Advanced ML PA4

Muhammad Huraira Anwar [1] **Adeen Ali Khan** [2]

Abstract

Federated Learning (FL) has emerged as a critical paradigm for distributed machine learning, enabling collaborative model training while preserving data privacy. This report explores the challenges of heterogeneity, robustness, and fairness in FL, with a focus on addressing the generalization issues arising from non-IID data distributions across clients. Key algorithms, including FedAvg, FedSGD, SCAFFOLD, Gradient Harmonization (FedGH), and Sharpness-Aware Minimization (FedSAM), are implemented and evaluated under varying degrees of data heterogeneity. Experimental results demonstrate the performance degradation of FedAvg with increasing heterogeneity and highlight the effectiveness of advanced approaches like SCAFFOLD and FedSAM in mitigating local drift and improving convergence. Novel insights into gradient conflicts and flatness measures are also provided, offering a comprehensive understanding of their implications for model generalization. This work underscores the importance of algorithmic adaptations to enhance the robustness and efficiency of FL systems under real-world constraints.

Introduction

Federated Learning (FL) has transformed distributed machine learning by offering a privacy-preserving framework for collaborative model training without requiring centralized data aggregation. This approach addresses critical challenges associated with data sensitivity, privacy regulations, and ownership constraints, making FL a vital solution for applications such as personalized healthcare, predictive text, and smart city systems. By enabling multiple decentralized entities to collaboratively learn a global model while keeping data localized, FL represents a significant advancement in the democratization of machine learning. However, despite its promise, FL faces significant obstacles stemming from the non-IID nature of data across clients, which introduces inconsistencies, client drift, and difficulties in achieving robust generalization across diverse datasets.

This report focuses on investigating and addressing the challenges posed by data heterogeneity in FL, with an emphasis on improving the generalization capabilities of the global model. The non-IID nature of FL datasets creates complex scenarios where classical algorithms like FedAvg, while effective under homogeneous conditions, fail to maintain performance when faced with varying data distributions. To address these limitations, we evaluate foundational methods such as FedSGD alongside advanced algorithms like SCAFFOLD, Gradient Harmonization (FedGH), and Sharpness-Aware Minimization (FedSAM). Each of these methods introduces novel strategies to mitigate challenges such as client drift, gradient conflicts, and the trade-offs between local and global optimization objectives.

Through theoretical analysis and rigorous experimentation, this report systematically examines the performance of these algorithms under diverse heterogeneity settings. Controlled scenarios, including variations in label skew using Dirichlet distributions, provide insights into the effectiveness of each approach. Additionally, this report explores the inductive biases, communication efficiency, and convergence properties of the algorithms, offering a holistic view of their strengths and weaknesses. By addressing open questions such as the impact of sharpness-aware optimization, the role of gradient harmonization in conflict resolution, and the effectiveness of control variates in alignment, this work aims to advance the field of FL.

The findings presented in this report not only highlight the limitations of existing FL methods under real-world conditions but also offer practical insights for improving their robustness and generalization. By bridging theoretical understanding and empirical validation, this study contributes to the development of more resilient FL systems, paving the way for their deployment in heterogeneous and privacy-sensitive environments.

1. FedSGD vs Centralized

1.1. Methodology

We investigated the equivalence of centralized training and Federated SGD (FedSGD) by comparing gradient updates under various scenarios within a tolerance of 1×10^{-2}.
All experiments were conducted on the MNIST dataset. Across all scenarios, the same hyperparameters were maintained for fair comparisons, including a batch size of 1, a learning rate of 1e-4, and 1 epochs per client per round and a total of 4 rounds with 5 clients. Keeping these parameters constant was crucial to isolate the effects of data distribution, aggregation techniques, and structural differences between

centralized and federated training. By standardizing the training setup, any observed deviations in update magnitudes can be attributed solely to the experimental conditions (e.g., uniform vs. non-uniform distributions or weighted aggregation) rather than differences in hyperparameter configurations. This consistency ensures a robust evaluation of the equivalence between centralized and federated training approaches.

Initially, data was partitioned non-uniformly using a Dirichlet distribution ($\alpha = 0.8$) to simulate heterogeneity. Centralized training involved training a global model on the combined dataset, while FedSGD aggregated client updates without weighting. Gradient magnitudes were computed and compared after each round.

Next, the data was partitioned equally among clients to remove heterogeneity, ensuring deterministic and uniform distributions. Both centralized and federated training were repeated, comparing gradients for equivalence.

Weighted aggregation was introduced to FedSGD, where client updates were scaled based on their data proportions. Finally, weighted aggregation was applied to a uniform data distribution, with all clients having equal weights. In all setups, gradient magnitudes were analyzed after each round, visually compared, and validated for equivalence. This streamlined approach ensured a thorough evaluation of the factors affecting gradient equivalence between centralized and federated training.

1.2. Results

1.2.1. Results for Non-Uniform Data Distribution (Dirichlet)

Round	Centralized Update Magnitude	FedSGD Update Magnitude
1	336.18	336.18
2	460.45	460.45
3	16963.09	16963.18
4	7113.69	7113.47

Table 1. Comparison of centralized and FedSGD update magnitudes for non-uniform data distribution.

1.2.2. Results for Uniform Data Distribution

Round	Centralized Update Magnitude	FedSGD Update Magnitude
1	274.65	274.65
2	374.67	374.67
3	4553.07	4553.08
4	265180.93	265179.38

Table 2. Comparison of centralized and FedSGD update magnitudes for uniform data distribution.

1.2.3. Results for Weighted Aggregation

Round	Centralized Update Magnitude	FedSGD Update Magnitude
1	336.18	84.01
2	460.45	66.58
3	16963.09	87.81
4	7113.69	143.29

Table 3. Comparison of centralized and FedSGD update magnitudes for weighted aggregation.

1.2.4. Results for Weighted Aggregation with Uniform Data Distribution

Round	Centralized Update Magnitude	FedSGD Update Magnitude
1	336.18	67.24
2	460.45	58.96
3	16963.12	75.37
4	7113.70	113.89

Table 4. Comparison of centralized and FedSGD update magnitudes for weighted aggregation with uniform data distribution.

1.3. Discussion

With uniform data distribution (without weighted aggregation), the update magnitudes from FedSGD and centralized training consistently fell within the defined tolerance value of 1×10^{-2}.This outcome aligns with theoretical expectations, as uniform data distribution ensures each client contributes an equal proportion of data, replicating the centralized training scenario. Additionally, deterministic partitioning eliminated heterogeneity, resulting in equivalence between the two methods. By uniformly distributing data, the client datasets are brought closer to an IID (Independent and Identically Distributed) condition, reducing variability across clients and ensuring consistency in gradient updates (see Fig 1).

For non-uniform data distribution (Table 1), the update magnitudes for centralized and FedSGD training were very close but still exceeded the tolerance value. This discrepancy arises due to floating-point precision errors, which accumulate over iterative gradient updates and averaging, and intrinsic randomness in PyTorch operations, where low-level operations (e.g., CuDNN functions) introduce non-deterministic behavior even when seeds are set. Furthermore, data heterogeneity results in variations in data proportions across clients, leading to imbalanced gradient contributions in FedSGD compared to centralized training. These factors collectively highlight the challenges of reproducing centralized training dynamics under heterogeneous conditions (Fig 2).

To address the imbalance caused by data heterogeneity, weighted aggregation was introduced. By scaling client updates based on their data sizes, the method theoretically aligns FedSGD with centralized gradient averaging. However, significant deviations persisted(Table 3) in practice

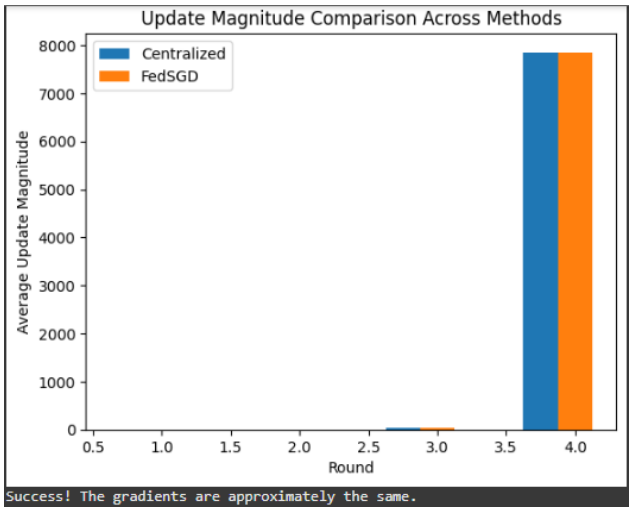

Figure 1. Comparison of update magnitudes between centralized training and FedSGD under uniform data distribution. The results show equivalence, with update magnitudes consistently within the tolerance value, validating the theoretical alignment between the two methods.

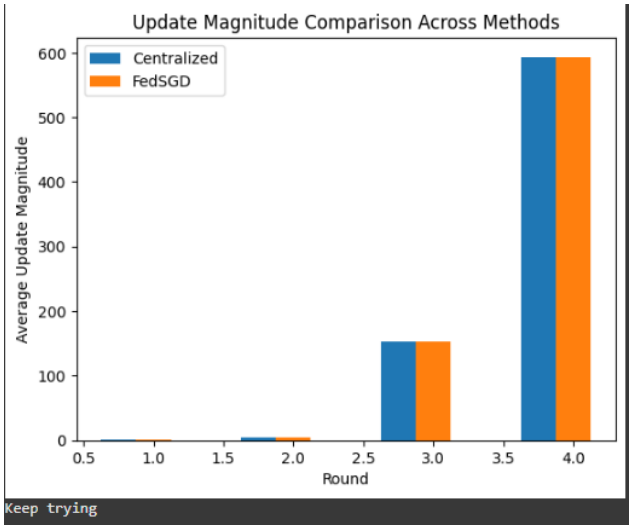

Figure 2. Comparison of update magnitudes between centralized training and FedSGD under non-uniform data distribution, showing close alignment but not within the tolerance value(the tolerance value is too small too should be increased to allow slight randomness due to the intrinsic non deterministic operations)

due to gradient distribution disparities, where gradients from heterogeneous data differ across clients, failing to fully replicate centralized training. Accumulated rounding and floating-point errors during scaling amplified these discrepancies over successive rounds. Additionally, the sequential nature of centralized training, where data is processed deterministically, contrasts with the parallel updates in FedSGD, introducing structural differences in parameter updates (Fig 3). These factors made it impossible to achieve equivalence between FedSGD and centralized training under non-uniform data distribution.

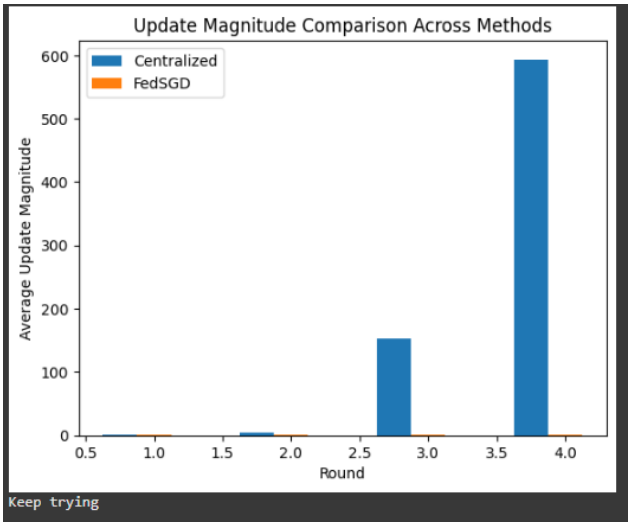

Figure 3. Comparison of update magnitudes for centralized training and FedSGD with weighted aggregation, highlighting significant deviations due to gradient disparities and accumulated errors.

Weighted aggregation combined with uniform data distribution, where clients had equal data proportions and weights, was theoretically equivalent to centralized training as uniformity eliminates heterogeneity and equal weights align with centralized gradient averaging. However, significant deviations between centralized and FedSGD update magnitudes were observed (Table 4) and (Fig 4).

These deviations arose primarily due to structural differences in how gradients are computed and applied. In FedSGD, gradients are computed independently across clients and aggregated, lacking the sequential processing inherent in centralized training. This independence introduces compounded floating-point discrepancies during gradient summation and scaling, which accumulate over multiple rounds. Additionally, minor variations in gradient directions and magnitudes across clients amplified over successive updates, particularly with more clients. Despite aligning FedSGD closer to centralized training in theory, these structural and computational differences prevented complete equivalence in practice.

The overall thought process began with uniform data distribution, hypothesizing that uniformity eliminates heterogeneity, and results validated this by showing equivalence between FedSGD and centralized training. Under non-uniform data distribution, observed deviations due to heterogeneity and intrinsic randomness highlighted the need for a correction mechanism. Weighted aggregation was introduced to

address client imbalances, but practical deviations occurred due to gradient disparities, floating-point errors, and non-determinism. Finally, combining weighted aggregation with uniform distribution aimed to make FedSGD equivalent to centralized training. While this approach theoretically aligned with centralized training, practical deviations persisted due to computational and structural differences.

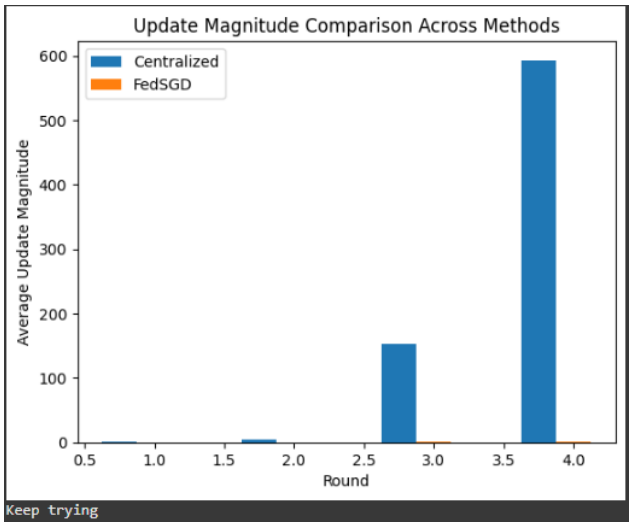

Figure 4. Centralized vs Weighted Aggregation with Uniform Data Distribution.

The experiments emphasize the impact of both data distribution and computational randomness on the equivalence of centralized and federated training. Uniform data distribution without weighted aggregation proved sufficient to produce results within the tolerance value, closely mirroring centralized training. However, even theoretically equivalent setups, such as weighted aggregation with uniform distribution, exhibited deviations due to computational limitations and intrinsic randomness in federated systems. These findings underscore the practical challenges of achieving complete equivalence in federated learning setups.

2. FedAvg

2.1. Methodology

In this task, we implemented FedAvg which is a widely used algorithm in federated learning that enables collaborative training of machine learning models across multiple decentralized devices without sharing raw data. The method operates by selecting a subset of clients in each communication round to perform local training on their private data using the current global model parameters received from the server. Each client updates the model by optimizing its local objective using a small number of stochastic gradient descent (SGD) steps. The locally trained model parameters are then sent back to the server, which aggregates them by computing a weighted average based on the clients' data sizes. This updated global model is then broadcast to the clients for the next round of training. By leveraging local computation and periodic communication, it achieves efficient model training while preserving data privacy, though it faces challenges in scenarios with heterogeneous data distributions due to gradient drift and client variability.

We used MNIST dataset for training and testing with **5** clients for **3** communication rounds and **20** epochs local training at each client, with a batch size of 128 and learning rate of **0.001** for **3** different heterogeneity levels which were determined by alpha parameters in the Dirichlet data distribution i.e. **2,0.5,0.1**.

For the extreme scenario, we implemented that with the same hyperparameters except that there were **10** clients, and each was trained on only one digit of the MNIST dataset (all were trained on a different digit). Then, we used GRAD-CAM for visualizing the features / local areas of the image , the separately trained models are focusing on, to give a more visual interpretation.

2.2. Results

Heterogeneity level	Round 1	Round 2	Round 3
$\alpha = 2$	20.05	48.75	61.02
$\alpha = 0.5$	21.5	42.14	53.06
$\alpha = 0.1$	9.95	22.34	32.56
Extreme	10.64	10.75	10.63

Table 5. Comparison of normal FedAvg roundwise at different heterogeneity levels with FedAvg in the extreme scenario roundwise (10 clients)

2.3. Discussion

As shown in the figure below, in IID case ($\alpha = 2$), the model performs the best, but as the heterogeneity level increases, the model accuracy drastically drops. This happens because the divergence between local updates grows with data heterogeneity, causing the global model to converge sub-optimally or more slowly. This divergence results in a mismatch between the aggregated global model and the underlying distribution of each client's data, further exacerbating performance degradation in heterogeneous settings. Additionally, the global model's inability to capture the nuances of each client's local data distribution leads to increased variance in updates, hindering the model's ability to generalize effectively across clients.

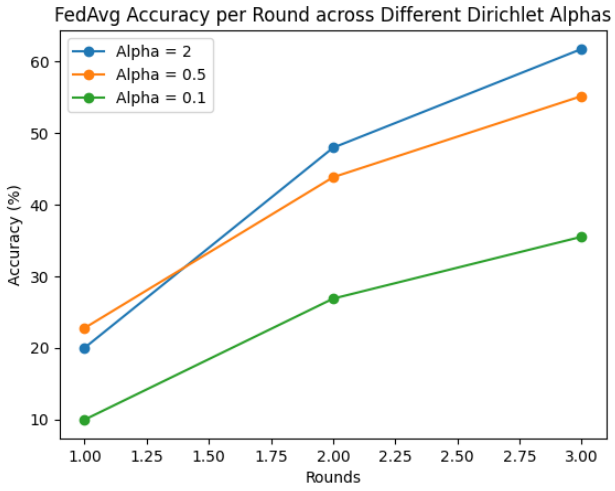

Figure 5. Model accuracies for 3 rounds of FedAvg for different values of Dirichlet alpha which corresponds to heterogeneity in the data distribution

2.3.1. Extreme scenario with each client having data of only one class

In this extreme scenario, where each client trains only on a single class of data, the local models will likely overfit to class-specific features rather than learning generalized representations as each client trains for only a single class. During federated training, the global model will aggregate weights from all the local models. However, since each local model is heavily biased toward a single class, the aggregated model may struggle to converge to a generalized representation.

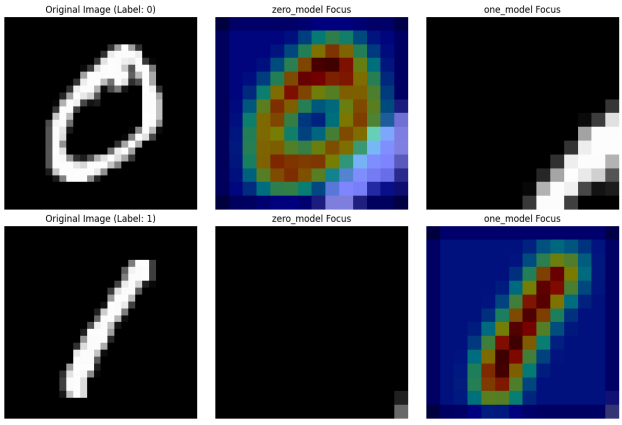

Figure 6. Areas where the different clients (first client trained on '0's , other trained on '1's) focus on

As shown in the above figure, the zero_model client focuses on the whole '0' digit image while the one_model client focuses on the area that is relatively straight, and might resemble the digit '1'. Moreover, in such a scenario, with balanced test sets across all clients, as each of the 10 clients is trained on one of the 10 digits, on average, they yield **10%** accuracy as shown in the Table 5, hence showing that they can only classify correctly the digit they were trained on.

2.3.2. Impact of Permutation Invariance in FedAvg

Yann's argument highlights a valid concern about permutation invariance and how it could complicate the averaging of model weights in FedAvg. The key point is that the filters or weights in the layers of two models trained on different clients might not align semantically, even if they approximate the same function. For example, in a CNN, one client might have a filter that learns edges, while another client learns textures, with the corresponding filters reversed. If these models are averaged, their filters won't align properly, potentially resulting in a meaningless or suboptimal averaged model.

However, the flaw in Yann's argument lies in the assumption that the exact correspondence of filters or weights between models is always necessary. In FedAvg, the goal isn't just about blindly averaging weights, but about finding a meaningful global model representation that generalizes across different data distributions. The optimization landscape pushes different clients' models towards a common, meaningful representation. It does not assume perfect alignment of the weights between clients. Instead, it averages them based on the assumption that, despite the potential mismatch of individual model parameters, the global function learned by each client should be approximately the same. In practice, the individual local minima (filter arrangements) learned by the clients may differ in terms of the exact weights, but they can still share similar global characteristics. The averaging of these parameters can still result in a useful global model, especially if the models are trained on similar distributions of data. This aligns with the findings from Federated Learning literature, where empirical results show that FedAvg works effectively even when models diverge in their specific weight configurations. Models might have different local optima (like different filters), but FedAvg tends to converge to a global model that represents a compromise between the local models' learned functions.

Moreover, in many cases, the high-level features (e.g. learned representations of edges, textures, etc.) across different clients' models will be similar, even if the exact weights differ. These high-level features tend to generalize well under averaging because they capture the same underlying patterns in the data. While specific filters may be permuted or ordered differently across clients, the global model often learns a robust approximation of the data distribution across all clients. This is why FedAvg can still succeed in practice, even if it doesn't rely on perfect alignment of weights across

clients.

In summary, while Yann's concern about weight mismatch due to permutation invariance is legitimate in a strict theoretical sense, FedAvg works effectively in practice because the local minima learned by different clients, even if permuted, tend to converge to similar global patterns. This empirical success suggests that exact alignment of model weights is not always necessary for effective model aggregation in Federated Learning.

3. Scaffold

3.1. Methodology

In this task, we implemented the Scaffold algorithm as mentioned in the paper, that basically is FedAvg with some control variates that are being learned (if they stay constant at 0, it is FedAvg) . These control variates are used to correct the client drift by accounting for the discrepancies between local updates and the global objective. During each communication round, the server distributes the global model parameters along with the control variates to the participating clients. The clients, in turn, perform local updates while leveraging the control variates to adjust their optimization direction, ensuring that the updates remain aligned with the global optimization goal.

We used MNIST dataset for training and testing with **5** clients for **3** communication rounds and **20** epochs local training at each client, with a batch size of **128** and learning rate of **0.001** for 3 different heterogeneity levels which were determined by alpha parameters in the Dirichlet data distribution i.e. **2,0.5,0.1**.

For the modified Scaffold algorithm, at the start of each communication round, we set each client's local control variate c_i equal to the global control variate c received from the server, instead of maintaining a client specific c_i throughout the rounds. All other parameters were kept the same for fair comparison.

3.2. Results

Heterogeneity lvl	FedAvg	Scaffold	Modified Scaffold
$\alpha = 2$	61.02	69.08	65.68
$\alpha = 0.5$	53.06	57.49	61.25
$\alpha = 0.1$	32.56	40.92	47.25

Table 6. Comparison of model accuracies after 3 rounds for different heterogeneity levels of FedAvg, Scaffold, and modified Scaffold algorithms

3.3. Discussion

As shown in the figure below, in IID case ($\alpha = 2$), the model performs the best, but as the heterogeneity level increases, the model accuracy drastically drops due to some models possibly overfitting to the data they see, and that reflects in accuracy of global model.

However, Scaffold performs way better than FedAvg, especially at greater levels of heterogeneity, it experiences an increase of approx **8.5** % at $\alpha = 0.1$ due to its ability to correct client drift through the use of control variates. These control variates help align the local updates with the global direction, mitigating the impact of non-iid data distributions. Consequently, Scaffold maintains more stable and accurate convergence even in extreme scenarios compared to FedAvg.

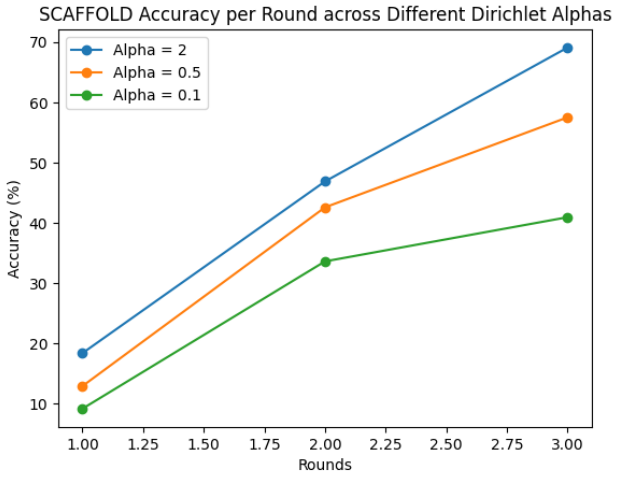

Figure 7. Model accuracies for 3 rounds of Scaffold for different values of Dirichlet alpha which corresponds to heterogeneity in the data distribution

3.3.1. Modified Scaffold Algorithm

As shown in the figure below, in IID case ($\alpha = 2$), the model performs the best, but as the heterogeneity level increases, the model accuracy drastically drops due to some models possibly overfitting to the data they see, and that reflects in accuracy of global model.

The modified Scaffold algorithm also performs better than FedAvg, with an even higher increase in accuracy of approx **15** % at $\alpha = 0.1$ due to its simplified control variate mechanism, where the local control variates are reset to match the global control variate at each round. This modification reduces the overhead of maintaining client-specific control variates while ensuring a consistent alignment with the global updates. As a result, it demonstrates improved performance in heterogeneous scenarios by effectively mitigating client drift, though it may slightly compromise the ability to adapt to highly IID data distributions compared to the original Scaffold algorithm, as we observe a drop in accuracy of about **4** % at ($\alpha = 2$) in the modified algorithm.

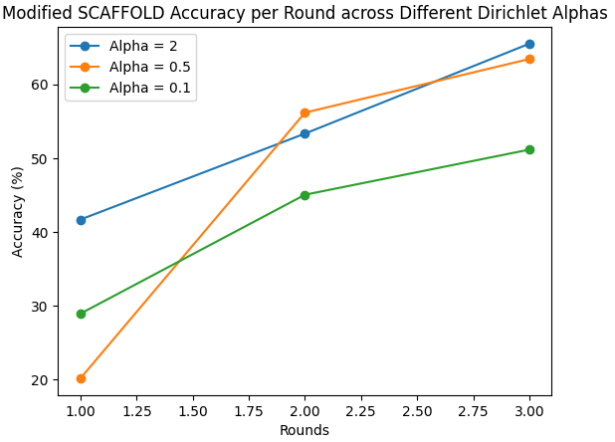

Figure 8. Model accuracies for 3 rounds of modified Scaffold for different values of Dirichlet alpha which corresponds to heterogeneity in the data distribution

4. Gradient Harmonization

4.1. Methodology

In this task, we implemented the Gradient Harmonization algorithm as mentioned in the paper, that basically is FedAvg in addition to harmonizing the gradient conflicts after getting the local client updates per round. It addresses conflicts that arise, where gradients from different tasks point in conflicting directions (¿90 degree difference in gradients). Such conflicts can hinder the optimization process, leading to suboptimal performance across clients. It dynamically aligns gradients to ensure that the updates to the model parameters are more harmonious. This harmonization is particularly beneficial in scenarios involving diverse data or imbalanced data at clients, as it helps to maintain a fair balance in learning the global model. Hence, it improves convergence and enhances the model's ability to generalize across multiple clients with heterogeneous distributions effectively.

We used MNIST dataset for training and testing with **5** clients for **3** communication rounds and **20** epochs local training at each client, with a batch size of **128** and learning rate of **0.001** for 3 different heterogeneity levels which were determined by alpha parameters in the Dirichlet data distribution i.e. **2,0.5,0.1**.

Moreover, we also plot the gradient conflicts pre and post harmonization across multiple rounds, and for different heterogeneity levels in order to understand and make sense of the gradient harmonization algorithm, and its process, and how it leads to convergence.

4.2. Results

Heterogeneity lvl	FedAvg	Gradient Harmonization
$\alpha = 2$	61.02	63.81
$\alpha = 0.5$	53.06	60.32
$\alpha = 0.1$	32.56	60.52

Table 7. Comparison of model accuracies after 3 rounds for different heterogeneity levels of FedAvg, and FedAvg with Gradient Harmonization

4.3. Discussion

As shown in the figure below, in IID case ($\alpha = 2$), the model performs the best, but as the heterogeneity level increases, the model accuracy drops very little , about **2.5**% only, in comparison to the stark drops in almost all other algorithms.

The Gradient Harmonization approach performs better than FedAvg, with a huge increase in accuracy of approx **28** % at $\alpha = 0.1$ due to its ability to align and balance gradient updates from clients with heterogeneous data distributions. This ensures that the global model update direction is not dominated by any particular client, effectively mitigating the adverse effects of heterogeneity. For IID data, however, the increase in accuracy is less significant as the clients already contribute well-aligned gradient updates, hence there are lesser conflicts, making the benefits of harmonization less pronounced. Nonetheless, the method demonstrates its strength in handling challenging non-IID scenarios, achieving robust convergence and improved generalization.

As shown in the graph, the pre harmonization conflicts generally go on decreasing as the rounds increase for IID data, as it is reaching convergence. However, that is not the case for extreme heterogeneous data distributions, the conflicts increase pre harmonization. On the other hand, for all alphas, the post harmonization conflicts always decrease as the rounds go by since the client's gradients are being harmonized, and the global model starts converging.

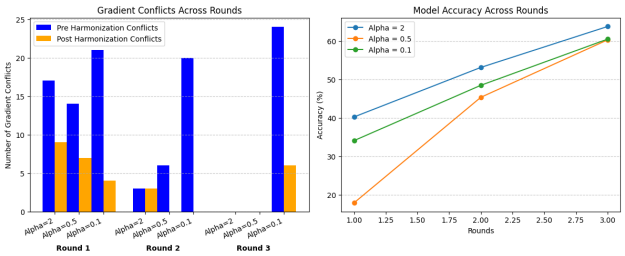

Figure 9. Model accuracies for 3 rounds of Gradient Harmonization for different values of Dirichlet alpha which corresponds to heterogeneity in the data distribution, and pre and post harmonization gradient conflicts at each round for each value of alpha

4.3.1. Does harmonized gradient always lead to minimizing local loss?

In Gradient Harmonization, the goal is to align conflicting gradients to reduce divergence in the direction of updates from multiple clients. However, whether the harmonized gradients will always lead to a minimization of each client's local objective is not guaranteed.

Since it adjusts the gradients to reduce conflicts, it may minimize the global objective in a way that this new direction may not always correspond to a decrease in the local loss for all clients. Moreover, the loss function of a client typically has a complex landscape, and the client's local gradient direction is ideally pointing toward a local minimum (or at least a descent direction). By adjusting this gradient to harmonize it with others, you might move the update direction away from the local minimum, causing an increase in the local objective for that client. If the local objective is heavily affected by local features or noise that the global model doesn't account for, the harmonized direction could lead to an increase in the client's local loss.

In some cases where the clients have significantly different data distributions or the harmonization is too aggressive as shown in diagram (a) of Figure 10, the new gradient direction might lead to a higher local loss for some clients, as it might pull the model away from a local minimum of the client's objective.

Hence, no, it is not guaranteed that the harmonized gradient direction will always lead to a minimization of the client's local objective.

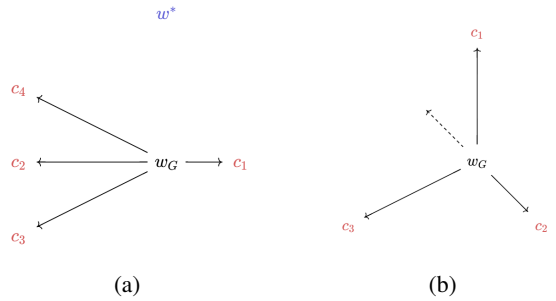

Figure 10. Example images of different gradient conflicts. (a) shows a case where the harmonized gradient may result in increase of local loss for a client whereas (b) shows how 3 clients with significantly different gradients can result in a relatively small gradient update.

4.3.2. Effects and Implications of harmonizing extremely conflicting gradients

In the scenario where each client's data distribution is vastly different, causing each client to learn a highly unique model, the effect of gradient harmonization can be significant. If the gradients are heavily conflicted (i.e., the angle between them is large), it will have important implications on the magnitude of the final harmonized gradient. Consider diagram (b) in Figure 10 which represents 3 clients, each having gradients that are separated by about 120° from one another.

If we harmonize these gradients, the overall direction of the harmonized gradient is likely to be a compromise between the three client gradients. The magnitude of the final harmonized gradient would likely be relatively small because the conflicting gradients are pushing in very different directions. It would essentially be the average of the three gradients, which can result in the components canceling each other out to some extent, leading to a smaller resultant vector. The exact magnitude will depend on the number of clients, the specific angle between their gradients, and how well the harmonization algorithm aligns them.

It has multiple implications. For example, if the harmonized gradient is small, the update to the global model will be smaller, hence resulting in slower convergence of the global model because the harmonized gradients don't point strongly in one direction. If each client's local model is highly specialized to their own data, the harmonized gradients might cancel each other out, resulting in no substantial global update. Over several rounds of federated training, this might lead to the global model stagnating or converging to a poor solution because it is unable to effectively integrate the unique knowledge from each client.

Furthermore, clients with highly unique models might experience distorted updates to their local models due to harmonization, causing them to deviate from their own optimal paths. This could lead to an overall increase in local losses for some clients, as they are forced to adapt to a global direction that doesn't fully align with their local data distribution.

To summarize, when gradients are highly conflicted, the final harmonized gradient is likely to have a small magnitude, which in turn reduces the effectiveness of the global model update. This can slow down convergence and prevent the global model from effectively integrating information from all clients, especially when their local data distributions are very different.

5. Sharpness Aware Minimization

5.1. Methodology

In this task, we implemented the FedSAM algorithm as mentioned in the paper, that basically is FedAvg with Sharpness-Aware Minimization (SAM) applied locally on each client to improve generalization by smoothing the loss landscape. By applying SAM locally on each client, it adjusts model updates to minimize both the loss and the sharpness of the loss landscape, resulting in smoother and more robust optimization. This approach mitigates the adverse effects of non-IID data distributions and helps achieve better conver-

gence across diverse clients in federated learning.

We used MNIST dataset for training and testing with **5** clients for **3** communication rounds and **20** epochs local training at each client, with a batch size of **128** and learning rate of **0.001** for 3 different heterogeneity levels which were determined by alpha parameters in the Dirichlet data distribution i.e. **2,0.5,0.1**.

In exploring alternative measures of sharpness for model generalization, one promising avenue is to use the surrogate gap as a measure of sharpness which quantifies the difference between the maximum loss within a neighborhood of a minimum and the loss at the minimum itself. Unlike conventional sharpness metrics that are based on the Hessian or gradient magnitude, the surrogate gap better captures the flatness of the minimum, providing a clearer indication of generalization potential. In the context of FedSAM, which uses a minimax optimization approach, incorporating a surrogate gap measure could potentially improve the estimation of "flatness" and lead to better generalization by more effectively avoiding sharp minima. This approach considers not just the steepness or sharpness of the minimum but also the spread of the loss function in the surrounding area, which can provide a better understanding of how the model might perform in diverse data environments (such as federated learning with heterogeneous clients).

To this end, we used the loss function as mentioned in the Sharpness Aware Gradient Matching (SAGM) paper by P. Wang et al. [6], and we kept all the parameters same (for fair comparison) except the way we do the weight perturbations according to the new loss function. We finetuned to find the optimal other approach-specific hyperparameters as well. For both Fed-SAM, and Fed-SAGM, the perturbation radius ρ was set to **0.001** for fair comparison. Moreover, there is another additional hyperparameter α used only in FedSAGM which limits the perturbation magnitude and finetunes it, providing a more nuanced control over the sharpness penalty. We found optimal results when α was set to **0.01**. (more on this in the discussion section).

5.2. Results

Heterogeneity lvl	FedAvg	FedSAM	FedSAGM
$\alpha = 2$	61.02	66.38	83.35
$\alpha = 0.5$	53.06	63.32	74.95
$\alpha = 0.1$	32.56	49.09	60.94

Table 8. Comparison of model accuracies after 3 rounds for different heterogeneity levels of FedAvg, and FedSAM, and FEDSAGM

5.3. Discussion

As shown in the figure below, in IID case ($\alpha = 2$), FedSAM performs the best, but as the heterogeneity level increases, the model accuracy drops quite a lot , about **17**%. It performs better than FedAvg, with a significant increase in accuracy of approx **16.5** % at $\alpha = 0.1$ due to its ability to better handle data heterogeneity by adapting to uneven distributions across clients. FedSAM's use of a more sophisticated local update mechanism which finds flatter minima in the loss landscape, in addition to the incorporation of a server-side model correction step significantly reduces the effects of data distribution disparities, allowing it to maintain better performance under non-IID conditions.

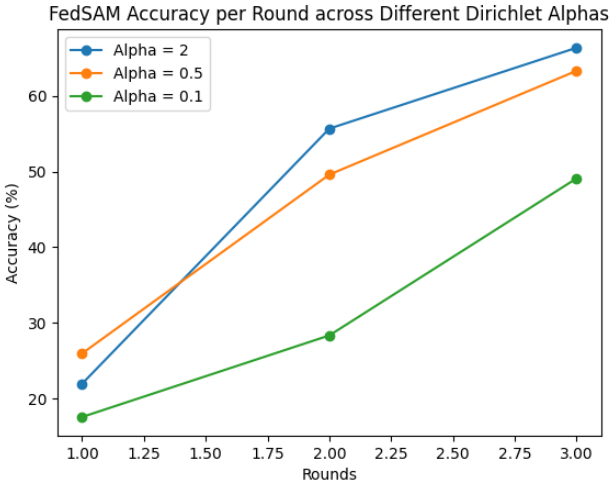

Figure 11. Model accuracies for 3 rounds of FedSAM for different values of Dirichlet alpha which corresponds to heterogeneity in the data distribution

5.3.1. Why might a flatter minimum lead to a more generalizable solution than a sharp minimum?

In the context of FedSAM, the approach of perturbing the model weights in the direction of gradient ascent followed by gradient descent serves to improve generalization by avoiding sharp minima. This idea hinges on the fact that the sharpness of a minimum in the loss landscape has a strong impact on the model's ability to generalize. Here's why a flatter minimum can lead to a more generalizable solution compared to a sharp minimum.

In a flatter minimum, the loss function doesn't change drastically when the model weights are slightly perturbed. The loss surface is more "broad" with a relatively smaller gradient, which means small perturbations in the input or model parameters won't cause significant changes in the model's predictions. This indicates that the model has stabilized and is more robust to small variations in input data or noise.

In contrast, a sharp minimum occurs when the loss function exhibits a sharp, steep drop-off in a narrow region. This means that small perturbations in the input or model weights lead to large changes in the loss or the model's output. Models that converge to sharp minima overfit to the training data,

as they rely too heavily on specific, narrow features of the training data and are sensitive to slight changes.

Therefore, a flatter minimum is more generalizable than a sharp minimum because it leads to less sensitivity to perturbations, avoiding overfitting to specific features of the training data, and improving robustness to noise. In FedSAM, the minimax optimization process helps ensure the model converges to such a flat, broad minimum, improving its ability to generalize to unseen data and perform robustly in real-world scenarios.

5.3.2. How does a flatter minimum help in data heterogeneity problem?

Data heterogeneity refers to the situation where different clients hold different distributions of data, which can lead to significant challenges in training a shared global model. The flatter minimum property of FedSAM can help address several of these challenges.

In federated learning, each client's data can have different distributions due to factors such as demographic variations, different sensor types, or different environments (e.g., in healthcare, one hospital's data may differ from another's). This heterogeneity often leads to divergent gradients and inconsistent updates during local training, which can cause poor model convergence or suboptimal performance when aggregating the updates. A model trained in a flatter region of the loss landscape tends to be less sensitive to local variations in data. This helps mitigate the problem of overfitting to specific local data distributions and makes the model more robust to data heterogeneity. By avoiding sharp minima, the model avoids becoming overly specialized to any one client's data, enabling it to perform better across a variety of clients with different data distributions. Since flatter regions tend to have more consistent gradient directions across different parts of the loss surface, which leads to more stable updates during the federated training process, and ultimately, smoother convergence.

Some clients may have more data or more representative data, and they may dominate the training process, causing the model to converge toward their specific data distribution. This can lead to a biased global model that does not represent the other clients well. A flatter minimum encourages the model to balance the contributions from all clients by avoiding sharp local minima that are biased toward the data of any one client. FedSAM's perturbation strategy prevents any single client's local data from having too much influence on the global model, ensuring that the global model is more representative of the overall client population.

To conclude, the flatter minimum property in FedSAM is particularly beneficial in addressing the challenges of data heterogeneity in federated learning. By encouraging the model to avoid sharp minima and instead focus on broader, more stable regions of the loss surface, FedSAM improves generalization robustness and stability across clients with diverse data distributions. This results in a more balanced representative and efficient global model even when clients' data distributions differ significantly.

5.3.3. Comparison with alternate measure of sharpness to measure flatness of the region

As mentioned by P. Wang et al. [6], the alternate measure of sharpness is through Sharpness Aware Gradient Matching (SAGM), which minimizes the empiricial loss, perturbed loss, as well as surrogate gap using this loss function (where α is a hyperparameter unique to FEDSAGM which enables fine-tuning of the perturbation magnitude, providing a more nuanced control over the sharpness penalty):

$$\min_w \mathcal{L}_i(w) + \mathcal{L}_i\left(w + \left(\frac{\rho}{\|\nabla\mathcal{L}_i(w)\|} - \alpha\right)\nabla\mathcal{L}_i(w)\right).$$

So basically, the perturbation / flatness measure, in FEDSAGM, is simplified as follows:

$$\rho\frac{\nabla\mathcal{L}_i(w)}{\|\nabla\mathcal{L}_i(w)\|} - \alpha\nabla\mathcal{L}_i(w).$$

Whereas, the first order approximation of flatness measure in FEDSAM is as follows:

$$\rho\frac{\nabla\mathcal{L}_i(w)}{\|\nabla\mathcal{L}_i(w)\|}$$

In FedSAM, the perturbation is determined solely by the normalized gradient direction, scaled by the perturbation radius ρ. This simple approach does not account for variations in the gradient magnitude. In contrast, FedSAGM introduces a corrective term $-\alpha\nabla\mathcal{L}_i(w)$, which adjusts the flatness measure, particularly reducing the perturbation when the gradient magnitude is large. FedSAGM also explicitly incorporates the surrogate gap into its optimization objective, ensuring smoother convergence by aligning the perturbed loss with the empirical loss, unlike FedSAM, which does not explicitly address this gap.

While FedSAM relies only on ρ, FedSAGM introduces an additional hyperparameter α, providing more control over the optimization process. This makes FedSAGM more rigorous in minimizing sharpness and achieving better generalization, particularly in non-IID federated settings with diverse client loss landscapes. Moreover, the corrective term in FedSAGM enhances training stability by mitigating overshooting from large gradients. However, these advantages come at the cost of increased computational complexity and hyperparameter tuning. In contrast, FedSAM is computationally efficient and simpler, making it practical in

resource-constrained scenarios, though it may struggle in capturing the intricacies of heterogeneous loss landscapes.

As shown in the figure below, in IID case ($\alpha = 2$), FedSAGM performs the best, but as the heterogeneity level increases, the model accuracy drops quite significantly , about **23**%. It performs better than FedAvg, with a significant increase in accuracy of approx **32.5** % at $\alpha = 0.1$ due to its ability to better handle data heterogeneity by adapting to uneven distributions across clients, and doing even better than FedSAM due to its enhanced loss function that also minimizes the surrogate gap in addition to minimizing empirical loss and perturbed loss which FedSAM was doing already. Moreover, FedSAGM explicitly takes into account the sharpness of the loss landscape, encouraging flatter minima, which contributes to improved generalization and stability in heterogeneous scenarios, compared to FedSAM's more basic adaptation mechanisms. This further enables FedSAGM to maintain high performance even with higher data distribution imbalances.

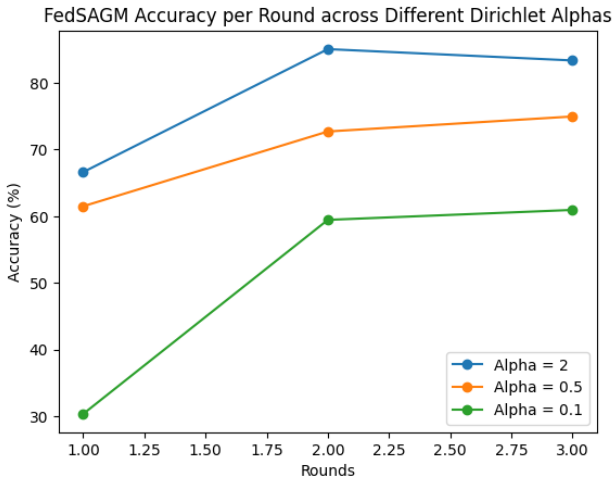

Figure 12. Model accuracies for 3 rounds of FedSAGM for different values of Dirichlet alpha which corresponds to heterogeneity in the data distribution

6. Analysis

6.1. Centralized

Centralized training aggregates the entire dataset into a single location, ensuring direct access to all data points during training. This approach is naturally robust to data heterogeneity as it eliminates client-specific biases, enabling the model to generalize effectively across diverse data distributions. Centralized training is highly efficient in terms of convergence, as gradient updates are computed globally, representing the overall objective directly. This leads to faster and more stable convergence compared to distributed methods. Moreover, since it isn't a distributed method, there are no communication rounds, hence it is completely efficient in those terms . However, centralized training is often impractical in federated scenarios due to privacy and data ownership concerns, as clients cannot share their raw data. One other concern is the initial data transfer to a central server can be expensive in terms of bandwidth and storage, making it infeasible for large-scale or privacy-sensitive applications.

6.2. FedSGD

FedSGD preserves client privacy by transmitting only gradient updates instead of raw data, making it a suitable approach for federated learning. It indirectly addresses heterogeneity by aggregating gradients computed locally on client datasets. While FedSGD works well in scenarios with uniform data distribution, it struggles as data heterogeneity increases. Divergent client gradients lead to inconsistent updates when aggregated, slowing convergence and reducing the global model's generalization. Communication efficiency is another challenge for FedSGD. Although it avoids raw data transfer, frequent synchronization between clients and the central server introduces significant communication overhead, especially with many clients or communication rounds. In terms of convergence, FedSGD is slower than centralized training, particularly under heterogeneous conditions, as it cannot fully represent the global objective due to inconsistent client contributions. This highlights the need for additional mechanisms, such as weighted aggregation or gradient harmonization, to address these challenges in federated learning.

6.3. FedAVG

FedAvg, as the baseline federated learning algorithm, struggles with handling increased levels of data heterogeneity. As the heterogeneity level increases (i.e., as the data distribution becomes more uneven across clients), the performance of FedAvg drops significantly. This is because FedAvg assumes that all clients have the same distribution of data, leading to mismatches in the local updates and poor model generalization. In terms of communication efficiency, FedAvg can be relatively efficient since it only requires clients to send model updates (i.e., weight updates) to the server after each round as compared to other federated approaches. However, this communication is less efficient when there is significant heterogeneity, as the server must aggregate highly divergent updates from clients, which could lead to slower convergence. Regarding convergence rates, FedAvg typically exhibits slower convergence, especially in heterogeneous scenarios. Empirical evaluations show that as the heterogeneity increases, convergence becomes slower due to the model's struggle to reach an optimal global state. In highly heterogeneous environments, FedAvg may require more rounds of communication to achieve satisfactory per-

formance compared to algorithms designed to handle such data imbalances.

6.4. Scaffold

Scaffold outperforms FedAvg when handling increased levels of data heterogeneity due to its use of control variates, which help mitigate the variance in client updates caused by skewed local data distributions. By maintaining client-specific control variates across communication rounds, SCAFFOLD reduces the discrepancy between local updates and ensures more stable and consistent model updates, resulting in better performance in heterogeneous environments. In terms of communication efficiency, SCAFFOLD introduces additional communication overhead compared to FedAvg, as it requires each client to send control variates along with model updates. While this extra communication may seem inefficient, it is crucial for improving the accuracy and robustness of the model, especially in settings with significant data heterogeneity. Regarding convergence rates, SCAFFOLD typically converges faster than FedAvg, especially in heterogeneous scenarios. Empirical results show that SCAFFOLD achieves faster convergence and better final accuracy in environments with uneven data distributions, thanks to the reduction in update variance and the more effective aggregation of updates across clients. Although the extra communication burden slightly slows down the convergence compared to FedAvg in scenarios with low heterogeneity, the gains in convergence speed in highly heterogeneous environments outweigh this drawback.

6.5. FedGH

FedGH (Federated Gradient Harmonization) is designed to handle increased levels of data heterogeneity more effectively than traditional algorithms like FedAvg. It utilizes a gradient harmonization technique which results in better model performance in heterogeneous environments, as the algorithm can better cope with the variations in local data distributions. In terms of communication efficiency, FedGH requires the same communication compared to FedAvg, more than Scaffold. When it comes to convergence rates, FedGH shows faster convergence compared to FedAvg, but slower compared to Scaffold particularly in scenarios with severe data heterogeneity, solely due to the extra computational costs of harmonizing the gradients with each other. Due to its gradient harmonization mechanism, it helps mitigate the local fluctuations in model updates, leading to faster stabilization of the global model. However, this advantage comes with a slightly higher computational cost and convergence speed.

6.6. FedSAM

FedSAM excels in handling increased levels of data heterogeneity, as it adapts better to diverse data distributions across clients by reducing the sensitivity of the model to local minima, which is crucial when there are significant differences in data across clients, and settling at flatter minima. In terms of communication efficiency, FedSAM requires same communication compared to FedAvg, more efficiency than Scaffold as the only information exchange is the client updates at end of a round. FedSAM's improved handling of data heterogeneity often results in better overall model performance, justifying the higher convergence rate. Regarding convergence rates, FedSAM typically exhibits slower convergence than all other methods, especially in highly heterogeneous settings. While it converges more steadily and can avoid poor local optima, it requires more rounds of communication to reach convergence compared to simpler methods like FedAvg. Also, the computational costs and time almost double as compared to other cases in FedSAM since it modifies the weight updates during training by considering not only the loss at the current parameters but also how the loss behaves when the parameters are perturbed in the direction of the gradients. This double computation of gradients — first for the unperturbed model and then for the perturbed one — doubles the time of each communication round. While FedSAM might have a slower convergence speed in the early rounds, its long-term performance, especially in heterogeneous scenarios, is superior, leading to higher accuracy in later stages.

Conclusion

In summary, while traditional methods like FedAvg and FedSGD are effective in simpler, homogeneous settings, they struggle significantly as data heterogeneity increases. FedAvg, in particular, suffers from slow convergence and reduced performance when data distributions become uneven across clients. On the other hand, advanced algorithms like Scaffold, FedGH, and FedSAM are specifically designed to handle heterogeneity more effectively. These methods employ sophisticated techniques such as control variates and gradient harmonization to mitigate the impact of diverse data distributions, leading to improved convergence rates and better final accuracy. However, these improvements come with additional communication and computational overheads, making them more resource-intensive compared to FedAvg.

FedSAM stands out in terms of its robustness to data heterogeneity, providing the most stable and accurate results in highly uneven environments. However, its increased computational cost, particularly due to the double gradient computation, results in slower convergence compared to methods like Scaffold and FedGH. Despite this, its ability to

avoid sharp minima and achieve higher accuracy in the long term justifies the extra cost. Overall, the choice of algorithm depends on the specific trade-offs between handling data heterogeneity, communication efficiency, and convergence rates. While more sophisticated methods provide better performance in heterogeneous settings, they come at the expense of increased computational complexity, making them less ideal in resource-constrained environments.

Contributions

Task	Done by
task 1	Adeen
task 2	Adeen
task 3	Huraira
task 4	Huraira
task 5	Huraira
task 6	Adeen

References

1. Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 5132–5143. PMLR, 2020.

2. Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

3. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282. PMLR, 2017.

4. Z. Qu, X. Li, R. Duan, Y. Liu, B. Tang, and Z. Lu. Generalized federated learning via sharpness aware minimization. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*. PMLR, 2022.

5. X. Zhang, W. Sun, and Y. Chen. Tackling the non-iid issue in heterogeneous federated learning by gradient harmonization. *IEEE Signal Processing Letters*, 2024.

6. P. Wang, Z. Zhang, Z. Lei and L Zhang. Sharpness-Aware Gradient Matching for Domain Generalization. *arXiv preprint arXiv:2303.10353*, 2023.