LOCAL LEARNING MATTERS: RETHINKING DATA HETEROGENEITY IN FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning (FL) is a promising strategy for performing privacy-preserving, distributed learning with a network of clients (i.e., edge devices). However, the data distribution among clients is often non-IID in nature, making efficient optimization difficult. To alleviate this issue, many FL algorithms focus on mitigating the effects of data heterogeneity across clients by introducing a variety of proximal terms, some incurring considerable compute and/or memory overheads, to restrain local updates with respect to the global model. Instead, we consider rethinking solutions to data heterogeneity in FL with a focus on local learning generality rather than proximal restriction. Inspired by findings from generalization literature, we employ second-order information to better understand algorithm effectiveness in FL, and find that in many cases standard regularization methods are surprisingly strong performers in mitigating data heterogeneity effects. Armed with key insights from our analysis, we propose a simple and effective method, FedAlign, to overcome data heterogeneity and the pitfalls of previous methods. FedAlign achieves comparable accuracy with state-of-the-art FL methods across a variety of settings while minimizing computation and memory overhead.

1 INTRODUCTION

Federated learning (FL) (Kairouz et al., 2021) enables a large number of clients to perform collaborative training of machine learning models without compromising data privacy. In the FL setting, participating clients are typically deployed in a variety of environments or owned by a diverse set of users. Therefore, the distribution of each client's local data can vary considerably (i.e., data heterogeneity). This non-IID data distribution among participating devices in FL makes optimization particularly challenging. As each client trains locally on their own data, they step towards their respective local minimum. However, this local convergence point may not be well aligned with the objective of the global model (that is, the model being learned though aggregation at the central server). *Therefore, the client model often drifts away from the ideal global optimization point and overfits to its local objective.* When such client drifting occurs, the performance of the central aggregated model is hindered (Kairouz et al., 2021; Li et al., 2021a).

One straight-forward solution to this phenomenon is to simply limit the number of local training epochs performed between central aggregation steps. However, this severely hinders the convergence speed of the FL system, and many communication rounds are required to achieve adequate performance. The time to convergence and immense network overhead incurred by such an approach are often not tolerable for real-world distributed systems. Therefore, effectively addressing data heterogeneity is of paramount concern in federated learning.

Many algorithmic solutions to this problem have been proposed in the literature (Sahu et al., 2018; Li et al., 2021b; Karimireddy et al., 2020; Acar et al., 2021). These strategies typically focus on mitigating the effects of data heterogeneity across clients by introducing a variety of *proximal terms* to restrain local updates with respect to the global model. *However, by restraining the drift, they also inherently limit the local convergence potential; less novel information is gathered per communication round.* Consequently, many current FL algorithms do not provide stable performance improvements across different non-IID settings in comparison to classic baselines (Li et al., 2021a). Furthermore, existing methods have paid little attention to the resource constraints of the client, typ-

ically scarce for deployed FL edge devices, and in some cases incur considerable compute and/or memory overheads on the client in their effort to alleviate client drift.

Motivation. In the centralized training paradigm, network generalization capability has been well studied to combat overfitting. Even in standard settings where the training and test data are drawn from a similar distribution, models still overfit on the training data if no precautions are taken. This effect is further intensified when the training and test data are of different distributions. Various regularization techniques are introduced to enforce learning generality during training and preserve suitable test performance. Similarly, overfitting to the local training data of each device in FL is detrimental to overall network performance, as the client drifting effect creates conflicting objectives among local models. *Thus, a focus on improving model generality should be of primary concern in the presence of data heterogeneity. Improving local learning generality during training would inherently position the objective of the clients closer to the overall global objective.* However, despite its intuitive motivations, this perspective has been overlooked by the bulk of current FL literature.

Therefore, in this paper, we propose rethinking approaches to data heterogeneity in terms of **local learning generality** rather than proximal restriction. Specifically, we carefully analyze the effectiveness of various data and structural regularization methods at reducing client drift and improving FL performance (Section 3). Utilizing second-order information and insights from out-of-distribution generality literature (Rame et al., 2021; Parascandolo et al., 2020), we identify theoretical indicators for successful FL optimization, and evaluate across a variety of FL settings for empirical validation.

Although some of the regularization methods perform well at mitigating client drift, *significant resource overheads* are still incurred to achieve the best performance (see Section 4). Therefore, we propose **FedAlign**, a distillation-based regularization method that promotes local learning generality while maintaining excellent resource efficiency. Specifically, FedAlign focuses on regularizing the Lipschitz constants of the final block in a network with respect to its representations. By focusing solely on the last block, we effectively regularize the portion of the network most prone to overfitting and keep additional resource needs to a minimum. Therefore, FedAlign achieves state-of-the-art accuracy on multiple datasets across a variety of FL settings, while requiring significantly less computation and memory overhead in comparison to other state-of-the-art methods.

Our contributions are as follows:

- We approach one of the most troublesome FL challenges (i.e., client drift caused by data heterogeneity) from a unique angle than any other previous work. We do not focus on reparameterization tricks to maintain closeness to the central model, or adjust the aggregation scheme to mitigate the effects of non-IID data distributions. *Rather, we propose the rethinking of this problem from fundamental machine learning training principles.* In this way, we analyze the performance of standard regularization methods on FL and their effectiveness against data heterogeneity.
- Not only do we empirically analyze the performance of regularization methods in FL, we also propose to take a deeper look. Specifically, we inform our analysis with theoretical indicators of learning generality to provide insight into which methods are best and why. Our aim is to provide this valuable knowledge to the FL community to inspire new, productive research directions.
- Informed by our analysis and examining the pitfalls of previous methods, we propose FedAlign, which achieves state-of-the-art accuracy while maintaining memory and computational efficiency.

2 Related Work

Federated Learning. In general, federated learning algorithms aim to obtain a collective model which minimizes the training loss across all clients. This objective can be express as

$$\min_{w} F(w) = \sum_{c=1}^{C} \alpha_c F_c(w), \tag{1}$$

where $F^k(w)$ is the local loss of device c, and α_c is an arbitrary weight parameter with $\sum_{c=1}^{C} \alpha_c = 1$. One of the earliest algorithms proposed in FL is Federated Averaging, or FedAvg (McMahan et al., 2017). This approach simply optimizes the local training loss with standard SGD training, and aggregates using a weighted average approach with $a_c = \frac{n_c}{n}$, where n_c is equal to the number of training samples on client c, with a total of n training samples partitioned across all C clients.

Recent works attempt to improve over this baseline with two distinct focuses: improvements to the local training at the client, or improvements to the global aggregation process at the server. In this work, we focus on local training and client drift, and therefore we will first discuss methods of this nature. To mitigate data heterogeneity complications, a common approach is to introduce proximal terms to the local training loss. For instance, FedProx (Sahu et al., 2018) forms the local objective $F^k(w) + \frac{\mu}{2} \|w - w^t\|^2$, where μ is a hyperparameter, w is the current local model weights, and w^t is the global model weights from round t. The goal of this reparameterization is to minimize client drift by limiting the impact of local updates from becoming extreme. More recently, MOON (Li et al., 2021b) proposes a similar reparameterization idea inspired by contrastive learning. Specifically, the authors form a local model constrastive loss comparing representations of three models: the global model, the current local model, and a copy of the local model from the previous round. The goals of this term are similar to that of FedProx but in feature representation space; to push the current local representation closer to the global representation. At the same time, the current local model is being pushed away from the representations of the local model copy of the previous round. Other methods (Acar et al., 2021; Karimireddy et al., 2020) follow similar ideas; they aim to limit the impact of the local update or shift the update with a correction term.

However, these approaches have two main downsides. First, by restraining the drift, they also inherently limit the local convergence potential. With this, not as much new information is gathered per communication round. Second, many of these methods incur substantial overheads in memory and/or computation. For instance, because of its model constrastive loss, MOON (Li et al., 2021b) requires the storage of three full-size models in memory simultaneously during training, and forward passing through each of these every iteration. This requires a great deal of additional resources, which are often already scarce in FL client settings.

Other works focus on the server side of the system, aiming to improve the aggregation algorithm. Yurochkin et al. (2019) propose a Bayesian nonparametric method for matching neurons across local models at aggregation rather than naively averaging. However, the presented framework is limited in application to fully-connected networks, and therefore Wang et al. (2019) extend it to CNNs and LSTMs. FedNova (Wang et al., 2020) presents a normalized averaging method as an alternative to the simple FedAvg update. As we focus on the local training, these works are orthogonal to our work. A few approaches (Yoon et al., 2021; Oh et al., 2020; Shin et al., 2020) propose federated schemes inspired by the data augmentation method Mixup, using similar averaging techniques on the local data and sharing the augmented data with the global model or other devices. However, even though the data is augmented in some way prior to distribution, the sharing of private data from the client is less than ideal for privacy preservation. Furthermore, sharing additional data worsens the communication burden on the system, which is a principal concern in FL.

Learning Generality. In traditional centralized training, the practice of regularization of various forms is common practice for improving generality. Data-level regularization, such as basic data augmentation and other more advanced techniques (Zhang et al., 2018; Yun et al., 2019), are known to be quite effective. Other methods introduce a level of noise to the training process via structural modification; for instance, random or deliberate modifications to the network connectivity (Huang et al., 2016; Ghiasi et al., 2018; Tompson et al., 2015). Yang et al. (2020) proposes a hybrid approach that introduces self-guided gradient perturbation to the training process through the use of sub-network representations, knowledge distillation, and input transformations. As part of this work, we employ a variety of regularization methods in many FL settings and analyze their performance in comparison to state-of-the-art FL algorithms.

3 Empirical Study

We wish to assess the data heterogeneity challenge of FL from a simple yet unique perspective of local learning generality. Specifically, we first study the effectiveness of standard regularization techniques as FL solutions in comparison to state-of-the-art methods.

3.1 Preliminaries

We employ three FL algorithms, namely FedAvg, FedProx, and MOON. These works represent both classic baselines and current state-of-the-art, and are described in Section 2. For comparison, we

employ three state-of-the-art regularization methods: Mixup (Zhang et al., 2018), Stochastic Depth (Huang et al., 2016), and GradAug (Yang et al., 2020).

Mixup is a data-level augmentation technique that performs linear interpolation between two samples. Specifically, given two sample-label pairs (x_i, y_i) and (x_j, y_j), they are combined as $\tilde{x} = \beta x_i + (1-\beta)x_j$ and $\tilde{y} = \beta y_i + (1-\beta)y_j$, where $\beta \sim Beta(\gamma, \gamma)$.

Stochastic depth is a structural-based method that drops layers during training, thereby creating an implicit network ensemble of different effective lengths. Specifically, the output of layer (or residual block) ℓ is given by $\zeta_\ell = \sigma\left(\lambda \mathcal{F}_{\theta_\ell}\left(\zeta_{\ell-1}\right) + \mathcal{I}\left(\zeta_{\ell-1}\right)\right)$, where λ is a Bernoulli random variable, $\mathcal{F}_{\theta_\ell}$ is the operation within the network with parameter θ at layer ℓ, $\mathcal{I}$ is the identity mapping operation of residual connections, and σ is a non-linear activation function. The keep probability is defined as $\gamma = P(\lambda = 1)$, where in practice each layer has its own keep probability set with a linear decay rule $\gamma_\ell = 1 - \frac{\ell}{L}(1-\gamma_L)$, with L denoting the total number of layers (or blocks) in the network.

GradAug is a recent regularization approach that combines data-level and structural techniques in a distillation-based framework. Specifically, the training loss is defined as

$$L_{GA} = L_{CE}(\mathcal{F}_\theta(x), y) + \mu \sum_{i=1}^{n} L_{KD}\left(\mathcal{F}_{\theta^{\omega_i}}\left(T^i(x)\right), \mathcal{F}_\theta(x)\right), \tag{2}$$

where $\mathcal{F}_{\theta^{\omega_i}}$ denotes a slimmed sub-network of fractional width ω_i, T^i is a transformation performed on the input (e.g. resolution scaling), and μ is a balancing parameter between the cross-entropy loss L_{CE} and the summed Kullback–Leibler divergence (L_{KD}) loss on the sub-networks.

3.2 Experimental Setup

To begin our analysis, we test the accuracy of several state-of-the-art FL algorithms with several regularization methods in a common FL setting. We perform experiments using CIFAR-100 (Krizhevsky et al.), an image recognition dataset with 50,000 training images across 100 categories, and ResNet56 (He et al., 2016) as the model. As common in the literature (Li et al., 2021b; Acar et al., 2021; He et al., 2020), the dataset is partitioned into K unbalanced subsets using a Dirichlet distribution ($Dir(\alpha)$), with the default being $\alpha = 0.5$. With this data partitioning scheme, it is possible for a client to have no samples for one or multiple classes. Therefore, many clients will only see a portion of the total class instances. This makes the setting more realistic and challenging. For all methods and experiments we use an SGD optimizer with momentum, and a fixed learning rate of 0.01. In our basic setting, training is conducted for 25 rounds, with 16 clients and 20 local epochs per round. Any modifications to this setting in subsequent results will be stated clearly.

We compare the previously described FL algorithms and regularization methods. FedProx, MOON, and GradAug all have a hyperparameter μ to balance their additional loss terms. We report all results with the optimal μ for all approaches, being 0.0001, 1.0, and 1.75 for FedProx, MOON, and GradAug respectively. For Mixup and Stochastic Depth, γ and γ_L are set to 0.05 and 0.9 respectively. For GradAug specifically, the number of sub-networks $n = 2$. A two-layer projection layer is added to the model for MOON as specified in the original paper. Basic data augmentations (random crop, scale, and normalization) are kept consistent across all methods.

3.3 Results Comparison

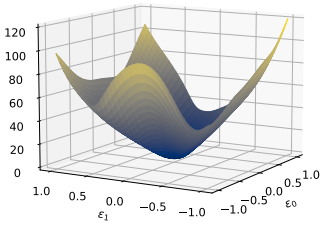
The accuracy results are shown in Table 1. Within the current state-of-the-art FL algorithms (upper portion of Table 1), MOON achieves the best accuracy. This is expected, as MOON is the most intricate of the FL methods, requiring the usage of three individual models for its contrastive learning technique. However, when we compare with standard regularization techniques (Mixup, StochDepth and GradAug in the lower portion of Table 1), we see that these perform similarly or substantially better. GradAug particularly stands out, achieving an

Table 1: Results for accuracy (%) on CIFAR-100 and second-order metrics indicating the smoothness of the loss space (λ_{max}, H_T) and cross-client consistency (H_N, H_D) for each method.

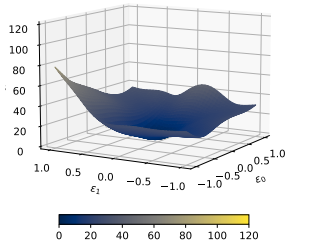
Method	Acc. ↑	λ_{max}↓	H_T↓	H_N↓	H_D↑
FedAvg	52.9	297	6240	11360	0.98
FedProx	53.0	270	6132	6522	0.98
MOON	55.3	252	5520	5712	0.97
Mixup	54.0	216	5468	15434	0.99
StochDepth	55.5	215	3970	8267	0.97
GradAug	**57.1**	**167**	**2597**	2924	0.96

accuracy ∼2% higher than MOON and ∼4% higher than FedAvg and FedProx. StochDepth also achieves similar accuracy to MOON. Furthermore, these regularization methods bring the same or better performance than MOON, **with much less memory and/or compute requirements**. *We find that regularization methods appear to have an advantage in this situation; however, we wish to further investigate why this could be the case.* Next, we present our in-depth analysis based on the second-order information in Section 3.4.

3.4 Algorithm Analysis based on Second-order Information



(a) FedAvg



(b) GradAug

Figure 1: Visualization of the parametric loss landscape with Hessian eigenvectors ϵ_0 and ϵ_1 for each resulting global model.

Recent works in the Neural Architecture Search domain (Chen & Hsieh, 2020; Zela et al., 2020), as well as in network generalization (Keskar et al., 2016; Yao et al., 2018; Jiang* et al., 2020), have noted the importance of the top Hessian eigenvalue (λ_{max}) and Hessian trace (H_T) as a predictor of performance and indicator of network generality. *Having a lower λ_{max} and H_T typically yields a network that is less sensitive to small perturbations in the networks weights. This has the beneficial effects of smoothing the loss space during training, reaching a flatter minima, and easing convergence.* These properties are particularly advantageous in federated learning, where extreme non-IID distributions and limited local data often make convergence difficult.

Motivated by these insights, we analyze the top Hessian eigenvalue and Hessian trace of the global models trained with each FL scheme to provide insight into the effectiveness of each method.

As described in Yao et al. (2020), the top Hessian eigenvalues can be approximated with the Power Iteration (Yao et al., 2018) method using a simple inner product and standard backpropagation. However, since neural networks are extremely high dimensional, just measuring the top few eigenvalues may not be sufficient. Therefore, Yao et al. (2020) also find a similar approximation for the trace utilizing the Hutchinson method (Hutchinson, 1989). We conduct our analysis with the top Hessian eigenvalues and trace of the final averaged models using these methods.

In Table 1, we include the results of the Hessian analysis. First, we find that FedAvg has the highest λ_{max} and H_T. FedProx and MOON each result in lower values, indicating some degree of improved generalization. However, interestingly, we find that regularization methods are most effective at reducing the λ_{max} and H_T, with GradAug having by far the lowest in both values. We visualize the effect of this reduction in λ_{max} and H_T in Fig. 1, where it can be seen that GradAug is able to smooth out the loss landscape considerably in comparison to FedAvg.

In the separate field of out-of-distribution (O.O.D.) generalization for centralized training, second-order information is being found quite useful as a theoretical indicator. Recent works (Parascandolo et al., 2020; Rame et al., 2021) find that forming representations that are "hard to vary" seem to result in better O.O.D. performance. More specifically, they show that the resulting loss landscapes across domains for the learned model should be consistent with each other. In terms of theoretical indicators, this translates to matching domain-level Hessians, as the Hessian provides an approximation of local curvature. Similarly, in federated learning, each client is essentially a separate domain. *Therefore, matching Hessians in norm and direction across clients reveals additional detail and reasoning behind the effectiveness of each method.* In light of these findings in O.O.D. literature, we analyze the difference in Hessian norm (H_N) and the Hessian direction across clients (H_D), where

$$H_N^{k,j} = \left(\|\text{Diag}\left(\mathbf{H}_k\right)\|_F^2 - \|\text{Diag}\left(\mathbf{H}_j\right)\|_F^2 \right)^2 \text{ and} \tag{3}$$

$$H_D^{k,j} = \frac{\text{Diag}\left(\mathbf{H}_k\right) \odot \text{Diag}\left(\mathbf{H}_j\right)}{\|\text{Diag}\left(\mathbf{H}_k\right)\|_F^2 \cdot \|\text{Diag}\left(\mathbf{H}_j\right)\|_F^2}. \tag{4}$$

Here, $\odot$ is the dot product, $\mathbf{H}_k$ and $\mathbf{H}_j$ are the Hessian matrices of clients k and j, and $\|\cdot\|_F$ is the Frobenius norm. $H_N^{k,j}$ and $H_D^{k,j}$ are averaged across all pairs of clients and reported as simply

H_N and H_D in Table 1. For these Hessian matching criteria, a lower H_N (less difference) and a higher H_D (essentially the cosine similarity) are desired.

As seen on the right side of Table 1, H_D is fairly consistent across all methods. In terms of λ_{max}, H_T, and H_D, most methods seem to correlate decently well between these values and performance. However, there are a couple of cases which require more information. First, Mixup has a similar H_T value as MOON, but lower accuracy. H_N provides an essentially detail; the Hessian norms of Mixup are not nearly as similar across clients as those of MOON. Second, MOON has both a higher λ_{max} and H_T; however, it achieves similar accuracy to StochDepth. We again find that the difference is in H_N; the Hessian norms of MOON are more similar across clients than those of StochDepth. In the end, MOON and StochDepth result in very similar performance.

Take-away. It appears that both the eigenvalue/trace analysis and Hessian matching criteria are important, and optimal methods should perform well across all these indicators. To understand how these differences will play out empirically, we conduct a variety of ablations in Section 3.5.

3.5 Ablation Study under Various FL Settings

Data Heterogeneity. Federated systems can be deployed with many different setups and diverse environments. Therefore, we conduct further analysis across a variety of FL settings to ensure the generality of our findings. First, we examine the effect of varying the degree of heterogeneity in the client data distributions. The results are shown in Table 2. All other settings are maintained from Table 1; only the data distribution $Dir(\alpha)$ is varied. *A lower α value indicates a more heterogeneous distribution.* As the degree of data heterogeneity decreases, the effect of client drift should also become less significant. Therefore, we expect that the accuracy for each method will increase, with peak performance in the homogeneous setting. All regularization methods, as well as FedAvg, perform as expected, and find consistent improvement across the degrees of data distribution. However, we see that the accuracy improvement of FedProx and MOON slows as the data approaches homogeneity, with the purely homogeneous setting ("homog" in Table 2) resulting in lower performance for MOON than $\alpha = 2$. In their attempt to mitigate client drift and keep local updates close to the global model, it appears that they also hinder their own ability to fully learn on minorly heterogeneous or even homogeneous data. This is not ideal for deployable FL systems, as the degree of heterogeneity is not known ahead of time. Additionally, even in the most heterogeneous cases, the structural regularization methods perform better than the standard FL algorithms. For instance, even though StochDepth and MOON achieve similar results at $\alpha = 0.5$, we see that StochDepth achieves $+1.9\%$ at $\alpha = 0.1$ as well as similar improvement in more homogeneous situations. In all settings, GradAug performs the best.

Table 2: Ablation results for data heterogeneity.

Method	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 2$	homog
FedAvg	45.1	52.9	54.2	56.2
FedProx	45.5	53.0	53.8	54.5
MOON	46.1	55.3	56.3	55.7
Mixup	43.5	54.0	55.5	56.3
StochDepth	48.0	55.5	57.4	58.6
GradAug	**48.8**	**57.1**	**59.4**	**60.7**

Number of Local Training Epochs. The main purpose for adequately handling data heterogeneity is to allow for more productive training on the client each round, therefore reducing the time to convergence and required communication cost. Therefore, to examine the training productivity of each method, we examine their accuracy with various allotted local training epochs per round (E). The results are shown in Table 3.

Table 3: Ablation results for number of local epochs (E).

Method	$E = 10$	$E = 20$	$E = 30$
FedAvg	50.6	52.9	53.2
FedProx	50.9	53.0	52.8
MOON	50.3	55.3	55.2
Mixup	50.5	54.0	54.2
StochDepth	51.1	55.5	56.7
GradAug	**53.7**	**57.1**	**57.9**

Ideally methods should continue to improve in accuracy with more allotted local training epochs. In Table 3, we see that all methods improve substantially from 10 epochs per round to 20. However, from 20 to 30, the trends vary substantially. As a baseline, FedAvg slightly improves by $+0.3\%$. Surprisingly, FedProx and MOON actually experience a slight reduction in accuracy from 20 to 30 epochs (-0.2% and -0.1%, respectively). Meanwhile, the standard (particularly structural) regularization methods continue to increase in accuracy. Therefore, these methods illustrate the ability to maintain productive training, even across a wide range of allotted local epochs.

Number of Clients. In real-world FL settings, the number of participating clients can vary widely. Moreover, only a portion of clients are potentially sampled per round, whether for connectivity reason or other capacity restrictions of the central system. Therefore, it is crucial that an FL method can converge under such conditions. We study the affect of client number and client sampling in Table 4. $C = 64 \times 0.25$ indicates that there are 64 total clients in the system, but only a fraction (0.25) are sampled each round. The rest of the presented results in Table 4 sample all K clients each round. $C = 64 \times 0.25$ (100) is run for 100 rounds, and all other settings for the default 25 rounds.

Table 4: Ablation results for varying number of clients K in synchronous and client sampling cases.

Method	$C = 16$	$C = 32$	$C = 64$	$C = 64 \times 0.25$	$C = 64 \times 0.25$ (100)
FedAvg	52.9	44.7	34.4	33.2	46.8
FedProx	53.0	45.1	34.8	32.1	46.0
MOON	55.3	45.5	35.2	34.3	49.8
Mixup	54.0	44.8	35.6	34.1	48.7
StochDepth	55.5	47.7	36.1	34.5	51.4
GradAug	**57.1**	**50.5**	**40.3**	**38.4**	**52.5**

FedProx and MOON have similar trends with increasing clients, having a slight edge on FedAvg in all synchronous cases. Interestingly, in the client sampling case ($C = 64 \times 0.25$), FedAvg actually performs better than FedProx. This is particularly insightful in terms of learning efficiency. When a small percentage of clients are sampled, only a portion of the dataset is effectively trained on each round. Therefore, learning efficiency becomes paramount for maintaining suitable convergence. However, methods that focus on staying close to the server representation may suffer from insufficient local learning and subsequently slower convergence.

The standard regularization methods in the lower portion of Table 4 have similar trends to those of the upper portion. However, these methods maintain better accuracy than FedAvg in all settings, even in the client sampling case. Starting from a similar accuracy as MOON at 16 clients, StochDepth maintains higher accuracy than MOON as the number of clients increases. Overall, GradAug performs the best in all cases. *Therefore, even though these regularization methods were not designed for the FL setting and partial client sampling, they still perform on par with or improve over current state-of-the-art FL algorithms.*

4 Proposed Method – FedAlign

Overall, we find that GradAug is particularly effective in the FL setting, having the highest accuracy in all tested scenarios along with the lowest λ_{max}, H_T, and H_N. However, while this method is quite memory efficient in comparison to many FL methods (only requires a single stored model during training), it does incur additional training time over the FedAvg baseline. This is because GradAug requires multiple forward passes through slimmed sub-networks for the distillation loss. It is possible to reduce the computation burden by using a smaller number of sub-networks during the knowledge distillation process. As seen in Table 5, the wall-clock time of GradAug can be reduced by 25% when using one sub-network ($n = 1$) versus two ($n = 2$). This change incurs only a 0.3% accuracy drop; nonetheless, a noticeable gap still remains between GradAug and vanilla FedAvg in wall-clock time. **Therefore, the question is, can we devise a method which provides similar effect and performance as GradAug in FL, but with substantially less computational overhead?** This is particularly important in the FL setting, where clients are typically deployed devices with minimal memory and computational resources.

Table 5: Analysis of wall-clock time per round on CIFAR-100 with $C = 16$ and $E = 20$ across four RTX-2080Ti GPUs.

Method	Accuracy (%)	Time (s)
FedAvg	52.9	142
FedProx	53.0	161
MOON	55.3	418
Mixup	54.0	142
StochDepth	55.5	**139**
GradAug ($n = 1$)	56.8	230
GradAug ($n = 2$)	**57.1**	325
GradAug ($n = 3$)	**57.1**	420
GradAug ($n = 4$)	56.6	513
FedAlign	56.6	167

To do so, we first take note of the following insights gathered during our analysis: 1) Second-order information is insightful for understanding the learning generality of neural networks. Particularly, we find that flatness and consistency in this realm are desirable traits. 2) In practice, we find that structural regularization, and especially distillation-based like GradAug, is quite ef-

fective. Furthermore, the weight sharing mechanisms of such approaches are memory efficient compared to other methods that rely on global model or previous model storage. Therefore, we combine these insights into a novel algorithm to optimize for performance and resource needs in FL.

We propose **FedAlign**, an distillation-based regularization method that aligns the Lipschitz constants (i.e. top Hessian eigenvalues) of the most critical network components through the use of slimmed sub-blocks. Fig. 2 shows an overview of FedAlign, whose design is based on two key principles. First, motivated by the insights of Section 3.4, we internally regularize the Liptschitz constants of network blocks to *promote smooth optimization and consistency within the model.* Recent work (Shang et al., 2021) presents a quick approximation of the Lipschitz constants for neural network layers in a differentiable manner. This enables the use of second-order information in the distillation process, traditionally between a fully trained teacher and a learning student. We adapt this technique for distillation-based regularization with an untrained network in place of the traditional logit-based loss.

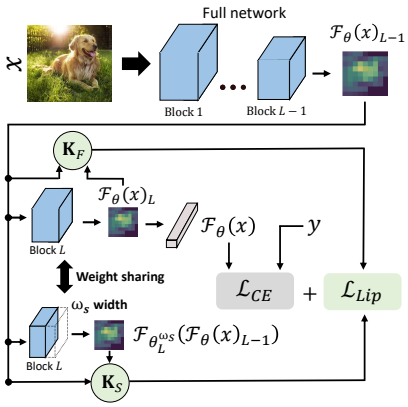


Figure 2: The proposed FedAlign for local client training in FL. Features $\mathcal{F}_\theta(x)_{L-1}$ are run through Block L as normal. The only additional inference in FedAlign is through Block L at a reduced width (i.e. sub-block), reusing features $\mathcal{F}_\theta(x)_{L-1}$ as input. The channels throughout the layers in the sub-block are a ω_S fraction of the original number. This is accomplished via temporary uniform pruning of Block L. Hyperparameter ablations are presented in Appendix A.1.

Second, in order to reduce computation in a purposeful manner, we take note of certain network properties. Particularly, it has been shown that the final layers of a neural network are most prone to overfit to the client distribution (Luo et al., 2021). Therefore, FedAlign is designed with a focus on these critical points in the network. *We then ask the question, if we aim to concentrate our regularization efforts on the final layers, why should we run all networks for distillation from start to finish like in GradAug?* Instead, we can reuse the intermediate features of the full network as input to just the final block at a reduced width, and therefore significantly reduce computation. In this way, we harness the benefits of distillation-based regularization in performance and memory footprint, while effectively mitigating computational overhead.

Combining these two key principles, we form the FedAlign local objective as

$$\mathcal{L}_{FA} = \mathcal{L}_{CE}(\mathcal{F}_\theta(x), y) + \mu \mathcal{L}_{Lip}\left(\mathbf{K}_S, \mathbf{K}_F\right), \tag{5}$$

where μ is a balancing constant, $\mathcal{L}_{CE}$ is the cross-entropy loss, and $\mathcal{L}_{Lip}$ is the mean squared error between the approximated Liptschitz constant vectors $\mathbf{K}_S$ and $\mathbf{K}_F$ for the reduced width (i.e. sub-block) and full width block L, respectively. Specifically, the Lipschitz approximations are calculated via the spectral norm of a transmitting matrix using feature maps to bypass the need for singular value decomposition as in Shang et al. (2021). Therefore, in our framework, we use the intermediate features for these transmitting matrices $\mathbf{X}_F$ and $\mathbf{X}_S$, where

$$\mathbf{X}_F = \left(\mathcal{F}_\theta(x)_{L-1}\right)^\top \mathcal{F}_\theta(x)_L, \text{and} \tag{6}$$

$$\mathbf{X}_S = \left(\mathcal{F}_\theta(x)_{L-1}\right)^\top \left(\mathcal{F}_{\theta_L^{\omega_S}}\left(\mathcal{F}_\theta(x)_{L-1}\right)\right). \tag{7}$$

$\mathcal{F}_{\theta_L^{\omega_S}}$ is the output feature map of the final block L at reduced width ω_S; $\mathcal{F}_\theta(x)_L$ and $\mathcal{F}_\theta(x)_{L-1}$ are the feature maps outputted by the last and prior to last blocks of the full network (see Fig. 2). Finally, the spectral norm (SN) of $\mathbf{X}_F$ and $\mathbf{X}_S$ are approximated using the Power Iteration method (Yao et al., 2018), and therefore $\mathbf{K}_F = \|\mathbf{X}_F\|_{SN}$ and $\mathbf{K}_S = \|\mathbf{X}_S\|_{SN}$. Looking back to Eq. 5, one could view $\mathcal{L}_{Lip}$ as a correction term; however, there is a key distinction between this form of regularization and that of traditional FL algorithms. *Our correction term promotes the local client models to learn well-generalized representations based on their own data, instead of forcing the local models to be close to the global model.*

As seen in Table 5, FedAlign achieves state-of-the-art accuracy in a resource-efficient manner. Specifically, FedAlign improves +3.7% over FedAvg while only incurring a small increase in wall-clock time. In comparison, FedProx and MOON have a much lower accuracy, while incurring similar or substantially more compute time. Also, both FedProx and MOON require the storage

of at least one additional model in memory, while FedAlign does not. Furthermore, FedAlign attains similar accuracy to GradAug while realizing a $\sim$48% and $\sim$37% reduction in wall-clock time compared to the $n = 2$ and $n = 1$ variants.

4.1 FEDALIGN EXPERIMENTS

We further verify the effectiveness of our method across various settings and datasets. In Table 6, we examine the performance of FedAlign with the same ablations as in Section 3.5. FedAlign maintains strong performance across all settings. It follows closely to GradAug in most cases, despite requiring substantially less computation. We further investigate the effectiveness of FedAlign and all other methods across two additional datasets: CIFAR-10 and ImageNet-200. For ImageNet-200, we randomly sample 200 classes from the classic ImageNet-1k (Russakovsky et al., 2015) dataset. We employ ResNet56 and ResNet18 as our models on CIFAR-10 and ImageNet-200, respectively. Hyperparameters for all methods are kept the same as those described in Section 3.2 and Table 6.

For CIFAR-10, we ran a 16 client synchronous and 64 client case with sampling. We note similar trends to CIFAR-100; regularization methods perform well, particularly in the more realistic client sampling case. FedAlign continues to provide similarly strong accuracy as GradAug ($n = 1$) in both settings. On ImageNet-200, we ran two settings as well. Based on the results in Table 7, both GradAug and FedAlign maintain higher performance than other methods, especially in the client sampling case. Interestingly, Stochastic Depth does not perform particularly well in the ImageNet-200 cases. As mentioned in the original paper (Huang et al., 2016), Stocahstic Depth performs better with deeper networks. However, with ResNet18, the overall depth of the network is reduced compared to that in the CIFAR cases. Therefore, as most deployable networks favor width over depth, regularizing with respect to the width of a network is more applicable to the FL setting. *This highlights an additional benefit of FedAlign, which operates using width reduction in the final block and maintains relatively high accuracy despite low resource needs.*

Discussion. Overall, FedAlign offers stable improvements over classic baselines in a vast array of settings with little resource overhead. While some methods perform well in specific situations, consistent performance is especially desirable in FL, as deployed settings can vary widely. Additionally, we note that GradAug is primarily designed for vision data, requiring that a transformation (e.g. scaling) be applied to the input of sub-networks. However, FedAlign is not specific to vision in any way, and therefore could be applied to other data formats. We leave this study for future work.

Table 6: FedAlign ablation results on CIFAR-100. Hyperparamters $\mu = 0.05$ and $\omega_S = 0.8$.

Method	$\alpha = 0.1$	$\alpha = 2$	homog	$E = 10$	$E = 30$	$C = 32$	$C = 64$	$C = 64 \times 0.25$	$C = 64 \times 0.25$ (100)
FedAlign	47.6	58.1	58.6	54.5	57.2	49.6	39.9	36.7	51.6

Table 7: CIFAR-10 and ImageNet-200 results for all methods.

Method	CIFAR-10 $C = 16$	CIFAR-10 $C = 64 \times 0.25$ (100)	ImageNet-200 $C = 16$	ImageNet-200 $C = 32 \times 0.125$ (25)
FedAvg	82.1	77.8	60.7	46.9
FedProx	81.8	76.9	61.3	46.7
MOON	83.4	78.7	61.0	47.0
Mixup	80.5	80.3	61.0	46.7
StochDepth	82.0	79.6	60.4	45.4
GradAug ($n = 2$)	84.3	83.1	63.8	48.5
GradAug ($n = 1$)	84.2	81.1	62.5	48.1
FedAlign	83.8	81.2	61.5	47.5

5 CONCLUSION

In this work, we study the data heterogeneity challenge of FL from a simple yet unique perspective of local learning generality. Specifically, we intuitively reason that regularization methods can effectively alleviate client drift, and confirm these ideas with empirical analysis. Based on our findings, we propose FedAlign, a distillation-based regularization method that promotes local learning generality rather than proximal restriction, and achieves comparable accuracy with SOTA methods while maintaining excellent resource efficiency.

6 Reproducibility Statement

In Section 3.2, we thoroughly describe our settings to ensure that the results of our empirical study can be easily replicated. Additional implementation details for FedAlign are discussed in Appendix A.2. We also plan to make our source code (implemented in PyTorch (Paszke et al., 2019)) publicly available upon acceptance to make replication simple and accessible. As mentioned in Section 4.1, we randomly sample 200 classes from ImageNet-1K (Russakovsky et al., 2015) to form ImageNet-200. The script for generating this dataset and the exact classes sampled will also be included in the code release.

References

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=B7v4QMR6Z9w. 1, 3, 4

Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *International Conference on Machine Learning*, pp. 1554–1565. PMLR, 2020. 5

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in Neural Information Processing Systems*, 31:10727–10737, 2018. 3

Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020. 4

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*, pp. 770–778, 2016. 4

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016. 3, 4, 9

M.F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communication in Statistics- Simulation and Computation*, 18:1059–1076, 01 1989. doi: 10.1080/03610919008812866. 5

Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH. 5

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, and et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2): 1–210, 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL http://dx.doi.org/10.1561/2200000083. 1

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*, pp. 5132–5143. PMLR, 2020. 1, 3

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. 2016. 5

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html. 4

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021a. 1

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021b. 1, 3, 4

Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *arXiv e-prints*, pp. arXiv–2106, 2021. 8

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*, pp. 1273–1282. PMLR, 2017. 2

Seungeun Oh, Jihong Park, Eunjeong Jeong, Hyesung Kim, Mehdi Bennis, and Seong-Lyun Kim. Mix2fld: Downlink federated learning after uplink federated distillation with two-way mixup. *IEEE Communications Letters*, 24(10):2211–2215, 2020. doi: 10.1109/LCOMM.2020.3003693. 3

Giambattista Parascandolo, Alexander Neitz, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf. Learning explanations that are hard to vary. *arXiv preprint arXiv:2009.00329*, 2020. 2, 5

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 10

Alexandre Rame, Corentin Dancette, and Matthieu Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. *arXiv preprint arXiv:2109.02934*, 2021. 2, 5

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. 9, 10

Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018. URL http://arxiv.org/abs/1812.06127. 1, 3

Yuzhang Shang, Bin Duan, Ziliang Zong, Liqiang Nie, and Yan Yan. Lipschitz continuity guided knowledge distillation, 2021. 8, 12

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. 3

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 648–656, 2015. 3

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2019. 3

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*, 2020. 3

Taojiannan Yang, Sijie Zhu, and Chen Chen. Gradaug: A new regularization method for deep neural networks. *Advances in Neural Information Processing Systems*, 33, 2020. 3, 4

Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. *Advances in Neural Information Processing Systems*, 31:4949–4959, 2018. 5, 8

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 581–590. IEEE, 2020. 5

Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ogga20D2HO-. 3

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019. 3

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*, pp. 7252–7261. PMLR, 2019. 3

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gDNyrKDS. 5

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 3, 4

A Appendix

A.1 Hyperparameter Ablations of FedAlign

The default hyperparamter setting used throughout the paper is $\mu = 0.05$ and $\omega_S = 0.8$. The performance of FedAlign with various hyperparamters on CIFAR-100 is shown in Table 8. We vary μ and ω_S independently, meaning $\omega_S = 0.8$ for the μ ablations, and $\mu = 0.05$ when varying ω_S. Table 8 shows that FedAlign is more sensitive to width, particularly below 0.8. Performance is relatively consistent across across a wide range of μ values.

Table 8: FedAlign hyperparameter ablations on CIFAR-100.

Method	$\mu = 0.005$	$\mu = 0.05$	$\mu = 0.1$	$\mu = 0.5$	$\omega_S = 0.6$	$\omega_S = 0.7$	$\omega_S = 0.8$	$\omega_S = 0.9$
FedAlign	56.4	**56.6**	56.3	55.6	55.7	55.1	**56.6**	56.2

A.2 Additional Implementation Details

When calculating $\mathbf{X}_F$ and $\mathbf{X}_S$, the input and output features involved will typically be of different spatial sizes in practice. Therefore, Shang et al. (2021) utilizes an adaptive average pool operation in PyTorch to reduce the spatial size of the larger feature map to that of the smaller one. We likewise employ this operation.

Prior to performing backpropagation, we apply a relative scale to $\mathcal{L}_{Lip}$ along with the μ scaling parameter. In PyTorch-style pseudocode: $loss\_lip = \mu^* \ (loss\_ce.item()/loss\_lip.item())^* \ loss\_lip$. This is to ensure that $\mathcal{L}_{Lip}$ is on relatively the same scale with $\mathcal{L}_{CE}$, as $\mathcal{L}_{Lip}$ is generally much larger on its own.