FedLPA: One-shot Federated Learning with Layer-Wise Posterior Aggregation

Xiang Liu[1,†], **Liangxi Liu**[2,†], **Feiyang Ye**[3], **Yunheng Shen**[4], **Xia Li**[5], **Linshan Jiang**[1,‡], **Jialin Li**[1,‡]
[1]National University of Singapore, [2]Northeastern University,
[3]University of Technology Sydney, [4]Tsinghua University, [5]ETH Zurich
{liuxiang,lijl}@comp.nus.edu.sg liu.liangx@northeastern.edu feiyang.ye.uts@gmail.com
shenyh19@mails.tsinghua.edu.cn ethlixia@gmail.com linshan@nus.edu.sg
[†]*Equal Contribution* [‡]*Correspondence Author*

Abstract

Efficiently aggregating trained neural networks from local clients into a global model on a server is a widely researched topic in federated learning. Recently, motivated by diminishing privacy concerns, mitigating potential attacks, and reducing communication overhead, one-shot federated learning (i.e., limiting client-server communication into a single round) has gained popularity among researchers. However, the one-shot aggregation performances are sensitively affected by the non-identical training data distribution, which exhibits high statistical heterogeneity in some real-world scenarios. To address this issue, we propose a novel one-shot aggregation method with layer-wise posterior aggregation, named FedLPA. FedLPA aggregates local models to obtain a more accurate global model without requiring extra auxiliary datasets or exposing any private label information, e.g., label distributions. To effectively capture the statistics maintained in the biased local datasets in the practical non-IID scenario, we efficiently infer the posteriors of each layer in each local model using layer-wise Laplace approximation and aggregate them to train the global parameters. Extensive experimental results demonstrate that FedLPA significantly improves learning performance over state-of-the-art methods across several metrics.

1 Introduction

Data privacy issues in Deep Learning [1, 2, 3, 4, 5, 6, 7] have grown to be a major global concern [8]. To safeguard data privacy, the conventional federated learning algorithm will use the aggregation methods and follow the data management rules of different institutions, which implies that the distribution of data exhibits variations among clients [8]. In the domain of machine learning, federated learning (FL) [9, 10, 11] has emerged as a prominent paradigm. The fundamental tenet of federated learning revolves around sharing machine learning models derived from decentralized data repositories, as opposed to divulging user raw data. This approach effectively preserves the confidentiality of individual data.

The standard federated learning framework, FedAvg [9, 12], applies local model training. These local models are then aggregated into a global model through parameter averaging. Existing FL algorithms, however, require many communication rounds to effectively train a global model, leading to substantial communication overhead, increased privacy concerns, and higher demand for fault tolerance throughout the rounds. One-shot FL, which reduces client-server communication into a single round as explored by prior work [13, 14, 15], is a promising yet challenging scheme to address these issues. One-shot FL proves particularly practical in scenarios where iterative communication is

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

not feasible. Moreover, a reduction in communication rounds translates to fewer opportunities for any potential eavesdropping attacks.

While one-shot FL shows promises, existing approaches often grapple with challenges such as inadequate handling of high statistical heterogeneity information [16, 17] or non-independent and non-identically distributed (non-IID) data [18, 19]. Moreover, some prior methods rely on an auxiliary public dataset to achieve satisfactory performance in one-shot FL [13, 14], or even on pre-trained large models [20], which may not be practical [21] in some sensitive scenarios. Additionally, some approaches, such as those [22, 19, 23, 15]), might expose private label information to both local and global models, e.g., the client label distribution, potentially violating General Data Protection Regulation (GDPR) rules. Furthermore, some prior methods [14, 18, 24] require substantial computing resources for dataset distillation, model distillation, or even training a generator capable of generating synthetic data for second-stage training on the server side, making them less practical.

Besides, the performance of one-shot FL often falls short when dealing with non-IID data. Non-IID data biases global updates, reducing the accuracy of the global model and slowing down convergence. In extreme non-IID cases, clients may be required to address distinct classes solely on their side. Several approaches to federated learning are proposed in multi-round settings to tackle this heterogeneity among clients. In the work [25], it allows each client to use a personalized model instead of a shared global model. With the personalized approach, a multi-round framework benefits from joint training while allowing each client to keep its unique model. However, one-shot aggregation on a local model is far from being resolved to address the concern of non-i.i.d data distributions.

In this paper, we introduce a novel one-shot aggregation approach to address these issues, named FedLPA (Federated Learning with Layer-wise Posterior Aggregation). FedLPA infers the posteriors of each layer in each local model using the empirical Fisher information matrix obtained by layer-wise Laplace Approximation. Laplace Approximations are widely used to compute the empirical Fisher information matrix for neural networks, conveying the data statistics in non-i.i.d settings. However, computing empirical Fisher information matrices of multiple local clients and aggregating their Fisher information matrices remains an ongoing challenge [17]. To mitigate it, FedLPA aggregates the posteriors of local models using the accurately computed block-diagonal empirical Fisher information matrices to measure the parameter space. This matrix captures essential parameter correlations and distinguishes itself from prior methods by being non-diagonal and non-low-rank, thereby conveying the statistics of biased local datasets. After that, the global model parameters are aggregated without any need for server-side knowledge distillation [26].

Our extensive experiments verify the efficiency and effectiveness of FedLPA, highlighting that FedLPA markedly enhances the test accuracy when compared to existing one-shot FL baseline approaches across various datasets. Our main contributions are summarized as follows:

- To the best of our knowledge, we are the first to propose an effective one-shot federated learning approach that trains global models using block-diagonal empirical Fisher information matrices. Our approach is data-free without any need for any auxiliary dataset and label information and significantly improves system performance, including negligible communication cost and moderate computing overhead.
- We are the first to train global model parameters via constructing a multi-variate linear objective function and optimizing its quadratic form, which allows us to formulate and solve the problem in a convex form efficiently, which has a linear convergence rate, ensuring good performance.
- We conduct extensive experiments to illustrate the effectiveness of FedLPA. Our approach consistently outperforms the baselines, showcasing substantial improvement across various settings and datasets. Even in some extreme scenarios where label skew is severe, e.g., each client has only one class, we achieve satisfactory results while other existing one-shot federated learning algorithms struggle.

2 Background and related works

2.1 Federated learning on non-iid data

Previous work FedAvg [9] first introduced the concept of FL and presented the algorithm, which

achieved competitive performance on i.i.d data, in comparison to several centralized techniques. However, it was observed in previous works [27, 28] that the convergence rate and ultimate accuracy of FedAvg on non-IID data distributions were significantly reduced, compared to the results observed with homogeneous data distributions.

Several methods have been developed to enhance performance in federated learning against non-IID data distributions. The SCAFFOLD method [29] leveraged control variates to reduce objective inconsistency in local updates. It estimated the drift of directions in local optimization and global optimization and incorporated this drift into local training to align the local optimization direction with the global optimization. FedNova [30] addressed objective inconsistency while maintaining rapid error convergence through a normalized averaging method. It scaled and normalized the local updates of each client based on the number of local optimization steps. FedProx [31] enhanced the local training process by introducing a global prior in the form of an $L2$ regularization term within the local objective function. Researchers introduced PFNM [32, 33], a Bayesian probabilistic framework specifically tailored for multilayer perceptrons. PFNM employed a Beta-Bernoulli process (BBP) [34] to aggregate local models, quantifying the degree of alignment between global and local parameters. The framework [17] proposed utilized a multivariate Gaussian product method to construct a global posterior by aggregating local posteriors estimated using an online Laplace approximation. FedPA [16] also applied the Gaussian product method but employed stochastic gradient Markov chain Monte Carlo for approximate inference of local posteriors. DAFL (Data-Free Learning) [35] introduced an innovative framework based on generative adversarial networks. ADI [36] utilized an image synthesis method that leveraged the image distribution to train deep neural networks without real data. The pFedHN method [37] incorporated HyperNetworks [38] to address federated learning applications.

However, all of these methods encountered challenges in the one-shot federated learning setting, as they required aggregating the model by multiple rounds and might be inaccurate due to the omission of critical information, such as posterior joint probabilities between different parameters.

2.2 One-shot federated learning

One-shot Federated Learning (FL) is an emerging and promising research direction characterized by its minimal communication cost. In the first study on one-shot FL [13], the approach involved the aggregation of local models, forming an ensemble to construct the final global model. Subsequently, knowledge distillation using public data was applied in the following step. FedKT [14] brought forward the concept of consistent voting to fortify the ensemble. Recent research endeavors [19, 24] proposed data-free knowledge distillation schemes tailored for one-shot FL. These methods adopted the basic ensemble distillation framework as FedDF [26]. XorMixFL [22] introduced the use of exclusive OR operation (XOR) for encoding and decoding samples in data sharing. It is important to note that XorMixFL assumed the possession of labeled samples from a global class by all clients and the server, which might not align with practical real-world scenarios. A noteworthy innovation of DENSE [24] was its utilization of a generator to create synthetic datasets on the server side, circumventing the need for a public dataset in the distillation process. Co-Boosting [39] improves the ensemble when doing the distillation to improve the performance. FedOV [15] delved into addressing comprehensive label skew cases. FEDCVAE [23] confronted this challenge by transmitting all label distributions from clients to servers. These schemes [22, 14, 19, 24, 23, 15] exposed some client-side private information, leading to additional communication overhead and potential privacy leakage, e.g., FEDCVAE [23] needed all the client label distribution to be transmitted to the server side and FedOV [15] needed the clients to know the labels which were unknown. Instead, MA-Echo [40] adopted a unique approach by emphasizing the addition of norms among layer-wide parameters during the aggregation of local models. The project [41] focused on the theoretic analysis of the error in its approximation method. However, their method grappled with limited experiments and lacked detailed explanations of the approach. FedDISC [20], on the other hand, relied on the pre-trained model CLIP from OpenAI, where their reliance might not always align with practicality or suitability for diverse scenarios.

While some of these techniques are orthogonal to FedLPA and can be integrated with it, it is worth noting that none of the previously mentioned algorithms possess the capability to train global model parameters using empirical Fisher information matrices on extensive experiment settings. Some

of them [13, 14] may require additional information, and may potentially entail the risk of label distribution leakages.

3 Methodology

3.1 Objective formulation

Generally, federated learning is defined as an optimization problem [31, 29, 30, 42] for maximizing a global objective function $\mathbb{F}(\boldsymbol{\theta})$ which is a mixture of local objective functions $\mathbb{F}_k(\boldsymbol{\theta}, \mathcal{D}_k)$:

$$\mathbb{F}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \mathbb{F}_k(\boldsymbol{\theta}, \mathcal{D}_k) \tag{1}$$

where $\boldsymbol{\theta} = [\mathrm{vec}(\mathbf{W}_1), \ldots, \mathrm{vec}(\mathbf{W}_l), \ldots, \mathrm{vec}(\mathbf{W}_L)]$ is the parameter vector of global model and $\mathbf{W}_l$ is the weight and bias of layer l for a L-layers neural network; $\mathcal{D}_k$ is the local dataset k-th client. $\mathbb{F}_k(\boldsymbol{\theta}, \mathcal{D}_k)$ is the expectation of the local objective function, which is proportional to the logarithm of likelihood $\log p(\mathcal{D}_k|\boldsymbol{\theta})$.

Previous works [16, 17] give a common formula of the global posterior which consists of local posteriors $p(\boldsymbol{\theta}|\mathcal{D}_k)$ under variational inference formulation.

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto \prod_{k=1}^{K} p(\mathcal{D}_k|\boldsymbol{\theta}) \propto \prod_{k=1}^{K} p(\boldsymbol{\theta}|\mathcal{D}_k) \tag{2}$$

$$\max_{\boldsymbol{\theta}} \mathbb{F}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \frac{|D_k|}{|\mathcal{D}|} \cdot \mathbb{E}_{s \in \mathcal{D}_k} \left[\log p(s|\boldsymbol{\theta})\right] \equiv \max_{\boldsymbol{\theta}} \prod_{k=1}^{K} p(\boldsymbol{\theta}|\mathcal{D}_k) \tag{3}$$

As we know, the objective function is the expectation of the likelihood, and the sum of the logarithms is equal to the logarithms of the product as Eq. 3. Therefore, globally variational inference using Eq. 2 is equivalent to optimization for Eq. 1. Correspondingly, we have:

$$\max_{\boldsymbol{\theta}} \mathbb{F}_k(\boldsymbol{\theta}, \mathcal{D}_k) \equiv \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}_k) \tag{4}$$

Following the same training pattern of federated learning, each client infers the local posterior $p(\boldsymbol{\theta}|\mathcal{D}_k)$ by using the local dataset $\mathcal{D}_k$. As a result, the server obtains the global posterior $p(\boldsymbol{\theta}|\mathcal{D})$ by aggregating local posteriors using Eq. 2.

However, both the global and local posterior are usually intractable because modern neural networks are usually non-linear and have a large number of parameters. Therefore, it is necessary to design an efficient and accurate aggregation method for one-shot federated learning.

3.2 Approximating posteriors

Although the posterior is usually intractable, the posterior can be approximated as a Gaussian distribution by performing a Taylor expansion on the logarithm of the posterior [43]:

$$\log p(\boldsymbol{\theta}|\mathcal{D}) \approx \log p\left(\boldsymbol{\theta}^*|\mathcal{D}\right) - \frac{1}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right)^\top \bar{\mathbf{H}} \left(\boldsymbol{\theta} - \boldsymbol{\theta}^*\right) \tag{5}$$

where $\boldsymbol{\theta}^*$ is the optimal parameter vector, $\bar{\mathbf{H}} = \mathbb{E}_{s \in \mathcal{D}}[\mathbf{H}]$ is the average Hessian of the negative log posterior over a dataset $\mathcal{D}$. It is reasonable to approximate global and local posteriors as multivariates Gaussian distributions with expectations $\bar{\boldsymbol{\mu}} = \boldsymbol{\theta}^*$ and $\boldsymbol{\mu_k} = \boldsymbol{\theta}_k^*$; co-variances $\bar{\boldsymbol{\Sigma}} = \bar{\mathbf{H}}^{-1}$ and $\boldsymbol{\Sigma}_k = \bar{\mathbf{H}}_k^{-1}$ [44].

$$p(\boldsymbol{\theta}|\mathcal{D}) \equiv \boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}), p(\boldsymbol{\theta}|\mathcal{D}_k) \quad \equiv \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu_k}, \boldsymbol{\Sigma}_k) \tag{6}$$

As a result, if given local expectation $\boldsymbol{\mu}_k$ and local co-variance $\boldsymbol{\Sigma}_k$, the global posterior is determined by Eq. 2 as below:

$$\bar{\boldsymbol{\mu}} = \bar{\boldsymbol{\Sigma}} \sum_{k}^{K} \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\mu}_k, \bar{\boldsymbol{\Sigma}}^{-1} = \sum_{k}^{K} \boldsymbol{\Sigma}_k^{-1} \tag{7}$$

Modern algorithms [45, 46] allow the local training process to obtain an optimal, regarded as the expectation $\boldsymbol{\mu}_k$ in the above equations. However, $\bar{\mathbf{H}}_k$ is intractable to compute due to a large number of parameters in modern neural networks. An efficient method is to approximate $\bar{\mathbf{H}}_k$ using the empirical Fisher information matrix [47].

3.3 Inferring the local layer-wise posteriors with the block-diagonal empirical Fisher information matrices

A empirical Fisher $\tilde{\mathbf{F}}$ is defined as below:

$$\tilde{\mathbf{F}} = \sum_{s \in \mathcal{D}} \left[\nabla \log p(s|\boldsymbol{\theta}) \nabla \log p(s|\boldsymbol{\theta})^\top \right] \tag{8}$$

where $p(s|\boldsymbol{\theta})$ is the likelihood on data point s. It is an approximate of the Fisher information matrix, the empirical Fisher information matrix is equivalent to the expectation of the Hessian of the negative log posterior if assuming $p(s|\boldsymbol{\theta})$ is identical for each $s \in \mathcal{D}$.

Therefore, the local co-variance $\boldsymbol{\Sigma}_k$ can be approximated by the empirical Fisher $\tilde{\mathbf{F}}_k$ [48, 49].

$$\boldsymbol{\Sigma}_k^{-1} \approx \tilde{\mathbf{F}}_k + \lambda \mathbf{I} \tag{9}$$

The works [50, 51, 17] ignore co-relations between different parameters and only consider the self-relations of parameters as computing all co-relations is impossible. Thus, their methods are inaccurate. Detailed discussions and the novelty compared to previous works are in Appendix B.

In order to capture co-relations between different parameters efficiently, previous works [46, 43] estimate a block empirical Fisher information matrix $\mathbf{F}$ instead of assuming parameters are independent and approximating the co-variance by the diagonal of the empirical Fisher. As pointed out, co-relations inner a layer are much more significant than others [46, 52, 53], while computing the co-relations between different layers brings slight improvement but much more computation [54, 43]. Therefore, assuming parameters are layer-independent is a good trade-off. As a result, the approximated layer-wise empirical Fisher is block-diagonal. For layer l on client k, its empirical Fisher $\mathbf{F}_{k_l}$ is one of the diagonal blocks in the whole empirical Fisher for the local model and is factored into two small matrices as below,

$$\boldsymbol{\Sigma}_{k_l}^{-1} \approx \mathbf{F}_{k_l} = \mathbf{A}_{k_l} \otimes \mathbf{B}_{k_l} \tag{10}$$

where $\otimes$ is the Kronecker product; $\mathbf{A}_{k_l} = \mathbb{E}\left[\hat{\mathbf{a}}_{k_{l-1}} \hat{\mathbf{a}}_{k_{l-1}}^\top\right] + \pi_l \sqrt{\lambda}\mathbf{I}$ and $\mathbf{B}_{k_l} = \mathbb{E}\left[\hat{\mathbf{b}}_{k_l} \hat{\mathbf{b}}_{k_l}^\top\right] + \frac{1}{\pi_l}\sqrt{\lambda}\mathbf{I}$ are two expectation factor matrices over the data samples; $\hat{\mathbf{a}}_{k_l}$ is the activations and $\hat{\mathbf{b}}_{k_l}$ is the gradient of the pre-activations of layer l on client k, λ is the hyperparameter and π_l is a factor minimizing approximation error in $\mathbf{F}_{k_l}$ [46, 49, 55]. $\mathbf{A}_{k_l}$ and $\mathbf{B}_{k_l}$ are symmetric positive definite matrices [45, 46].

We use $\boldsymbol{\theta}_{k_l}$ to denote the parameter vector of layer l and $\mathbf{M}_{k_l} = vec^{-1}(\boldsymbol{\mu}_{k_l})$ is the vectorized optimal weight matrix of layer l on client k. Thus, the resulting local layer-wise posterior approximation is $\boldsymbol{\theta}_{k_l} \sim \mathcal{N}(\boldsymbol{\mu}_{k_l}, \mathbf{F}_{k_l}^{-1})$.

3.4 Estimating the global expectation

Given the local posteriors, the global expectation could be aggregated by Eq. 7. With Eq. 10, the l-th layer's global expectation $\bar{\boldsymbol{\mu}}_l$ consists of Kronecker products:

$$\begin{aligned} \bar{\boldsymbol{\mu}}_l &= \bar{\boldsymbol{\Sigma}}_l \sum_k^K \boldsymbol{\Sigma}_{k_l}^{-1} \boldsymbol{\mu}_{k_l} = \bar{\boldsymbol{\Sigma}}_l \sum_k^K (\mathbf{A}_{k_l} \otimes \mathbf{B}_{k_l}) \boldsymbol{\mu}_{k_l} \\ &= \bar{\boldsymbol{\Sigma}}_l \sum_k^K \mathrm{vec}(\mathbf{B}_{k_l} \mathbf{M}_{k_l} \mathbf{A}_{k_l}) = \bar{\boldsymbol{\Sigma}}_l \sum_k^K \mathbf{z}_{k_l} = \bar{\boldsymbol{\Sigma}}_l \bar{\mathbf{z}}_l \end{aligned} \tag{11}$$

where $\bar{\mathbf{z}}_l = \sum_k^K \mathbf{z}_{k_l}$ and $\mathbf{z}_{k_l} = \mathrm{vec}(\mathbf{B}_{k_l} \mathbf{M}_{k_l} \mathbf{A}_{k_l})$ is a immediate notations for simplification. For the global expectation, we have $\bar{\boldsymbol{\mu}} = \bar{\boldsymbol{\Sigma}} \cdot \bar{\mathbf{z}}$. The corresponding global co-variance is an inverse of the

sum of Kronecker products:

$$\bar{\mathbf{\Sigma}}_l = (\sum_k^K \mathbf{A}_{k_l} \otimes \mathbf{B}_{k_l})^{-1} \tag{12}$$

As shown in Eq. 11, obtaining the global expectation $\bar{\boldsymbol{\mu}}_l$ requires calculating the inverse of $\bar{\mathbf{\Sigma}}_l^{-1}$ as Eq. 12, which is unacceptable and the details are in Appendix C. Thus, we propose our method to directly train the parameters of the global model on the server side.

3.5 Train the parameters of the global model

We use $\mathbb{E}[\mathbf{A}]$ denotes $\sum_k^K(\mathbf{A}_k)$, $\mathbb{E}[\mathbf{B}]$ denotes $\sum_k^K(\mathbf{B}_k)$, $\mathbb{E}[\mathbf{A} \otimes \mathbf{B}]$ denotes $\sum_k^K(\mathbf{A}_k \otimes \mathbf{B}_k)$. Previous works [46, 49] approximate the expectation of Kronecker products by a Kronecker product of expectations $\mathbb{E}[\mathbf{A} \otimes \mathbf{B}] \approx \mathbb{E}[\mathbf{A}] \otimes \mathbb{E}[\mathbf{B}]$ with an assumption of $\mathbf{A}_{k_l}$ and $\mathbf{B}_{k_l}$ are independent, which is called Expectation Approximation (EA). However, it may lead to a biased global expectation. The details are discussed in Appendix D. Instead, we could construct a linear objective after aggregating the approximation of local posteriors via using block-diagonal empirical Fisher information matrices. We denotes $\bar{\mathbf{M}}$ as the matrix formula of $\bar{\boldsymbol{\mu}} = \mathrm{vec}(\bar{\mathbf{M}})$, and the optimal solution of $f(\bar{\boldsymbol{\mu}})$ is $\bar{\boldsymbol{\mu}}^* = \mathrm{vec}(\bar{\mathbf{M}}^*)$. We construct $f(\bar{\boldsymbol{\mu}})$ as a multi-variates linear objective function. When $\bar{\boldsymbol{\mu}} = \bar{\boldsymbol{\mu}}^*$ is optimal solution, $f(\bar{\boldsymbol{\mu}}) = \mathbf{o}$, where $\mathbf{o}$ is a vector with all zero. Note that

$$\begin{aligned} f(\bar{\boldsymbol{\mu}}) = \bar{\mathbf{\Sigma}}^{-1}\bar{\boldsymbol{\mu}} - \bar{\mathbf{z}} &= \sum_k^K \mathrm{vec}(\mathbf{B}_k \bar{\mathbf{M}} \mathbf{A}_k) - \bar{\mathbf{z}} \\ &= \mathrm{vec}(\mathbb{E}\left[\mathbf{B}\bar{\mathbf{M}}\mathbf{A}\right]) - \bar{\mathbf{z}} \end{aligned} \tag{13}$$

To obtain the optimal solution, we minimize the following problem to obtain an approximate solution $\bar{\mathbf{M}}^*$ of $\bar{\mathbf{M}}$:

$$\bar{\mathbf{M}}^* = \min_{\bar{\mathbf{M}}} \frac{1}{2} \left\| \sum_k^K \mathrm{vec}(\mathbf{B}_k \bar{\mathbf{M}} \mathbf{A}_k) - \bar{\mathbf{z}} \right\|_2^2 \tag{14}$$

The above equation is a quadratic objective, and it can be solved by modern optimization tools efficiently and conveniently. Since the main objective of the above problem is both convex and Lipschitz smooth w.r.t $\mathrm{vec}(\bar{\mathbf{M}})$, we can use the gradient descent method to solve it with a linear convergence rate. Here, we use automatic differentiation to calculate the gradient w.r.t. $\bar{\mathbf{M}}$.

Algorithm 1 FedLPA Global Aggregation

Input: clients K, layers L
Initialize global weight $\bar{\mathbf{W}}_l$ of layer $l = 1, ..., L$
clients executes:
Initialize local model
for k = 1, ..., K **do**
    $\{\mathbf{M}_{k_l}, \mathbf{A}_{k_l}, \mathbf{B}_{k_l} | l = 1, ..., L\} \leftarrow$ local training
end for
Server executes:
for l = 1, ..., L **do**
    $\bar{\mathbf{A}}_l \leftarrow \sum_k^K \mathbf{A}_{k_l}$
    $\bar{\mathbf{B}}_l \leftarrow \sum_k^K \mathbf{B}_{k_l}$
    $\bar{\mathbf{Z}}_l \leftarrow \sum_k^K \mathbf{B}_{k_l} \mathbf{M}_{k_l} \mathbf{A}_{k_l}$
    $\bar{\mathbf{M}}_l \leftarrow$ Train the parameter of the global model
    $\bar{\mathbf{W}}_l \leftarrow \bar{\mathbf{M}}_l$
end for

3.6 Overall FedLPA algorithm and discussions

In summary, the proposed algorithm FedLPA follows the same paradigm as the standard one-shot federated learning framework. In FedLPA, the clients locally train their models to get $\mathbf{M}_k$ and calculate the local co-variance over its training dataset using the layer-wise Laplace approximation to compute $\mathbf{A}_k, \mathbf{B}_k$. Subsequently, each client transmits their local $\mathbf{A}_k, \mathbf{B}_k, \mathbf{M}_k$ to the server. Following Algorithm 1, the server aggregates these contributions to obtain the global expectation, as described in Eq. 7, then trains the global model parameters, as outlined in Eq. 14. Thus, the transmitted data

between the clients and the server is solely $\mathbf{A}_k, \mathbf{B}_k, \mathbf{M}_k$ without any extra auxiliary dataset and label information.

Note that FedLPA can be directly adopted in most common scenarios. For the special case that the neural model has enormous single-layer weight parameters, how to extend our proposed FedLPA is discussed in Appendix E.

3.7 t-SNE observation and discussions

To quickly demonstrate the effectiveness of FedLPA, we show the t-SNE visualization of our FedLPA global model on the MNIST dataset as an example with a biased training data setting among 10 local clients. The experiment details, t-SNE visualizations of the local models and the global models of other algorithms and discussions are in Appendix G.1. As shown in Figure 1, FedLPA generates the global model which can clearly distinguish these classes, meanwhile, the classes are separate.

3.8 Privacy Discussions

FedLPA is intuitively compatible with existing privacy-preserving techniques, such as differential privacy (DP) [56, 57], secure multiparty computation (SMC) [58, 59], and homomorphic encryption (HE) [60, 61, 62]. In Appendix F.1, we propose a naive DP-FedLPA with two different mechanisms to show the compatibility with differential privacy. Meanwhile, we mention that our proposed FedLPA has the same privacy-preserving level as the conventional federated learning algorithms (i.e, FedAvg, FedProx, FedNova and Dense). Compared with FedAvg, we have conducted a detailed analysis from a privacy attack perspective to show that our proposed FedLAP exhibits a security level consistent with FedAvg against several types of privacy attacks, where the details are shown in Appendix F.3. Note that the main focus of FedLPA is to improve the learning performance on the one-shot FL settings, thus, we leave the integration with other privacy-preserving techniques beyond DP as an open problem.

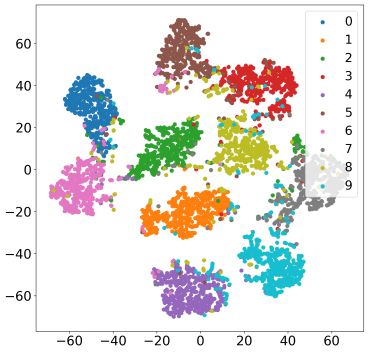


Figure 1: t-SNE visualization for our FedLPA global model.

4 Experiments

4.1 Experiments settings

Datasets. We conduct experiments on MNIST [63], Fashion-MNIST [64], CIFAR-10 [65], and SVHN [66] datasets. In most of the previous works and the most popular benchmark, the majority of their experiments use these datasets and these models. We choose these datasets and models to do the majority of our experiments following these established methods and benchmarks to fairly compare our method with the baselines. We use the data partitioning methods for non-IID settings of the benchmark [1] to simulate different label skews. Specifically, we try two different kinds of partition: 1) #C = k: each client only has data from k classes. We first assign k random class IDs for each client. Next, we randomly and equally divide samples of each class to their assigned clients; 2) p_k - Dir(β): for each class, we sample from Dirichlet distribution p_k and distribute $p_{k,j}$ portion of class k samples to client j. In this case, smaller β denotes worse skews.

Here's a brief overview of these datasets. MNIST Dataset: The MNIST dataset comprises binary images of handwritten digits. It consists of 60,000 28x28 training images and 10,000 testing images. FMNIST Dataset: Similar to MNIST, the FMNIST dataset also contains 60,000 28x28 training images and 10,000 testing images. SVHN Dataset: The SVHN dataset includes 73,257 32x32 color training images and 10,000 testing images. CIFAR-10 Dataset: CIFAR-10 consists of 60,000 32x32 color images distributed across ten classes, with each class containing 6,000 images. The input dimensions for MNIST, FMNIST, SVHN, and CIFAR-10 are 784, 784, 3,072, and 3,072, respectively.

[1] https://github.com/Xtra-Computing/NIID-Bench

Table 1: Comparison with various FL algorithms in one round.

Dataset	Partition	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	DENSE
FMNIST	β=0.01	21.20±0.67	10.13±0.00	15.97±0.12	18.17±0.15	13.37±0.19	15.23±0.14
	β=0.05	54.27±0.38	18.67±0.41	18.67±0.41	18.67±0.41	22.03±0.14	47.77±0.20
	β=0.1	55.33±0.06	30.47±0.59	31.40±0.25	30.93±0.58	31.00±0.52	52.93±0.67
	β=0.3	68.20±0.04	49.40±0.26	46.00±0.02	45.17±0.05	44.30±0.08	64.27±0.08
	β=0.5	73.33±0.06	57.03±0.28	56.03±0.28	59.10±0.63	58.10±0.47	72.87±0.13
	β=1.0	76.03±0.05	63.63±0.33	66.10±0.02	62.13±0.43	63.10±0.29	72.97±0.01
	#C=1	13.20±0.02	10.37±0.00	10.40±0.00	10.37±0.00	13.03±0.18	10.00±0.00
	#C=2	46.13±0.15	21.00±0.10	23.53±0.22	23.20±0.08	19.97±0.10	38.90±0.45
	#C=3	57.90±0.06	27.47±0.02	27.37±0.36	29.20±0.03	23.93±0.33	53.40±0.07
CIFAR-10	β=0.01	16.17±0.00	11.57±0.02	11.47±0.01	11.53±0.05	10.47±0.00	12.30±0.03
	β=0.05	18.37±0.00	10.30±0.00	10.73±0.01	10.23±0.00	10.97±0.02	17.87±0.31
	β=0.1	19.97±0.02	12.30±0.04	10.87±0.01	12.83±0.06	11.97±0.04	19.93±0.07
	β=0.3	26.60±0.01	11.77±0.02	10.93±0.01	10.53±0.00	10.97±0.00	25.57±0.84
	β=0.5	24.20±0.02	11.07±0.00	11.77±0.02	10.97±0.00	11.33±0.00	20.17±0.73
	β=1.0	29.33±0.00	12.00±0.00	13.00±0.00	13.23±0.00	13.63±0.01	28.23±0.34
	#C=1	10.70±0.01	10.50±0.00	10.27±0.00	10.23±0.00	10.37±0.01	10.00±0.00
	#C=2	16.40±0.00	10.07±0.00	12.03±0.08	10.07±0.00	10.03±0.00	14.13±0.22
	#C=3	18.97±0.01	11.30±0.01	11.00±0.01	11.53±0.01	10.77±0.00	14.77±0.11
MNIST	β=0.01	39.17±1.16	13.53±0.20	8.87±0.01	9.37±0.00	9.33±0.00	15.80±0.24
	β=0.05	70.07±0.05	31.60±0.71	41.07±0.46	38.57±0.28	32.23±0.18	57.83±1.55
	β=0.1	77.43±0.14	48.07±0.28	47.73±0.22	48.63±0.15	47.40±0.00	70.33±0.02
	β=0.3	85.77±0.02	67.6±0.40	67.07±0.15	66.17±0.21	63.40±0.41	84.50±0.01
	β=0.5	88.73±0.07	79.27±0.08	78.57±0.29	77.37±0.07	79.60±0.24	86.33±0.36
	β=1.0	93.37±0.08	84.93±0.18	85.33±0.15	85.10±0.13	86.50±0.16	91.43±0.02
	#C=1	11.43±0.01	10.27±0.02	10.10±0.01	10.10±0.01	10.13±0.01	9.93±0.00
	#C=2	69.63±0.29	20.90±0.49	25.23±1.08	16.47±0.23	14.30±0.34	52.73±0.46
	#C=3	77.13±0.24	29.53±1.65	31.83±2.45	33.13±2.60	29.00±2.05	58.90±0.31
SVHN	β=0.01	19.20±0.00	13.73±0.14	9.83±0.00	12.13±0.04	11.43±0.12	17.33±0.28
	β=0.05	22.93±0.38	14.90±0.43	15.77±0.14	16.60±0.23	15.90±0.12	21.47±0.20
	β=0.1	39.77±0.69	25.97±0.13	25.70±0.08	22.17±0.02	24.50±0.06	19.43±0.45
	β=0.3	52.23±0.26	34.40±0.28	34.03±0.06	33.93±0.26	34.70±0.20	47.13+7.14
	β=0.5	54.27±0.02	38.53±0.07	40.07±0.13	38.53±0.15	36.93±0.09	53.70±0.07
	β=1.0	67.80±0.01	55.60±0.08	54.03±0.14	55.97±0.04	55.23±0.12	54.40+9.43
	#C=1	19.60±0.00	10.43±0.00	13.73±0.18	13.77±0.17	18.27±0.03	7.70±0.03
	#C=2	47.03±4.63	12.90±0.27	24.47±0.08	20.17±0.04	17.47±0.13	37.67±0.76
	#C=3	48.00±0.22	20.87±0.12	28.37±0.09	27.60±0.03	24.93±0.10	47.43±0.40

Training Details. By default, we follow FedAvg [12] and other existing studies [67, 68, 15] to use a simple CNN with 5 layers in our experiments. The experiments with more complex neural network structures are in Appendix G.8. We set the batch size to 64, the learning rate to 0.001, and the $\lambda = 0.001$ for FedLPA. By default, we set 10 clients and run 200 local epochs for each client. For the various settings of the number of clients and local epochs, we refer to Section 4.3 and Section 4.5. For results with error bars, we run three experiments with 5 different random seeds. Note that all methods were evaluated under fair comparison settings. Due to the page limit, representative results are represented in the main paper. Refer to Appendix G for more experimental details and additional results.

Baselines. To ensure fair comparisons, we neglect the comparison with methods that require to download auxiliary models or datasets, such as FedBE [69], FedKT [14] and FedGen [21], or even pretrained large model, like FedDISC [20]. FedOV [15] and FEDCAVE [23] entail sharing more client-side label information or transmitting client label information to the server, which could jeopardize label privacy and are beyond the scope of this study. XorMixFL [22] may not be practical, as we mentioned before. FedFisher [41] is not publicly available. FedDF [26], DAFL [35] and ADI [36] are compared with the state-of-the-art data-free method DENSE [24]. Co-Boosting [39] requires too many computational resources[2]. In conclusion, we include one-shot FL algorithms as baselines including FedAvg [12], FedProx [31], FedNova [30], SCAFFOLD [29] and DENSE [24]. All the methods are fairly compared, and our implementation is available and the experiment details can be viewed in Appendix G.11.

4.2 An overall comparison

We compare the accuracy between FedLPA and the other baselines as shown in Table 1, the data in the green shadow shows the best results. FedLPA can achieve the best performance in all the dataset and partition settings. In extreme cases such as $\beta = \{0.01, 0.05\}$, #C = 1, #C = 2, FedLPA exhibits a significant performance advantage over the baseline algorithms. This demonstrates our framework's ability to effectively aggregate valuable information from local clients for global weight training.

[2]The experiments with more models, FedOV and Co-Boosting, are in Appendix G.5.

Table 2: Experimental results of varying number of clients on FMNIST dataset.

# of Clients	Partition	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	DENSE
20 Clients	β=0.01	33.57±0.38	10.00±0.00	13.13±0.24	13.23±0.21	13.93±0.08	10.30±0.00
	β=0.05	47.30±0.74	21.30±0.08	20.53±0.56	21.20±0.64	19.40±0.46	46.13±0.36
	β=0.1	57.37±0.05	31.50±0.29	29.23±0.60	32.43±0.99	28.80±1.26	57.20±0.12
	β=0.3	71.30±0.03	53.87±0.33	50.63±0.10	52.83±0.08	52.13±0.40	71.17±0.04
	β=0.5	74.07±0.00	62.83±0.03	58.60±0.08	60.17±0.03	59.47±0.06	74.10±0.04
	β=1.0	76.07±0.01	68.63±0.08	69.13±0.12	68.33±0.08	69.33±0.10	75.47±0.04
	#C=1	21.50±0.30	10.00±0.00	10.00±0.00	10.00±0.00	10.33±0.00	10.00±0.00
	#C=2	59.17±0.45	19.23±0.23	19.47±0.49	18.53±0.46	13.53±0.26	33.07±0.27
	#C=3	66.37±0.01	27.30±0.20	28.07±0.35	25.93±0.27	24.63±0.26	52.23±0.79
50 Clients	$\beta = 0.01$	15.91±0.01	10.00±0.00	10.00±0.00	10.00±0.00	10.27±0.00	10.00±0.00
	β=0.05	28.43±0.80	15.50±0.43	17.77±0.25	17.37±0.24	18.10±0.01	25.03±0.47
	β=0.1	57.03±0.00	34.33±0.04	30.17±0.03	28.90±0.05	31.00±0.27	55.83±0.49
	β=0.3	66.70±0.23	46.70±0.12	43.97±0.02	45.40±0.12	45.07±0.11	59.23±1.90
	β=0.5	71.13±0.00	57.93±0.40	52.93±0.22	53.67±0.26	53.80±0.20	69.57±0.02
	β=1.0	71.07±0.04	60.00±0.20	57.67±0.22	56.30±0.45	56.90±0.41	70.33±0.03
	#C=1	15.93±0.02	10.00±0.00	10.00±0.00	10.00±0.00	10.27±0.00	10.00±0.00
	#C=2	49.60±0.37	18.03±0.11	17.20±0.00	20.50±0.26	15.70±0.03	44.57±0.92
	#C=3	65.50±0.05	38.03±0.99	40.53±1.41	40.97±1.51	38.93±1.34	56.10±0.38

In summary, the state-of-the-art DENSE could be comparable with FedLPA when the skew level is small. However, with the increment of skewness, FedLPA shows significantly superior results.

4.3 Scalability

We assess the scalability of FedLPA by varying the number of clients. In this section, we show results on FMNIST in Table 2. From the table, we can observe that FedLPA still almost always achieves the best accuracy when increasing the number of clients. Notably, there is a slight exception highlighted in red, where DENSE outperforms us when we have 20 clients and $\beta = 0.5$, this may be attributed to the dataset being less biased and the DENSE only getting a marginal 0.03% higher test accuracy. Our method is generally much more robust in all kinds of settings.

Table 3: Experiments with different proportions of data samples.

Data sample proportion	Accuracy(β=0.1)	Accuracy(β=0.3)	Accuracy(β=0.5)
100%	55.33±0.06	68.20±0.04	73.33±0.06
80%	53.88±1.14	65.47±0.02	73.17±0.05
60%	53.15±0.82	64.80±0.71	72.40±0.29
40%	53.20±0.21	64.10±0.40	70.02±0.17
20%	45.71±0.13	62.15±0.03	68.54±2.02

4.4 Experiments with different proportions of data samples

We have added the experiments with our method on the same experiment setting with 10 clients. We conducted experiments on FMNIST datasets with β=0.1, 0.3 and 0.5. The performance changes w.r.t the number of data samples are shown in Table 3. We could see that our method FedLPA could yield satisfactory results even with only 20% data samples under multiple settings.

4.5 Ablation study

The hyper-parameter of our approach is λ, which controls variances of a priori normal distribution and guarantees $\mathbf{A}_k$ and $\mathbf{B}_k$ are positive semi-definite. In this part, we show results on FMNIST. All other Laplace Approximations are sensitive to the hyper-parameter λ based on their experimental results, Table 4 shows that our approach is relatively robust. Based on our numerical results, we set $\lambda = 0.001$ by default for our method FedLPA.

We also conduct the experiments when the local epochs are 10,20,50,100. More experiments are available in Appendix G.2, which shows that our methods outperform all the baselines in all kinds of scenarios without requiring extensive tuning.

4.6 Communication and computation overhead

We conduct experiments on CIFAR-10 on a single 2080Ti GPU to estimate the overall communication and computation overhead. We set the number of clients is 10. Table 5 shows the numerical results on

Table 4: Experimental results of different hyper-parameter λ on FMNIST dataset.

value of λ	0.01	0.001	0.0001
β=0.01	18.63±0.78	21.20±0.67	22.50±1.84
β=0.05	54.33±0.54	54.27±0.38	53.30±0.01
β=0.1	56.83±0.19	55.33±0.06	54.60±0.15
β=0.3	66.83±0.02	68.20±0.04	67.53±0.03
β=0.5	73.20±0.03	73.33±0.06	72.17±0.04
β=1.0	76.53±0.02	76.03±0.05	73.47±0.19
#C=1	12.73±0.01	13.20±0.02	14.17±0.02
#C=2	45.20±0.21	46.13±0.15	44.80±0.03
#C=3	58.97±0.07	57.90±0.06	55.60±0.06

Table 5: Communication and computation overhead evaluation.

	Overall Computation (mins)	Overall Communication (MB)
FedLPA	65	4.98
FedNova	50	2.47
SCAFFOLD	50	4.94
FedAvg	50	2.47
FedProx	75	2.47
DENSE	400	2.47

FedLPA and baselines. Details of the overhead evaluation are referred to Appendix G.6 and G.7. Our observations reveal that FedLPA is slightly slower than FedNova, SCAFFOLD, FedAvg, and FedProx, while much faster than DENSE. FedLPA also has significantly improved the one-shot learning performance of the above four approaches. Similarly, FedLPA performs moderately incremental communication overhead while outperforming other baseline approaches on learning performance, as one-shot FL introduces heavy computation overhead while communication overhead is usually small. It is noteworthy that FedLPA strikes a favorable balance between computation and communication overhead, making it the most promising approach for one-shot FL.

4.7 Extension to multiple rounds

We conduct experiments on MNIST with 10 clients and data partitioning p_k - Dir($\beta = 0.5$). The results are shown in Figure 2. As DENSE could not support multiple rounds, we compare our methods with FedAvg, FedNova, SCAFFOLD, and FedProx. FedLPA achieves the highest accuracy in the first round, denoting the strongest learning capabilities in a one-shot setting. With the increment in the number of rounds, the performances of FedLPA increase slower than the other baseline approaches. This figure shows that the joint approach (ours (one round) then FedAvg) that utilizes FedLPA in the first round and then adopts other baseline methods may be most promising to save communication and computation resources in the multiple-round federated learning scenario.

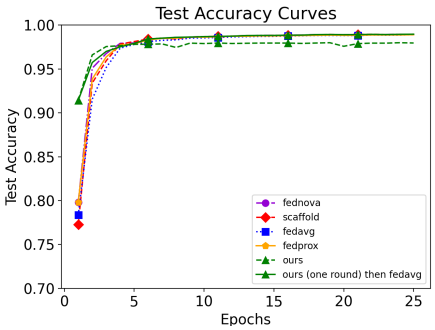


Figure 2: Extension to multiple rounds on MNIST dataset.

4.8 Supplementary experiments

Experiments for privacy concerns, experiments on different local epoch numbers, experiments in extreme settings (the number of clients=5, $\beta = 0.001$), experiments with more methods, experiments with more complex network structures, experiments with more complex datasets, ablation experiments analyzing the number of approximation iterations of FedLPA can be found in Appendix.

5 Conclusions

In this work, we design a novel one-shot FL algorithm FedLPA to better model the global parameters in effective one-shot federated learning. We propose a method that could aggregate the local clients in a layer-wise manner with their posterior approximation via block-diagonal empirical Fisher information matrices, which could effectively capture the accurate statistics of a locally biased dataset. Overall, FedLPA stands out as the most practical and efficient framework that conducts data-free one-shot FL, particularly well-suited for high data heterogeneity in various settings, considering it significantly outperforms other baselines with extensive experiments. Our FedLPA is available in https://github.com/lebronlambert/FedLPA_NeurIPS2024.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers. The authors would like to thank Yiqun Diao from National Univeristy of Singapore for his valuable comments to improve this work. Dr. Jialin Li is supported by the Singapore Ministry of Education Academic Research Fund Tier 1 (T1 251RES2104) and Tier 2 (MOE-T2EP20222-0016).

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR, 2016.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[5] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[6] Qingchen Zhang, Laurence T Yang, Zhikui Chen, and Peng Li. A survey on deep learning for big data. *Information Fusion*, 42:146–157, 2018.

[7] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[8] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[9] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2, 2016.

[10] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[11] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[12] 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.

[13] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

[14] Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. *arXiv preprint arXiv:2010.01017*, 2020.

[15] Yiqun Diao, Qinbin Li, and Bingsheng He. Towards addressing label skews in one-shot federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. *arXiv preprint arXiv:2010.05273*, 2020.

[17] Liangxi Liu, Xi Jiang, Feng Zheng, Hong Chen, Guo-Jun Qi, Heng Huang, and Ling Shao. A bayesian federated learning framework with online laplace approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[18] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.

[19] Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Jianghe Xu, Shouhong Ding, and Chao Wu. A practical data-free approach to one-shot federated learning with heterogeneity. *arXiv preprint arXiv:2112.12371*, 1, 2021.

[20] Mingzhao Yang, Shangchao Su, Bin Li, and Xiangyang Xue. Exploring one-shot semi-supervised federated learning with a pre-trained diffusion model. *arXiv preprint arXiv:2305.04063*, 2023.

[21] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pages 12878–12889. PMLR, 2021.

[22] MyungJae Shin, Chihoon Hwang, Joongheon Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Xor mixup: Privacy-preserving data augmentation for one-shot federated learning. *arXiv preprint arXiv:2006.05148*, 2020.

[23] Clare Elizabeth Heinbaugh, Emilio Luz-Ricca, and Huajie Shao. Data-free one-shot federated learning under very high statistical heterogeneity. In *The Eleventh International Conference on Learning Representations*, 2022.

[24] Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. Dense: Data-free one-shot federated learning. *Advances in Neural Information Processing Systems*, 35:21414–21428, 2022.

[25] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

[26] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[27] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[28] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[29] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[30] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[31] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[32] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.

[33] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[34] Romain Thibaux and Michael I Jordan. Hierarchical beta processes and the indian buffet process. In *Artificial intelligence and statistics*, pages 564–571. PMLR, 2007.

[35] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE international conference on computer vision*, pages 3514–3522, 2019.

[36] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[37] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, pages 9489–9502. PMLR, 2021.

[38] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

[39] Rong Dai, Yonggang Zhang, Ang Li, Tongliang Liu, Xun Yang, and Bo Han. Enhancing one-shot federated learning through data and ensemble co-boosting. *arXiv preprint arXiv:2402.15070*, 2024.

[40] Shangchao Su, Bin Li, and Xiangyang Xue. One-shot federated learning without server-side training. *Neural Networks*, 164:203–215, 2023.

[41] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedfisher: Leveraging fisher information for one-shot federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1612–1620. PMLR, 2024.

[42] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[43] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

[44] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

[45] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[46] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[47] Charles F Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1-2):85–100, 2000.

[48] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[49] Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2016.

[50] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[51] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[52] Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. In *International Conference on Machine Learning*, pages 1817–1853. PMLR, 2022.

[53] Lin Zhang, Shaohuai Shi, Wei Wang, and Bo Li. Scalable k-fac training for deep neural networks with distributed preconditioning. *IEEE Transactions on Cloud Computing*, 2022.

[54] James Martens. *Second-order optimization for neural networks*. University of Toronto (Canada), 2016.

[55] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical gauss-newton optimisation for deep learning. In *International Conference on Machine Learning*, pages 557–565. PMLR, 2017.

[56] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

[57] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and S Yu Philip. Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems*, 2022.

[58] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.

[59] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *Network and Distributed System Security Symposium*, 2015.

[60] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

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

[62] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[63] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[64] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[65] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[66] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[67] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A principled approach to data valuation for federated learning. *Federated Learning: Privacy and Incentive*, pages 153–167, 2020.

[68] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022.

[69] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

[70] Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2016.

[71] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical gauss-newton optimisation for deep learning. In *International Conference on Machine Learning*, pages 557–565. PMLR, 2017.

[72] Elaine Angelino, Matthew James Johnson, Ryan P Adams, et al. Patterns of scalable bayesian inference. *Foundations and Trends® in Machine Learning*, 9(2-3):119–247, 2016.

[73] Ward Cheney and David Kincaid. *Numerical mathematics and computing*. International Thomson Publishing, 1998.

[74] David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons, 2005.

[75] M Hashem Pesaran. *Time series and panel data econometrics*. Oxford University Press, 2015.

[76] Lloyd N Trefethen and David Bau. *Numerical linear algebra*. SIAM, 2022.

[77] Jongseok Lee, Matthias Humt, Jianxiang Feng, and Rudolph Triebel. Estimating model uncertainty of neural networks in sparse information form. In *International Conference on Machine Learning*, pages 5702–5713. PMLR, 2020.

[78] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. *Advances in Neural Information Processing Systems*, 31, 2018.

[79] Jiankai Sun, Xin Yang, Yuanshun Yao, and Chong Wang. Label leakage and protection from forward embedding in vertical federated learning. *arXiv preprint arXiv:2203.01451*, 2022.

[80] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User-level label leakage from gradients in federated learning. *Proceedings on Privacy Enhancing Technologies*, 2022(2):227–244, 2022.

[81] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[82] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[83] Cynthia Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340, 2011.

[84] 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.

[85] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[86] Yiqun Diao, Qinbin Li, and Bingsheng He. Exploiting label skews in federated learning with model concatenation. *arXiv preprint arXiv:2312.06290*, 2023.

[87] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[88] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 603–618, 2017.

[89] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Robust traceability from trace amounts. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 650–669. IEEE, 2015.

[90] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. *arXiv preprint arXiv:1708.06145*, 2017.

[91] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[92] Jamie Hayes, Luca Melis, George Danezis, and E LOGAN De Cristofaro. Membership inference attacks against generative models. *URL https://api. semanticscholar. org/CorpusID*, 202588705, 2018.

[93] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

[94] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Towards demystifying membership inference attacks. *arXiv preprint arXiv:1807.09173*, 2018.

[95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[96] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[97] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[98] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

[99] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

[100] James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*, 2018.

[101] Blair Bilodeau, Yanbo Tang, and Alex Stringer. On the tightness of the laplace approximation for statistical inference. *Statistics & Probability Letters*, 198:109839, 2023.

A The FedLPA algorithm

The proposed algorithm follows the same paradigm as the standard one-shot federated learning framework. Each client follows the local training procedure as shown in the paper. The global aggregation is illustrated in Algorithm 1.

With the Algorithms, let us assume the dimensionality list of each layer in a fully connected neural network is ([s_0, s_1, s_2, ..., s_l,..., s_L]), which means the size of the weight $\mathbf{W}_{k_l}$ of layer l is s_{l-1}xs_l. Consequently, the size of $\mathbf{A}_{k_l}$ for this layer would be s_{l-1}xs_{l-1}, and the size of $\mathbf{B}_{k_l}$ would be s_lxs_l. The size of $\mathbf{F}_{k_l}$ is (s_{l-1}xs_l)x(s_{l-1}xs_l).

Then, we give a concrete example to show the dimensions of different matrices using a fully-connected neural network model with architecture 784-256-64-10 as in Appendix G. Then, the $\mathbf{M}_{k_1}$ is 784x256+256, $\mathbf{M}_{k_2}$ is 256x64+64, $\mathbf{M}_{k_3}$ is 64x10+10. The $\mathbf{A}_{k_1}$ is 785x785, $\mathbf{A}_{k_2}$ is 257x257, $\mathbf{A}_{k_3}$ is 65x65. The $\mathbf{B}_{k_1}$ is 256x256, $\mathbf{B}_{k_2}$ is 64x64, $\mathbf{B}_{k_3}$ is 10x10. Then the $\mathbf{F}_{k_1}$ is (785x785)x(256x256), $\mathbf{F}_{k_2}$ is (257x257)x(64x64), $\mathbf{F}_{k_3}$ is (65x65)x(10x10). The $\mathbf{F}_k$ is (785x785+257x257+65x65)x(256x256+64x64+10x10).

However, in fact, we do not need to combine the $\mathbf{A}_{k_l}$, $\mathbf{B}_{k_l}$, $\mathbf{F}_{k_l}$ into $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{F}_k$. In this paper, we utilize the diagonal block property to compute each block in our method.

B Comparison with the previous methods

To the best of our knowledge, we are the first to consider the posterior inference problem in the one-shot scenario. Note that the approach [16] requires a lengthy burn-in period before conducting posterior inference, for instance, 400 rounds, and it updates global model parameters by modifying the covariance-aggregated local models. It means that the algorithm [16] necessarily requires multiple iterations and cannot be used in a one-shot scenario. In contrast, our method FedLPA only requires immediate variational inference after training the local model, ensuring higher flexibility and efficiency in the one-shot scenario.

Besides, in the algorithm [16], obtaining statistical information to compute local covariances is of low rank. In reality, it fails to acquire the posterior of the aggregated model and cannot perform variational inference on the aggregated model. However, our method yields full-rank covariances, and after employing an expectation approximation method for variational inference on the aggregated model, we can achieve a usable global posterior.

In both the domain of natural gradient optimization [48, 70, 16] and modeling output uncertainty in variational inference [71], using the Fisher approximation of the Hessian does not involve the issue of inverting covariance. However, in the context of federated learning, when performing variational inference on the aggregated model, the necessity of inverting covariance becomes unavoidable. To address this problem, we propose a novel algorithm that constructs a quadratic objective function. During aggregation, this algorithm directly trains the aggregated model using local covariances and expectations, thereby circumventing the need for inversion operations.

Note that the previous methods [51, 17] adopt the same core approach that utilizes the online Laplace approximation to obtain diagonal Fisher for model aggregation, in which they conduct experiments on different datasets and published on different venues. We mainly analyze our approach with the comparison of DiagonalFisher [17]. DiagonalFisher assumes independence among parameters, neglecting inter-parameter correlations, resulting in inaccurate posterior approximations. However, strong correlations exist among parameters within each layer, such as matching patterns in convolutional kernels within convolutional networks. This is a crucial factor that cannot be overlooked; otherwise, aggregation of the posterior would result in lower posterior regions, as compared to our method. In complex environments, employing diagonal Fisher for aggregation would prove to be entirely ineffective, whereas our method effectively leverages inter-parameter correlations at each layer, rendering it more robust. To demonstrate, we present results comparing aggregation using diagonal Fisher and our method. We have added experiments using the settings of our paper and an MLP model (784-256-64-10) on the FMNIST dataset with five random seeds for one-shot FL, the client number is 10, and the β=0.01. The results are in Table 6.

In the table, “Initial" denotes whether the client models were initialized using the same parameter values or independently.

Table 6: Experiments with DiagonalFisher using MLP.

Initial	FedAvg	FedProx	SCAFFOLD	DiagonalFisher	FedLPA
Same	42.35±0.16	24.80±0.10	42.10±0.15	56.34±0.34	76.63±0.04
Different	10.00±0.00	24.12±0.02	10.16±0.70	10.51±0.11	73.73±0.07

When “Initial" is set to “Same", all client models are trained on their respective datasets using identical parameter values for initialization. Consequently, there exists a strong correlation among the local models. Additionally, in this scenario, model aggregation is equivalent to aggregating updates of local models. Although DiagonalFisher performs reasonably well under this condition, our method demonstrates superior performance, exhibiting a 20.29% increase in global test accuracy.

When “Initial" is set to "Different", the models on different clients start training with distinct parameter values. Due to the high heterogeneity of local datasets, there is minimal correlation among local models. In this extreme scenario, DiagonalFisher completely fails, while our method maintains an accuracy of 73.73%, showcasing remarkable robustness.

It is essential to consider the indispensability of parameter correlations, which is why we compute correlations among parameters within layers to ensure the robustness and accuracy of model aggregation.

Now, we discuss some related works which directly utilize K-FAC to approximate the Fisher matrix and make a comparison with our proposed approach FedLPA. The works [48, 70, 71] have provided us with significant inspiration. However, methods like K-FAC do not require computing the inverse of covariance. Nevertheless, in the context of federated learning, the necessity of inverting covariance becomes unavoidable during variational inference on the aggregated model.

Methods like K-FAC assume direct independence among data samples to utilize expectation approximation. They obtain the inverse of Fisher from individual samples and then directly compute the expectation, thereby avoiding inverse operations. However, the expectation approximation inevitably leads to biased results during model aggregation. Detailed analysis can be found in Appendix D.

To address this issue, we propose a novel algorithm that constructs a quadratic objective function. During aggregation, this algorithm directly trains the aggregated model using local covariances and expectations, eliminating the need for inversion operations. This aims to minimize aggregation biases as much as possible.

Here, we provide a comparative analysis of different methods.

FedAvg and FedProx minimize the Kullback-Leibler (KL) divergence between the local and global posteriors: $\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}^{-1} = \min_{\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}^{-1}} KL\left((\sum_k^K p(\boldsymbol{\theta}|\mathcal{D}_k))|p(\boldsymbol{\theta}|\mathcal{D})\right)$. SCAFFOLD computes the bias term, and FedNova computes the correction term. None of these four methods consider the correlations between parameters. DENSE and FedOV, on the other hand, employ distillation methods, attempting to extract the distribution of non-iid data among clients through distillation. However, this itself leads to information loss due to dimensionality reduction and introduces additional variance of data.

Although the work [72] also uses the distributed Bayesian inference, however, it focuses on the dataset feature and could not be applied to train the global model parameters.

In conclusion, the reason our approach performs better in this scenario stems from our improved approximation of the global posterior. This approach signifies our novelty in addressing these challenges.

B.1 The efficiency of FedLPA

Although the number of uploaded bits increased per round of FedLPA, it resulted in a significant improvement in the final outcome. Additionally, the increase in transmitted bits enhanced the robustness of the aggregation method. Moreover, as indicated in Table 5 of the paper, we observe only a marginal increase in the amount of communication required.

A fully-connected neural network model with architecture 784-256-64-10, has $784 \cdot 256 + 256 + 256 \cdot 64 + 64 + 64 \cdot 10 + 10 = 217930$ floating point numbers, which is 6973760 bits or around 0.831

Table 7: Experiments with DiagonalFisher considering efficiency.

"Initial" Method	FedLPA (Global Test Acc / MB)	DiagonalFisher (Global Test Acc / MB)
"Same"	76.63/(2.272*10) = 3.37	56.34/(1.662*10) = 3.39
"Different"	73.73/(2.272*10) = 3.25	10.51/(1.662*10) = 0.63

MB. For one communication from a client to the server, our approach needs to upload additional $\mathbf{A}_k$ and $\mathbf{B}_k$, which have $785 \cdot 785 + 256 \cdot 256 + 257 \cdot 257 + 64 \cdot 64 + 65 \cdot 65 + 10 \cdot 10 = 756231$ floating point numbers. Note, $\mathbf{A}_k$ and $\mathbf{B}_k$ are symmetric matrices, so we only need to upload the upper triangular part of $\mathbf{A}_k$ and $\mathbf{B}_k$, which is around 756231/2 = 378115.5 floating point numbers and 1.442 MB. Therefore, our approach costs 2.272 MB for the directed communication, which is only 1.367 times than DiagonalFisher while DiagonalFisher costs 0.831*2 = 1.662 MB. We show the following Table 7 based on the previous experiment results. When "Initial" is set to "Same", the efficiency of every bit is almost the same. When "Initial" is set to "Different", the efficiency of every bit for our method is much higher than the DiagonalFisher.

C Detailed discussion for the time complexity of Eq. 12

A fully-connected neural network model with architecture 784-256-64-10 as an example is shown in Appendix K. We use this example to further explain this question. The size of $\mathbf{A}_{k_1}$ is 785x785 and the size of $\mathbf{B}_{k_1}$ are both 256x256. Then, we need to compute the inverse of the matrix (785x785)x(256x256), which is huge. The time complexity of calculating the inverse of a matrix is $O(n^3)$ (n is the dimension of the matrix), which is very slow. The accuracy of calculating it is decided by the condition number of the huge matrix [73, 74, 75, 76]. That's why calculating Eq. 12 is unacceptable, considering the time complexity, the size of the huge matrix and the accuracy.

Further, for example, in the machine learning field, to accelerate the training of the neural network, they use the Newton method. However, using this method, they need to compute the inverse of the Hessian matrix, which is also huge and unacceptable. That is why they introduce the KFAC [46, 49], KFRA [55] and KFLR [55] methods to avoid computing the inverse of the huge Hessian matrix.

In this paper, we avoid computing the inverse of the huge matrix via our method, and the time complexity is linear.

D Expectation approximation (EA)

Previous works [46, 49] approximate the expectation of Kronecker products by a Kronecker product of expectations $\mathbb{E}[\mathbf{A} \otimes \mathbf{B}] \approx \mathbb{E}[\mathbf{A}] \otimes \mathbb{E}[\mathbf{B}]$ with an assumption of $\mathbf{A}_{k_l}$ and $\mathbf{B}_{k_l}$ are independent, which is called Expectation Approximation (EA).

It is a simple and effective method to approximate the expectation of Kronecker products. As a result, the global co-variance $\bar{\mathbf{\Sigma}}$ is approximated by:

$$\bar{\mathbf{\Sigma}}_l \approx (\sum_k^K \mathbf{A}_{k_l})^{-1} \otimes (\sum_k^K \mathbf{B}_{k_l})^{-1} = \bar{\mathbf{A}}_l^{-1} \otimes \bar{\mathbf{B}}_l^{-1} \tag{15}$$

where $\bar{\mathbf{A}}_l = \sum_k^K \mathbf{A}_{k_l}$ and $\bar{\mathbf{B}}_l = \sum_k^K \mathbf{B}_{k_l}$. Denoting $\bar{\mathbf{Z}}_l$ as matrix formula of $\bar{\mathbf{z}}_l = \mathrm{vec}(\bar{\mathbf{Z}}_l)$, then $\bar{\boldsymbol{\mu}}_l$ can be computed efficiently as below:

$$\bar{\boldsymbol{\mu}}_l = \bar{\mathbf{\Sigma}}_l \cdot \bar{\mathbf{z}}_l \approx (\bar{\mathbf{A}}_l^{-1} \otimes \bar{\mathbf{B}}_l^{-1})\bar{\mathbf{z}}_l = \mathrm{vec}(\bar{\mathbf{B}}_l^{-1}\bar{\mathbf{Z}}_l\bar{\mathbf{A}}_l^{-1}) \tag{16}$$

However, Eq. 16 leads to a biased global expectation. The EA needs the independence assumption, but $\mathbf{A}_{k_l}$ and $\mathbf{B}_{k_l}$ are weakly related in back-propagation. Besides, even if they are independent, Eq. 16 still suffers from approximation error because the clients' number K is finite and always a small number but statistical independence can only be demonstrated when the sampling number is large enough. Eq. 17 shows the approximation error directly:

$$\begin{aligned}(\mathbf{A}_1 + \mathbf{A}_2) \otimes (\mathbf{B}_1 + \mathbf{B}_2) = &\mathbf{A}_1 \otimes \mathbf{B}_1 + \mathbf{A}_2 \otimes \mathbf{B}_2 \\ &+ \mathbf{A}_1 \otimes \mathbf{B}_2 + \mathbf{A}_2 \otimes \mathbf{B}_1 \\ \neq &\mathbf{A}_1 \otimes \mathbf{B}_1 + \mathbf{A}_2 \otimes \mathbf{B}_2\end{aligned} \tag{17}$$

E Extend FedLPA to the models with enormous single layer weight parameters

This implies a Fisher matrix with a large dimension and it significantly increases communication costs. In such cases, the most intuitive approach is to explore the possibility of dimensionality reduction for its Fisher matrix. A promising approach to enhance the efficiency of our method may employ some low-rank factorization techniques [77]. As described [44], the main idea involves performing an eigendecomposition on the Kronecker factors [78], while preserving only the eigenvectors corresponding to the top k largest eigenvalues. As a result, this approach drastically reduces space complexity, enabling communication costs to be compared favorably with diagonal Fisher matrices.

F Privacay discussion of FedLPA

In the FedLPA, $\mathbf{A}_k$ is computed via the activations while $\mathbf{B}_k$ is computed via the linear pre-activations of the layer. We note that $\mathbf{A}_k$, $\mathbf{B}_k$, and $\mathbf{M}_k$ do not carry any label information, thus the transmission of $\mathbf{A}_k$, $\mathbf{B}_k$, and $\mathbf{M}_k$ will not leak any label privacy. As a comparison, FedCAVE, which transmits client label information to the server, requires training in label distribution to do the distillation. Several papers [79, 80] have notified that label privacy, e.g., the concern of label distribution leakage and raw label leakage, is sensitive in federated learning. We believe that it has also been a concern in the one-shot FL scenario.

Besides, our t-SNE illustration in Fig 1 shows the classification capability on the global model, which can separate the classes. However, our figures of the t-SNE illustrations on local models in Appendix G.1 show that for the data belonging to the same class, their t-SNE illustrations are erratically distributed on different local nodes. For instance, for node 2, its training data only has 3 classes while most of the training data locates in class 5. However, it is hard for the server to infer that label distribution since the t-SNE illustration both on node 2 and other nodes also seems irregular.

$\mathbf{A}_k$, $\mathbf{B}_k$, and $\mathbf{M}_k$ are a function of data that may contain privacy-sensitive information of the local training data. However, in this case, our privacy-preserving level is similar to FedAvg, which means that FedLPA has the same privacy-preserving level as the conventional federated learning algorithms (i.e, FedAvg, FedProx, FedNova, and Dense), which are all vulnerable to some privacy attacks (e.g, membership inference [81] or reconstruction attacks [82]). Our approach FedLPA provides more information than FedAvg, However, the additional information we provide is the mean of each sample in each dimension, the mean of squares of each sample in each dimension, and the mean of square gradients. These solely marginally enrich the attack capability of several reconstruction attacks.

FedLPA is intuitively compatible with existing privacy-preserving techniques, such as differential privacy (DP) [56, 57], secure multiparty computation (SMC) [58, 59], and homomorphic encryption (HE) [60, 61, 62]. In Appendix F.1, we propose a naive DP-FedLPA with two different mechanisms to show the compatibility with differential privacy. In Appendix F.2, using iDLG attack [82], we show that our proposed FedLPA has the same privacy-preserving level as the conventional federated learning algorithms (i.e, FedAvg, FedProx, FedNova and Dense). Compared with FedAvg, we have conducted a detailed analysis from a privacy attack perspective to show that our proposed FedLAP exhibits a security level consistent with FedAvg against several types of privacy attacks, where the details are shown in Appendix F.3. Note that the main focus of FedLPA is to improve the learning performance on the one-shot FL settings, thus, we leave the integration with other privacy-preserving techniques beyond DP as an open problem.

F.1 Experiments with differential privacy

We first list the definitions and techniques for differential privacy [83]. (ϵ-DP) For $\epsilon > 0$, a randomized function f provides ϵ-differential privacy if, for any datasets D, D' that have only one single record different, for any possible output O,

$$Pr[f(D) \in O] \leq e^{\epsilon} \cdot Pr[f(D') \in O] \tag{18}$$

Suppose f is a function and D, D' have only one record different. The sensitivity of f is defined as

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\|_1 \tag{19}$$

Table 8: Experiments with Differential Privacy using two mechanisms.

ϵ	Partitions	FedAvg	DP-FedLPA (mechanism 1)	DP-FedLPA (mechanism 2)
8	β=0.1	31.90±0.58	50.01±0.07	57.15±1.23
	β=0.3	44.37±0.05	68.30±0.41	66.21±0.14
	β=0.5	57.92±0.63	71.17±0.27	73.50±0.06
5	β=0.1	28.17±0.16	48.51±0.07	55.87± 0.88
	β=0.3	43.91±0.05	67.34±0.92	66.02±0.71
	β=0.5	57.14±0.63	70.89±0.80	73.44±0.20
3	β=0.1	27.85±0.79	48.39±0.07	54.31±0.44
	β=0.3	42.80±0.05	65.08±0.45	65.22±0.46
	β=0.5	54.80±0.63	70.28±1.30	72.19±0.62

Table 9: Experiments with Differential Privacy for Round Numbers.

β ϵ	8	5	3
0.1	11	11	12
0.3	11	9	8
0.5	8	8	7

Here one record different means a database has one more record than another. We utilize the Laplace mechanism [84] to achieve the $\epsilon-$DP.

Laplace Mechanism: For function $f : \mathcal{D} \to R^d$, function:

$$F(D) = f(D) + Lap(0, \Delta f/\epsilon) \tag{20}$$

provides ϵ-DP, where $Lap(0, \Delta f/\epsilon)$ is sampled from Laplace distribution.

Following the differential privacy (DP) mechanisms [56, 85, 57, 86] to protect privacy, we conduct the two mechanisms of DP-FedLPA: (1) adding Laplace random noise to the training data samples, (2) adding Laplace random noise to the parameters to be transmitted. DP is a rigorous and popular privacy metric, which guarantees that the output does not change with a high probability even though an input data record changes. Specifically, since the sensitivity of the data sample distribution after the normalization is 1, we add Laplacian noises with $\lambda = \frac{1}{\epsilon}$. We set $\epsilon = \{3, 5, 8\}$ that provides modest privacy guarantees since normally $\epsilon \in (1, 10)$ is viewed as a suitable choice. We have added the experiments using the same experiment setting in the paper with five random seeds and 10 clients on the FMNIST dataset. Results are shown in Table 8. DP-FedLPA under both mechanisms outperforms FedAvg, which shows that it is compatible with combining our proposed FedLPA with DP to enhance privacy protection levels. Note that the smaller ϵ is, the larger noises we add. We find that when the ϵ gets smaller, the performance drops simultaneously, while the privacy protection level is increased.

Besides, we have added the experiments using the same experiment setting to show the round results of how many rounds DP-FedAvg needs to achieve the same test performance with the first mechanism. The results in Table 9 show that DP-FedAvg needs about 10 rounds of communication to achieve the same test performance, compared to our one-round FedLPA. Combined with our previous results in Table 5 and Table 8, our FedLPA could save the communication and computation overhead and combine with the DP method to mitigate the potential privacy leakage. Based on the above settings, DP-FedAvg needs at least 3x communication overhead and 5x computation overhead. While DP-FedAvg needs multiple rounds to get similar accuracy, DP-FedAvg maybe vulnerable to more privacy attack methods due to the multiple queries, such as curvature-based privacy attacks.

F.2 Experiments with iDLG attack

We also add experiments with iDLG attack [82] following the link (https://github.com/PatrickZH/Improved-Deep-Leakage-from-Gradients/blob/master/iDLG.py). We did the experiments with the setting of the paper [82]: in each single experiment, the client is trained with one random picked image in FMNIST, then we use the iDLG attack to recover the image based on the model from FedAvg and FedLPA. We randomly selected 500 training examples to collect 500 MSEs between the recovered and the original image. The larger the MSE is, the better

Table 10: iDLG attack results of FedLPA and FedAvg.

Percentile	12.5	25.0	37.5	50.0	62.5	75.0	87.5	100.0
FedLPA	0.60	1.00	1.10	1.39	2.75	50.71	40736.88	>=1e9
FedAvg	0.09	0.16	0.71	1.56	26.24	950.89	54307.91	>=1e9

the privacy-preserving level for the method. Due to the rebuttal limitation, we cannot show the figure for the cumulative distribution function considering the MSE of the iDLG attack. We provide the results in Table 10 to show MSE considering the percentile for these 500 experiments.

Based on the Table, we could see that from 12.5 to 50.0 percentile, regarding the privacy-preserving aspect, FedLPA behaves better than FedAvg on these samples. However, from 50.0 to 87.5 percentile, FedAvg behaves better than FedLPA on such samples. Thus, no clear evidence exists of which one performs better when referring to the privacy level. Considering the overall 500 data samples, we roughly concluded that FedLPA and FedAvg share a similar privacy level.

F.3 Concrete examples of privacy attack

For privacy attacks, we start by assuming the simplest scenario where each client has only one sample, and the model comprises a single layer, such as a multi-layer perceptron.

Let $\mathbf{y} = \mathbf{W} * \mathbf{x}$, ($\mathbf{x}$ is $n+1$ dimensional, with the last dimension being a unit value 1), $\mathbf{g} = Df(\mathbf{y})/D\mathbf{x}$ (where f is the loss function). In this case, $\mathbf{A} = \mathbf{x}\mathbf{x}^T, \mathbf{B} = \mathbf{g}\mathbf{g}^T$.

In this single-sample scenario, an attacker can directly obtain $\mathbf{x}$ from the last column of $\mathbf{A}$. With $\mathbf{x}$ and $\mathbf{W}$, the attacker can acquire the model's output. Furthermore, utilizing the Loss and $\mathbf{g}$, it's possible to get the label information.

FedAvg would also be vulnerable to a reconstruction attack in this scenario, allowing the attacker to obtain sample and label information.

When each client has two samples ($\mathbf{x}_1\mathbf{x}_2 \in Dataset$), then: $\mathbf{A} = 1/2 * \mathbf{x}_1 * \mathbf{x}_1^T + 1/2 * \mathbf{x}_2 * \mathbf{x}_2^T, \mathbf{B} = 1/2 * \mathbf{g}_1 * \mathbf{g}_1^T + 1/2 * \mathbf{g}_2 * \mathbf{g}_2^T$. The last column $\mathbf{c}$ of $\mathbf{A}$ equals $1/2\mathbf{x}_1 + 1/2\mathbf{x}_2$. The diagonal elements $\mathbf{d}$ of $\mathbf{A}$ equal $1/2\mathbf{x}_1^2 + 1/2\mathbf{x}_2^2$. In the case of these two samples, an attacker can utilize the information from $\mathbf{A}$ and $\mathbf{B}$ to get the two samples $\mathbf{x}_1$ and $\mathbf{x}_2$. Using the same methodology, they can also obtain $\mathbf{g}_1$ and $\mathbf{g}_2$. Consequently, the attacker can reverse-engineer the labels as well.

FedAvg could also potentially succumb to a reconstruction attack in this scenario, providing the attacker with sample and label information, although the obtained information might be more ambiguous.

When each client has three or more samples ($\mathbf{x} \in Dataset$), $\mathbf{A} = \mathbb{E}_{\mathbf{x} \in Dataset}(\mathbf{x} * \mathbf{x}^T), \mathbf{B} = \mathbb{E}_{\mathbf{x} \in Dataset}(\mathbf{g} * \mathbf{g}^T)$. In this situation, the last column $\mathbf{c}$ of $\mathbf{A}$, $\mathbf{c} = \mathbb{E}_{\mathbf{x} \in Dataset}(\mathbf{x})$ represents the average of the sample dataset, depicting the projection of the data distribution in the sample space on various coordinate axes. Furthermore, the diagonal elements of $\mathbf{A}(\mathbb{E}_{\mathbf{x} \in Dataset}(\mathbf{x} * \mathbf{x}^T))$ offer the attacker statistical information about this local dataset.

Generally, solely using the statistical information of these datasets cannot reconstruct the entire dataset. Similarly, it's not possible to obtain gradients for the output of each sample, thereby preventing the reconstruction of individual sample labels. The results obtained by using $\mathbf{c}$ and $\mathbf{W}$ to gather statistical label information are unreliable.

Additionally, for structures such as CNNs and RNNs/LSTMs, the difficulty of attacks increases due to weight sharing. For CNNs, since convolutional kernels only accept local samples as input, information in $\mathbf{A}$ encompasses statistical information from all localities of the samples. For RNNs/LSTMs, information in $\mathbf{A}$ includes statistics of each word vector in a sentence. These network structures make it possible for attackers to fail even in single-sample scenarios. For MLPs, the information contained in the intermediate layer $\mathbf{A}$ is almost equivalent to the information encoded in the parameters of the BN (Batch Normalization) layer. The mean output of the Batch Normalization (BN) layer is equivalent to the last column of $\mathbf{A}$, whereas the variances differ between the BN layer and $\mathbf{A}$'s diagonal but both contain statistical information related to squared values.

It's worth noting that the parameters acquired by the BN layer using the sliding-window average method are also frequently used during the computation of $\mathbf{A}$ and $\mathbf{B}$, as mentioned in the paper [54].

FedAvg provides model parameter values, the average of gradients, and BN layer parameters. Compared to FedAvg, the additional information we offer is actually limited to: the mean of each sample in each dimension, the mean of squares of each sample in each dimension, and the mean of square gradients. Utilizing this information, attacking becomes highly challenging when the number of samples exceeds three. Although we don't rule out the possibility of successful methods in practice due to the data's own correlations, the limitations are significant based on our analysis, and our security level is quite close to that of FedAvg.

We discuss two common attacks here. Inferring class representatives:

i) Model inversion attacks [87] exploit the confidence information provided by machine learning applications or services. Our method does not provide confidence information, nor does it compute the information required for it. Therefore, our method's defense level against these attacks aligns with FedAvg's defense level.

ii) Attacks using GANs to construct class representatives [88] utilize the client-uploaded model as a discriminator and its output as labels to train a generator to generate similar data. The additional statistical information we provide might be used to constrain the distribution of inputs for GANs, specifically their mean values. Since the statistical information of the dataset may contain some common features among samples, it might potentially aid in speeding up the convergence of training GANs but may not significantly enhance the accuracy of generated data after GAN optimization. It's worth noting that if the BN layer parameters uploaded by FedAvg could be used to constrain the statistical information of GANs' inputs, they would be equivalent to the information provided by our method.

Additionally, these attack methods against FedAvg only yield favorable results when class members are similar, meaning the dataset has clear common features that allow the constructed representatives to resemble the training data. When class members are dissimilar, these shared features tend to be confounded, rendering the constraints imposed by the sample mean ineffective, hence not enhancing the effectiveness of GANs attacks.

In summary, our method exhibits a security level consistent with FedAvg against these types of attacks. Even in cases where the BN layer is not required, our method's security is similar to that of FedAvg.

Membership inference attacks against aggregate statistics [87, 88] and Membership inference attacks against ML models [89, 90, 91, 92, 93, 94] aim to infer whether a sample belongs to the training dataset using appropriate prior distributions and statistical data. These attack methods impose specific requirements on the dataset. In such attack scenarios, whether the sample mean information our method can provide is exploitable by the attacker depends on whether this information can reveal the inherent distribution correlations within the dataset. However, for high-dimensional complex data, sample mean information often falls short in achieving this.

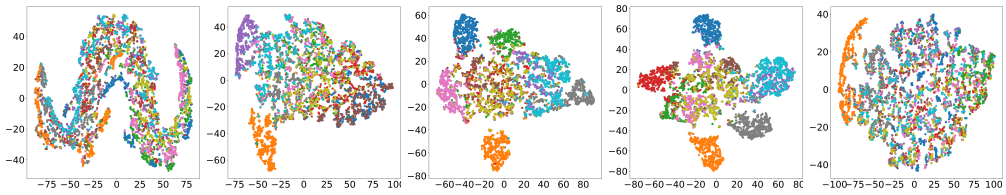
The inference attack towards client model is a complex topic. Other inference attack methods and defense mechanisms against them fall outside this paper's scope. It is an interesting topic to explore more robust measures to prevent such breaches in future works.

Therefore, in the case of these attacks we mentioned, our method exhibits the same level of security as FedAvg (since FedAvg requires uploading statistically equivalent information within the BN layer). For scenarios without a BN layer, whether our method reduces security depends on the characteristics of the dataset itself. Real-world data is often high-dimensional and complex, making successful attacks challenging.

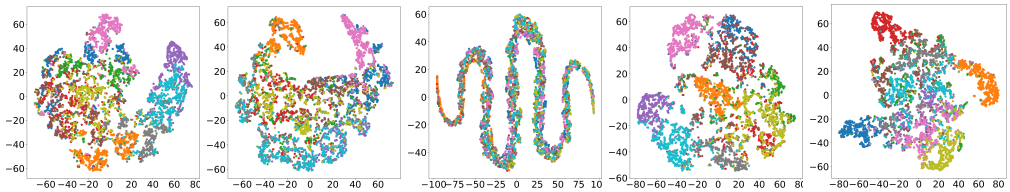
G Additional experiments

G.1 t-SNE visualization

We conduct experiments using MNIST dataset with a β value of 0.05, training 10 local clients over 200 local epochs with random seed 0. In this biased local dataset setting, local clients could only distinguish a subset of the classes, as illustrated in Figure 3.

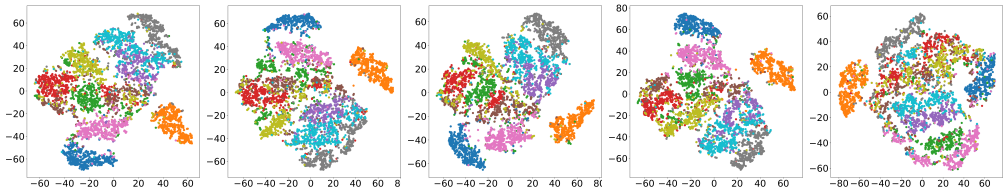


(a) local client #1 (b) local client #2 (c) local client #3 (d) local client #4 (e) local client #5



(f) local client #6 (g) local client #7 (h) local client #8 (i) local client #9 (j) local client #10

Figure 3: t-SNE visualizations of 10 local clients.



(a) FedAvg (b) FedNova (c) SCAFFOLD (d) FedProx (e) DENSE

Figure 4: t-SNE visualizations of the baseline approaches on the global model.

Based on seed 0, we partition the training data for the 10 local clients with the following form (label:# of the data) as:

local client #1: {4: 2, 5: 12, 6: 2847, 9: 16}
local client #2: {1: 20, 4: 189, 5: 5349}
local client #3: {0: 669, 1: 476, 2: 67, 6: 15, 7: 6068}
local client #4: {0: 266, 1: 375, 3: 3956, 7: 196, 9: 5932}
local client #5: {0: 4, 1: 418, 2: 5862}
local client #6: {1: 2, 2: 25, 4: 5195, 5: 24, 6: 80, 8: 28}
local client #7: {1: 5034, 2: 3, 4: 22, 5: 6, 6: 2669}
local client #8: {0: 4914}
local client #9: {4: 433, 5: 29, 6: 307, 8: 5373}
local client #10: {0: 70, 1: 417, 2: 1, 3: 2175, 4: 1, 5: 1, 7: 1, 8: 450, 9: 1}

It is worth noting that local client #2 has the training data mostly with label number 5, and as the corresponding t-SNE visualization shows in Figure 3b, the local train model could mainly cluster the data with label 5 (marked as purple). As data for label 1 (marked as orange) is different from other data with all other labels, some local clients may be able to cluster the data with label 1 with good results. Other local clients, such as local client #3, #4, #6, #7, #9, #10, show the similar results like local client #2.

Figure 4 displays the t-SNE visualization for the global models of FedAvg, FedNova, SCAFFOLD, FedProx, and DENSE using the training data, with the figure legends identical to those in Figure 1. It's evident from Figure 1 that FedLPA outperforms the baselines in classifying the ten classes.

FedLPA's superiority is not only demonstrated by its ability to cluster the ten classes but also by the distinct separation between classes, as observed in Figure 1, compared to the baselines.

G.2 Experiments on different local epoch numbers

Table 11: Comparison with various FL algorithms in one round with 10 local epochs settings.

Dataset	Partition	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	DENSE
FMNIST	β=0.01	24.47±1.02	11.43±0.04	13.37±0.11	11.83±0.03	12.30±0.10	10.00±0.00
	β=0.05	30.77±0.94	15.67±0.28	20.07±0.46	19.93±0.30	20.07±0.37	23.60±0.14
	β=0.1	42.83±0.33	25.10±1.32	21.03±1.08	22.57±1.08	22.20±1.17	34.83±0.16
	β=0.3	61.43±0.17	40.90±0.05	40.70±0.09	38.20±0.10	37.50±0.03	43.17±0.05
	β=0.5	67.63±0.36	52.43±0.60	51.77±0.46	54.67±0.73	54.33±0.64	54.30±0.07
	β=1.0	71.90±0.09	51.03±0.62	52.30±0.53	51.50±0.62	50.90±0.57	52.30±0.15
	#C=1	13.03±0.04	10.90±0.02	11.27±0.03	10.90±0.02	11.43±0.04	10.00±0.00
	#C=2	28.93±0.74	16.93±0.19	22.07±0.01	23.83±0.20	23.33±0.03	22.60±0.88
	#C=3	37.73±0.09	22.20±0.23	26.60±0.03	23.67±0.27	22.80±0.40	23.03±0.86
CIFAR-10	β=0.01	15.80±0.00	10.07±0.00	12.13±0.09	11.90±0.07	11.93±0.07	10.00±0.00
	β=0.05	20.23±0.01	10.90±0.02	10.00±0.00	10.00±0.00	10.00±0.00	13.33±0.04
	β=0.1	20.20±0.07	10.27±0.00	10.93±0.02	10.37±0.00	10.27±0.00	14.77±0.09
	β=0.3	25.60±0.01	18.13±0.33	14.97±0.05	14.77±0.05	15.67±0.03	20.33±0.06
	β=0.5	25.60±0.08	14.87±0.05	16.77±0.01	15.73±0.01	13.93±0.08	23.20±0.16
	β=1.0	28.93±0.01	15.63±0.03	19.10±0.14	15.30±0.05	15.43±0.05	22.30±0.46
	#C=1	11.00±0.01	10.30±0.00	10.23±0.00	10.30±0.00	10.33±0.00	10.00±0.00
	#C=2	20.40±0.03	11.37±0.04	11.67±0.06	11.00±0.02	11.80±0.06	10.40±0.00
	#C=3	22.30±0.03	12.23±0.04	14.37±0.13	14.10±0.14	14.00±0.12	18.50±0.14
MNIST	β=0.01	32.20±0.50	9.53±0.00	9.37±0.00	9.00±0.01	9.40±0.00	9.53±0.00
	β=0.05	60.60±0.07	20.80±0.13	35.17±0.66	35.10±0.87	34.13±0.91	50.37±1.57
	β=0.1	78.07±0.09	45.07±0.37	43.23±0.10	43.83±0.13	44.27±0.21	65.53±0.85
	β=0.3	85.60±0.17	64.40±0.24	64.03±0.11	64.17±0.09	64.07±0.11	75.53±0.22
	β=0.5	91.77±0.00	79.43±0.13	77.37±0.22	78.17±0.25	77.90±0.30	87.93±0.17
	β=1.0	94.70±0.00	85.00±0.10	85.10±0.06	84.40±0.08	84.63±0.08	89.30±0.03
	#C=1	11.87±0.00	10.43±0.02	10.13±0.01	10.13±0.01	10.13±0.01	9.93±0.00
	#C=2	47.93±0.89	13.20±0.09	16.47±0.21	12.97±0.16	12.23±0.07	32.57±0.26
	#C=3	65.97±0.98	26.70±2.24	31.67±2.60	31.63±3.03	31.20±3.24	53.80±0.09
SVHN	β=0.01	17.00±0.03	13.93±0.16	16.57±0.15	16.27±0.22	13.30±0.20	13.97±0.17
	β=0.05	20.23±0.05	15.40±0.11	15.53±0.12	15.53±0.12	15.53±0.12	17.90±1.01
	β=0.1	32.57±0.53	15.17±0.18	18.37±0.03	18.37±0.03	18.33±0.03	24.20±0.28
	β=0.3	35.47±0.54	18.23±0.29	20.77±0.02	21.63±0.03	21.17±0.01	29.23±0.03
	β=0.5	41.17±0.01	26.07±0.13	27.40±0.00	26.27±0.00	27.80±0.00	36.80±0.13
	β=1.0	44.33±0.01	30.77±0.01	32.27±0.01	30.43±0.03	31.97±0.00	29.47±2.86
	#C=1	19.60±0.00	10.10±0.03	16.60±0.11	16.77±0.13	15.53±0.12	8.90±0.02
	#C=2	31.20±0.01	11.80±0.33	15.77±0.29	15.67±0.31	15.60±0.32	14.00±0.24
	#C=3	34.43±0.40	8.93±0.01	22.03±0.05	18.03±0.12	17.50±0.17	23.57±0.90

Table 12: Comparison with various FL algorithms in one round with 20 local epochs settings.

Dataset	Partition	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	DENSE
FMNIST	β=0.01	24.17±1.13	11.90±0.07	15.33±0.15	13.40±0.09	10.57±0.00	12.67±0.14
	β=0.05	36.97±0.78	18.83±0.55	19.93±0.11	19.37±0.14	19.70±0.17	33.13±0.47
	β=0.1	41.83±0.02	28.13±1.34	23.00±0.41	24.63±0.75	24.10±1.29	36.40±0.02
	β=0.3	60.83±0.38	42.50±0.12	42.83±0.01	40.47±0.22	40.63±0.11	40.67±0.04
	β=0.5	67.80±0.25	53.17±0.25	55.23±0.67	53.27±0.34	54.20±0.52	64.60±0.45
	β=1.0	75.47±0.03	55.47±0.57	54.53±0.52	53.57±0.46	54.73±0.41	70.97±0.02
	#C=1	14.07±0.02	10.43±0.00	11.03±0.02	10.43±0.00	11.13±0.03	10.00±0.00
	#C=2	29.67±0.37	16.83±0.21	22.67±0.27	23.27±0.24	25.43±0.17	24.77±0.14
	#C=3	34.37±0.79	24.93±0.02	26.17±0.01	27.70±0.02	27.70±0.03	29.43±0.22
CIFAR-10	β=0.01	15.13±0.01	10.10±0.00	12.50±0.11	11.90±0.07	11.83±0.07	10.50±0.00
	β=0.05	23.37±0.01	11.33±0.04	10.20±0.00	10.00±0.00	10.77±0.01	14.67±0.02
	β=0.1	25.07±0.00	11.60±0.05	12.33±0.10	12.67±0.14	13.23±0.21	18.50±0.06
	β=0.3	26.00±0.02	16.63±0.13	13.10±0.01	13.87±0.01	14.43±0.02	24.97±1.13
	β=0.5	30.60±0.00	14.20±0.02	13.30±0.04	13.13±0.03	14.23±0.04	27.60±0.06
	β=1.0	26.77±0.10	17.60±0.01	17.50±0.05	18.13±0.04	18.20±0.01	26.07±0.18
	#C=1	10.67±0.01	10.20±0.00	10.23±0.00	10.20±0.00	10.27±0.00	10.00±0.00
	#C=2	22.00±0.00	12.03±0.08	10.23±0.00	11.10±0.02	11.40±0.04	15.33±0.14
	#C=3	23.60±0.05	11.97±0.03	14.93±0.17	13.37±0.09	13.63±0.09	21.17±0.05
MNIST	β=0.01	32.43±0.86	11.00±0.06	9.30±0.00	9.37±0.00	10.33±0.02	12.40±0.19
	β=0.05	68.73±0.45	26.73±0.24	37.77±0.68	37.70±0.84	36.57±0.75	62.03±0.54
	β=0.1	71.77±0.20	48.57±0.60	45.67±0.22	46.63±0.23	45.83±0.23	66.93±0.25
	β=0.3	90.83±0.01	68.17±0.33	66.90±0.03	66.60±0.15	66.03±0.21	85.37±0.11
	β=0.5	89.43±0.09	80.90±0.14	76.63±0.13	79.57±0.22	79.47±0.27	86.07±0.22
	1.0	96.17±0.01	86.60±0.13	85.90±0.14	86.03±0.14	86.57±0.15	88.40±0.00
	#C=1	11.47±0.01	10.27±0.01	10.13±0.01	10.13±0.01	10.13±0.01	9.87±0.00
	#C=2	53.37±0.61	17.47±0.48	20.70±0.58	14.77±0.14	13.47±0.02	43.33±0.10
	#C=3	72.27±0.44	28.63±1.61	32.93±2.76	31.40±2.01	30.97±2.50	44.30±0.62
SVHN	β=0.01	19.03±0.00	14.83±0.13	9.33±0.02	9.30±0.02	9.30±0.02	18.23±0.03
	β=0.05	26.27±0.19	13.37±0.04	15.53±0.12	15.53±0.12	15.57±0.12	24.63±0.41
	β=0.1	28.8±0.70	17.47±0.05	19.33±0.00	19.30±0.01	19.70±0.06	26.63±0.42
	β=0.3	45.03±0.17	27.83±0.04	26.53±0.05	26.90±0.00	27.57±0.01	38.27±4.74
	β=0.5	48.00±0.21	30.80±0.19	32.20±0.33	30.07±0.21	30.97±0.14	43.33±0.08
	β=1.0	62.23±0.05	49.07±0.25	47.83±0.32	48.03±0.25	48.53±0.10	60.03±0.55
	#C=1	16.23±0.23	9.83±0.03	17.07±0.11	16.60±0.12	15.50±0.12	7.70±0.03
	#C=2	27.87±0.49	11.83±0.32	20.57±0.02	19.17±0.01	16.00±0.27	18.53±0.57
	#C=3	42.97±0.02	14.10±0.12	23.70±0.20	25.80±0.39	23.37±0.00	36.73±0.07

Table 13: Comparison with various FL algorithms in one round with 50 local epochs settings.

Dataset	Partition	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	DENSE
FMNIST	β=0.01	19.33±0.43	10.13±0.00	15.87±0.16	18.53±0.35	12.97±0.15	10.70±0.01
	β=0.05	32.70±0.40	19.47±0.53	24.10±0.02	23.93±0.11	22.63±0.20	31.33±1.34
	β=0.1	40.00±0.01	30.40±1.05	27.37±0.23	25.83±0.35	25.50±0.72	39.93±1.10
	β=0.3	62.80±0.41	43.67±0.01	42.50±0.09	41.50±0.10	42.23±0.11	57.80±0.04
	β=0.5	68.27±0.00	55.97±0.23	55.27±0.38	53.95±0.27	55.00±0.26	63.50±0.11
	β=1.0	73.47±0.27	61.20±0.09	60.67±0.19	60.77±0.12	61.40±0.15	66.03±0.06
	#C=1	13.30±0.03	10.50±0.00	11.03±0.02	10.50±0.00	11.87±0.07	10.00±0.00
	#C=2	27.60±0.02	16.37±0.08	23.00±0.12	18.37±0.38	18.60±0.32	29.33±0.44
	#C=3	38.13±0.01	25.47±0.02	25.23±0.29	26.70±0.05	26.40±0.18	37.53±0.17
CIFAR-10	β=0.01	16.23±0.01	10.23±0.00	12.27±0.07	13.07±0.08	12.17±0.09	10.33±0.00
	β=0.05	17.93±0.06	11.00±0.02	10.33±0.00	10.13±0.00	11.20±0.03	7.63±0.01
	β=0.1	19.20±0.02	13.17±0.09	13.63±0.21	12.00±0.03	12.43±0.06	19.13±0.09
	β=0.3	27.57±0.03	12.53±0.05	12.33±0.02	11.93±0.01	12.90±0.01	26.03±0.52
	β=0.5	27.57±0.29	13.47±0.03	12.30±0.00	12.47±0.02	13.47±0.02	26.40±0.23
	β=1.0	30.27±0.02	15.47±0.12	15.30±0.14	15.23±0.13	15.53±0.10	29.17±2.06
	#C=1	10.90±0.00	10.30±0.00	10.30±0.00	10.30±0.00	10.33±0.00	10.00±0.00
	#C=2	21.17±0.06	10.13±0.00	11.93±0.02	10.57±0.01	11.27±0.01	15.87±0.08
	#C=3	23.80±0.01	12.00±0.06	12.07±0.02	12.97±0.04	11.90±0.02	21.53±0.29
MNIST	β=0.01	34.10±0.88	10.57±0.02	9.50±0.00	9.33±0.02	10.13±0.01	12.53±0.19
	β=0.05	66.23±0.32	32.00±0.78	39.70±0.50	39.60±0.31	39.87±0.15	56.63±0.65
	β=0.1	72.90±0.27	49.17±0.62	47.20±0.22	47.07±0.23	46.30±0.10	69.93±0.27
	β=0.3	87.03±0.02	68.30±0.33	66.40±0.16	67.10±0.09	66.17±0.22	82.47±0.01
	β=0.5	90.43±0.07	80.70±0.13	78.13±0.19	79.37±0.19	79.50±0.22	88.30±0.05
	β=1.0	94.47±0.04	86.73±0.15	85.43±0.11	86.07±0.12	86.20±0.16	89.23±0.01
	#C=1	11.37±0.01	10.17±0.01	10.13±0.01	10.13±0.01	10.10±0.01	9.80±0.00
	#C=2	71.07±0.02	23.53±1.00	22.93±0.80	22.63±0.99	17.53±0.12	42.97±0.17
	#C=3	76.17±0.38	29.60±1.80	33.50±2.91	32.77±2.27	23.40±3.38	57.30±0.12
SVHN	β=0.01	19.60±0.00	13.93±0.16	13.57±0.19	9.50±0.03	9.27±0.02	19.10±0.52
	β=0.05	22.97±0.01	14.87±0.12	15.83±0.13	15.67±0.12	14.67±0.14	19.97±0.54
	β=0.1	45.83±1.70	22.40±0.21	22.97±0.12	22.47±0.01	24.30±0.04	41.47±4.63
	β=0.3	36.30±2.02	33.90±0.15	33.87±0.20	33.50±0.13	34.43±0.21	29.90±0.56
	β=0.5	51.77±0.01	39.70±0.19	39.93±0.13	38.03±0.14	38.33±0.12	50.10±1.71
	β=1.0	57.97±0.10	56.70±0.15	54.03±0.28	55.33±0.23	55.80±0.15	47.80+8.91
	#C=1	19.37±0.00	9.90±0.03	16.57±0.12	16.53±0.12	15.53±0.12	10.00±0.00
	#C=2	36.93±0.02	12.53±0.25	20.30±0.07	20.70±0.12	15.57±0.22	40.77±2.21
	#C=3	42.43±0.02	21.07±0.31	29.63±0.16	27.10±0.06	24.73±0.00	38.50±0.60

Table 14: Comparison with various FL algorithms in one round with 100 local epochs settings.

Dataset	Partition	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	DENSE
FMNIST	β=0.01	19.17±0.01	11.73±0.06	16.10±0.18	19.00±0.27	12.67±0.12	10.07±0.00
	β=0.05	36.77±0.51	18.07±0.35	22.67±0.03	22.73±0.04	21.20±0.17	31.77±1.14
	β=0.1	35.90±0.12	32.83±0.58	29.87±0.49	30.80±0.28	29.33±0.60	33.23±1.22
	β=0.3	64.07±0.28	47.77±0.07	42.20±0.01	43.33±0.06	46.03±0.06	60.30±0.17
	β=0.5	68.73±0.10	57.03±0.20	55.87±0.38	56.10±0.31	58.60±0.43	64.60±0.01
	β=1.0	76.27±0.00	65.00±0.05	61.67±0.35	65.13±0.14	65.03±0.14	75.80±0.05
	#C=1	13.37±0.04	10.87±0.02	10.47±0.00	10.87±0.02	13.23±0.21	10.00±0.00
	#C=2	31.40±1.16	20.93±0.23	24.97±0.19	23.13±0.25	21.50±0.29	26.30±1.56
	#C=3	49.73±0.24	26.97±0.00	25.57±0.27	26.17±0.22	25.50±0.12	46.87±0.10
CIFAR-10	β=0.01	16.93±0.01	10.33±0.00	10.97±0.02	9.57±0.41	11.10±0.02	11.23±0.01
	β=0.05	19.07±0.01	12.33±0.11	12.50±0.12	10.33±0.00	12.60±0.13	18.63±0.11
	β=0.1	20.80±0.08	12.53±0.05	10.33±0.00	10.67±0.00	11.87±0.03	24.30±0.05
	β=0.3	28.33±0.00	11.63±0.02	11.03±0.01	11.07±0.00	11.70±0.01	28.23±0.36
	β=0.5	29.37±0.01	12.07±0.01	12.13±0.01	11.80±0.01	13.17±0.01	28.90±0.49
	β=1.0	30.57±0.00	14.53±0.09	13.93±0.01	13.97±0.10	15.93±0.11	29.37±1.73
	#C=1	11.03±0.02	10.23±0.00	10.23±0.00	10.23±0.00	10.57±0.01	10.00±0.00
	#C=2	16.70±0.13	10.00±0.00	12.90±0.03	11.00±0.01	11.97±0.03	13.67±0.03
	#C=3	18.87±0.01	11.33±0.03	10.70±0.00	11.77±0.02	11.67±0.02	15.97±0.10
MNIST	β=0.01	34.10±0.66	13.60±0.32	9.33±0.00	9.30±0.00	9.30±0.00	16.63±0.33
	β=0.05	72.47±0.07	32.30±0.66	41.37±0.37	38.57±0.35	40.70±0.49	55.30±1.88
	β=0.1	78.53±0.20	48.20±0.39	47.87±0.26	47.57±0.19	46.93±0.04	76.47±0.20
	β=0.3	85.83±0.04	68.77±0.28	67.43±0.11	67.13±0.12	65.67±0.36	84.23±0.08
	β=0.5	89.03±0.12	80.53±0.19	79.13±0.23	79.00±0.28	79.50±0.30	88.30±0.31
	β=1.0	94.13±0.03	86.53±0.09	85.87±0.09	85.63±0.08	86.17±0.14	92.57±0.02
	#C=1	11.27±0.01	10.30±0.02	10.10±0.01	10.10±0.01	10.13±0.01	9.93±0.00
	#C=2	71.07±0.35	21.00±0.61	22.47±0.89	18.83±0.55	14.50±0.12	45.47±0.14
	#C=3	76.83±0.32	29.63±2.43	35.17±2.54	32.47±3.15	29.2±2.22	67.33±0.95
SVHN	β=0.01	19.50±0.00	13.90±0.16	9.37±0.02	12.57±0.22	11.60±0.09	19.10±0.13
	β=0.05	32.90±0.05	13.50±0.03	16.03±0.15	15.90±0.18	16.83±0.10	25.80±1.64
	β=0.1	36.63±0.27	22.37±0.62	24.17±0.20	24.83±0.07	25.93±0.09	26.97±0.23
	β=0.3	56.40±0.01	35.43±0.10	34.40±0.05	35.17±0.10	34.40±0.07	55.67±1.85
	β=0.5	55.63±0.16	39.07±0.03	40.33±0.05	37.47±0.01	37.07±0.12	55.53±0.62
	β=1.0	65.57±0.01	55.87±0.27	55.30±0.26	54.80±0.19	54.17±0.34	62.50±0.12
	#C=1	16.27±0.22	10.33±0.00	13.83±0.17	15.67±0.11	15.63±0.11	12.10±0.30
	#C=2	41.87±0.01	14.80±0.12	22.53±0.07	20.77±0.06	13.87±0.86	41.43±1.77
	#C=3	48.70±0.02	23.50±0.04	30.20±0.08	29.20±0.10	25.30±0.02	48.60±0.49

Here, we present experiments similar to those in Table 1 but with different numbers of epochs (10, 20, 50, 100). The performance of our methods outperforms other approaches, as shown in Table 11, Table 12, Table 13, and Table 14. Without tuning the number of local epochs, our method consistently achieves high performance compared to other baselines. In almost all the settings, our method can outperform the state-of-the-art baseline approach DENSE. We also note that DENSE

consumes more computing resources, as shown in Table 5. Besides, it needs an extra data generation stage and an extra model distillation stage. Our method could get better results and consume fewer resources. What's more, in Section 4.7, we also show that our method has the potential to extend to multiple-round settings, while it is hard to extend the DENSE into multi-round settings.

G.3 Extreme setting, 5 clients

Table 15: Comparison with various FL algorithms in one round when client number is 5.

Dataset	Partition	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	DENSE
FMNIST	β=0.01	48.13±0.28	26.03±0.07	30.77±0.49	30.80±0.34	17.83±0.07	44.23±0.14
	β=0.05	55.20±0.17	23.40±0.16	30.80±0.67	29.90±0.12	20.43±0.16	46.17±0.09
	β=0.1	59.27±0.12	33.47±0.16	37.77±0.45	35.43±0.86	32.57±0.98	58.73±0.15
	β=0.3	73.13±0.00	53.13±0.42	52.57±0.46	52.03±0.59	49.90±0.33	63.40±0.06
	β=0.5	74.17±0.02	60.27±0.53	60.13±0.57	59.97±1.14	61.67±0.35	72.03±0.05
	β=1.0	75.30±0.00	63.00±0.05	60.87±0.24	62.63±0.05	60.37±0.01	74.93±0.04

When the number of clients is set to 5, the experimental results for the FMNIST dataset are shown in Table 15. These results demonstrate that our framework performs well even in extreme situations when the number of clients is relatively small.

G.4 Extreme setting, $\beta = 0.001$

Table 16: Comparison with various FL algorithms in one round with different epoch numbers and $\beta = 0.001$.

Dataset	epochs number	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	DENSE
FMNIST	10	14.57±0.04	10.60±0.01	10.53±0.01	10.60±0.01	13.10±0.01	10.00±0.01
	20	15.33±0.04	10.13±0.00	10.23±0.00	10.13±0.00	12.87±0.16	10.00±0.00
	50	13.77±0.02	10.57±0.01	10.17±0.00	10.57±0.01	12.30±0.11	10.00±0.00
	100	15.83±0.03	10.17±0.00	10.73±0.01	10.17±0.00	13.23±0.21	10.00±0.00
	200	14.53±0.00	10.07±0.00	10.10±0.00	10.07±0.00	12.50±0.12	10.00±0.00
CIFAR-10	10	11.50±0.00	10.27±0.00	10.17±0.00	10.27±0.00	10.33±0.00	10.00±0.00
	20	10.57±0.01	10.27±0.00	10.13±0.00	10.27±0.00	10.30±0.00	10.00±0.00
	50	10.77±0.01	10.23±0.00	10.33±0.00	10.23±0.00	10.33±0.00	10.00±0.00
	100	10.90±0.01	10.20±0.00	10.30±0.00	10.23±0.00	10.57±0.01	10.00±0.00
	200	10.87±0.02	10.27±0.00	10.23±0.00	10.27±0.00	10.37±0.01	10.00±0.00
MNIST	10	24.10±0.17	10.07±0.01	12.17±0.07	11.83±0.05	12.17±0.12	9.90±0.00
	20	19.53±0.33	10.07±0.01	12.07±0.07	13.37±0.08	12.37±0.12	9.27±0.00
	50	16.93±0.37	10.07±0.01	10.80±0.04	13.17±0.09	13.13±0.25	11.40±0.08
	100	19.07±0.41	10.13±0.01	10.97±0.00	11.37±0.02	12.90±0.13	12.83±0.17
	200	15.63±0.03	10.07±0.01	11.13±0.06	12.50±0.04	11.83±0.11	9.27±0.00
SVHN	10	17.50±0.02	15.90±0.00	15.53±0.12	15.53±0.12	15.53±0.12	17.13±0.03
	20	20.10±0.21	15.90±0.00	15.53±0.12	15.53±0.12	14.00±0.11	17.13±0.03
	50	20.07±0.71	16.30±0.00	15.50±0.12	15.13±0.16	14.03±0.07	15.17±0.16
	100	19.70±0.00	15.90±0.00	15.10±0.16	15.53±0.12	13.77±0.10	18.47±0.05
	200	19.13±0.00	13.90±0.11	14.90±0.19	15.13±0.16	13.27±0.06	15.23±0.16

Here, we demonstrate that even when $\beta = 0.001$ and with different dataset and local epoch number settings, FedLPA has the potential to aggregate models effectively in extreme situations and produce superior results. These results are presented in Table 16.

G.5 Experiments with FedOV and Co-Boosting

We compare with FedOV[3], the state-of-the-art method which addresses label skews in one-shot federated learning. We run the experiments with fair comparison (same model size) on MNIST dataset with #C=2 partition setting. Table 17 shows that our method could be comparable with FedOV in some scenarios even when FedOV transmits the unknown label information to the clients and utilizes the knowledge distillation. As the epoch number of local clients equals to 50,100,200, FedLPA outperforms FedOV.

We also compare with Co-Boosting[4], the state-of-the-art distillation method. We run the experiments with fair comparison (same model size) on the FMNIST dataset, and the rest of the settings are the

[3]https://github.com/Xtra-Computing/FedOV

[4]https://github.com/rong-dai/Co-Boosting

Table 17: Comparison with FedOV on MNIST with #C=2.

epoch number	10	20	50	100	200
FedLPA	47.93±0.89	53.37±0.61	71.07±0.02	71.07±0.35	69.63±0.29
FedOV	71.0±0.25	70.27±0.39	69.23±0.31	65.83±0.23	64.50±0.38

Table 18: Comparison with Co-Boosting on FMNIST.

β	0.01	0.05	0.1	0.3	0.5
FedLPA	21.20±0.67	54.27±0.38	55.33±0.06	68.20±0.04	73.33±0.06
Co-Boosting	17.31±0.24	48.97±1.44	73.15±1.86	83.37±0.44	86.21±0.31

same as the default. The results in Table 18 show that when the β is smaller than 0.1, our method outperforms Co-Boosting. Thus, with the increment of skewness, FedLPA shows significantly superior results.

In conclusion, our method could be comparable with FedOV and Co-Boosting in some settings, even when they consume more computational resources as shown in Appendix G.7.

G.6 Communication overhead evaluation

Table 5 shows the communication overhead evaluation of a simple CNN with 5 layers on CIFAR-10 dataset. The results are given based on the experiments. In this section, we will give a concrete example to show the details.

The communication bits are the number of bits that are transmitted between a server and a client in a directed communication. It reflects the communication efficiency of federated learning algorithms. Better algorithms should have lower communication bits. The default floating point precision is 32 bits in Pytorch.

A fully-connected neural network model example: We use a fully-connected neural network model with architecture 784-256-64-10 as an example to show the calculation, which has $784 \cdot 256 + 256 + 256 \cdot 64 + 64 + 64 \cdot 10 + 10 = 217930$ floating point numbers, which is 6973760 bits or around 0.831 MB.

For a single directed communication from a client to the server or vice versa, the cost for FedAvg, FedProx, FedNova, and DENSE is 0.831 MB each. SCAFFOLD costs 1.662 MB for the same communication, which is double the amount of the others.

For a single communication from a client to the server, our method requires additional upload of $\mathbf{A}_k$ and $\mathbf{B}_k$, which contain $785 \cdot 785 + 256 \cdot 256 + 257 \cdot 257 + 64 \cdot 64 + 65 \cdot 65 + 10 \cdot 10 = 756231$ floating point numbers in total. Note, as $\mathbf{A}_k$ and $\mathbf{B}_k$ are symmetric matrices, we only need to upload the upper triangular part of them, reducing the total to roughly $756231/2 = 378115.5$ floating point numbers as about 1.442 MB. Therefore, our approach costs 2.272 MB for the one directed communication, which is 2.734 times as FedAvg, FedProx, and DENSE, and 1.367 times as SCAFFOLD.

A CNN model example: We use another example using CNN to show the communication overhead. For example, we have one model, the first layer is nn.Conv2d(1, 6, 5), means there are 3 input channels, 6 output channels, and a 5x5 kernel size; the second layer is nn.Conv2d(6, 8, 5), means there are 6 input channels, 8 output channels, and a 5x5 kernel size.

The parameter count for the first layer is 1x6x5x5+6=156. Note that $\mathbf{A}$ and $\mathbf{B}$ are both symmetric matrices. Thus, the additional parameters for $\mathbf{A}$ and $\mathbf{B}$ for each kernel would be 5x5, and estimating the covariance for biases without decomposition results in a size of 6x6. Therefore, the additional parameters for this layer are 201 ($\mathbf{A}_{k_1}$= 6x((5x5-5)/2+5), $\mathbf{B}_{k_1}$=6x((5x5-5)/2+5)+(6x6-6)/2+6.

The parameter count for the second layer is 8x6x5x5+8=1208. Therefore, the additional parameters for this layer are 1476 ($\mathbf{A}_{k_2}$= 6x8x((5x5-5)/2+5),$\mathbf{B}_{k_2}$=6x8x((5x5-5)/2+5)+(8x8-8)/2+8).

These two examples all follow the theory: the communicated parameters $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{M}$ are approximately 2x of the number of all parameters in the model $\boldsymbol{\theta}$.

Table 19: Running time and computation overhead evaluation.

FedLPA	FedNova/SCAFFOLD/FedAvg	FedProx	DENSE	FedOV	Co-Boosting
65mins	50mins	75mins	400mins	150mins	700mins

Table 20: Experiments with ResNet-18.

β	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	Dense
0.1	23.62±0.51	12.16±0.23	10.07±0.04	13.87±0.26	12.04±0.16	21.45±0.60
0.3	27.43±0.04	11.75±0.09	10.08±0.32	10.01±0.07	12.97±0.17	27.10±0.25
0.5	31.70±0.14	13.81±0.31	12.75±0.11	10.61±0.21	11.45±0.23	29.04±0.30

The communication overhead will increase linearly using FedLPA when we change the client numbers to 20 and 50.

However, as Figure 2 demonstrates, to achieve the same performance as FedLPA, FedAvg, FedNova, SCAFFOLD, and FedProx require more communication rounds, resulting in a heavier data transfer burden on the system.

G.7 Running time and computation overhead evaluation

The running times of different algorithms, using a simple CNN on the CIFAR-10 dataset, are summarized in Table 19. In this experiment, there are 10 clients, each running 200 local epochs with only one communication round. Our device is a single 2080Ti GPU. Compared to the state-of-the-art methods FedOV, DENSE and Co-Boosting, our method is efficient and slightly slower than the fastest algorithm. Notably, DENSE consumes almost 7 times the computational resources, as the knowledge distillation method is computationally intensive and resource-demanding. Co-Boosting even uses more time. It's important to note that while our method is efficient, it also yields almost always the best results. In our paper, we mainly adopt the most-cited non-IID FL benchmark (https://github.com/Xtra-Computing/NIID-Bench) to get a fair comparison of FedLPA and other baselines. The reason why the computation cost of FedProx is higher than FedAvg may be that the FedProx adds a l_2 regular term to make local updates around the global mode, which adds more computing overhead. Besides, using the original codebase (https://github.com/litian96/FedProx) from FedPorx also consumes more time than FedAvg and FedNova, under the above non-IID FL benchmark.

Our methods guarantee that the computation result overhead will increase almost linearly using FedLPA when we change the client numbers to 20 and 50.

G.8 Experiments with more complex neural network structure

We do the experiments with FedLPA on the same experiment setting in the paper using the more complex network, ResNet-18 [95], with five random seeds on the CIFAR-10 dataset. We set the parameters with β=0.1, 0.3 and 0.5 with 10 clients. The results are shown in Table 20. From the results, we could see that using ResNet-18, our method still gets better performance compared to other baselines.

We do the experiments with FedLPA on the same experiment setting in the paper using the more complex network, VGG-9 [96], with five random seeds on the FMNIST dataset. We set the parameters with β=0.1, 0.3 and 0.5 with 10 clients. The results are shown in Table 21. From the results, we could see that using VGG-9, our method still performs better than other baselines.

Based on the results of ResNet-18 and VGG-9, Our method has the potential to be applied to more complex models.

G.9 Experiments with more complex datasets

We do the experiments with FedLPA on the same experiment setting in the paper using ResNet-18 with five random seeds on the CIFAR-100 [65] dataset. The results are shown in Table 22. We can see that even with the complicated dataset CIFAR-100, our method could also get satisfactory results in the federated one-shot setting. Besides, we also have added the experiments on EMNST [97] using

Table 21: Experiments with VGG-9.

β	FedLPA	FedNova	SCAFFOLD	FedAvg	FedProx	Dense
0.1	58.48±1.33	28.77±2.03	32.45±0.12	33.71±0.16	31.78±0.40	51.76±0.28
0.3	75.98±1.72	53.78±0.32	55.00±1.07	54.61±0.02	52.79±0.80	70.10±1.45
0.5	79.02±0.81	62.31±0.90	61.75±0.34	63.18±1.70	62.55±0.17	76.20±1.10

Table 22: Experiments with CIFAR-100 using FedLPA.

β	FedAvg	FedLPA
0.1	1.31±0.05	15.11±0.38
0.3	1.75±0.10	18.82±0.71
0.5	1.38±0.11	21.77±0.03

simple-CNN with 10 clients and five random seeds. We do the experiments on EMNIST-mnist and EMNIST-letters. The results are shown in Table 23.

In addition to these, we conduct experiments with Tiny-ImangeNet [98] with ResNet-18 with 10 clients and five random seeds. The results are shown in Table 24.

G.10 Ablation experiments analyzing the number of approximation iterations of FedLPA

The proposed method is composed of multiple approximations: 1) empirical Fisher to approximate the Hessian, 2) block-diagonal Fisher matrix instead of full, 3) approximating global model parameter $\bar{\mathbf{M}}$ with optimization problem in Eq. 14.

1) Empirical Fisher to approximate the Hessian:

Although empirical Fisher has been successfully applied in many methods and yielded good results, discussions concerning the approximation error of empirical Fisher are limited. Fortunately, previous work [99] provides a detailed critical discussion of the empirical Fisher approximation.

i. Fisher to approximate the Hessian:

When the loss function represents an exponential family distribution, the Fisher is a well-justified approximation of the Hessian, and its approximation error can be bounded in terms of residuals. The accuracy of this approximation improves as the residuals diminish and is exact when the data is perfectly fitted.

ii. Empirical Fisher to Fisher:

It's noted that the Fisher and empirical Fisher coincide near minima of the loss function under two conditions:

A. The model distribution closely approximates the data distribution.

B. A sufficiently large number of samples allows both the Fisher and empirical Fisher to converge to their respective average values in the population.

In practical environments, especially condition 1, might not hold, causing bias between empirical Fisher and Fisher. However, empirical Fisher still contains effective covariance information. In second-order optimization methods, the covariance information in empirical Fisher can adapt to

Table 23: Experimental with EMNIST.

Dataset	Partitions	FedLPA	FedAvg
EMNIT-mnist (10 classes)	β=0.1	74.23±3.10	57.63±2.30
	β=0.3	86.55±0.24	62.32±1.77
	β=0.5	91.75±0.26	82.71±0.96
EMNIT-letters (37 classes)	β=0.1	26.34±0.71	16.22±0.38
	β=0.3	31.75±0.03	25.51±0.44
	β=0.5	33.78±0.14	26.34±0.07

Table 24: Experiments with Tiny-ImangeNet using ResNet-18.

β	FedLPA	Dense	FedAvg
0.1	17.02±1.40	15.88±1.96	3.72±1.44
0.3	27.80±2.10	24.91 ±1.65	8.41±0.87
0.5	30.14±1.25	29.43±0.72	12.07 ±1.92

Table 25: Experiments for the approximation study.

Number of iterations	Accuracy(β=0.1)	Accuracy(β=0.3)	Accuracy(β=0.5)	Computation(s)
1000	52.81±0.71	60.31±0.23	72.11±0.57	3
5000	59.70±0.32	68.09±0.30	74.27±0.12	15
10000	55.33±0.06	68.20±0.04	73.33±0.06	30
20000	58.41±0.05	68.11±0.07	73.51±0.02	60

the gradient noise in stochastic optimization. Nevertheless, referencing the work [54], we can use the model's predictive distribution to obtain an unbiased estimate of the true Fisher at the same computational cost as empirical Fisher.

(2) Block-diagonal Fisher matrix to approximate the full one: The work [100] provides a detailed evaluation and testing of using block-diagonal Fisher to approximate the full one. Firstly, Chapter 6.3.1 "Interpretations of this approximation" in the paper [100] indicates that using a block-wise Kronecker-factored Fisher closely approximates the full Fisher. Although there is a bias term, this term approximates zero when there are sufficient samples. Furthermore, the paper examines the approximation quality of block-diagonal Fisher compared with the true Fisher and suggests that block-diagonal Fisher captures the main correlations, while the remaining correlations have a minimal impact on the experimental results.

(3) Besides, we have added some experiments for more ablation studies with our method on the same experiment setting in the paper with five random seeds with 10 clients. We conducted experiments on FMNIST datasets with β=0.1, 0.3 and 0.5. The results are shown in Table 25. We show the experiments analyzing the number of approximation iterations. With the experiment results, we could know that 5000 iterations are enough to get the ideal results. By default, we use 10000 iterations.

We also show that the computation time for the approximation is linear with the number of approximation iterations in the last column of Table 25.

Additionally, it's worth noting that concerning Laplace approximation, the analysis [101] suggests that the error of Laplace approximation is inversely proportional to the input dimension n with $O(n^{-1})$. According to this conclusion, it can be inferred that in our method, for each layer of the neural network, the error of Laplace approximation is inversely proportional to its width. When the neural network is infinitely wide, the approximation error tends towards zero.

G.11 Artifact details

We have uploaded the codebase containing all the methods compared in our paper. Setting up the environment is relatively straightforward with the provided readme file. If you refer to the scripts folder, you will find all the bash scripts necessary to reproduce the tables and figures from our experiments.

The experiments.sh script covers the experiments in Table 1, Table 11, Table 12, Table 13, and Table 14. Running these experiments on a single 2080Ti GPU will take approximately 81 days. Specifically, Table 1 itself will take about 35 days.

The experiments_client.sh script covers the experiments in Table 2, requiring approximately 40 days on a single 2080Ti GPU.

The experiments_coor.sh script covers the experiments in Table 4, which can be completed in 2 days.

The experiments_dp.sh script covers the experiments in Table 8, requiring approximately 1 day on a single 2080Ti GPU.

The experiments_fedavg_with_attack.py and experiments_fedlpa_with_attack.py covers the experiments in Table 10 requiring approximately 1 day on a single 2080Ti GPU.

The experiments_extreme_clients.sh script covers the experiments in Table 15 and requires approximately 4 days of GPU processing.

The experiments_extreme.sh script reproduces the experiments in Table 16 and takes about 10 days.

The experiments of Table 17 and Table 18 take about 8 days.

The experiments_emnist.sh script covers the experiments in Table 23 and takes about 1 day.

Running experiments_multiple_round.sh will yield the results as shown in Figure 2, and this process takes about 1 day.

The experiments for Table 6 and Table 7 will take about 1 day. The experiments for Table 25 and Table 3 will also take about 1 day.

To generate the t-SNE visualizations shown in Figure 1, Figure 3, and Figure 4, you can use the experiments.py script with the “alg=tsne" option.

In total, reproducing all the experiment results in this paper will require about 185 days for GPU processing.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes] .

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes] .

   Justification: The paper discusses the limitations of the work performed by the authors.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes] .

Justification: The paper provides the full set of assumptions and a complete (and correct) proof.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes] .

   Justification: The paper fully discloses all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper.

   Guidelines:

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

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

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

Answer: [Yes] .

Justification: The paper provides open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material.

Guidelines:

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

6. **Experimental Setting/Details**

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

Answer: [Yes] .

Justification: The paper specifies all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results.

Guidelines:

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

7. **Experiment Statistical Significance**

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

Answer: [Yes] .

Justification: The results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes] .

   Justification: For each experiment, the paper provides sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes] .

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [NA] .

    Justification: There is no societal impact of the work performed.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA] .

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [NA] .

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [NA] .

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA] .

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA] .

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.