Improving Convergence in Federated Learning through Personalized Model Editing Techniques for Computer Vision Models

Emanuele Romito
s339170@studenti.polito.it

Farnood Sotoudeh
s341275@studenti.polito.it

Pardis Poostpardaz
s343187@studenti.polito.it

Giuseppe Ventrella
s348517@studenti.polito.it

Abstract

*Federated Learning (FL) enables decentralized model training without data sharing, making it well-suited for privacy-sensitive applications. However, convergence remains a major challenge, especially under non-IID (non-independent and identically distributed) data distributions. In this work, we propose **pTaLoS**, a hybrid strategy that combines global model editing (`TaLoS`) with client-level personalization (`pFedEdit`) to accelerate convergence and improve performance in heterogeneous federated environments. Our approach applies personalized model editing on a selected number of clients and a fixed number of layers, leveraging a dynamic, hybrid editing strategy that balances global model alignment with local client-specific data distributions, enabling a smooth transition from global to personalized editing across training rounds (or vice-versa). We evaluate `pTaLoS` using Vision Transformer backbones (ViT-S/16) pretrained with DINO and conduct extensive experiments on CIFAR-100 under varying degrees of statistical heterogeneity. Results suggest that `pTaLoS` consistently improves convergence in non-IID settings, particularly during the early training rounds compared to `TaLoS`. Code and training logs are available in the Appendix.*

1. Introduction

Federated Learning [11] enables decentralized model training while preserving data privacy, making it suitable for sensitive domains. Despite its advantages, FL suffers from slow convergence and performance degradation when client data is *non-IID*, a common real-world scenario [10, 15]. Classical methods like `FedAvg` [11] are known to struggle in these settings due to client drift and inconsistent local updates. To address this, model editing has emerged as an efficient alternative to full retraining [12, 14]. Techniques like `TaLoS` [5] and `pFedEdit` [6] introduce selective parameter updates to adapt models more effectively and with lower computational cost. Building on these ideas, we propose a hybrid approach that combines `TaLoS` and `pFedEdit` to simultaneously address global model alignment and client-level personalization within the non-IID setting, called `pTaLoS`.

2. Background and Related Work

Federated Learning enables multiple clients to collaboratively train a shared global model without sharing local data [11]. The typical architecture of FL involves iterative training rounds comprising local training on client-side data and periodic aggregation at a central server. The `FedAvg` algorithm [11] remains one of the most widely adopted aggregation methods due to its simplicity and efficiency. `FedAvg` aggregates client models by averaging their parameters, significantly reducing the communication overhead compared to classical distributed learning methods. However, despite its popularity, it often struggles with issues of convergence in non-identically distributed (non-IID) data scenarios, which are prevalent in real-world applications [10, 15].

Heterogeneity Issues Real-world federated systems often encounter *statistical and system heterogeneity* [7]. Statistical heterogeneity arises from non-IID data distributions across clients, significantly affecting the convergence speed and stability of FL algorithms [10]. System heterogeneity, on the other hand, refers to variations in client hardware, network conditions, and computational capabilities, potentially slowing down global model synchronization and introducing stragglers [1]. Addressing heterogeneity is crucial as it often leads to divergence in training or suboptimal global performance. Numerous solutions have been proposed, including *proximal terms* that constrain local updates. `FedProx` [9] incorporates a proximal term into the local objective function to explicitly regularize local updates toward the global model, mitigating the negative ef-

fects of data heterogeneity. Additionally, custom *adaptive optimization strategies* like YOGI [13] have been proposed, employing adaptive learning rates that adjust dynamically to varying local data distributions and training progress, further enhancing model convergence in heterogeneous environments. Despite these advances, fully resolving issues caused by severe heterogeneity remains an open challenge.

Personalized FL Training Given the intrinsic heterogeneity among client datasets, personalized FL training methods have emerged as viable solutions. Methods such as pFedEdit [6] propose training strategies where part of the model parameters are personalized per client, thus accommodating unique local data distributions while maintaining global knowledge sharing.

Model Editing Techniques *Model editing* refers to techniques that directly adjust specific model parameters or structures to rapidly adapt to new information or correct undesirable behaviors without full retraining [12,14]. The motivations behind model editing include reducing retraining costs, efficiently correcting model errors, and quickly adapting models to new data distributions. Existing model editing techniques include *sparse fine-tuning*, which selectively adjusts only a subset of model parameters, significantly reducing the computational cost [4]. *Fisher-based* sensitivity analyses have also been proposed, leveraging Fisher information to identify the most impactful parameters to edit, optimizing the effectiveness of updates [8]. Techniques specifically adapted for federated learning scenarios, such as TaLoS [5] which leverages targeted parameter updates to correct specific behaviors.

3. Methodology

To address convergence issues and data heterogeneity in FL, we propose a hybrid methodology that leverages the complementary strengths of two model editing techniques: TaLoS and pFedEdit. Our main idea is to combine these editing approaches across training rounds, initially leveraging TaLoS across the entire set of client models and progressively integrating personalized model edits with pFedEdit as training advances (or vice-versa). TaLoS performs targeted parameter updates, efficiently modifying a subset of model parameters to rapidly correct erroneous behaviours. In contrast, pFedEdit specifically addresses *client-level data divergences* by introducing personalized edits directly during local training. Our hybrid approach, pTaLoS, is motivated by the hypothesis that initiating training with a broad and uniform editing strategy (via TaLoS) effectively captures global model corrections meanwhile introducing increasingly personalized edits (via pFedEdit) allows for fine-grained adaptation to local data distributions.

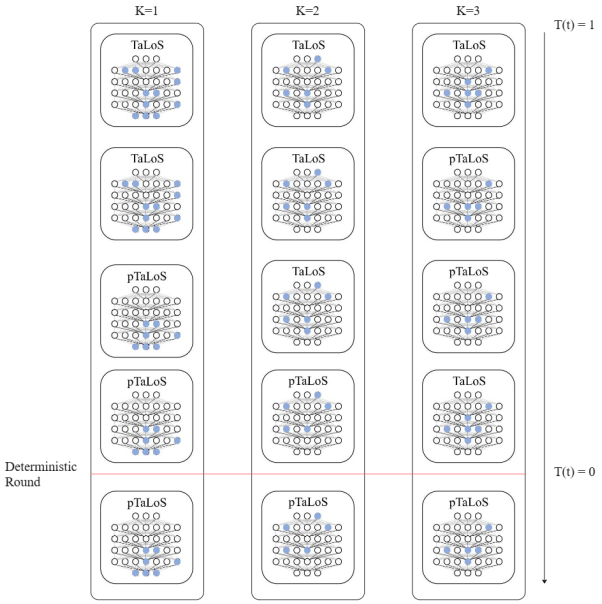

Figure 1. Overview of pTalos in the non-reversed setting. Each column represents a distinct client and each row a communication round. Early rounds (top) would most likely use Talos, while later rounds gradually introduce pTalos based on a decaying threshold function $T(t)$. After the deterministic round (marked in red), clients are going to use exclusively pTalos. Blue nodes represent the subset of layers selected for training.

Alternatively, this paradigm can be reversed: by applying pTaLoS in the initial rounds, we can inject client-specific updates into the global model through selective layer-wise updates. Once a stable and more representative global model has been established, TaLoS can then be used across all clients to consolidate alignment and edit the global model.

Formulation Consider a federated setting with K clients, each with local dataset D_k. Let θ denote the global model parameters, and $\theta_k^{(t)}$ the local model parameters for client k at communication round t. The local optimization problem under FedProx can be defined as:

$$\min_{\theta_k^{(t)}} \mathcal{L}_k(\theta_k^{(t)}, D_k) + \mu \left\| \theta_k^{(t)} - \theta^{(t-1)} \right\|^2$$

where μ is the regularization coefficient and the proximity term $\left\| \theta_k^{(t)} - \theta^{(t-1)} \right\|^2$ penalizes deviations from the global model to stabilize training in non-IID settings. During the initial training rounds, TaLoS is applied to all clients:

$$\theta_k^{(t)} \leftarrow \theta_k^{(t-1)} - \eta \nabla_\theta \mathcal{L}_k(\theta_k^{(t-1)}, D_k)$$

As training advances, we introduce personalized edits using pFedEdit in a selective fashion. To determine which clients apply the hybrid pTaLoS strategy, we define a linearly *decaying* threshold function that starts from 1 and decreases to 0 over time:

$$T(t) = 1 - \frac{t}{T_{\text{max}}}$$

This function controls the probability of applying layer-wise personalized edits: clients are more likely to perform global editing in the early rounds, and increasingly likely to be selected for personalized editing as training progresses. In the reversed paradigm instead, the threshold is instead updated in an *ascending* fashion:

$$T(t) = \frac{t}{T_{\text{max}}}$$

At each round t, a client k samples a random number $u_k \sim \mathcal{U}(0, 1)$. T_{max} is a hyperparameter that defines the deterministic round from which all clients stop stochastically selecting the strategy to use and instead start applying one strategy or another in a deterministic way (i.e., with 100% probability). Beyond this point, the threshold value becomes either 0 or 1, fully enforcing the use of either TaLoS or pTaLoS depending on the chosen scheduling mode. If $u_k > T(t)$, pTaLoS is applied.

Algorithm 1 pTaLoS with Random Sampling

Input: K clients, rounds T, learning rate η, threshold function $T(t)$, selection size k, batch size B
Initialize: global model θ_0

for $t = 1$ to T **do**
  $S_t \leftarrow$ random subset of m clients
  for each client $k \in S_t$ **do**
    Sample $u_k \sim \mathcal{U}(0, 1)$
    if $u_k > T(t)$ **then**
      ClientUpdate(k, θ_t, mode=pTaLoS)
    else
      ClientUpdate(k, θ_t, mode=TaLoS)
    end if
  end for
  $\theta_{t+1} \leftarrow$ Aggregate($\{\theta_k\}_{k \in S_t}$)
  Update $T(t)$
end for

In pTaLoS, the client still minimizes the same local objective function as in FedProx with TaLoS. However, the key difference lies in the fact that we use a *layer-wise selection strategy* inspired by pFedEdit. The client evaluates the average training loss for each layer across a fixed number of mini-batches and selects the top-k layers with the lowest contribution to the loss. These selected layers are then fine-tuned while the remaining layers are frozen. Formally, let L be the set of candidate layers in the model, and $\mathcal{J} \subseteq L$ be the top-k selected layers with lowest per-layer loss, The resulting training is restricted to:

$$\mathcal{J} = \text{argmin}_{j \in L} \ \text{AvgLoss}_j, \quad |\mathcal{J}| = k$$
$$\theta_k^{(t)} \leftarrow \theta_k^{(t-1)} - \eta \nabla_{\theta_{\mathcal{J}}} \left[\mathcal{L}_k(\theta_k^{(t-1)}, D_k) \right]$$

4. Experimental Setup

Model Architecture We adopt the DINO ViT-S/16 model [2], a small Vision Transformer trained using the DINO self-supervised method on ImageNet. Since the default DINO head has no classification output, we place a new fully connected layer with 100 output units as the head, corresponding to the CIFAR-100 classes.

Dataset and Preprocessing We use the CIFAR-100 dataset, which consists of 60,000 RGB images of size 32×32 distributed across 100 classes, with 50,000 training and 10,000 test images (i.e., 500 training and 100 test samples per class). Since the pretrained model expects inputs of size 224×224, all images are resized accordingly. Furthermore, we normalize all images using the standard ImageNet statistics to match the requirements of the DINO ViT-S/16 model. During training, we apply a set of DINO-inspired augmentations including AutoAugment [3] with the CIFAR-10 policy. To simulate federated learning settings, we first split the CIFAR-100 training set into a training and validation subset (90%–10%), then we distributed the training portion among clients using two partitioning strategies:

- **IID Partitioning:** Each client receives a randomly sampled, equally sized subset of the global dataset, preserving the overall class distribution.
- **Non-IID Partitioning:** Each client is assigned N_c distinct classes (sampled with replacement) and only receives examples from those classes.

Hyperparameters and Training Configuration Unless otherwise stated, all experiments follow a consistent training configuration, summarized in Table 1, 2 and 3.

Table 1. Fixed general training settings used across experiments.

Component	Value
Batch Size	256
Scheduler	Cosine Annealing
Loss Function	Cross-Entropy
Rounds	8
Local Epochs	4

Table 2. Fixed TaLoS and pTaLoS hyperparameters used across experiments. The parameter k (`pTaLoS`) indicates the number of top layers retained for local personalization; *Batches* and *Rounds* refer to the number of batches used for Fisher estimation and the number of calibration rounds in TaLoS pruning, respectively.

Hyperparameter	Value
k	3
Batches	4
Rounds	4

Table 3. Fixed optimizers hyperparameters used across experiments.

Hyperparameter	SGDM	AdamW
Learning Rate	0.00005	0.00005
Weight Decay	0.05	0.05
Momentum	0.9	–
Nesterov	false	–
Betas	–	(0.9, 0.999)

Additionally, we use a warmup phase at the beginning of training, during which the learning rate is gradually increased from a small value to the initial learning rate defined by the scheduler. Regarding the values of rounds and local epochs, they were selected empirically to balance performance and runtime on limited hardware.

Metrics and Evaluation We evaluate all methods mainly using validation loss and accuracy curves to analyze convergence speed and generalization capacity of the models, since computational constraints make comparisons based on final accuracy almost invalid in the earlier experimentations.

4.1. Experimental Protocol

Centralized Baseline To ground our analysis and provide a point of comparison for the federated experiments, we implement a centralized training baseline together with an unified data pipeline that is reused in both centralized and federated settings. This pipeline ensures consistent data normalization, augmentation, and image resizing. It is important to emphasize that our objective is not to maximize performance on the centralized model. While fine-tuning the data augmentation policy, learning rate schedule, or architectural components could push the accuracy higher, our goal is instead to obtain results that are *comparable in scale and behavior* to those observed under realistic federated learning constraints. This approach helps us isolate and analyze the true impact of model editing and personalization strategies in subsequent experiments.

FL Baseline We establish a baseline by first training with IID data partitions, where each client receives a balanced subset of the training data. To simulate real-world scenarios, we progressively introduce non-IID settings through label-skewed partitions. As expected, this leads to slower convergence and lower final accuracy due to *client drift*, where heterogeneous local updates hinder effective aggregation. To address this, we integrate regularization techniques such as `FedProx`, which adds a proximal term to local objectives to limit deviation from the global model. This improves stability and performance in heterogeneous settings. Our FL baseline thus serves both as a reference under ideal conditions and as a foundation for testing more robust, personalized strategies under non-IID constraints.

Centralized Model Editing Baseline To assess the potential of model editing and neural network personalization in federated scenarios, we implemented a baseline grounded on *sparsified optimizers* and *sensitivity-guided* pruning techniques. Specifically, we introduced two custom optimizers: `SparseSGDM (SSGDM)` and `SparseAdamW (SADAMW)`, which are extensions of `SGDM` with momentum and `AdamW`, respectively. These optimizers incorporate *gradient masking* during the parameter update step. This masking mechanism is essential to enable *selective fine-tuning*, where only a subset of the model (e.g., the least sensitive or most sensitive parameters) is modified. To determine which parameters to update, we used our implementation of the `TaLoS` framework, which supports these operating modes:

- **Head**: only the parameters of the model's classification head are pruned;
- **Full**: pruning is applied across the entire ViT backbone;

Once parameter sensitivity is estimated over a small number of batches, it performs a *multi-round calibration* of the masks, selecting the parameters to update according to two strategies:

- **Least sensitive**: only the least sensitive parameters—those whose change has minimal impact on the model's output—are updated. This is useful when local updates should preserve the global model behavior.
- **Most sensitive**: conversely, the most sensitive parameters are updated. This strategy is intended to maximize the effect of local fine-tuning.

Federated Editing and Regularization Techniques We then extended our analysis to the federated setting by experimenting with `TaLoS` under various configurations such as in combination with our custom implementation of

FedProx, which we refer to as TaLoSProx, as well as with the adaptive optimizer YOGI. These experiments allowed us to assess the stability and convergence behavior of TaLoS-based editing when integrated with different federated optimization baselines.

Personalized Model Editing with pTaLoS Building on the editing baseline, we implemented a personalized variant called pTaLoS, which combines the sensitivity-guided pruning of TaLoS with layer-wise stochastic personalization of pFedEdit. This method relies on dynamic selection of which layers to fine-tune locally and clients sampling strategy as discussed in section 3. To control this behavior, we define two key scheduling parameters:

- **Deterministic Round**: after this round, the behavior becomes deterministic: the client always utilize one strategy over the other.
- **Reverse Mode**: when enabled, this flag inverts the scheduling direction. Instead of starting with full personalization and gradually reducing it, the model begins with selective personalization and progressively allows more full-model training.

5. Results and Discussion

Centralized Baseline This section compares centralized training with either a frozen backbone or full fine-tuning. All models were trained for 15 epochs. Freezing the backbone speeds up training and saves memory but yields lower accuracy. Fine-tuning the entire model improves performance at higher computational cost. We evaluated SGDM and AdamW in both cases. With fine-tuning, AdamW outperformed SGDM (88.15% vs 74.19%, loss 1.18 vs 1.67). When the backbone was frozen, SGDM slightly outperformed AdamW (77.57% vs 74.9%, loss 1.61 vs 1.70). The best results for each setup are summarized in the Table 4.

Table 4. Comparison of accuracy (%) and loss for SGDM and AdamW under frozen (F) and non-frozen (NF) backbones. Best values per row are highlighted.

Backbone	Metric	SGDM	AdamW
Non-Frozen	Accuracy (%)	74.19	**88.15**
	Loss	1.67	**1.18**
Frozen	Accuracy (%)	**77.57**	74.90
	Loss	**1.61**	1.70

To accelerate the large number of training runs, we froze the backbone in the next experiments. In addition, since SGDM slightly outperformed AdamW in this setting, we adopted SGDM for the experiments with the frozen backbone.

Centralized Model Editing Baseline To evaluate the impact of model editing in a centralized setting, we tested whether fine-tuning only a subset of parameters—rather than the entire model—can lead to meaningful improvements. The goal was to personalize the model efficiently by updating only selected weights based on their sensitivity. All tests were conducted under fixed hyperparameters of Tables 2 and 3 to ensure consistency across configurations, the only variable changed across runs was the sparsity level tested at three values: low, medium, and high. Notably, when the backbone remained unfrozen (e.g in the full pruning mode), we used SparseAdamW. In contrast, for the head pruning configuration, we employed SparseSGDM, which yielded better performance under frozen backbone conditions. Results are shown in Table 5.

Table 5. Accuracy (%) and loss under different sparsity levels using least and most sensitive parameter selection, evaluated in two pruning modes. Best values per row are bolded.

Mode	Sparsity	Acc Least	Loss	Acc Most	Loss
Head	0.3	69.55	1.91	**74.68**	**1.70**
	0.6	54.86	2.41	**69.20**	**1.85**
	0.9	16.80	3.94	**38.94**	**3.03**
Full	0.3	**87.12**	**1.24**	86.39	1.26
	0.6	83.56	1.40	**86.86**	**1.26**
	0.9	69.51	1.84	**84.34**	**1.31**

In both the head and full pruning modes, a sparsity level of 0.3 yields the best overall performance in terms of accuracy and loss. Within the head mode, updating the most sensitive parameters consistently outperforms the least sensitive strategy. This trend also holds in the full pruning mode, although at the 0.3 sparsity level, the least sensitive strategy slightly surpasses the most sensitive one. As sparsity increases, the performance of the least sensitive approach degrades more sharply, while the most sensitive strategy demonstrates greater robustness, maintaining higher accuracy and lower loss across settings.

FL IID Baseline To evaluate how local training and communication frequency affect performance in a federated setup, experiments were conducted on IID data using different combinations of local epochs while keeping the number of communication rounds fixed at 20. All experiments were run using the SGDM optimizer with a frozen backbone. To account for the highly heterogeneous settings explored later in the experiments we temporarily increased the learning rate to 0.03 in the FL baselines to accelerate convergence under computational constraints. The results shown in Table 6 suggests that increasing the number of local epochs, while keeping the number of rounds fixed, led to higher ac-

curacy.

Table 6. FedAvg results on IID data.

Local Epochs	Accuracy (%)	Loss
4	68.65	1.1019
8	72.64	**0.9647**
16	**73.43**	0.9901

This suggests that giving clients more time to train on their local data before communicating with the server allows the model to learn more meaningful representations. Although high local epochs can sometimes lead to instability due to client drift, the IID setting used here avoids that issue, as all clients have a similar data distribution. This makes deeper local training more effective and stable.

FL Non-IID Baseline To evaluate the effect of data heterogeneity and local training effort in federated learning, experiments were conducted using `SGDM` on a frozen backbone across various configurations of class distribution (N_c), local epochs, and communication rounds. The goal was to simulate non-IID conditions by varying N_c, the number of unique classes assigned to each client, and to observe how model performance changes. The optimizer hyperparameters used in these experiments were the same as those used in the IID setting to ensure consistent comparison. Results are shown in Table 7.

Table 7. FedAvg results on non-IID data using `SGDM` with a frozen backbone. We vary the number of local epochs, communication rounds, and number of classes per client (N_c). Best accuracy and loss per N_c block are bolded.

N_c	Local Epochs	Rounds	Accuracy (%)	Loss
1	4	8	**5.52**	**22.59**
	8	4	4.91	26.87
	16	2	4.32	29.54
5	4	8	**35.06**	**7.91**
	8	4	24.02	8.27
	16	2	11.21	9.96
10	4	8	**47.03**	**7.83**
	8	4	35.80	8.10
	16	2	19.64	9.56
50	4	8	**66.91**	5.85
	8	4	61.87	**5.17**
	16	2	39.36	9.45

As expected, performance improved significantly as N_c increased. When each client had access to a larger number of classes ($N_c = 50$), the distribution became closer to IID, resulting in better accuracy and lower loss. In contrast, at lower N_c values (5 or 10), the model struggled due to highly skewed local data. This led to conflicting updates and degraded global model performance. A key observation was that increasing local epochs while reducing communication rounds did not improve performance. For lower N_c values, fewer rounds with more local training (16 local epochs and only 2 rounds) caused noticeable drops in accuracy. This is likely due to *client drift*, where local models diverge too far from the global model when trained extensively on biased data. On the other hand, configurations with shorter local training and more frequent communication (4 local epochs, 8 rounds) generally produced better results, especially at higher N_c. This supports the idea that in non-IID settings, more frequent synchronization can help maintain global consistency.

Regularization Techniques To address the convergence and generalization challenges in federated learning, particularly under non-IID conditions, several regularization techniques were evaluated at an heterogeneity level of $N_c = 50$. The goal was to assess whether these methods could improve stability and performance compared to standard `FedAvg`. The most significant improvement was observed when combining `FedProx` with `Yogi`, resulting in the highest accuracy and lowest loss across all test cases. Results are shown in Table 8.

Table 8. Performance of different regularization strategies under fixed hyperparameters. The best results are highlighted in bold.

Test Case	Accuracy (%)	Loss
FedProx with FedAvg	64.82	2.3968
Yogi with standard clients	65.73	6.2961
FedProx with Yogi	**74.62**	**2.3621**

TaLoS, TaLoSProx, pTaLoS To assess the impact of statistical heterogeneity on model performance in federated editing settings, we evaluated three variants of our editing-based framework: `TaLoS`, `TaLoSProx`, and `pTaLoS`. Each method was tested under varying degrees of non-IID-ness, using $N_c \in \{5, 10, 20, 50\}$. All experiments in this section were conducted using the `SparseAdamW` optimizer and a sparsity level of 60% as we used the unfrozen backbone configuration (unlike previous experiments) and they were the ones who yielded the best results in the this setup. The results are shown in Table 9. `TaLoSProx` shows improved performance with increasing N_c outperforming `pTaLoS`: when data is nearly IID ($N_c = 50$), accuracy reaches 76.82% (most sensitive) and 66.04% (least sensitive), with minimal loss. However, in highly non-IID settings ($N_c = 5$), `TaLoSProx` performance drops

sharply.

Table 9. Comparison of accuracy and loss for all methods across different values of N_c and sensitivity . Best results per block are highlighted in bold.

N_c	Method	Sensitivity	Accuracy (%)	Loss
5	TaLoSProx	Least	2.14	5.27
	TaLoSProx	Most	2.72	5.41
	TaLoS	Least	8.43	4.92
	TaLoS	Most	4.52	5.20
	pTaLoS	Least	1.82	7.13
	pTaLoS	Most	**11.00**	**4.59**
10	TaLoSProx	Least	10.68	4.56
	TaLoSProx	Most	19.18	4.10
	TaLoS	Least	21.85	3.65
	TaLoS	Most	18.16	3.88
	pTaLoS	Least	4.25	5.50
	pTaLoS	Most	**26.30**	**3.54**
20	TaLoSProx	Least	30.64	3.05
	TaLoSProx	Most	49.30	**2.13**
	TaLoS	Least	47.81	2.14
	TaLoS	Most	49.06	2.14
	pTaLoS	Least	13.39	4.27
	pTaLoS	Most	**49.54**	2.15
50	TaLoSProx	Least	66.04	1.24
	TaLoSProx	Most	**76.82**	**0.84**
	TaLoS	Least	75.65	0.91
	TaLoS	Most	74.69	0.90
	pTaLoS	Least	42.97	2.35
	pTaLoS	Most	76.08	1.01

Comparing `TaLoS` and `TaLoSProx` with respect to `pTaLoS`, we find that `pTaLoS` consistently improves performance in skewed scenarios. For example, with $N_c = 5$ and most-sensitive pruning, accuracy increases from 4.52% (`TaLoS`) to 11.00% (`pTaLoS`), and loss decreases from 5.20 to 4.59. When the data becomes more IID, both methods converge in performance, though `TaLoSProx` slightly outperforms in this regime. `TaLoSProx` shows robust performance in moderately heterogeneous regimes, while `pTaLoS` excels in more skewed scenarios thanks to its personalization schedule.

pTaLoS in a Realistic Scenario To validate the effectiveness of our hybrid strategy in a more realistic long-term training scenario, we extended the number of communication rounds to 50 under the non-IID condition with $N_c = 10$. All configurations adopted the `TaLoS` most-sensitive pruning approach. We tested both the standard and reverse scheduling modes of `pTaLoS`, varying the deterministic round (DR) to assess how the timing of the global-to-personalized transition influences model convergence and final performance. Results are shown in Table 10. As shown in Image 2, the normal schedule initially performed well, learning in early rounds thanks to a mix of global editing and localized editing.

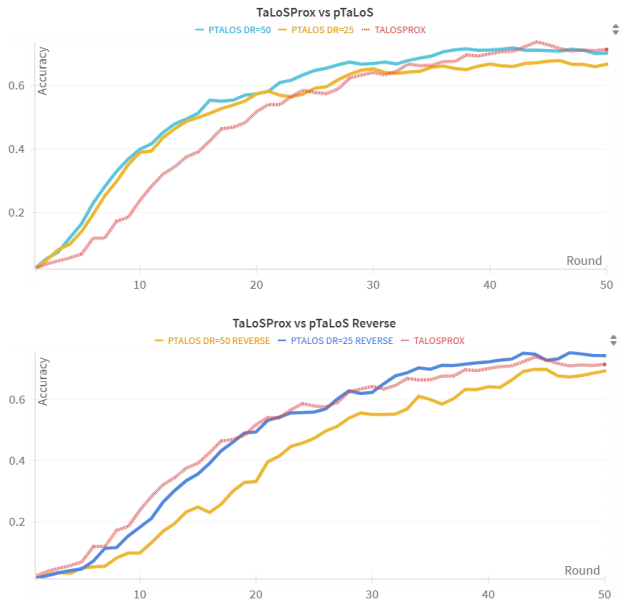

Figure 2. Convergence comparison across editing strategies under non-IID settings. The top plot compares `TaLoS` and `TaLoSProx`, showing that `TaLoS` converges faster in early rounds but suffers degradation later, while `TaLoSProx` is more stable. The bottom plot illustrates the behavior of `pTaLoS` with a reverse schedule: although slower at first, it gradually improves and ultimately surpasses both baselines in later rounds.

However, its ability to improve further diminished in the later stages, likely due to the localized nature of `pTaLoS`, which struggled to generalize across clients in the later rounds with the same pace as `TaLoS`.

Table 10. Comparison of `TaLoSProx` and various `pTaLoS` configurations under $N_c = 10$ on 50 rounds runs. DR indicates the deterministic round.

Method	DR	Accuracy (%)	Loss
TaLoSProx Baseline	–	71.36	1.19
pTaLoS (Reverse)	25	**74.20**	**0.98**
pTaLoS (Reverse)	50	69.00	1.20
pTaLoS (Normal)	25	66.00	1.34
pTaLoS (Normal)	50	70.00	1.21

In contrast, as we can see in the second plot of Figure 2,

the reverse schedule exhibited weaker performance during early training but steadily improved over time. By enabling local adaptation early and enforcing global alignment later, this configuration produced a model better aligned with client distributions, ultimately outperforming both the standard `pTaLoS` schedule and the `TaLoSProx` baseline in terms of final accuracy and loss.

6. Conclusion

This work introduced `pTaLoS`, a hybrid model editing strategy that integrates the global alignment capabilities of `TaLoS` with the personalization benefits of `pFedEdit`. Experimental results, especially under challenging heterogeneous scenarios, demonstrate that `pTaLoS` provides a favorable trade-off between convergence speed and final performance. Notably, the reversed scheduling mode showed slower initial learning but superior long-term accuracy and generalization, as it better accommodated local data distributions over time. These findings suggest that controlling the timing and application of personalized updates can substantially impact learning dynamics. However, our study is not without limitations. Due to computational constraints, we were unable to explore longer training horizons, larger client populations, or more diverse datasets. Additionally, while we fixed the client sampling to uniform random, more intelligent sampling strategies—such as selecting clients whose models are more disaligned from the global one—could further improve performance and stability.

References

[1] Keith et al. Bonawitz. Towards federated learning at scale: System design. *Proceedings of the 2nd SysML Conference*, 2019.

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021.

[3] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[4] Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4884–4896, 2021.

[5] Leonardo Iurada, Marco Ciccone, and Tatiana Tommasi. Efficient model editing with task-localized sparse fine-tuning, 2025.

[6] Pengcheng et al. Jiang. pfededit: Personalized model editing for federated learning. *arXiv preprint arXiv:2302.12345*, 2023.

[7] Peter et al. Kairouz. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

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

[9] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

[10] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2020.

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

[12] Eric et al. Mitchell. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022.

[13] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations (ICLR)*, 2021.

[14] Anton et al. Sinitsin. Editable neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

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

Appendix: Resources

Code Repository:
`https://github.com/Federated-Learning-Machine-Learning/federated-learning-project`

Training Logs and Visualizations (WandB):
`https://wandb.ai/polito-fl/federated-learning-project/overview`