Analyzing TaLoS in Federated Learning Scenarios

Cerbelli Federico Depetro Fabio Lalli Duccio Tozzi Niccolò
Data Science and Engineering, Politecnico di Torino
{s346453, s347102, s345062, s345809}@studenti.polito.it

Abstract

The increasing widespread use of large-scale pre-trained models, alongside growing concerns around data privacy, is leading to the adoption of the Federated Learning (FL) framework and the implementation of novel model editing techniques. In this work, we explore the combination of these methods to address an image classification task using the pre-trained ViT-S/16 model, on the CIFAR-100 dataset.

As a baseline, we employed Task-Localized Sparse Fine-Tuning (`TaLoS`) to update our models and evaluated performance across centralized and federated scenarios. In the federated setting, we adopted the Federated Averaging (FedAvg) algorithm under both IID and non-IID data distributions across clients.

In addition, this work focuses on an alternative parameter partitioning scheme in the federated context with `TaLoS` in order to leverage its intrinsic properties of minimal interference and weight disentanglement across tasks. In this approach, parameter subsets are pre-assigned into mutually exclusive partitions, which are distributed among clients so that each client updates only its unique section of the model. The goal is to determine whether this scheme can serve as a viable alternative to traditional `FedAvg` aggregation, particularly in challenging non-IID settings.[1]

1. Introduction

Federated Learning (FL) has emerged as a privacy-preserving framework in machine learning, enabling collaborative model training across a multitude of decentralized devices (also called *clients*): such as smartphones, IoT devices or medical centers. In the canonical FL framework, each client performs multiple steps of local training on its private dataset and periodically transmits model updates (e.g., weight vectors) to a central server, without exposing sensitive raw data. In the vanilla scenario, the server aggregates these updates using the Federated Averaging (`FedAvg`) algorithm, which computes a weighted average of the locally updated models to iteratively refine the global model [3].

At the same time, in the field of efficient model editing, recent works have explored strategies that allow fine-tuning large-scale pre-trained models with minimal parameter updates. This is especially relevant for large-scale models such as large-language models (LLMs) or vision transformers. Among these techniques, Task Arithmetic [1] has gained attention as a solution for editing models by encoding task-specific knowledge into vector representations that can be combined and reused. Those vectors are obtained by computing the difference between the parameters of a fine-tuned model and its original pre-trained counterpart, effectively capturing the updates needed for a specific task. In this context, one of the main challenges is task interference, which can disrupt previously acquired knowledge when combining task vectors obtained in decentralized settings. To mitigate this issue, recent techniques such as Task-Localized Sparse Fine-Tuning (`TaLoS`) have been proposed [2], leveraging sparse fine-tuning (over the subset of least sensitive parameters, according to fisher scores) to limit interference during fine-tuning, while trying to preserve weight disentanglement [4]. It refers to the phenomenon in which distinct directions in a model's weight space correspond to non-overlapping and task-specific behaviors in the input space.

One of the main objectives of this work is to bridge the two frameworks by applying Task-Localized Sparse Fine-Tuning within a Federated Learning setting, thereby benefiting from both paradigms' ability to aggregate distributed expertise, without sharing sensitive data.

Alongside this, we propose and investigate two alternative strategies for combining `TaLoS` and `FL`, which are tested to better understand the trade-offs between communication efficiency, parameter sharing, and task interference.

In the first method, we propose a variant of the previus `FL+TaLoS` protocol where the set of least sensitive parameters (as identified by Fisher scores) is partitioned into mutually exclusive disjoint masks, and each client is assigned a distinct slice of the sparse parameter space. Each client fine-tunes only its dedicated subset, ensuring that no pa-

[1]Code available at: https://github.com/depetrofabio/FederatedLearningProject#

rameter is updated by more than one client per round. This design allows the server to aggregate client updates by direct concatenation rather than averaging, effectively eliminating destructive interference and reducing communication cost. We present this approach as an alternative to the vanilla `FedAvg`, specifically for settings characterized by sparse and task-local parameter updates.

In the second method, we remove the communication rounds entirely. Clients independently fine-tune their assigned disjoint subset of sparse parameters on local data, without synchronization or coordination with other clients. Each client thus produces a lightweight, task-specific update vector, which we later combine to obtain a global task-edited model. This strategy aims to compose localized task knowledge captured under IID and non-IID settings (e.g., one class per client), leveraging the composability property of task vectors to aggregate decentralized expertise post-hoc.

With these experiments, we want to verify whether parameter partitioning can induce any beneficial behaviors with respect to interference among updates from different clients in this context.

1.1. Benchmark Baselines

Before evaluating the proposed integration of `TaLoS` within the FL setting, as well as our additional approaches, we establish a set of baseline benchmarks to contextualize performance. These benchmarks are obtained under controlled conditions and varying degrees of data heterogeneity. The benchmarks include:

- **Centralized Baseline (Head-only):** The model is trained in a centralized setting, with only the classification head being updated.
- **Centralized with TaLoS Baseline:** This model is obtained by applying `TaLoS` to the backbone, using the classification head fine-tuned during centralized baseline training.
- **Federated Baseline (`FedAvg`):** To contextualize performance in a distributed environment, we replicate the centralized "head-only" approach using the `FedAvg` aggregation algorithm: the classification head is trained federatively across clients, while the model's backbone remain frozen.

For all federated experiments, we consider two data distribution settings:

- **IID Partitioning:** The dataset is partitioned across K clients which receives an approximately equal number of samples, with a local class distribution that mirrors the global one.
- **non-IID Partitioning:** Every client dataset contains examples corresponding to only a subset of the complete set of labels.

After establishing these benchmarks, we proceed to evaluate our proposed approach.

2. Methods

To conduct our experiments, we use the CIFAR-100 dataset, originally composed of 50,000 training and 10,000 test examples. We set aside 10,000 samples from the original training set to build a validation set, resulting in a 40,000/10,000/10,000 split. The resulting split preserves the original class distribution of CIFAR-100. The validation set is necessary for hyperparameter tuning and early stopping, ensuring that model selection is performed without bias from the test set.

We employ DINO ViT-S/16, a Vision Transformer pretrained on the ImageNet dataset. The model produces a 384-dimensional embedding that encodes high-level visual features of the input image. To adapt the pretrained backbone to the CIFAR-100 classification task, we append a linear classification head that projects the embedding onto the 100 target classes.

2.1. Centralized Model Baseline

In this initial phase, we focused exclusively on training the classification head, keeping the weights of the pretrained backbone frozen in order to preserve the pre-trained model's acquired knowledge. This "warm-up" of the head was performed over 30 training epochs.

The training used the **Cross-Entropy Loss** as the loss function, a standard choice for multi-class classification tasks. As optimizer, we adopted **Stochastic Gradient Descent with Momentum (SGDM)**, setting the momentum coefficient to $\mu = 0.9$.

Learning rate and **weight decay** were treated as tunable hyperparameters and varied across experiments to identify optimal configurations. To facilitate model convergence, we experimented with two learning rate schedulers:

- **CosineAnnealingLR**, which smoothly decays the learning rate following a cosine schedule;
- **StepLR**, which in our case applies a multiplicative decay factor of 0.1 at fixed intervals of 10 epochs.

Tables 1 and 2 summarize the validation accuracies obtained under various combinations of learning rate and weight decay for the CosineAnnealingLR and StepLR schedulers, respectively.

From the experimental results, the best hyperparameter combination is obtained using the CosineAnnealingLR

Table 1. Validation accuracy (%) for learning rate (γ) and weight decay (λ) combinations using **CosineAnnealingLR**.

λ \ γ	0.001	0.005	0.01
0.0001	46.54	46.30	**47.07**
0.001	45.59	46.59	46.83

Table 2. Validation accuracy (%) for learning rate (γ) and weight decay (λ) combinations using **StepLR**.

λ \ γ	0.001	0.005	0.01
0.0001	45.99	46.34	**46.65**
0.001	45.40	46.22	46.50

scheduler with a weight decay of $\lambda = 0.0001$ and a learning rate of $\gamma = 0.01$, yielding a validation accuracy of 47.07%. For this configuration, fig. 1 illustrates the evolution of test loss and test accuracy over the 30 training epochs. Notably, the final test accuracy reaches 46.50%.

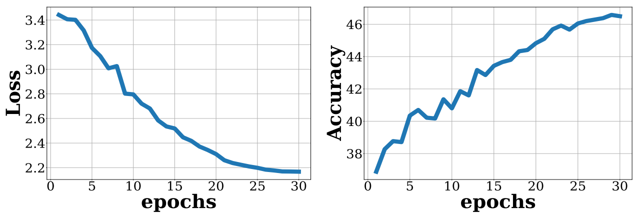


Figure 1. Performance metrics for the Centralized Head-Only baseline, respectively test loss (on the left) and test accuracy (on the right).

2.2. Federated Approach

We evaluated both IID and non-IID scenarios by keeping the number of examples per client fixed while varying the number of distinct classes per client. The parameter $N_c \in \{1, 5, 10, 50\}$ defines how many of the 100 classes are locally available to each client, thus controlling the degree of data heterogeneity. To synthesize these pathological non-identical user splits, data for each class was first split into shards, which were then assigned to clients according to their designated subset of classes, similarly to the approach adopted by McMahan et al. [3]. This ensures the desired label distribution.

To begin, we trained only the classification head, which was added to the pretrained model, within this distributed scenario. We implemented the SGDM in accordance with the `FedAvg` algorithm, proposed by McMahan et al. [3]. In each communication round $t \in [T]$, the central server distributes the current global model parameters, $\theta^{(t-1)}$, to a subset of clients. A fixed fraction $C = 0.1$ of the total $K = 100$ clients is selected uniformly at random to participate in the round. Acknowledging that in real world scenarios rarely all clients are available in every round, C serves to control the degree of multi-client parallelism. Each selected client $k \in [K]$ then performs local training on its private dataset for a specified number of local SGDM steps (J) and sends the updated weights to the server for aggregation. The server combines locally updated parameters through an average, weighted by the cardinality of local datasets:

$$\theta^{(t)} \leftarrow \sum_{k=1}^{C \cdot K} \frac{n_k}{n} \theta_k^{(t)} \tag{1}$$

Where:

- n_k is the cardinality of local datasets
- $\theta^{(t)}$ represents the updated global model parameters.
- $\theta_k^{(t)}$ represents the locally updated model parameters from client k.
- $n = \sum_{k=1}^{C \cdot K} n_k$.

Initially, we set $J = 4$ and conducted 600 communication rounds (fig. 2) in the IID configuration. This allowed us to identify a good trade-off between achieved accuracy and computational cost at 300 rounds. For subsequent experiments, we kept the product

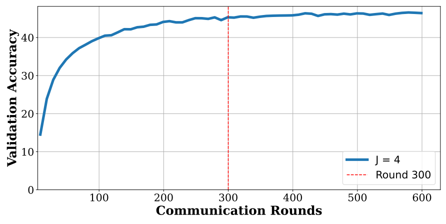


Figure 2. Server validation accuracy over 600 communication rounds for the FedAvg algorithm, where only the classification head was trained in an IID configuration.

of $J \times$ (number of communication rounds) constant across varying numbers of local steps ($J = 4, 8, 16$). This approach ensured comparability between experimental runs, since the total number of steps per-model is the same.

To comprehensively assess performance, we track two metrics: (i) the **validation accuracy** of the aggregated global model, evaluated **server-side** on a held-out validation set (Sec.3 fig.8); (ii) the **average training accuracy** computed across **clients** during each training round (Sec.3 fig.9). The second metric is particularly insightful in non-IID scenarios, where clients' private datasets can be viewed as distinct tasks. It helps reveal the degree of clients specialization after a round.

We experimentally determined that the optimal number of local steps, across different heterogeneity levels, is $J = 4$ (fig. 3), which we adopt in the subsequent experiments involving sparse fine-tuning. Further analysis of these intermediate results can be found in Section 3 fig. 8,9 .

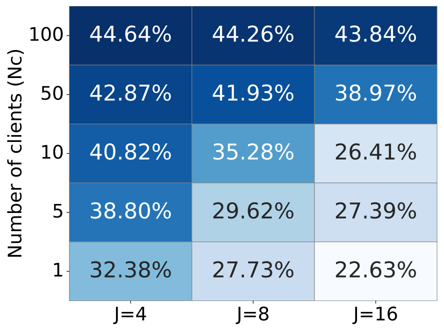


Figure 3. Server validation accuracy across different levels of data heterogeneity (N_c) and number of local steps (J). Increased color saturation indicates higher accuracy.

Using the optimal $J = 4$, we computed the accuracy on the test set, as reported in Table 3. These values are the ”FL Head-Only benchmarks” and will be used for comparison with the subsequent experiments.

Table 3. Test accuracy (%) with $J = 4$ for varying N_c

	IID	$N_c = 1$	$N_c = 5$	$N_c = 10$	$N_c = 50$
Accuracy	44.51	31.96	38.70	41.11	42.88

2.3. Model Editing with `TaLoS`

2.3.1 Mask calibration

To compute the mask of the least sensitive parameters of the model, we used the Fisher Information Matrix (FIM) [2]. The sensitivity score of parameter $j \in \{1, \dots, m\}$ is defined as the j-th diagonal entry of the FIM:

$$F_{[j,j]}(\boldsymbol{\theta}_0, \mathcal{D}_t) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{y \sim p_{\boldsymbol{\theta}_0}(y|\boldsymbol{x}_i)} \left[\nabla_{\theta_{[j]}} \log p_{\boldsymbol{\theta}_0}(y|\boldsymbol{x}_i) \right]^2 \tag{2}$$

Where $\mathcal{D}_t$ is the data support. Since computing expectation is computationally expensive, we approximate it by sampling from the model’s output distribution for each input. This allows us to compute **per-example** gradients and accumulate them to estimate the Fisher diagonal. Following the recommendations in Iurada et al. [2], we restrict the computation of the mask and the subsequent fine-tuning to a subset of the model’s parameters, by excluding the embedding layers and final projection layers.

Calibration Choices and Justifications

- **Iterative mask refinement:**

  The mask is not built ”one-shot”. Instead, we perform multiple calibration rounds (R) to improve numerical stability. At each round we increase the sparsity as:

  $$\rho^{(r)} = 1 - (1 - \rho^*)^{\frac{r}{R}} \tag{3}$$

  to achieve the desired sparsity ρ^*. We tested three different values for R : 3, 5, 10. Although $R = 10$ was more computationally expensive, it yields significantly more stable masks, with better localization in the self-attention layers (Q,K,V) and reduced noise across other layers of the model, as shown in fig. 4.

- **Soft-Zeroing:**

  To improve next-round sensitivity estimation, the weights of the most sensitive parameters are *softly zeroed*. A soft zero is preferred over a hard zero to prevent *layer collapse*. We tested different soft-zero values: 0.1, 0.01, and 0.001. Among them, **0.01** provided the most stable and well-localized masks, with active weights primarily in the Q,K,V layers. In contrast, higher values such as 0.1 caused broader activation across many layers (fig. 4).

- **Calibration examples:**

  To compute parameter sensitivity, we provide the model with a subset of input training data:

  – In the IID setting, we select **1 example per class**, for a total of 100 examples (CIFAR-100).

  – In the non-IID setting, we select 100 examples, **stratified** to match the client’s local label distribution. For instance, in non-IID with $Nc = 5$, we use 20 samples per class from 5 classes.

  This choice ensures that the model is exposed to representative samples, even though parameter sensitivity is not strictly dependent on data distribution. Empirically, we observed that using 100 examples yielded masks comparable to those computed with 500 examples, while achieving a 4× speedup.

2.3.2 SparseSGDM

In the fine-tuning phase with sparse masks, we extend the standard SGDM optimizer by implementing a custom variant, referred to as *SparseSGDM*. This optimizer takes as input a binary gradient mask that selectively disables updates: weights corresponding to zero entries in the mask receive no gradient update. This allows us to fine-tune only the targeted subset of least sensitive parameters.

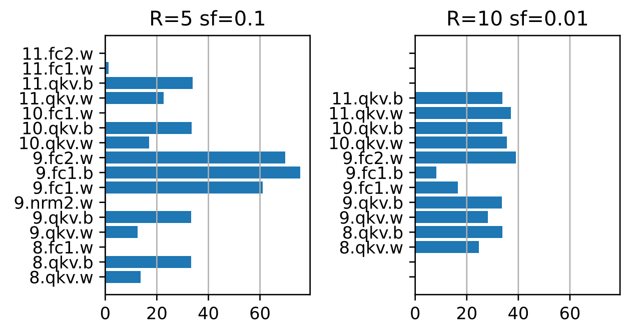


Figure 4. Comparison of masks ($\rho^* = 0.9$) with R=5, soft-zero= 0.1 (left) and R=10, soft-zero= 0.01 (right). Only backbone layers from blocks 8–11 are shown for visual reasons, and only labels of layers containing at least one active weight (mask = 1) are included.

2.3.3 Centralized Model with TaLoS

Building on the best-performing model from the "head-only" centralized training, we proceed to fine-tune it using the *SparseSGDM* optimizer to apply the mask previously computed in the centralized setting.

We retain the same hyperparameter configuration used during the head warm-up phase: a learning rate of $\gamma = 0.01$, a weight decay of $\lambda = 0.0001$ and the CosineAnnealingLR scheduler.

Model performance is monitored on the validation set to analyze the evolution of loss and accuracy across different training durations. Based on this analysis, we select 30 epochs as the optimal stopping point. As we can observe in fig. 5 beyond this threshold, training accuracy continues to increase significantly, while validation accuracy improves only marginally, indicating the beginning of overfitting.

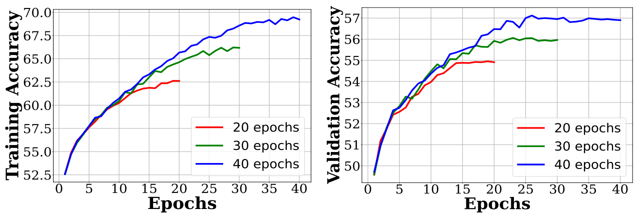


Figure 5. Training (left) and validation (right) accuracy across different training durations (20, 30, and 40 epochs). Additional runs were conducted because a cosine scheduler was employed.

For this configuration, fig. 6 illustrates the evolution of test loss and test accuracy over 30 training epochs using TaLoS. Notably, test accuracy improves from an initial 46.50%, obtained with head-only, to 58.38% after applying the masked fine-tuning strategy.

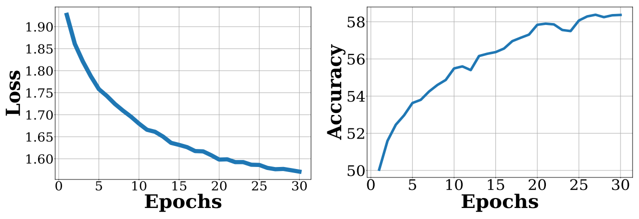


Figure 6. Performance metrics for the Centralized training fine tuned with TaLoS.

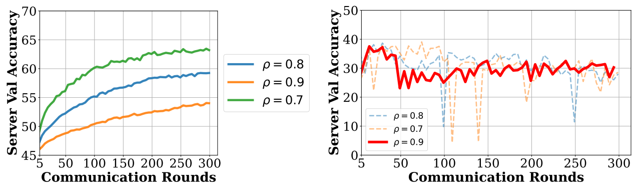
2.3.4 FL with TaLoS

To begin, we calculated the masks for all local clients using their private data, for all heterogeneity configurations. From now on, we will denote the local mask of client k as c_k. Subsequently, using the classification head already trained with the federated approach under the same heterogeneity configuration, we updated only the selected parameters of the backbone according to Algorithm 1, which integrates TaLoS into the FedAvg procedure.

Algorithm 1 FedAvg with TaLoS

```
1:  Server executes:
2:  Initialize global model $\theta^{(0)}$
3:  for each communication round from 1 to T do
4:      $S_t \leftarrow$ (Random set of $C \cdot K$ clients)
5:      for each client $k \in S_t$ in parallel do
6:          $\theta_k^{(t+1)} \leftarrow \text{ClientUpdate}(k, \theta^{(t)})$
7:      end for
8:      $n \leftarrow \sum_{k \in S_t} n_k$
9:      $\theta^{(t+1)} \leftarrow \frac{1}{n} \sum_{k \in S_t} n_k \theta_k^{(t+1)}$
10: end for
    ClientUpdate($k$, $\theta$)                     {Run on client $k$}
11: Initialize local momentum buffer $b \leftarrow 0$.
12: for each local step $j$ from 1 to $J$ do
13:     Sample a mini-batch $d_j$ from client $k$'s local data.
14:     $g \leftarrow \nabla_\theta \mathcal{L}(\theta_{j-1}; d_j)$          {Compute gradient}
15:     $v \leftarrow \mu b_{j-1} + g + \lambda\theta_{j-1}$
16:     $\theta_j \leftarrow \theta_{j-1} - \gamma(c_k \odot v)$
17:     $b_j \leftarrow v$
18: end for
19: return $\theta_J$                              {to the server} =0
```

We experimented with different sparsity levels. As expected, decreasing ρ improves validation accuracy in the IID setting, as it allows a greater number of parameters to be updated. However, this improvement comes at the cost of increased instability in non-IID configurations, as shown in fig. 7. For this reason, we set $\rho = 0.9$ to compare test accuracy curves consistently across all heterogeneity levels (Sec. 3.2, fig. 10).



(a) IID case (b) non-IID case, with $N_c = 1$

Figure 7. Validation Accuracy in Federated Fine-Tuning, varying the sparsity level of the mask, ρ.

2.4. Personal Contribution

We introduce a radical alternative to the canonical `FedAvg` algorithm. To begin, we identify the set of least sensitive parameters ($\hat{\theta}$), under a specified sparsity constraint (ρ).

We then perform a one-time, static partitioning of $\hat{\theta}$ into N mutually exclusive and exhaustive subsets:

$$\{\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_N\},$$

$$\bigcup_{i=1}^{N} \hat{\theta}_i = \hat{\theta},\ \hat{\theta}_i \cap \hat{\theta}_j = \emptyset\ \forall i \neq j$$

Where N is the total number of clients. The partitioning is performed pseudo-randomly.

Since an exclusive partitioning is required, a global view of the least sensitive parameters must be established. To achieve this, each client first computes a local binary mask (with the methodology previously described) indicating its own set of least sensitive parameters. These boolean masks are then sent to the server, which aggregates them via elementwise summation. The result is a global sensitivity score for each parameter, representing how many clients consider it among the least sensitive. To meet the desired sparsity level, the server prunes the parameters with the lowest scores (i.e., those least frequently selected across clients).

Each client k is assigned a unique subset $\hat{\theta}_k$ and is solely responsible for updating it. This fundamentally redefines the dynamics of client collaboration and model aggregation. At each communication round t, a subset of clients S_t is selected, $|S_t| \leq N$. These clients receive the current global model parameters $\theta^{(t)}$. During local training, each client $k \in S_t$ freezes all parameters except those in its assigned subset $\hat{\theta}_k$ and performs a fixed number of local update steps on its private dataset D_k. We denote the locally updated subset of parameters as $\hat{\theta}^*_k$.

After local training, client k sends the difference to the server:

$$\delta_k = \hat{\theta}^*_k - \hat{\theta}_k. \tag{4}$$

The server then updates the set of least sensitive parameters of the global model. The aggregation of the disjoint updates is represented by the union of the client deltas, which are then applied via addition:

$$\hat{\theta}^{(t+1)} = \hat{\theta}^{(t)} + \bigcup_{k \in S_t} \delta_k^{(t)}. \tag{5}$$

In Equations 4 and 5, the operators $-$ and $+$ denote elementwise operations between ordered sets.

This approach shifts the objective from updating a fully shared model to building a composite model composed of specialized, non-overlapping parts. A practical advantage is a significant reduction in communication cost, as each client transmits only $|\hat{\theta}_k|$ values.

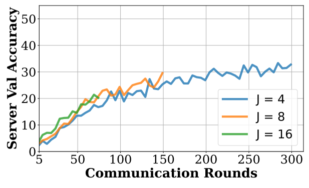
We hypothesized that the most practical method to implement this idea is to involve frequent server-side aggregations (Eq. 5), as this enable each client to receive regular updates on the model partitions it cannot modify. However, from the perspective of Task Arithmetic, an interesting alternative is to enable clients to specialize their parameter partitions for their respective tasks by performing many local update steps, followed by a single, final aggregation of all clients' contributions. In real-world applications, not all clients are available at all times. However, since only a single aggregation is required, we can assume that the server waits for all clients to aggregate their updates. In this scenario, the client differences δ can be interpreted as non-overlapping task vectors. In the results section, we evaluate both proposed approaches.

3. Results

3.1. Varying number of local steps in FL Only-Head

A comparison between IID and non-IID behavior as J varies is shown in fig. 8. In the IID case, the final accuracy depends mainly on the total number of local steps, and is largely unaffected by the number of communication rounds. In contrast, the non-IID setting shows a general drop in accuracy, which worsens as J increases. Fewer central aggregations in this case lead to performance degradation.

We also observed that increasing J generally improves the average local training accuracy (fig. 9). This is expected, as more local steps allow each client to train more thoroughly and better specialize on its own dataset. In the IID case, this local specialization aligns well with the global objective, resulting in non-disruptive behaviors. In contrast, in the non-IID scenario ($N_c = 1$), specialization on skewed local data can harm generalization, leading to lower validation accuracy and unstable global model behavior. This phenomenon is known in the literature as **client drift**. It results in conflicting local updates during aggregation, ultimately degrading both the final performance of the global model and the stability of convergence.


(a) non-IID, $Nc = 1$

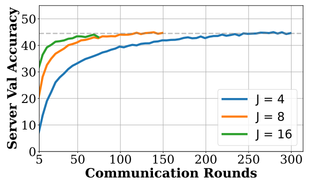
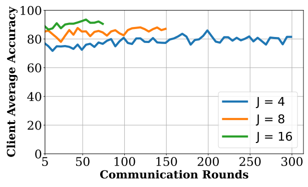

(b) IID

Figure 8. Server validation accuracy with $(J \times \text{Comm. Rounds}) = \texttt{constant}$ using the FedAvg algorithm, training only the classification head.


(a) non-IID, $N_c = 1$

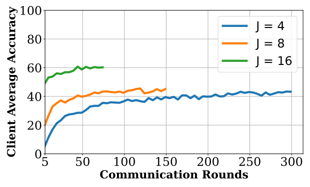

(b) IID

Figure 9. Average client training accuracy with $(J \times \text{Comm. Rounds}) = \texttt{constant}$ using the FedAvg algorithm, training only the classification head.

3.2. Test set performance with varying N_c ($\texttt{FL+TaLoS}$, $J = 4$, $\rho = 0.9$)

A comparison of the performance across different label distributions is shown in fig. 10. The results were obtained by applying TaLoS to the model backbone, using the corresponding classification head trained during the FL baseline. As client dataset becomes more heterogeneous, their local data are less representative of the overall population, causing their model updates to diverge more easily. Aggregating these conflicting updates at the server impairs convergence and reduces the global model's accuracy. In the IID case, we observe accuracy increasing from 44.51% to 53.90%. With $N_c = 50$, accuracy improves from 42.88% to 52.14%. As N_c decreases further, we observe general instability and no clear upward trend.

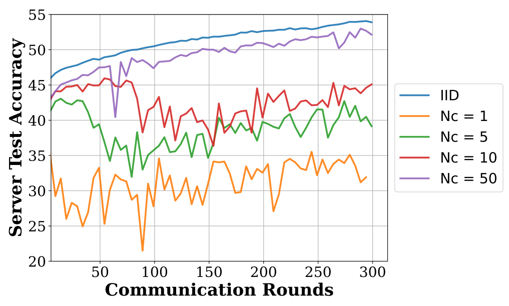


Figure 10. Test Accuracy curves varying N_c using TaLoS on the backbone (FedAvg)

3.3. Test set performance with our approach

Here we report the results of our proposed aggregation method. To assess its behavior, we ran experiments under different levels of heterogeneity and compared the results directly to those obtained with FedAvg (fig. 10). The outcomes, shown in fig. 11, illustrate how our method performs in each scenario.

In the IID case, we observe a slow but stable improvement in test accuracy, increasing from 44.51% to 47.58% at round 300. For $N_c = 50$, the improvement is less stable and less pronounced, rising from 42.88% to 44.83%. As heterogeneity increases further, the curve becomes more unstable and shows a general decline in accuracy.

The generally slow improvement for the IID and $N_c = 50$ cases is partly due to the smaller number of parameters updated in each round compared to FedAvg. Indeed, increasing client participation (C) and therefore the number of updated parameters per round results in a faster improvement rate (in the IID scenario) (Fig. 12). With half of the clients involved in each round, the accuracy reaches 47.29% after 50 rounds; with a quarter of the clients it reaches 46.51%, while in the standard case ($C = 0.1$) reach 45.77%.

Finally, we tested how our approach performs when interpreting the differences $\delta_k | k \in S_t$ as task vectors and performing only one communication round (and thus only one aggregation). Due to limited computational resources, we were able to test only a few configurations. We chose the most extreme case, with $N_c = 1$, where each client performed ~ 25 local epochs (in practice 80 local steps) to specialize fully on its own task. A task vector is obtained by subtracting the pre-trained model parameters from the fine-tuned model parameters. Summing these task vectors yields a direction that improves the performance of the pre-trained model across multiple tasks for which the fine-tuned models were trained [1]. In our case, however, the δ_k represent differences computed on disjoint subsets of parameters. Therefore, rather than a true sum, our aggregation is better described as an assembly.

The updated server model using this approach achieved only 1.08% accuracy on the test set, indicating a complete loss of its predictive ability, since in this case 1% accuracy is equivalent to random guessing.

3.4. Performance Comparison with Baselines

Here we present a summary of the test accuracies obtained across all settings (Tab. 4).

4. Discussion & Conclusion

Overall, our approach underperformed compared to FL - TaLoS. However, due to limited computational resources, we were unable to extend training beyond 300

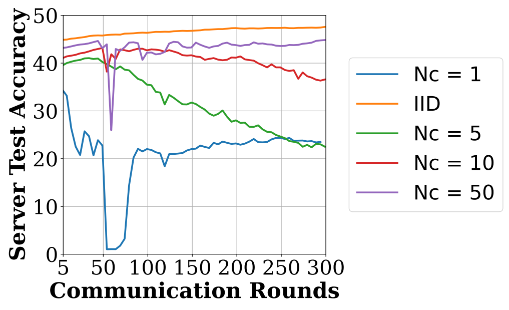


Figure 11. Test Accuracy curves varying N_c using TaLoS on the backbone, $J = 4$, $\rho = 0.9$, using our proposed approach

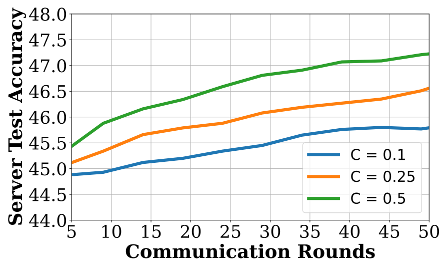


Figure 12. Test Accuracy curves varying C using TaLoS on the backbone, $J = 4$, $\rho = 0.9$, using our proposed approach, in a IID setting

	IID	$N_c = 50$	10	5	1
CB - HO	46.50	—	—	—	—
CB - TaLoS	58.38	—	—	—	—
FLB - HO	44.51	42.88	41.11	38.70	31.96
FL - TaLoS	53.90	52.14	45.09	39.15	31.92
Our	47.58	44.83	36.58	22.50	23.52

Table 4. Test accuracies across different heterogeneity levels. CB: Centralized Baseline; HO: Head Only; FLB: Federated Learning Baseline; Our: Proposed approach. In red updates that decreased accuracy. Where applicable $J = 4$, $\rho = 0.9$

communication rounds. Given its slower convergence, it is possible that with more rounds, the method could reach accuracy levels comparable to FedAvg (in the IID setting). In this case we observed that increasing the fraction of participating clients accelerates convergence without introducing additional instability. Nonetheless, in practical scenarios, the fraction of available clients cannot be easily controlled. For this reason, our strategy of partitioning the updatable weights may be more appropriate for a cross-silo setting (rather than cross-device), where clients are generally more reliable and fewer in number, making it feasible to involve all or most clients in each round and thus guarantee that all selected parameters are updated.

Nevertheless, this method exhibits significant limitations, particularly as data heterogeneity increases, where a clear degradation in performance is observed.

Regarding the attempt to perform only one round of communication, in line with the idea of one-shot FL, the bad results obtained with the most extreme of the pathological distributions ($N_c = 1$) were predictable, since such severe local data skew prevents the model from learning a coherent global representation. As highlighted in [5], the one-shot approach can exacerbate the challenges posed by non-IID data even more. In this scenario, we also observed that the ℓ_2 norm of the update vectors δ_k ($k \in S_t$) was about an order of magnitude larger than in the case with $J = 4$. This is expected: with more local epochs, each client can accumulate larger updates on its private subset of parameters.

Finally, although our method offers an alternative to FedAvg, it inherits the same structural limitations. In particular, our aggregation approach, while providing a more flexible and modular interpretation of updates, is not able to mitigate the effects of statistical heterogeneity. Our approach currently lacks any explicit mechanism to enforce alignment between client models and the global model.

To overcome this limitation, a promising future direction could be to intervene at the level of the local loss function, by introducing a regularization term that penalizes excessive deviation from the central model. Such a strategy would encourage clients to converge toward stationary points of the global loss, improving both training stability and the final performance of the federated model. Well-established methods like `SCAFFOLD` follow this principle and could potentially be integrated into the fine tuning with `TaLoS` in future work, although with the cost of increasing the communications.

References

[1] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023. 1, 7

[2] Leonardo Iurada, Marco Ciccone, and Tatiana Tommasi. Efficient model editing with task-localized sparse fine-tuning, 2025. 1, 4

[3] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023. 1, 3

[4] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models, 2023. 1

[5] Zhixu Tao, Sanjeev Kulkarni, Ian Mason, and Xavier Boix. Task arithmetic through the lens of one-shot federated learning. *arXiv preprint arXiv:2411.18607*, 2024. 8