\mathbf{w}_t)||^2$$

By setting $\eta \le 1/(12EL)$, we get $1-6\eta LE \le 1/2$, then we have

$$\frac{E\eta}{4}\mathbb{E}||\nabla f(\mathbf{w}_t)||^2 \le \mathbb{E}[f(\mathbf{w}_t) - f(\mathbf{w}_{t+1})] + \frac{\eta^2E^2L\sigma_l^2}{M}$$
$$\frac{3}{2}\eta^3E^3L^2G^2(2K^2+1)(1+6\eta L) + \frac{3L\eta^2E^2\sigma_g^2}{M}$$

By applying the telescoping sum to from $t=0$ to $t=T-1$,

$$\min_{t\in[0,T-1]}\mathbb{E}||\nabla f(\mathbf{w}_t)||^2 \le \frac{4\mathbb{E}[f(\mathbf{w}_0) - f(\mathbf{w}_T)]}{\eta ET}$$
$$+6\eta^2E^2L^2G^2(2K^2+1)(1+6\eta L) + \frac{4\eta EL\sigma_l^2}{M} + \frac{12L\eta E\sigma_g^2}{M}$$

For sufficiently large T, we can set $\eta = \sqrt{M/ET}$,

$$\min_{t\in[0,T-1]}||\nabla f(\mathbf{w}_t)||^2 \le O\Big(\frac{f(\mathbf{w}_0) - f(\mathbf{w}_T)}{\sqrt{MET}}\Big)$$
$$+O\Big(\frac{L^2(2K^2+1)G^2M}{T}\Big) + O\Big(\frac{\sqrt{E}L\sigma_l^2}{\sqrt{MT}}\Big) + O\Big(\frac{\sqrt{E}L\sigma_g^2}{\sqrt{MT}}\Big)$$