Article in Press

AdaCOS: adaptive differential privacy shuffle model based on cosine similarity

Received: 09 Sep 2025

Accepted: 18 Nov 2025

Published online: 06 January 2026

Zi Ye, Guangyao Wang, Chao Ma, Ziming Guo & Hai Huang

Cite this article as: Ye, Z., Wang, G., Ma, C. *et al.* AdaCOS: adaptive differential privacy shuffle model based on cosine similarity. *J. King Saud Univ. Comput. Inf. Sci.* (2026). https://doi.org/10.1007/s44443-025-00397-5

AdaCOS: Adaptive Differential Privacy Shuffle Model Based on Cosine Similarity

Zi Ye[1]*, Guangyao Wang[1], Chao Ma[1], Ziming Guo[1], Hai Huang[1]

[1]Harbin University of Science and Technology Harbin, 150080, China.

*Corresponding author(s). E-mail(s): killparsley@163.com;
Contributing authors: ruruaeaeba@outlook.com; machao8396@163.com;
orlosziming@outlook.com; hust_hh@vip.163.com;

CRediT authorship contribution statement

Zi Ye: Writing – original draft, investigation, software, conceptualization, methodology, formal analysis

Guangyao Wang: Writing – review and editing, investigation, software, conceptualization, validation, formal analysis

Chao Ma: Resources, funding acquisition, conceptualization

Ziming Guo: Data curation, writing – review and editing

Hai Huang: Validation, project administration

AdaCOS: Adaptive Differential Privacy Shuffle Model Based on Cosine Similarity

Abstract

This paper proposes AdaCOS, a novel adaptive differentially private randomization model for privacy protection in federated learning. AdaCOS addresses the issues of accuracy loss and inadequate parameter importance assessment in traditional SDP-FL models, which stem from fixed Top-K parameter selection strategies. The proposed model introduces a dynamic Top-K adjustment mechanism based on cosine similarity, marking the first incorporation of modified cosine similarity as a smooth indicator of training stability for adaptive privacy perturbation control. Additionally, a novel network pruning technique integrating weight magnitude and Hessian values is developed to achieve more precise quantification of parameter importance. Experiments on three real-world datasets—MNIST, CIFAR-10, and CIFAR-100—demonstrate that under the same privacy budget of $\varepsilon_l = 4000$, AdaCOS improves model accuracy by an average of 3.6% (std ±0.42%, 95% CI) compared to fixed Top-K methods. In non-IID data scenarios, it reduces accuracy degradation by approximately 18%. Furthermore, to achieve the same privacy protection level, AdaCOS requires approximately 22% less Laplace noise standard deviation, significantly mitigating the interference of noise injection with model convergence. This research provides a reliable solution with high performance and strong privacy guarantees for balancing privacy protection and model efficiency in practical federated learning applications.

Keywords: Federated Learning, Differential Privacy, Shuffle Model, Cosine Similarity, Network Pruning

1 Introduction

Federated learning is a distributed machine learning paradigm that effectively tackles the data silo problem[1]. However, the model parameters exchanged during training carry risks of privacy leakage [2, 3].

To address this problem, differential privacy (DP) has been proposed to protect privacy by adding noise to model parameters [4]. Currently, there are three main types of differential privacy-based federated learning models: centralized federated learning model (DP-FL), local federated learning model (LDP-FL), and secure shuffle model (SDP-FL), as shown in Fig. 1.

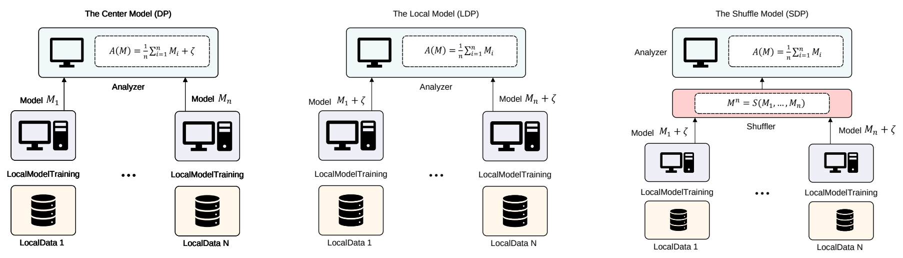

Fig. 1: Comparison of three model structures for federated learning

DP-FL streamlines privacy preservation by introducing a trusted third-party aggregator: clients upload raw weights, and the aggregator adds noise after central aggregation [5, 6]. While this model ideally balances security and efficiency, its entire privacy guarantee relies on the absolute trustworthiness and security of the centralized server. This strong trust assumption is often impractical in real-world applications. If the server is compromised or becomes untrustworthy, it becomes a single point of failure for privacy leakage.

LDP-FL enhances privacy by applying local randomization on the client side, eliminating the need to trust the server [7, 8]. However, this localized processing comes at the cost of significant utility loss. Since each local model must add strong noise sufficient to meet privacy requirements independently, the accuracy of the final aggregated global model is severely degraded.

SDP-FL introduces a shuffler into the process, leveraging the privacy amplification effect to achieve the same privacy guarantees as LDP-FL with much less noise, thereby greatly improving accuracy [9, 10, 11]. This design opens new optimization directions. However, its effectiveness heavily depends on parameter processing strategies before and after shuffling. The lack of adaptive optimization based on training dynamics limits further performance improvements in complex scenarios.

The accuracy advantage of SDP-FL mainly comes from the privacy amplification effect [12]. To optimize this model, various strategies have been proposed. Regarding parameter importance evaluation, the FLAME framework [13] uses a fixed Top-K sub-sampling strategy. However, a fixed sampling ratio cannot adapt to varying noise sensitivity across different training stages, ultimately harming model accuracy. Moreover, ranking parameters solely by their absolute values fails to accurately reflect their true contribution [13, 14].

In terms of dynamic adjustment strategies, AdaSTopK [15] attempts to dynamically adjust the Top-K ratio. Yet, its reliance on historical accuracy as a feedback signal often leads to significant training instability and affects convergence. Furthermore, most existing methods rank importance based only on weight magnitude, ignoring the parameters' influence on the loss curvature. This oversight may result in the incorrect pruning of critical parameters.

To address these issues, this paper proposes the AdaCOS model. Its core contributions are threefold:

1.AdaCOS is the first shuffler-based FL framework that employs cosine similarity for adaptive Top-K selection. It enables stable and state-aware privacy allocation by automatically adjusting the proportion of parameters uploaded from clients based on the training status.

2.This work is the first to integrate Hessian significance from network pruning into SDP-FL. This method more accurately evaluates parameter importance by considering the Hessian matrix values, going beyond just the absolute values of model parameters. It not only identifies critical parameters more precisely but also provides more effective privacy protection for them.

3. The effectiveness of the proposed strategy is validated through experiments conducted on three benchmark datasets: CIFAR-10, MNIST, and CIFAR-100. Results demonstrate that AdaCOS significantly reduces the amount of noise added while maintaining model prediction accuracy, thereby achieving an effective balance between privacy protection and model performance.

The remainder of this paper is organized as follows. Section 2 briefly summarizes related work on shuffle models. Section 3 presents the definitions of differential privacy, the privacy budget calculation, parameter importance assessment, and the basic workflow of shuffle models. Section 4 provides a detailed analysis and introduction of the AdaCOS model. Section 5 describes the experimental setup and results. Finally, Section 6 concludes the paper.

2 Related work

In recent years, differential privacy (DP) has been widely adopted in federated learning due to its mathematical rigor, aiming to optimize the privacy-utility tradeoff. The core idea of these studies is to replace fixed, worst-case noise injection strategies with intelligent privacy budget allocation based on data characteristics, model states, or training dynamics. Meanwhile, exploration of the theoretical boundaries of privacy amplification has provided a solid foundation for these adaptive mechanisms.

In the development of SDP-FL models, Bittau et al. [16] pioneered the integration of a data shuffler into the LDP-FL model, achieving significant privacy amplification through anonymization. Erlingsson et al.[17] subsequently provided a rigorous theoretical proof and quantification of the privacy amplification effect brought by shuffling. This design significantly enhances privacy protection without adding extra noise, but their analysis and evaluation were based on an idealized IID data assumption. Under highly heterogeneous data distributions, fixed noise injection and parameter selection strategies can severely amplify data bias, leading to a sharp decline in model utility.

To enhance model utility without excessively increasing overhead, research has shifted toward more efficient protocol design. Balle et al.'s "Privacy Blanket" protocol[18] and Li et al.'s virtual point-based enhancement method [19] further raised the theoretical upper bound of privacy protection by introducing virtual messages. However, a common limitation of these methods is that the privacy gains come at the cost of significantly increased computational and communication overhead for clients, which may hinder their practical application in resource-constrained federated learning scenarios.

To improve model utility without overburdening the system, research has focused on more efficient protocol designs. The works of Cheu et al. [20] and Ghazi et al.[21] demonstrated the accuracy advantages of SDP-FL from the perspectives of single-message and multi-message protocols, respectively. Such methods combine input data-dependent perturbations with data-independent noise in multi-message shuffle models. While they significantly enhance privacy protection, the algorithmic complexity also increases substantially. The multi-message protocol later proposed by Balle et al. [22] achieved privacy protection without relying on privacy amplification. Although these protocols represent theoretical advances, their complex communication frameworks (especially the multi-message protocols [21, 22]) introduce high communication costs and potential synchronization issues in practice, posing challenges for large-scale deployment.

In recent years, adaptive differential privacy mechanisms have become an important research direction for optimizing the privacy-utility tradeoff, further expanding the capabilities of SDP-FL. In adaptive noise injection, Xue et al. [23] proposed a sensitivity-based adaptive noise mechanism that adjusts noise scale by analyzing the dynamic characteristics of model updates. While effective in centralized DP frameworks, its reliance on server-side sensitivity calculation limits its application in shuffle models.

In personalized differential privacy, Liu et al. [24] introduced a personalized privacy protection framework into shuffle models, allowing users to set different local privacy budgets. Although this work enhances flexibility, it mainly optimizes horizontal privacy allocation among users and fails to address vertical adaptation at the parameter level within a single model. Similarly, regarding improvements based on gradient clipping, Xia et al. [25] proposed a dynamic gradient clipping algorithm. It adaptively adjusts the clipping threshold. This significantly improves model performance under the same privacy budget. Andrew et al. [26] introduced Adaptive Clipping. It dynamically adjusts the clipping norm C using online quantile estimation. This avoids the over-perturbation caused by fixed clipping. Its effectiveness was validated under the DP-SGD framework, with almost no communication overhead. However, its privacy analysis still uses the moments accountant method. It does not integrate the privacy amplification theorem of the shuffle model. Furthermore, the varying clipping thresholds across rounds complicate the cumulative privacy budget calculation. A dual-end collaborative implementation is also lacking. While these methods are not directly designed for the shuffle model, their idea of dynamic sensitivity adjustment provides valuable insights for improving SDP-FL.

In privacy amplification theory, Balle et al. [12] proposed the Random Check-Ins mechanism. It provides an amplification bound suitable for non-uniform participation scenarios. However, its theoretical bound performs poorly under extreme Non-IID conditions. This limits further improvement in noise efficiency. Building on this, Balle et al. [27] further proposed a multi-message shuffle protocol. It achieves privacy amplification without using the amplification lemma. It also reduces the communication rounds to a constant level. But each client needs to send O(log n) messages. This directly leads to a 2 to 3-fold increase in communication overhead and complex synchronization issues. The resulting communication inflation and synchronization delay make deployment in resource-constrained scenarios difficult.

In summary, existing research has laid a solid foundation for the privacy amplification theory and protocol design of SDP-FL. However, a gap remains in achieving a balance among "high utility, low overhead, and strong adaptability". Some approaches sacrifice efficiency for security and utility. Others fail to deeply integrate their adaptive mechanisms with the model's internal state and training dynamics. There is a lack of an end-to-end adaptive framework that can collaboratively optimize training state awareness and parameter-level importance evaluation in an environment with an untrusted server under the shuffle model.

The AdaCOS model proposed in this paper aims to fill this gap. It integrates adaptive noise control, structured sparsity, and privacy-enhancing technologies into a unified framework. By incorporating a dynamic Top-K strategy with cosine similarity and Hessian-based parameter importance evaluation, it achieves a more stable and efficient balance between privacy and utility.

3 Preliminaries

This subsection introduces the specific concepts of differential privacy techniques. In the shuffle model, differential privacy techniques are consistent with local differential privacy but benefit from the privacy amplification effect provided by the shuffle model.

3.1 Differential privacy

In 2006, Dwork introduced differential privacy [4], defining privacy as aggregate information that can be publicly disclosed while ensuring that specific individual information remains concealed. In Differentially Private Federated Learning (DP-FL), a trusted central authority collects users' raw models and adds noise to the aggregated model to achieve indistinguishability of any output. Let the dataset $D = \{d_1, \ldots, d_n\}$ consist of n data points, and let D' be an adjacent dataset. The datasets D and D' differ by the replacement of a single sensitive data point. The privacy objective is expressed as $D \simeq_r D'$. A more detailed definition is provided in Definition 1.

Definition 1. *For $\epsilon, \delta \geq 0$, a mechanism $M : \mathcal{X}^n \to \mathcal{Y}$ is (ϵ, δ)-differentially private ((ϵ, δ)-DP), on any neighboring datasets $X \simeq_r X' \in \mathcal{X}^n$, any subset $S \subseteq \mathcal{Y}$ will satisfy $Pr[M(X) \in S] \leq e^{\epsilon} Pr[M(X') \in S] + \delta$.*

Local differential privacy (LDP) fully considers the possibility of data collectors stealing or leaking users' privacy during the data collection process. In LDP, each user first privatizes their data and then sends the processed data to the data collector. The data collector performs statistical analysis on the collected data to obtain effective analytical results. In other words, while conducting statistical analysis on the data, individual privacy information is not disclosed. The formal definition of local differential privacy is provided in Definition 2.

Definition 2. *Suppose there are n users, each corresponding to a record, and given a random mechanism $R : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the definition domain and $\mathcal{Y}$ is the value domain. If the mechanism M obtains the same output $d^* \in \mathcal{X}$ on any two records*

$d, d' \in \mathcal{X}$ and satisfies the inequality: $Pr[R(d) = d^*] \leq e^{\epsilon} Pr[R(d') = d^*]$, then M satisfies ϵ-locally differentially private ($\epsilon - LDP$).

3.2 SDP-FL

SDP-FL is primarily composed of three components: the local encoder $\mathcal{R}^n$, the shuffler $\mathcal{S}$, and the analyzer $\mathcal{A}$, such that $\mathcal{P} = \mathcal{A} \circ \mathcal{S} \circ \mathcal{R}^n$. Since $\mathcal{A}$ is an untrusted third-party server, the overall privacy objective of the model can be described as the existence of $\mathcal{M} = \mathcal{S} \circ \mathcal{R}^n$ satisfying (ϵ_c, δ_c)-differential privacy (DP). In 2019, Balle et al. introduced the "privacy blanket" [18], which provides optimal privacy amplification bounds for single-message protocols. Let $[b]$ denote the target domain of the encoder $\mathcal{R}$, and let γ represent the probability of outputting elements from the blanket distribution. Based on Lemma 1, the privacy bounds (ϵ_c, δ_c) for $\mathcal{M}$ can be derived.

Lemma 1. *For $\frac{\sqrt{14\log(2/\delta_c)(b-1)}}{n-1} < \epsilon_c \leq 1$, if $\mathcal{R}_{\gamma,b}$ satisfies ϵ_l-LDP, we have (ϵ_c, δ_c) for $S \circ \mathcal{R}^n$, where $\epsilon_c = \frac{14\log(2/\delta_c)(e^{\epsilon_l}+b-1)}{n-1}$.*

In 2014, Dwork proposed that differential privacy possesses compositional properties [28], as demonstrated in Lemma 2, Lemma 3, and Lemma 4.

Lemma 2. *Assuming that $A_1(\cdot)$ is an algorithm that satisfies the (ϵ, δ)-differentially private ((ϵ, δ)-DP) mechanism. After combining algorithm $A_2(\cdot)$ and algorithm $A_1(\cdot)$ into algorithm $A = A_2(A_1(\cdot))$, the algorithm A will also satisfy the (ϵ, δ)-DP mechanism.*

Lemma 3. *An ϵ-DP mechanism will satisfy $k\epsilon$-DP under k-fold adaptive composition.*

Lemma 4. *For all $\epsilon, \delta, \delta' > 0$, under k-fold adaptive composition, the group (ϵ, δ)-DP mechanism will satisfy the $(\epsilon', k\delta + \delta')$-DP mechanism, where $\epsilon' = \sqrt{2k \ln\left(\frac{1}{\delta'}\right)}\,\epsilon + k\epsilon(e^{\epsilon} - 1)$.*

In 2023, Liu et al. [13] were the first to propose the fixed sub-sampling privacy amplification technique, which further reduces the noise added to local model parameters under the same privacy budget by selecting and uploading the top K most important model parameters. Let the client's model weight parameters be W, and the selection ratio during sub-sampling be tkr. According to Lemma 3, the local single-dimensional privacy budget is given by $\epsilon_{lt} = \frac{\epsilon_l}{\sum_j^n W_j tkr}$. Based on Lemma 1 and Lemma 4, Theorem 1 is used to calculate the specific privacy budget bounds within the entire TopK shuffle model.

Theorem 1. *With $\gamma = \frac{b}{e^{\epsilon_{lt}}+b-1}$ and $\delta_{cd} < 2tkr$, $\mathcal{M} = \mathcal{S} \circ \mathcal{R}_{\gamma,b}$ satisfies $(\epsilon_{cd}, \delta_{cd})$-DP, where:*

$$\epsilon_{ck} = \sqrt{\frac{14\log\left(\frac{2tkr}{\delta_{cd}}\right)(e^{\epsilon_{lt}} + b - 1)}{|W| - 1}}, \tag{1}$$

$$\epsilon_{cd} = \log(1 + tkr(e^{\epsilon_{ck}} - 1)). \tag{2}$$

The SDP-FL framework demonstrates a clear defensive effect against inference attacks. The attacker's goal is to determine whether a specific data record exists in the model's training set. Acting as a malicious server, the attacker utilizes the generated global model and queries specific data points. By observing the model's outputs, the attacker attempts to infer whether a data point belongs to the training set. Differential privacy inherently defends against membership inference attacks. Under the condition that the SDP-FL framework satisfies (ϵ_c, δ_c)-DP, the features of any specific data record from local clients are obscured in the global model. The experimental results in Section 5 further confirm this protective effect.

3.3 Network pruning

Network pruning is a method used to reduce the size of large neural networks [29], as specifically defined in Definition 3. In this paper, the importance evaluation of the model parameters for sub-sampling employs the same strategy as the network pruning method, thereby identifying the top K most significant model parameters.

Definition 3. *Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, and a desired sparsity level tkr, neural network pruning can be written as the following constrained optimization problem:*

$$\min_{\mathbf{W}} L(\mathbf{W}; \mathcal{D}) = \min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{W}; (x_i, y_i)) \tag{3}$$

$$s.t. \quad \mathbf{W} \in \mathbb{R}^m, \quad \|\mathbf{W}\|_0 \leq tkr.$$

In the definition, l represents the standard loss function, W denotes the set of neural network parameters, m is the total number of parameters, and $\|\cdot\|_0$ is the standard L_0 norm. An effective method to optimize the aforementioned problem is based on saliency, which approaches the problem by selectively removing redundant parameters from the neural network. Popular criteria for determining parameter importance include the magnitude of the weights and the Hessian values of the loss with respect to the weights. Parameters with larger weights or higher Hessian values are considered more important.

The Hessian matrix of the model parameters comprises all second-order partial derivatives of the objective function with respect to the parameters, i.e., the Hessian values. The computation method is illustrated in Definition 4.

Definition 4. *For a function $f : \mathbb{R}^n \to \mathbb{R}$, the Hessian matrix H is calculated as follows, where $x_1, x_2, \ldots, x_n$ are the input variables:*

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{4}$$

The aforementioned saliency-based methods constitute the primary approach of many network pruning strategies, such as the ODS and OBS pruning strategies that rely on Hessian values [30, 31]. Existing SDP-FL sub-sampling strategies [13, 15] select parameters based on the magnitude of their weights using a greedy approach. This strategy assumes that smaller parameter weights have a lesser impact on model performance. However, research [32] indicates that this assumption is not entirely accurate, as the absolute magnitude does not fully represent the importance of a model parameter. Compared to directly using model parameters, the Hessian values reflect the local curvature of the loss function, thereby helping to identify which parameters have a greater impact on model performance.

In 2017, Molchanov et al. proposed a network pruning strategy based on the Taylor expansion [33]. This strategy uses the absolute value of the first-order term of the Taylor expansion of the objective function with respect to the activation functions as the evaluation criterion. Compared to OBD and OBS, this paper employs a first-order derivative estimate while ignoring higher-order terms. In 2018, Lee et al. [34] used the absolute value of the normalized first derivative of the objective function with respect to the parameters as a measure of importance.

Building on existing research, this paper proposes a novel strategy that applies network pruning techniques to evaluate the importance of model parameters. This approach optimizes the selection process of the Top-K parameters. Specifically, this paper uses a combination of Hessian values and the absolute values of model parameters as a comprehensive metric for evaluating parameter importance.

Experimental results demonstrate that, compared to traditional Top-K selection methods based solely on the absolute values of model parameters, the proposed strategy can more accurately identify parameters that have a greater impact on model performance. Compared to traditional SDP-FL, this strategy increases the model prediction accuracy by an average of 3.6% on three real-world datasets under the same privacy budget conditions, effectively balancing privacy protection and model utility.

4 AdaCOS Framework

This section primarily introduces the AdaCOS model. Fig. 2 illustrates the model framework, encompassing the local encoder $\mathcal{R}^n$, the shuffler $\mathcal{S}$, and the analyzer $\mathcal{A}$.

4.1 AdaCOS Framework

Fig. 2 illustrates the AdaCOS model, which consists of three core components: the local client, shuffler, and analyzer. This structure corresponds to the three elements of the ESA framework. The design incorporates a parallel branch mechanism that adaptively selects the optimal path by comparing training outcomes under different strategies, thereby enhancing overall robustness. This concept draws on the core idea in ensemble learning of improving model performance through diverse selection and fusion [35, 36]. Here, the two branches represent different perspectives on the parameter space. Through dynamic selection, the model aims to strengthen its resistance to noise interference, thereby balancing privacy and utility more intelligently.

The operational workflow of the framework is as follows. The local client obtains the initial weight parameters W_0, the Top-K ratio tkr, and the branch strategy for the current round from the server. After training the model using its local dataset, the client employs a randomization procedure to select the indexes of the top-K most important weights according to the specified branch strategy. Noise is then added to the selected parameters.

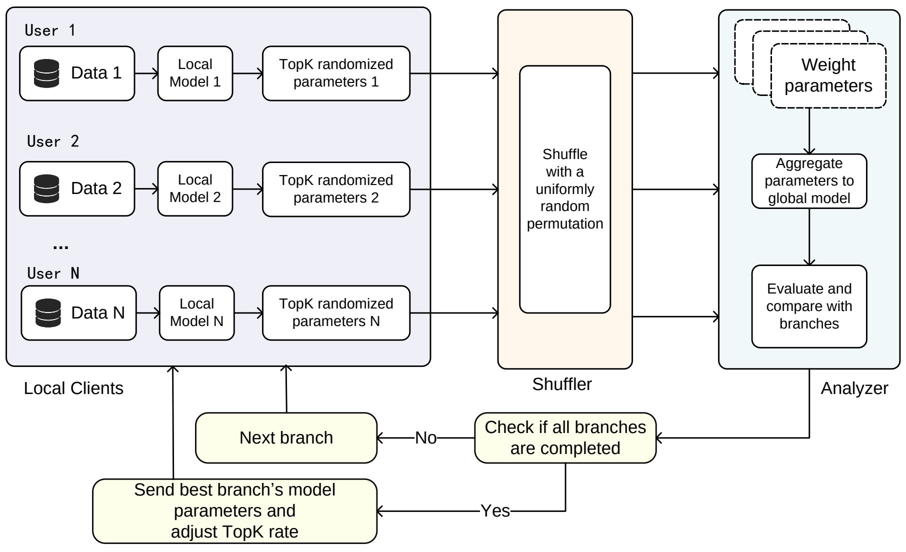
User 1
Data 1
Local Model 1
TopK randomized parameters 1
User 2
Data 2
Local Model 2
TopK randomized parameters 2
...
User N
Data N
Local Model N
TopK randomized parameters N
Local Clients
Shuffle with a uniformly random permutation
Shuffler
Weight parameters
Aggregate parameters to global model
Evaluate and compare with branches
Analyzer
Check if all branches are completed
No
Next branch
Yes
Send best branch's model parameters and adjust TopK rate

Fig. 2: AdaCOS: adaptive differential privacy shuffle model based on cosine similarity

The shuffler receives the noise-added weight parameters from the clients and assigns a fixed ID to each client's weight parameters. Subsequently, the shuffler randomly and uniformly rearranges the data to break any associations between the data points. Once the shuffling operation is complete, the weight parameters are sent to the analyzer.

The analyzer receives the weight parameters from the shuffler and performs aggregation and analysis on these data. According to Lemma 2, the analyzer satisfies differential privacy without adding any noise. The main functions of the analyzer include:

1. Model Aggregation: Aggregates the weight parameters from all clients to generate the global model parameters.

2. Performance Evaluation: Evaluates the accuracy of the current branch model using the evaluation function $Evaluate$ and selects the model parameters with the highest performance as the average weight parameters W_{t+1} for the next iteration.

3. Top-K Ratio Update: Calls the cosine similarity evaluation function $CosineTopkStrategy$ to assess the training progress based on the model's parameters and dynamically updates the Top-K ratio tkr.

Algorithm 1 outlines the detailed workflow of the AdaCOS framework. Initially, the analyzer $\mathcal{A}$ initializes the model weights W_0, which are then disseminated to the participating clients for training. During each global training iteration, $\mathcal{A}$ randomly selects a subset of clients c^t from the entire client pool N based on the client selection rate cr to participate in the t-th epoch of training. Following the selection, $\mathcal{A}$ broadcasts the weight parameters W_0 and the Top-K ratio tkr to the chosen clients.

Algorithm 1 AdaCOS.

```
1:  Input: global epoch $T$; local epoch $E$; $N$ is the number of users, $B$ is the local batch size, $\epsilon_l$ is the local differential privacy budget, $lr$ is the learning rate, $cr$ is the client participation rate, $tkr$ is the top-k rate.
2:  Output: $W^T$
3:  Analyzer: initialize weights $W^0$
4:  for each global epoch $t = 1, \dots, T$ do
5:      for each branch $br = 1, 2$ do
6:          $c^t \leftarrow RandomSelectClients(N, cr)$
7:          $\epsilon_{lt} \leftarrow \frac{\epsilon_l}{\sum_{j=1}^{n} W_j tkr}$
8:          for all local client $c_i^t \in c^t$ in parallel do
9:              $W_i^{t,br} \leftarrow W^{t-1}$
10:             for each local epoch $e = 1, \dots, E$ do
11:                 for each batch $b \in B$ do do
12:                     $W_i^{t,br} \leftarrow W_i^{t,br} - lr\nabla L(W_i^t; b)$
13:                 end for
14:             end for
15:             $W_i^{t,br} \leftarrow Clip(W_i^{t,br}, 0, C)$
16:             $W_i^{t,br,*} \leftarrow RandomTopkData(W_i^{t,br}, tkr, \epsilon_{lt}, br)$
17:             Send $W_i^{t,br,*}$ to Shuffler
18:         end for
19:         Shuffler:ShuffleParameter($W_{i \in c^t}^{t,br}$)
20:         Send $W_{i \in c_n^t}^{t,br}$ to Analyzer
21:     end for
22:     Analyzer:
23:     for each branch $br = 1, 2$ do
24:         aggregate model $W^{t+1,br} \leftarrow \frac{1}{c^t} \sum_i^{c_n^t} W_i^{t,br}$
25:         $Acc^{t,br} \leftarrow Evaluate(W^{t,br})$
26:     end for
27:     $W^{t,best} \leftarrow (Acc^{t,1} > Acc^{t,2})?W^{t,1} : W^{t,2}$
28:     $Acc^t \leftarrow (Acc^{t,1} > Acc^{t,2})?Acc^{t,1} : Acc^{t,2}$
29:     update model $W^t \leftarrow W^{t,best}$
30:     update $tkr \leftarrow CosineTopkStrategy(W, tkr, \epsilon_l)$
31: end for
32: return $W^T$
```

During the parallel training phase on the clients, each client ${c^t}_i \in c^t$ involved in the t-th global iteration and the br-th branch training epoch utilizes the Stochastic Gradient Descent (SGD) optimizer to update the model parameters W_{t-1} using their local dataset. To mitigate the risk of gradient explosion caused by excessively large gradients, the *Clip* function is employed to perform gradient clipping based on the gradient norm threshold C.

Subsequently, the local clients invoke the *RandomTopkData* function to add noise to the subsampled model parameters according to the specified Top-K ratio *tkr* for the current epoch. The resulting perturbed model parameters W are then transmitted to the shuffler $\mathcal{S}$ for the shuffling process. The shuffler executes the $ShuffleParameter$ operation, which randomly and uniformly rearranges the model data to eliminate any potential associations between data points. After shuffling, the randomized model parameters are forwarded to the analyzer $\mathcal{A}$ for aggregation and evaluation.

Within the analyzer, the aggregated model undergoes evaluation using the $Evaluate$ function to determine the accuracy of the current branch model. Based on this evaluation, the model parameters exhibiting higher accuracy across two consecutive branch epochs are selected as the average weight parameters W_{t+1} for the subsequent training iteration. Finally, the analyzer employs the cosine similarity evaluation function $CosineTopkStrategy$ to assess the training progress by evaluating the model's parameters and dynamically updates the Top-K ratio tkr. The updated weight parameters W_t and the Top-K ratio tkr are then broadcasted to the clients at the commencement of the next training epoch. This iterative process continues until all training epochs are completed.

4.2 Adaptive top-k perturbation

Existing Top-K mechanisms [13, 15] primarily rely on a greedy strategy that selects only model parameters with the largest absolute magnitudes. However, this reliance on a single metric is regarded as a heuristic rule lacking theoretical guidance, which may fail to accurately assess parameter importance—particularly during the dynamic process of model convergence.

Recent studies in neural network pruning have clearly demonstrated that parameter significance should not be determined solely by absolute values, but should be comprehensively evaluated by considering both first-order (gradient) and second-order (Hessian) influences on the loss function [37, 38].

Inspired by these findings, this paper introduces a novel mechanism that integrates network pruning principles into Top-K selection. By simultaneously considering both the absolute values and Hessian information of model parameters, the proposed approach enables more precise quantification of parameter importance.

Since computing the full Hessian matrix is typically computationally intensive, this paper approximates the Hessian matrix by using only its diagonal elements. The importance of a model parameter W_j is defined as $M_j = \frac{W_j^2 H_{jj}}{2}$, where $H_{jj} = \frac{\partial^2 l}{\partial W_j^2}$ and l is the loss function.

From Definition 4, it can be seen that The Lipschitz continuity of H implies that small perturbations in W lead to bounded changes in H. The Laplace noise has mean zero and variance $2\lambda^2$, so the bias in $\tilde{H}_{jj}$ is proportional to λ. The importance score M_j depends on W_j^2 and H_{jj}, both of which are perturbed by $O(\lambda)$. Thus, the relative order of top parameters is preserved when noise is small. Computationally, using only the diagonal Hessian reduces cost from $O(m^2)$ to $O(m)$, where m is the number of parameters. Therefore, Theorem 2 holds.

Theorem 2. *Let $\tilde{W} = W + Z$ be the noisy weight parameters with Laplace noise $Z \sim Lap(0, \lambda)$, where $\lambda = \frac{\Delta f}{\epsilon}$. Assume the loss function $L(W)$ is twice continuously differentiable, and the Hessian matrix $H(W)$ is γ-Lipschitz continuous. Then, the* *diagonal Hessian estimate $\tilde{H}_{jj}$ computed from $\tilde{W}$ satisfies:*

$$|\tilde{H}_{jj} - H_{jj}| \leq \gamma\lambda + O(\lambda^2). \tag{5}$$

Moreover, the parameter importance score $M_j = \frac{W_j^2 H_{jj}}{2}$ has an estimation error of $O(\lambda)$, and for sufficiently small λ, the ranking of parameters by importance remains unchanged. Therefore, the effect of differential privacy noise on the Hessian estimation is considered negligible.

Algorithm 2 details the operation of the *RandomizeTopkData* function. When the i-th client completes training for the t-th epoch and generates the model parameters W to be sent, the function first calculates the number of model layers and traverses them layer by layer. For different training branches br, one branch traverses each layer and computes the diagonal elements of the Hessian matrix H_i using the *GetHessian* function, subsequently calculating the parameter importance M_i. The other branch directly uses W_i as M_i.

Algorithm 2 RandomizeTopkData.

```
1:  Input: $W, tkr, \epsilon_{lt}, br$
2:  Output: $W^*$
3:  $N \leftarrow layers(W)$
4:  for each layers $i = 1, \ldots, N$ do
5:      for each index $idx \in W_i$ do
6:          $H_i[idx] \leftarrow GetHessian(W_i[idx])$
7:          $M_i[idx] \leftarrow ||H_i[idx]|| * ||W_i[idx]^2||$
8:      end for
9:      if $br = 1$ then
10:         $S \leftarrow Top(W_i, tkr)$
11:     else
12:         $S \leftarrow Top(M_i, tkr)$
13:     end if
14:     for each index $idx \in W_i$ do
15:         if $idx \in S$ then
16:             $W_i^*[idx] \leftarrow AddNoise(W_i[idx], laplace(\epsilon_{lt}))$
17:         end if
18:     end for
19: end for
20: return $W^*$
```

Subsequently, M_i is used to select the top K most important model parameters, which are then marked as S. For the parameters in S, the *AddNoise* function is invoked to add Laplacian noise equivalent to a privacy budget of ϵ_{lt}, resulting in the processed model weights W_i^*. The noise-added model parameters are then sent to the shuffler. The Hessian values involved in the model importance evaluation reflect the local curvature of the loss function, effectively identifying which parameters have a greater impact on model performance. This allows for the addition of differential privacy noise to protect these critical parameters. By combining with the cosine similarity strategy, the Top-K strategy can better select the most important parameters in each epoch through the dynamically changing tkr values.

Algorithm 1 and Algorithm 2 provide substantial support for defending against poisoning attacks from malicious clients. The Top-K selection mechanism in Algorithm 2 filters out a large majority $(1 - tkr)$ of parameters. If a malicious client attempts to poison all dimensions, our parameter importance evaluation identifies the parameters that are truly critical to model performance. As a result, malicious updates that do not align with the parameter importance assessment will be ineffective in disrupting the server's model aggregation from client models.

4.3 Cosine top-k strategy

In the early stages of training, model parameters exhibit significant fluctuations. At this point, adding more differential privacy noise does not significantly affect the model's accuracy [14]. As training progresses, the loss function gradually converges, and model parameters stabilize. Consequently, the amount of added noise should be progressively reduced. Since the Top-K ratio (tkr) directly determines the noise added when clients upload their models, its dynamic adjustment must follow this trend.

The key to implementing a dynamic Top-K strategy lies in identifying an evaluation metric that robustly and smoothly reflects the training progress. Compared to fixed-magnitude subsampling strategies, a dynamic Top-K approach driven by stable signals achieves higher model accuracy under the same privacy budget.

Instead of relying on highly volatile loss values or delayed accuracy feedback, we adopt the cosine similarity between model parameters. This metric directly captures the evolution of the internal model state and demonstrates excellent smoothness and stability throughout the training process. This approach aligns closely with the concept in Gaussian process regression of utilizing smoothly varying covariance to guide decision-making [39, 40]. Thus, cosine similarity serves as a more reliable feedback signal, ensuring robust dynamic adjustment and enabling higher accuracy than fixed strategies under identical privacy constraints.

Implementing a dynamic Top-K strategy requires a stable evaluation metric to monitor the progress of model training. Cosine similarity is used to measure the similarity between two non-zero vectors in an inner product space. It is equal to the cosine of the angle between the two vectors, which is the dot product of the vectors divided by the product of their magnitudes. In this paper, the two vectors used to calculate cosine similarity are obtained by flattening two neural network models into one-dimensional vectors. Experiments show that the cosine similarity between the global models W^t of the current epoch and W^{t-1} of the previous epoch exhibits greater stability under the given experimental conditions compared to the loss function and prediction accuracy. Therefore, cosine similarity is used as the primary reference attribute for evaluating the degree of model training in this paper. By combining cosine similarity with network pruning strategies, the entire AdaCOS framework can intelligently adjust the parameters selected in the Top-K step, further enhancing the model's privacy protection level under the same privacy budget.

Algorithm 3 CosineTopkStrategy.

```
1: Input: accuracy list $Acc$;loss list $Loss$; Model parameters $W$; Cosine list similarity $C$; global epoch $T$; current epoch $t$; number of users $N$ and old top-k rate $tkr$.
2: Output: updated $tkr$
3: $C^t \leftarrow \frac{dot(W^t, W^{t-1})}{||W^t|| \times ||W||}$
4: if $Loss_{t-1} < Loss_t$ then
5:     $score\_loss \leftarrow 1$
6: else
7:     $score\_loss \leftarrow 0$
8: end if
9: $score\_acc \leftarrow 1$
10: for $i = 1$ to $N$ do
11:     if $t - N - 1 + i < 0$ then
12:         $temp_{acc} \leftarrow temp_{acc}$
13:     else
14:         $temp_{acc} \leftarrow temp_{acc} + Acc_{t-N-1+i}$
15:     end if
16: end for
17: if $temp_{acc} \geq Acc_t$ then
18:     $score\_acc \leftarrow 1$
19: else
20:     $score\_acc \leftarrow 0$
21: end if
22: if $t/T \geq 1/2$ then
23:     $score\_t \leftarrow 1$
24: else
25:     $score\_t \leftarrow 2 \times t/T$
26: end if
27: $score\_total \leftarrow (score\_loss + score\_acc + score\_t)/3$
28: if $score\_total < 0.5$ then
29:     $d \leftarrow |1 - (\frac{C_t - C_{t-1}}{C_{t-1} - C_0} \times 0.1)|$
30:     $tkr \leftarrow tkr \times d$
31: end if
32: return $tkr$
```

Algorithm 3 provides the detailed mechanism of the pseudocode for calculating Top-K using cosine similarity and model training parameters. When the function *CosineTopkStrategy* is invoked, it retrieves all training information of the model with the highest accuracy from the previous epoch, including historical accuracy Acc, historical loss function values $Loss$, historical cosine similarity C, current model parameters W, total training epochs T, current training epoch t, number of participating clients N, and the Top-K ratio tkr from the previous epoch. At the beginning of the evaluation, the analyzer $\mathcal{A}$ calculates the cosine similarity between the current epoch's model

W^t and the previous epoch's model W^{t-1}, and stores the cosine similarity in the historical cosine similarity list. Subsequently, the analyzer comprehensively evaluates the training progress based on the loss function value, accuracy, and training epochs.

The loss function represents the difference between the model's predicted values and the actual values. During training, it should gradually decrease and eventually stabilize within a small fluctuation range. This algorithm first checks whether the loss function of the current epoch is lower than that of the previous epoch. If so, it sets the $Score\_loss$ of W^t and the previous epoch's model to 1. Model accuracy should also show an increasing trend during training and eventually stabilize at a relatively steady state. Due to accuracy fluctuations, the accuracy of adjacent epochs may vary significantly. Therefore, this algorithm calculates the average accuracy for comparative evaluation by summing and storing the accuracies of the recent $t-N-1+i$ epochs. If the sum is greater than or equal to the current epoch's accuracy Acc_t, it sets $Score\_acc$ to 1. This method smooths short-term fluctuations and provides a more robust performance evaluation. Generally, as the number of training epochs increases, accuracy tends to improve, and the loss function value decreases. Therefore, the algorithm includes an evaluation of the training epochs. When t/T is greater than $1/2$, it sets $Score\_t$ to 1; otherwise, it sets $Score\_t$ to $2t/T$.

By averaging the scores corresponding to the three factors, if the average score is greater than 0.5, the Top-K ratio tkr is reassigned using the coefficient $|1-(\frac{C_t-C_{t-1}}{C_{t-1}-C_0}\times 0.1)|$. During training, the difference in cosine similarity generated in each training epoch becomes increasingly smaller, allowing tkr to contract rapidly in the early stages of training and gradually stabilize in the later stages.

4.4 Privacy budget under cosine similarity

It can be easily seen from Definition 3 that the loss function $L(W)$ is β-smooth and μ-strongly convex. Lemma 5 and Lemma 6 are given by the standard properties of optimization theory.

Lemma 5. *Let L be a β-smooth function. Then for any points W^t and W^{t-1}, $\|\nabla L(W^t)-\nabla L(W^{t-1})\| \le \beta\|W^t-W^{t-1}\|$.*

Lemma 6. *Let L be a μ-strongly convex function near the optimum W^*. Then for any points W^t and W^{t-1}, $\langle W^t-W^*, W^{t-1}-W^*\rangle \ge \left(1-\frac{\beta}{\mu}\right)\|W^t-W^*\|\|W^{t-1}-W^*\|$.*

Based on Lemma 5, Lemma 6, and the definition of cosine similarity provided in Algorithm 3, Theorem 3 can be deduced:

Theorem 3. *For a loss function L that is β-smooth and μ-strongly convex, the cosine similarity C_t satisfies*

$$|1-C_t| \le \frac{\beta}{\mu}\cdot\frac{\|W^t-W^{t-1}\|}{\max(\|W^t\|,\|W^{t-1}\|)} \tag{6}$$

Furthermore, when $|1-C_t|\le\eta$, where $\eta>0$ is a small positive constant, then the gradient change $\|\nabla L(W^t)-\nabla L(W^{t-1})\|$ is bounded by $O(\eta)$, indicating stable training progress.

For any fixed Top-K ratio tkr, Theorem 1 establishes that the shuffle mechanism satisfies (ϵ_t,δ_{cd})-DP per epoch. The cosine similarity-driven adjustment ensures $tkr_t \ge$

$tkr_{min} > 0$ for all t, since $C_t \in [-1, 1]$ and the adjustment factor is bounded, preventing privacy budget divergence. Applying Lemma 4 across T training epochs, Corollary 1 can be deduced:

Corollary 1. *Let $\epsilon_{\max} = \max_{t=1}^{T} \epsilon_t$ and $\delta = T\delta_c + \delta'$. Then the following bound holds:*

$$\epsilon \leq \sqrt{2T \ln(1/\delta')}\, \epsilon_{\max} + T\epsilon_{\max} \left(e^{\epsilon_{\max}} - 1\right). \tag{7}$$

The adjustment rule $tkr_t = tkr_{t-1} \cdot \left[1 - \left(\frac{C_t - C_{t-1}}{C_{t-1} - C_0} \times \alpha\right)\right]$ ensures gradual reduction of privacy cost as training stabilizes, with smaller ΔC_t values leading to decreased tkr_t and consequently lower overall privacy expenditure while maintaining model utility. Consider AdaCOS framework with dynamic Top-K ratio tkr_t adjusted based on cosine similarity C_t. Let each client satisfy ϵ_l-LDP locally. Combining Corollary 1, Theorem 4 can be derived:

Theorem 4. *After T training epochs, the overall mechanism satisfies (ϵ, δ)-DP with:*

$$\epsilon = \min \left\{ \sqrt{2T \ln(1/\delta')} \cdot \epsilon_{max} + T\epsilon_{max}(e^{\epsilon_{max}} - 1), \sum_{t=1}^{T} \epsilon_t \right\} \tag{8}$$

$$\delta = T\delta_c + \delta' \tag{9}$$

where $\epsilon_t = \log\left(1 + tkr_t \cdot (e^{\epsilon_{ck_t}} - 1)\right)$, and ϵ_{ck_t} is the per-epoch privacy budget after shuffling:

$$\epsilon_{ck_t} = \sqrt{\frac{14 \log\left(\frac{2tkr_t}{\delta_{cd}}\right) \left(e^{\epsilon_{lt}} + b - 1\right)}{|W| - 1}} \tag{10}$$

The dynamic adjustment follows: $tkr_t = tkr_{t-1} \cdot \left[1 - \left(\frac{C_t - C_{t-1}}{C_{t-1} - C_0} \times \alpha\right)\right]$, with $\alpha = 0.1$.The ϵ obtained in Theorem 4 represents the privacy budget consumption of the cosine similarity-driven dynamic Top-K strategy under the AdaCOS framework.

5 Experiments

This study conducts experiments on three well-recognized image datasets: MNIST, CIFAR-10, and CIFAR-100.

Each dataset is partitioned into three configurations: Independent and Identically Distributed (IID), non-Independent and Identically Distributed 1 (non-IID-1), and non-Independent and Identically Distributed 2 (non-IID-2). In the non-IID-1 setting, a class-preferential approximate Dirichlet distribution with a concentration parameter of $\alpha = 0.3$ is employed, accompanied by dynamic allocation to ensure that each client receives data samples from all classes. For the non-IID-2 setting, an unbalanced Dirichlet distribution with $\alpha = 2$ is used to evaluate the effectiveness of the proposed strategy under strong-non-IID conditions. The dataset partitioning procedure follows the Dirichlet-based approach proposed by Hsu et al.[41]

Further details on the data distributions are provided in Appendix Figures 9, 10, and 11.

Table 1: Model structure of MNIST and CIFAR-10

Model	conv1	conv2	pool1/2	linear1	linear2	linear3
MNIST	10*5	20*5	2*2	320*256	256*50	50*10
Cifar-10	6*5	16*5	2*2	400*200	200*84	84*10

Table 2: Model structure of modified ResNet on CIFAR-100

Model	conv1	layer1	layer2	layer3	layer4	avgpool/fc
Cifar-100	3*3,64	2×(3*3,16)	2*(3*3,32)	2(3*3,64)	2*(3*3,128)	128*100

~~Each dataset is divided into three configurations. These configurations are Independent and Identically Distributed (IID), non-IID-1, and non-IID-2. In non-IID-1 and non-IID-2, a Dirichlet distribution with $\alpha = 0.3$ is used. For non-IID-2, the data distribution is unbalanced. In other words, different clients have very different amounts of data. More details about the data distributions are provided in Fig. 9, Fig. 10, and Fig. 11 in the Appendix.~~

Under these three distribution setups, this paper use a smaller model for MNIST and CIFAR-10. The MNIST model has $W = 100,816$ parameters. The CIFAR-10 model has $W = 100,806$ parameters. Detailed model parameters for these datasets are listed in Table 1. To evaluate AdaCOS on larger models and more complex tasks, a small ResNet model is used for CIFAR-100. This model has $W = 320,436$ parameters. Detailed parameters are provided in Table 2.

A comprehensive set of experiments was conducted to evaluate the performance of the AdaCOS model under realistic conditions. As a baseline, the SDP-FedAvg model [1] was employed for comparison. To ensure effective training results, all experiments in this study were performed using a Geforce RTX4090 GPU.

5.1 Training epoch analysis

This subsection presents the variation in model accuracy throughout the entire training process, as illustrated in Fig. 3. In this experiment, the local privacy budget ϵ_l is set to 4000, the client selection rate cr is set to 0.8, the number of global training epochs T for CIFAR-10 and MNIST is set to 15 and for CIFAR-100 is set to 50. The number of local training epochs E is set to 10. The total number of clients N for CIFAR-10 and MNIST is set to 100, and for CIFAR-100 is set to 50. The learning rate lr is set to 0.005, and the initial Top-K ratio tkr for AdaCOS is set to 1. To incorporate as much uniform noise as possible into the local encoder, the tkr for SDP-FedAvg is fixed at 0.9. Detailed metrics variation during training are presented in Table 3.

Tests on three datasets show that AdaCOS achieves higher accuracy than SDP-FedAvg under the same differential privacy budget. On the non-IID-1 and non-IID-2 conditions, AdaCOS experiences a smaller accuracy drop compared to the IID dataset than SDP-FedAvg does. In contrast, non-IID-2 exhibited significant accuracy degradation across all three datasets, with the CIFAR-100 dataset showing the most pronounced decline. Comparatively, AdaCOS achieved an earlier convergence point under non-IID-2 conditions and demonstrated a slight accuracy advantage. This demonstrates that AdaCOS has a greater advantage in handling non-IID conditions. Notably, on the CIFAR10 and CIFAR-100 datasets, AdaCOS shows larger fluctuations in accuracy per epoch. This is attributed to the continuously decreasing tkr values that affect the noise levels. On the CIFAR-100 dataset, the non-IID-2 configuration may lead to a more noticeable drop in accuracy due to the larger imbalance in the data distribution.

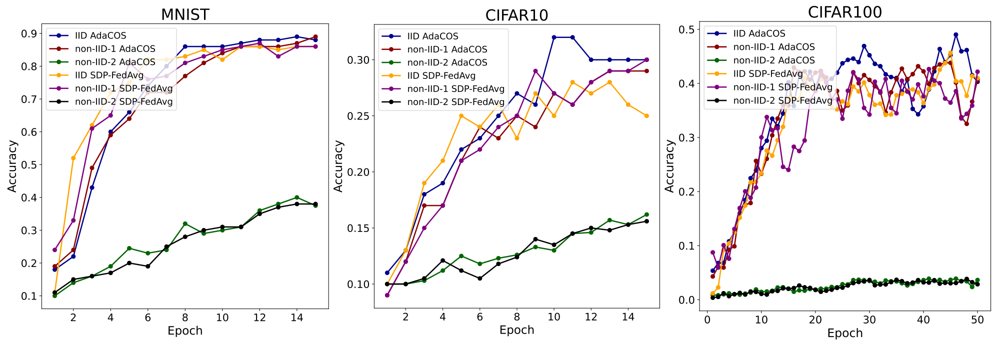

Fig. 3: Model training accuracy under different data distributions on different datasets

When the model converges, the number of global communication epoch reflects the overall communication efficiency. Fewer epochs mean lower overall communication costs. Our experiments show that AdaCOS converges in 15 epochs for MNIST and CIFAR-10 under IID and non-IID-1 conditions. For CIFAR-100 under IID and non-IID-1, it takes 50 rounds.

Under IID and non-IID-1, data is evenly distributed among clients. Each client communicates the same amount with the central server.

Under the non-IID-2 condition, the uneven data distribution results in the absence of certain classes on some clients. Consequently, both FedAvg and AdaCOS exhibit reduced accuracy. Furthermore, the imbalance in data distribution degrades communication efficiency, thereby requiring a greater number of rounds for the model to achieve convergence. ~~In the non-IID-2 condition, the uneven data distribution reduces communication efficiency. This causes more epochs to converge and lower accuracy in the same number of epochs.~~

The Fig.4. below shows the accuracy changes in 15 epochs (50 epochs for CIFAR-100) under different client ratios. When more clients participate in training, the server receives more gradients from clients. This increases the upload communication burden. Also, when more clients take part, broadcasting the global model to all clients increases the download communication load.

It can be seen that in all datasets, the model performs best when the client ratio is around 0.5 to 0.8. With a low client ratio, communication cost is reduced. However, the training progress slows down and more rounds are needed to converge. With a high client ratio, communication costs increase and training efficiency drops.

Table 3: Changes in metrics during training

Dataset	Data Dist	Metric	SDP-FL				AdaCOS			
			R1	R5	R10	R15	R1	R5	R10	R15
MNIST	IID	Accuracy	0.12	0.75	0.88	0.91	0.11	0.68	0.87	0.92
		Loss	2.59	1.20	0.09	0.03	2.68	1.15	0.10	0.02
		tkr	0.90	0.90	0.90	0.90	1.00	0.91	0.86	0.80
		cos	0.68	0.91	0.95	0.97	0.63	0.93	0.97	0.98
	non-IID-1	Accuracy	0.11	0.68	0.83	0.91	0.12	0.57	0.86	0.91
		Loss	2.66	1.24	0.09	0.02	2.57	1.23	0.06	0.02
		tkr	0.90	0.90	0.90	0.90	1.00	0.92	0.85	0.82
		cos	0.69	0.91	0.94	0.97	0.63	0.92	0.97	0.98
	non-IID-2	Accuracy	0.10	0.15	0.26	0.34	0.10	0.20	0.24	0.35
		Loss	2.66	2.18	1.88	1.57	2.60	2.20	1.92	1.53
		tkr	0.90	0.90	0.90	0.90	1.00	0.90	0.83	0.76
		cos	0.66	0.91	0.95	0.97	0.60	0.92	0.96	0.97
CIFAR-10	IID	Accuracy	0.09	0.21	0.25	0.26	0.10	0.22	0.32	0.30
		Loss	2.30	2.20	1.99	2.07	2.30	2.15	2.06	1.92
		tkr	0.90	0.90	0.90	0.90	1.00	0.91	0.82	0.77
		cos	0.68	0.91	0.95	0.97	0.63	0.93	0.97	0.98
	non-IID-1	Accuracy	0.10	0.18	0.23	0.32	0.09	0.19	0.28	0.29
		Loss	2.30	2.17	1.96	1.90	2.29	2.22	2.02	2.13
		tkr	0.90	0.90	0.90	0.90	1.00	0.92	0.83	0.79
		cos	0.69	0.91	0.94	0.97	0.63	0.92	0.97	0.98
	non-IID-2	Accuracy	0.10	0.11	0.13	0.15	0.10	0.12	0.13	0.16
		Loss	2.31	2.26	2.23	2.15	2.30	2.20	2.18	2.08
		tkr	0.90	0.90	0.90	0.90	1.00	0.90	0.83	0.76
		cos	0.66	0.91	0.95	0.97	0.60	0.92	0.96	0.97
CIFAR-100	IID	Accuracy	0.01	0.11	0.28	0.31	0.05	0.11	0.24	0.32
		Loss	3.36	2.6	2.33	1.73	3.01	2.45	2.26	1.63
		tkr	0.90	0.90	0.90	0.90	1.00	0.98	0.93	0.85
		cos	0.32	0.71	0.95	0.97	0.43	0.63	0.93	0.97
	non-IID-1	Accuracy	0.09	0.14	0.30	0.24	0.03	0.10	0.26	0.36
		Loss	2.78	2.17	1.86	1.90	2.99	2.78	2.03	1.56
		tkr	0.90	0.90	0.90	0.90	1.00	0.98	0.93	0.85
		cos	0.56	0.89	0.94	0.97	0.45	0.88	0.95	0.96
	non-IID-2	Accuracy	0.01	0.01	0.02	0.03	0.01	0.01	0.02	0.03
		Loss	3.00	2.97	2.96	1.93	3.00	2.98	2.96	2.91
		tkr	0.90	0.90	0.90	0.90	1.00	0.97	0.75	0.85
		cos	0.48	0.87	0.95	0.95	0.33	0.92	0.94	0.96

To prove the effectiveness of the cosine similarity-based strategy, this paper introduced AdaSTopK for comparison[15]. AdaSTopK also uses a dynamic TopK strategy. All hyperparameters are kept consistent with those of AdaStopK, as shown in the Fig. 5.

Results indicate that AdaCOS and AdaSTopK achieve comparable accuracy on both datasets. On the MNIST dataset in the IID setting, AdaCOS has slightly lower average accuracy than AdaSTopK. In the non-IID setting, their performances differ significantly.

The difference is likely due to the data distribution in the experiments. AdaSTopK's training strategy largely depends on the previous round's performance. As a result, it shows a wider range of fluctuations than AdaCOS. AdaCOS is more stable. On the

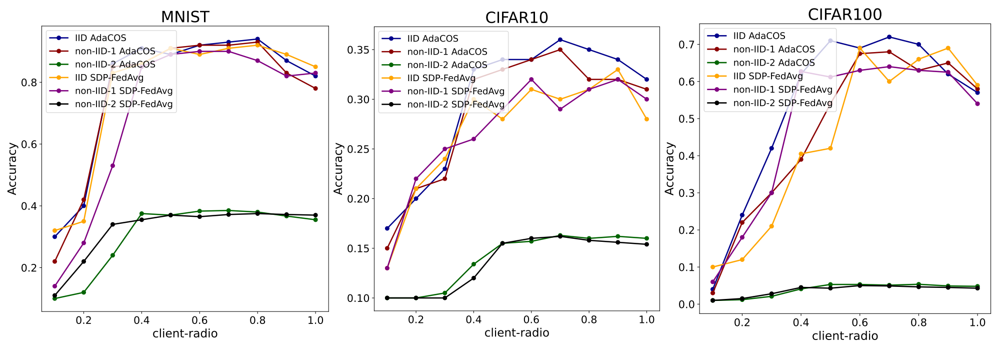

Fig. 4: The impact of changes in client radio on model performance

CIFAR10 dataset, both methods exhibit training fluctuations. However, AdaCOS has a smaller fluctuation amplitude. This further proves the effectiveness of the cosine similarity-based strategy.

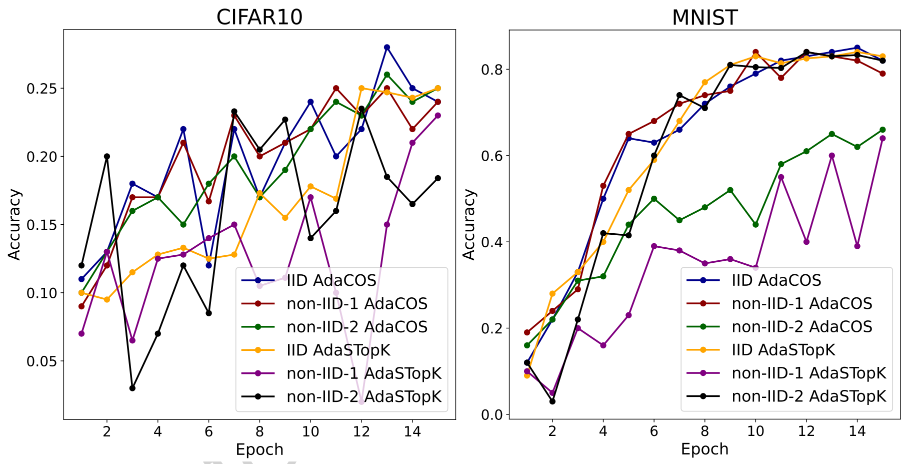

Fig. 5: Model training accuracy compared with AdaStopK

5.2 Privacy budget analysis

To investigate the impact of the local privacy budget ϵ_l on the overall performance of the model, Following the setup from subsection 5.1, the same parameter settings were employed, while varying the independent variable ϵ_l within the range [2000, 7000]. The results are presented in Fig. 6.

As observed, with an increase in the privacy budget ϵ_l, the performance of the AdaCOS model positively correlates across all three data distributions: IID, non-IID-1, and non-IID-2.

On the MNIST dataset, when $\epsilon_l > 3000$, the model's accuracy gradually stabilizes. Both IID and non-IID-1 distributions experience a rapid increase in accuracy, whereas the accuracy for non-IID-2 grows at a slower rate. This slower growth in non-IID-2 is attributed to the imbalanced distribution of sample sizes. When $\epsilon_l < 3000$, the model's performance is influenced by the noise added; a substantial amount of noise causes the model parameters to lose their inherent features. In complex tasks, neural networks may have high dimensionality, which can lead to biases in the evaluation of parameter importance. AdaCOS leverages both Hessian values and model parameters to assess parameter importance, thereby better preserving the characteristics of the model parameters. When $\epsilon_l > 6000$, the accuracy improvements of both AdaCOS and SDP-FL start to plateau, with AdaCOS achieving slightly higher accuracy than SDP-FL under the IID distribution.

On the CIFAR10 dataset, AdaCOS exhibits performance patterns similar to those observed on the MNIST dataset. Under different privacy budgets, the accuracy under the non-IID-2 distribution remains consistently lower, which can be attributed to the significant performance gap caused by the uneven data distribution during training. ~~However, the inherent complexity of the CIFAR10 dataset and the imbalance in its data distribution result in more significant fluctuations in accuracy for the non-IID-2 distribution across different privacy budgets.~~ Additionally, the performance gap between the SDP-FL model and the AdaCOS model becomes more pronounced under higher differential privacy budgets. Overall, the AdaCOS model consistently achieves superior performance across various privacy budgets and data distributions.

On the CIFAR100 dataset, both AdaCOS and SDL-FL converge smoothly. However, the overall graph shows a large accuracy gap as ϵ_l varies. This suggests that in deep models and complex tasks, the model is more sensitive to the addition of global noise. The performance in both non-IID-1 and non-IID-2 cases is generally lower than that in the IID scenario.Under the non-IID-2 scenario, both frameworks fail to obtain effective training information. This is primarily due to the higher task complexity of CIFAR-100, which imposes stricter requirements on data balance for effective model learning.

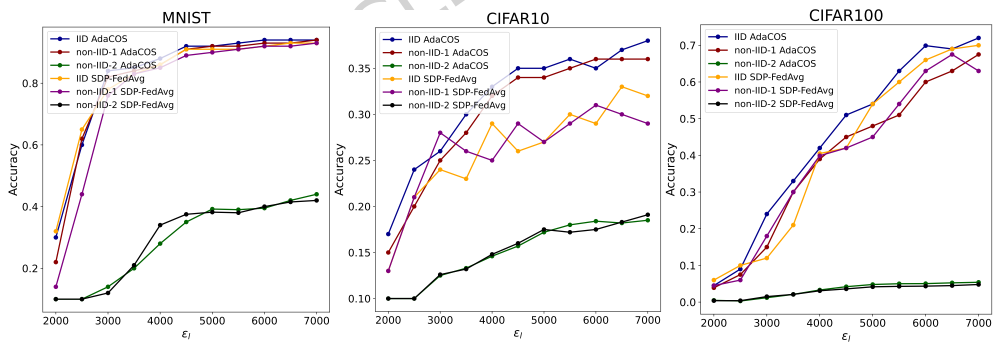

Fig. 6: The impact of changes in privacy budget on model performance

5.3 Top-K rate analysis

To further validate the superiority of Algorithm 3 introduced in Section 4, this study additionally records the dynamic adjustments of the Top-K parameter (tkr) by AdaCOS across all training epochs. The results are illustrated in Fig. 7. All experimental parameters in this subsection remain consistent with those outlined in the first subsection of this section.

As observed, in the IID distribution of the MNIST dataset, the convergence degree of the tkr parameter is minimal. As the degree of data heterogeneity increases, the reduction in tkr becomes more pronounced. This behavior results from the dynamic adjustment in response to the rate of accuracy improvement.

In the CIFAR10 dataset, the convergence of tkr is more evident. This is attributed to the CIFAR10 dataset being more challenging to train compared to MNIST. Additionally, due to significant fluctuations in accuracy during the training process of CIFAR10, it is unable to generate a sufficiently long window for accuracy improvement. Consequently, the amount of noise added needs to be appropriately reduced to maintain a stable training process and enhance the final model's usability. In the last training batch of the non-IID-2 distribution, tkr reaches its lowest value, which is due to the high degree of data heterogeneity in non-IID-2.

In the CIFAR100 dataset, the performance of tkr is roughly similar to that on CIFAR10. However, due to a longer training period, tkr converges at a relatively early stage and reaches a value close to 0.7. Compared with simpler models, ResNet experiences more accuracy loss in the later stages of training due to noise perturbation, so it is adjusted to a lower level.

As a control, Fig. 7 also includes the tkr values of FixTopK for comparison. When the global training batch $T < 7$, AdaCOS adds more differential privacy noise to the model than the fixed Top-K approach. When $T > 8$, the added noise gradually decreases, and at this point, the model accuracy of AdaCOS surpasses that of FixTopK. This demonstrates the effectiveness of the dynamic Top-K adjustment strategy discussed in Section 4. It effectively enhances the final model's usability while ensuring adherence to the privacy budget.

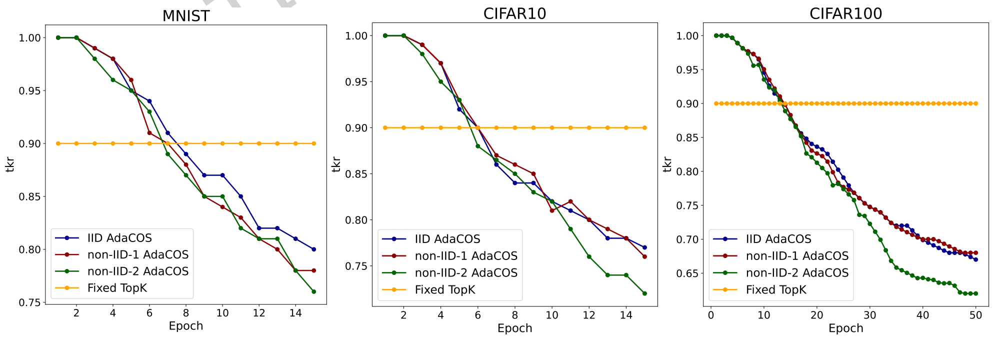

Fig. 7: Changes in tkr during training

During training, the amount of noise added to each parameter is fixed, which means that the total noise amount is directly related to tkr. Based on the conclusion from Section 3, each selected model parameter is perturbed with Laplace noise having a scale of $\frac{\epsilon_l tkr}{\sum_j^n W_j}$ and a mean of 0. The overall noise scale added in each round cannot be obtained simply by summing the noise for each parameter; however, the total privacy budget consumed by all the noise remains equal to ϵ_l. In the Fig. 7, the scale of the noise added over all training rounds is equivalent to that produced by a privacy budget of 4000.

The main computations during the training process involve local model training and the addition of noise during communication. The time consumed by the noise addition is directly proportional to tkr. The Fig. 8 presents the training time for each global round during the model training process. The fewer the parameters that have noise added, the less time each round of training will take.

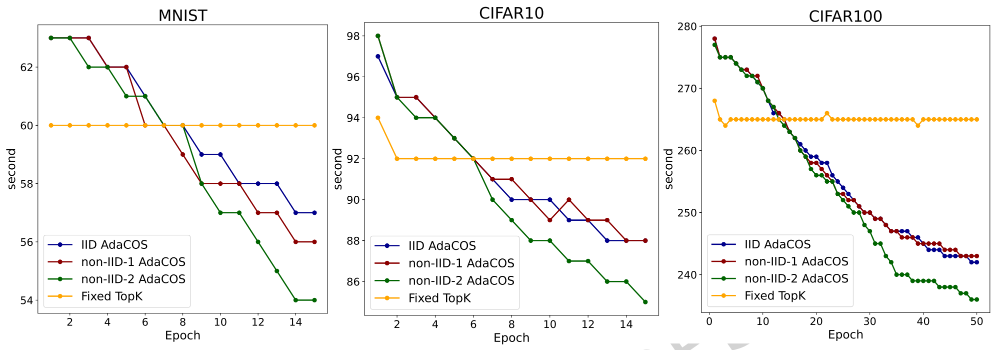

Fig. 8: Global epoch training time during training

6 Conclusion

This paper proposes a novel adaptive differentially private stochastic model, AdaCOS. It effectively addresses the inherent trade-off between privacy protection and model accuracy in traditional SDP-FL frameworks. This is achieved through an innovative dynamic Top-K adjustment strategy and a precise parameter importance quantification method. Furthermore, AdaCOS integrates a new network pruning technique. This technique enables accurate parameter importance quantification by considering both weight magnitudes and their Hessian values.

Experiments on three datasets demonstrate that AdaCOS outperforms traditional SDP-FL methods. It achieves an average improvement of 3.6% in model accuracy under the same privacy budget. The model also shows significant robustness in Non-IID data scenarios. This effectively achieves a superior balance between privacy protection and model utility.

Although AdaCOS demonstrates superior performance, this study has several limitations. These point to directions for future work: The experimental validation was primarily conducted on relatively small-scale image datasets (MNIST, CIFAR-10, CIFAR-100). The generalization capability of AdaCOS to larger-scale datasets (e.g., ImageNet) or different data modalities (e.g., text, graph-structured data) remains to be verified. Future work will validate AdaCOS's effectiveness in fine-tuning scenarios for large pre-trained models. We also plan to extend it to more complex domains like federated graph neural networks.Regarding data distribution, this paper only investigates two non-IID scenarios (non-IID-1 and non-IID-2), without exploring the model performance under extreme non-IID conditions, which represents a direction for our future extensions.

Additionally, while the Hessian diagonal approximation avoids substantial computational overhead, its introduced extra cost warrants attention for extremely resource-constrained edge devices. A promising future direction is to explore lighter-weight proxy metrics for parameter importance. Alternatively, developing asynchronous execution strategies to offload importance evaluation from the critical training path could be beneficial.

Furthermore, we will explore integrating personalized differential privacy with our model's internal adaptive mechanisms to suit various application scenarios.

Statements and Declarations

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Data availability

The authors do not have permission to share data.

Acknowledgement

This work was supported by the National Natural Science Foundation, China (No. 62172123) and the Key Research and Development Program of Heilongjiang (Grant No. 2022ZX01A36).

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[3] Hanna M Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily B Fox, and Roman Garnett. Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019, neurips 2019, 8-14 december 2019, vancouver, bc, canada, 2019. *URL https://proceedings. neurips. cc/paper*, 2019.

[4] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[5] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models, 2018.

[6] Robin C. Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective, 2018.

[7] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649. IEEE, 2019.

[8] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. Fedsel: Federated sgd under local differential privacy with top-k dimension selection. In *Database Systems for Advanced Applications: 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24–27, 2020, Proceedings, Part I 25*, pages 485–501. Springer, 2020.

[9] Albert Cheu. Differential privacy in the shuffle model: A survey of separations, 2022.

[10] Badih Ghazi, Rasmus Pagh, and Ameya Velingker. Scalable and differentially private distributed aggregation in the shuffled model, 2019.

[11] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation, 2020.

[12] Borja Balle, Peter Kairouz, Brendan McMahan, Om Thakkar, and Abhradeep Guha Thakurta. Privacy amplification via random check-ins. *Advances in Neural Information Processing Systems*, 33:4623–4634, 2020.

[13] Ruixuan Liu, Yang Cao, Hong Chen, Ruoyang Guo, and Masatoshi Yoshikawa. Flame: Differentially private federated learning in the shuffle model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages

8688–8696, 2021.

[14] Zhiqiang Wang, Xinyue Yu, Qianli Huang, and Yongguang Gong. An adaptive differential privacy method based on federated learning, 2024.

[15] Qiantao Yang, Xuehui Du, Aodi Liu, Na Wang, Wenjuan Wang, and Xiangyu Wu. Adastopk: Adaptive federated shuffle model based on differential privacy. *Information Sciences*, 642:119186, 2023.

[16] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th symposium on operating systems principles*, pages 441–459, 2017.

[17] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[18] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 638–667. Springer, 2019.

[19] Xiaochen Li, Weiran Liu, Hanwen Feng, Kunzhe Huang, Yuke Hu, Jinfei Liu, Kui Ren, and Zhan Qin. Privacy enhancement via dummy points in the shuffle model. *IEEE Transactions on Dependable and Secure Computing*, 2023.

[20] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*, pages 375–403. Springer, 2019.

[21] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Pure differentially private summation from anonymous messages, 2020.

[22] Borja Balle, James Bell, Adria Gascón, and Kobbi Nissim. Private summation in the multi-message shuffle model. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 657–676, 2020.

[23] Rui Xue, Kaiping Xue, Bin Zhu, Xinyi Luo, Tianwei Zhang, Qibin Sun, and Jun Lu. Differentially private federated learning with an adaptive noise mechanism. *IEEE Transactions on Information Forensics and Security*, 2023.

[24] Yixuan Liu, Yuhan Liu, Li Xiong, Yujie Gu, and Hong Chen. Enhanced privacy bound for shuffle model with personalized privacy. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 3907–3911, 2024.

[25] Tianyu Xia, Shuheng Shen, Su Yao, Xinyi Fu, Ke Xu, Xiaolong Xu, and Xing Fu. Differentially private learning with per-sample adaptive clipping. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 10444–10452, 2023.

[26] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

[27] Borja Balle, James Bell, Adria Gascón, and Kobbi Nissim. Private summation in the multi-message shuffle model. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 657–676, 2020.

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

[29] Sunil Vadera and Salem Ameen. Methods for pruning deep neural networks. *IEEE Access*, 10:63280–63300, 2022.

[30] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[31] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE, 1993.

[32] Jianbo Ye, Xin Lu, Zhe Lin, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers, 2018.

[33] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, 2016.

[34] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity, 2019.

[35] Xiaojie Xu and Yun Zhang. Individual time series and composite forecasting of the chinese stock index. *Machine Learning with Applications*, 5:100035, 2021.

[36] Xiaojie Xu. Corn cash price forecasting. *American Journal of Agricultural Economics*, 102(4):1297–1320, 2020.

[37] Bingzi Jin, Xiaojie Xu, and Yun Zhang. Peanut oil price change forecasts through the neural network. *foresight*, 27(3):595–612, 2025.

[38] Erkang Li, Man Jiang, Duidui Li, Ruiduo Wang, Xin Kang, Tianqi Wang, Xiaoxin Yan, Beibei Liu, and Zhaoyu Ren. All-optical ti3c2tx modulator based on a sandwich structure. *Applied Optics*, 61(4):925–930, 2022.

[39] Bingzi Jin and Xiaojie Xu. Forecasts of thermal coal prices through gaussian process regressions. *Ironmaking & Steelmaking*, 51(8):819–834, 2024.

[40] Bingzi Jin and Xiaojie Xu. China commodity price index (ccpi) forecasting via the neural network. *International Journal of Financial Engineering*, pages 1–27, 2025.

[41] 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.

A Dataset Distribution

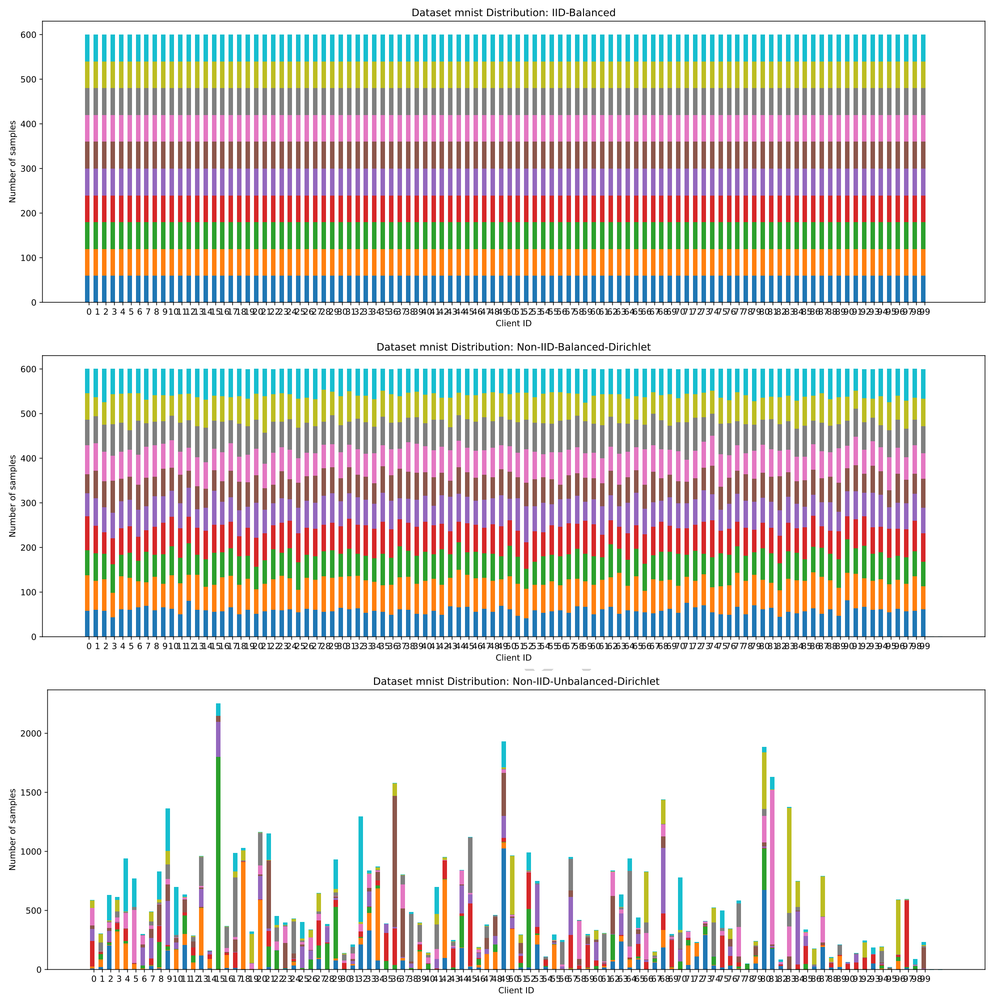

Fig. 9: Dataset Distribution of MNIST

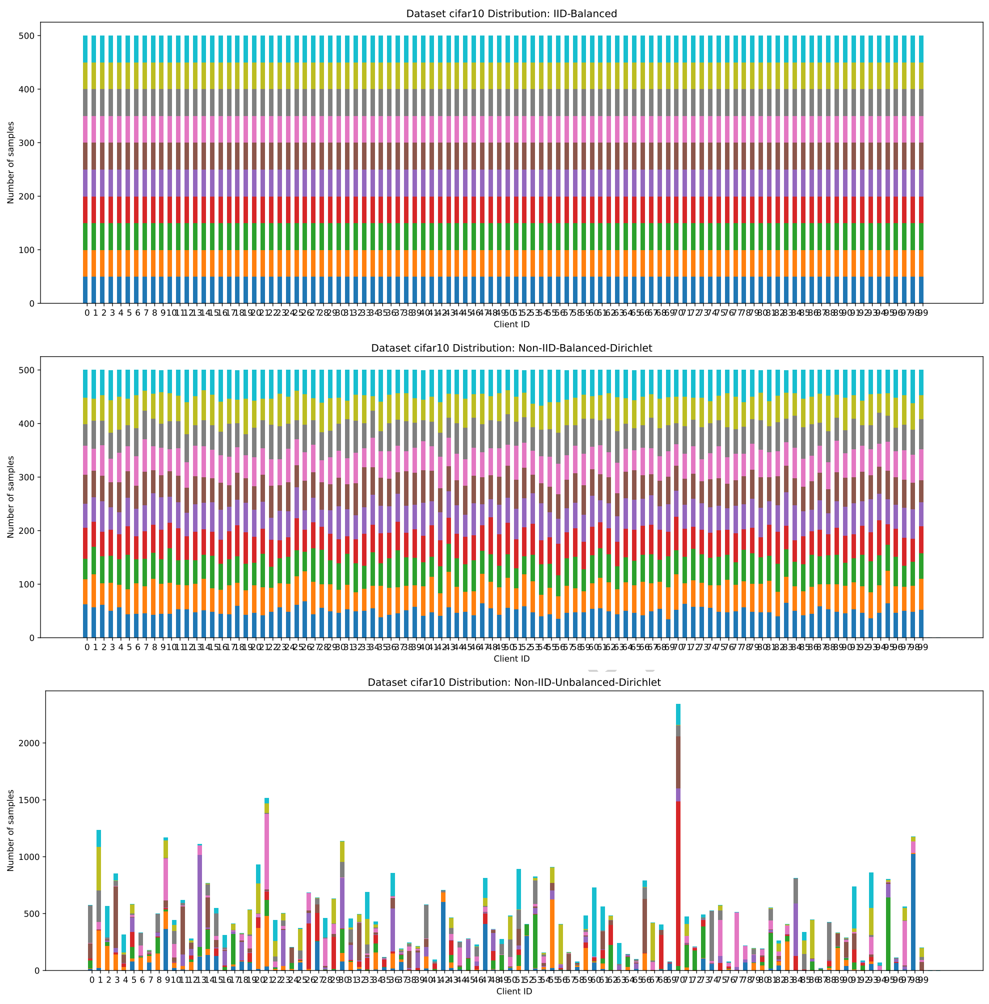

Fig. 10: Dataset Distribution of CIFAR10

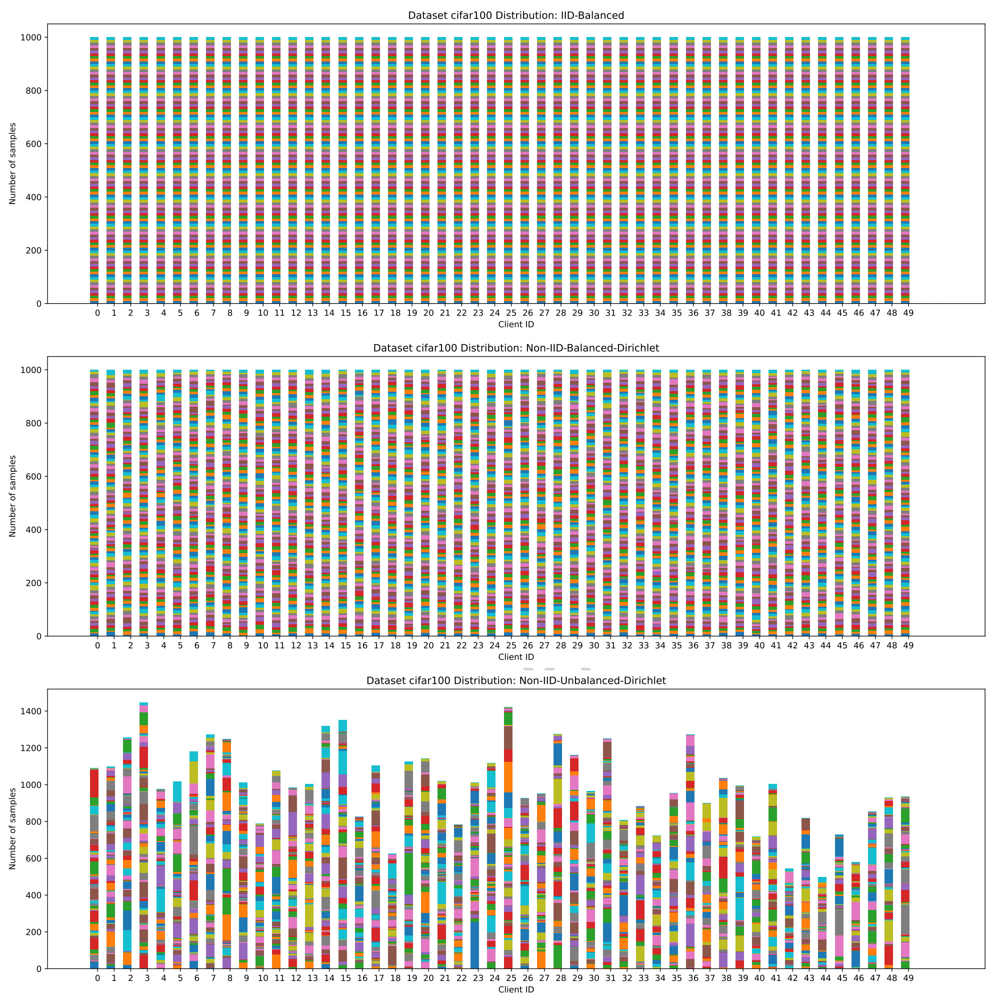

Fig. 11: Dataset Distribution of CIFAR100