Federated Model Adaptation via Sparse Fine-Tuning and Low-Rank Decomposition

Gabriele Adorni Taha Atabay Ahmet Berk Calisir Alessandro Casadei

Abstract

Federated Learning (FL) enables collaborative training across decentralized clients while ensuring that private data remains local, but suffers from challenges such as statistical heterogeneity, communication bottlenecks, and client drift—that can impair generalization and convergence. We investigate the use of model editing—specifically sparse fine-tuning via Fisher Information as a communication-efficient alternative to full model updates in FL. Using a DINO-pretrained ViT-S/16 on CIFAR-100, we implement an extensible FL pipeline that supports IID and non-IID data settings. We introduce a custom `SparseSGDM` optimizer to apply Fisher-based gradient masks and compare it with LoRA, and analyze multiple masking strategies and their impact on accuracy, sparsity, and communication. Our results show that sparse editing, particularly with Fisher masks and the TaLoS framework, matches or exceeds dense performance while updating fewer parameters. More broadly, our findings show that adapting model editing for FL can mitigate key limitations of standard methods while preserving the privacy and personalization benefits of decentralized training. These findings highlight model editing as a viable strategy for scalable, modular federated optimization. Code available at: Github source code.

1. Introduction

Federated Learning (FL) is an emerging machine learning paradigm that enables multiple decentralized clients to collaboratively train a shared global model without exchanging their private local data [16]. The decentralized approach is particularly valuable in domains where data privacy and security are paramount, such as healthcare, finance, and mobile applications [9]. Despite its promise, FL presents several significant challenges, including statistical heterogeneity (non-IID data distributions across clients), system heterogeneity (variations in computational capabilities of clients), communication bottlenecks, and client drift [10, 12]. Moreover, while FL enhances privacy by design, it is not completely immune to security threats and adversarial attacks [1, 17].

Model editing is a complementary line of research in Machine Learning that aims to modify or enhance the behavior of pre-trained models without full retraining [6]. Recent work suggests that sparse fine-tuning, modifying only a small, carefully selected subset of model parameters, can enable targeted updates with minimal interference, a property especially valuable in the federated setting [20]. Notably, model-editing techniques that focus on low-sensitivity parameters (as characterized by metrics provided by the Fisher Information Matrices) promise a near-additive merging behavior, aligning well with the iterative aggregation of FL [7, 8]. Inspired by this, we explore sparse model editing via Fisher masking.

This study follows the recent advances and acquired knowledge in the field, and attempts to create an experimenting pipeline to verify and acknowledge them. In detail:

- Implementing and evaluating a centralized baseline for vision classification using DINO ViT-S/16 on CIFAR-100, using the SGDM optimizer [2, 4].
- Testing learning schedulers such as ReduceLrOnPlateau and CosineAnnealingLR, in order to achieve optimal results [14].
- Extending the model to a federated setting, simulating both IID and non-IID client distributions, and comparing the performance of standard FedAvg under varying local update steps and heterogeneity levels.
- Exploring model editing in the FL context, implementing sparse fine-tuning through a custom SparseSGDM optimizer using sensitivity-based gradient masks obtained from the fisher information matrix.
- Designing and implementing a novel aggregation method (**FedAlignAvg**) that adapts client weights based on the alignment of their internal representations with the global model, using SVCCA scores over a fixed probe batch. This approach dynamically down-weights clients whose updates deviate significantly from the global objective, mitigating client drift under non-IID settings.

2. Related Work

Federated Learning (FL) has been extensively studied as a decentralized paradigm that enables privacy-preserving model training across multiple clients without sharing raw data. [9] The seminal work of McMahan *et al*. [16] introduced *Federated Averaging (FedAvg)*, which averages local model updates to construct a global model. FedAvg laid the foundation for many subsequent FL algorithms [10, 12].

Model editing, a closely related field [6]that aims to fine-tune pre-trained models with minimal changes, provides valuable insights for FL. Iurada *et al*. [7] proposed sparse fine-tuning using low-sensitivity parameters identified via Fisher Information, achieving efficient task updates with improved mergeability—an essential property for federated aggregation.

Our work builds upon these prior efforts by exploring how sparse model editing and LoRA-based fine-tuning can be integrated into federated systems to improve communication efficiency, mergeability, and performance under heterogeneous conditions.

3. Methodology

In this section, we provide an overview of our dataset, network, and configuration used in our experiments, along with a baseline centralized assesment.

3.1. Dataset

We use the CIFAR-100 dataset [11], a widely adopted benchmark in image classification. It consists of 60,000 32×32 color images in 100 classes, with 600 images per class. The dataset is divided into 50,000 training and 10,000 test images. For our experiments, we resize all images to 224×224 to match the input resolution of the the ViT models and normalize them using ImageNet statistics [3].

To simulate federated learning, we apply both IID and non-IID sharding strategies. IID partitioning distributes data uniformly between clients. For non-IID settings, we use label-based partitioning where each client receives samples from a limited number of classes, controlled by the hyperparameter N_c). This setup introduces varying degrees of heterogeneity across clients.

3.2. Network Details

The baseline architecture used in this project is the DINO ViT-S/16, a small Vision Transformer trained in self-supervised fashion. Vision Transformers (ViTs) [4]differ from convolutional architectures by applying self-attention mechanisms to patch embeddings of the input image, which enables the model to capture long-range dependencies without inductive biases such as locality or translation invariance.

DINO ViT-S/16 processes the CIFAR-100 images (resized to 224×224) by splitting them into 16×16 patches, linearly projecting them, and applying multiple transformer layers that consist of Multi-Head Self-Attention (MHSA) and Feed-Forward Networks (FFNs). The CLS token output of the final transformer layer is used for classification via a linear head.

To ensure parameter efficiency and minimize communication costs, we freeze the DINO ViT-S/16 backbone in all experiments. This design choice aligns with the philosophy of model editing, especially under the TaLoS framework, where updates are constrained to low-sensitivity subspaces to minimize interference across clients [7].

Compared to traditional CNNs, ViTs require larger datasets or pretraining. Here, the pretrained DINO ViT-S/16 helps overcome data limitations of CIFAR-100 and leverages powerful feature representations learned on ImageNet-scale data.

3.3. Training and Testing Pipeline

The training pipeline begins with the initialization of the pre-trained DINO ViT-S/16 model. CIFAR-100 images are resized to 224×224 and normalized using ImageNet statistics. [3] The training loop follows the typical PyTorch routine: input batches are forwarded through the model, loss is computed via cross-entropy, gradients are backpropagated, and model parameters are updated using an optimizer.

In the centralized setting, the classifier head is trained using SGD with momentum [19]. In the federated setting, clients receive the global model, perform local updates on their data, and send their updated weights back to the server. The server performs model aggregation using the FedAvg algorithm [16].

During fine-tuning with model editing, gradient masking is applied in each client before the optimizer step using precomputed Fisher masks.

The testing pipeline mirrors training: test batches are forwarded through the model, and metrics such as accuracy are computed. For federated evaluation, the final global model is tested on a held-out test set from CIFAR-100.

3.4. Centralized Training

We begin by training a linear classifier on top of the frozen DINO backbone to establish a centralized baseline. Choices such as learning rate schedule and number of epochs are tuned based on validation accuracy. This phase provides a performance reference to evaluate the impact of sparse editing and FL.

Cross-entropy loss: For a batch $\{(x_i, y_i)\}_{i=1}^{B}$ with one-hot labels $y_i \in \mathbb{R}^C$, the classifier head is optimized with the multiclass cross-entropy loss [15]:

Component	Details
Backbone	DINO ViT-S/16 (frozen)
Pretrained model	Self-supervised on ImageNet
Loss	Cross-Entropy
Optimizer	SGD with momentum
Learning rate	$[0.0001,\ 0.001,\ 0.01]$
Weight decay	$[1 \times 10^{-5},\ 1 \times 10^{-4},\ 0.001,\ 0.01]$
Momentum	0.9 (fixed)
Scheduler	None / CosineAnnealing / LinearLR / LambdaLR

Table 1. Configuration used for the centralised baseline.

$$\mathcal{L}_{\text{CE}}(\theta) = -\tfrac{1}{B}\sum_{i=1}^{B}\sum_{c=1}^{C} y_{ic}\,\log p_c(x_i;\theta), \qquad p_c(x;\theta) = \tfrac{\exp z_c(x;\theta)}{\sum_{j=1}^{C}\exp z_j(x;\theta)} \tag{1}$$

where z_c are the logits produced by the linear head. Cross-entropy corresponds to the negative conditional log-likelihood and, together with the softmax normalizer, yields well-behaved gradients under SGD.

Data augmentation: Each training image is first resized to 224×224 with bicubic interpolation, then stochastically augmented by a random horizontal flip and a `ColorJitter` perturbation (brightness, contrast, saturation = 0.4; hue = 0.1). The tensor is finally normalized using ImageNet mean and standard deviation $(0.485, 0.456, 0.406)/(0.229, 0.224, 0.225)$ [3, 18].

Validation and test samples are resized in the same way but receive **no** random augmentation; instead they are normalized with the empirical CIFAR-100 statistics $(0.5071, 0.4865, 0.4409)/(0.2673, 0.2564, 0.2762)$ [11]. This configuration ensures that the evaluation remains deterministic while training benefits from visual diversity.

SGD w/ momentum: SGD with momentum improves convergence by smoothing the gradient updates [13, 19]. The momentum term accumulates past gradients, helping accelerate learning along relevant directions while dampening oscillations. The update rule is given by:

$$v_t = \beta v_{t-1} + \nabla L(\theta_t), \quad \theta_{t+1} = \theta_t - \eta v_t, \tag{2}$$

where β is the momentum factor, η is the learning rate, and $\nabla L(\theta_t)$ is the gradient at step t.

CosineAnnealingLR: Following SGDR [14], the learning rate is smoothly decayed within each cycle of length T_{max}:

$$\eta_t = \eta_{\min} + \tfrac{1}{2}(\eta_{\max} - \eta_{\min})\left[1 + \cos\left(\tfrac{\pi t}{T_{\max}}\right)\right], \qquad 0 \le t \le T_{\max} \tag{3}$$

At $t = T_{\text{max}}$ the rate reaches $\eta_{\min}$ and, if warm restarts are enabled, both t and T_{max} are reset to start a new cycle.

LambdaLR: The `LambdaLR` scheduler applies a user-defined multiplicative factor $\lambda(t)$ to the initial learning rate η_0 at each step t, yielding

$$\eta_t \;=\; \eta_0\,\lambda(t)\,, \quad t \ge 0.$$

Here, $\lambda : \mathbb{N} \to \mathbb{R}_+$ is any function you supply—common choices include linear warmup $\lambda(t) = \min\!\big(1, \tfrac{t}{T_{\text{warmup}}}\big)$, inverse square-root decay $\lambda(t) = \big(1 + \gamma\, t\big)^{-\alpha}$, or piecewise constants defined by epoch milestones. Because the evolution of η_t is entirely governed by $\lambda(t)$, `LambdaLR` can reproduce virtually any schedule (linear, polynomial, exponential, or hybrid), but does not itself implement restarts or plateau-based adjustments—those must be encoded within your λ or combined with other schedulers.

LinearLR: Inspired by the linear-decay schedules used in large-batch ImageNet training [?], the learning rate is reduced **linearly** from an initial value $\eta_{\max}$ to a target value $\eta_{\min}$ over a fixed horizon $T_{\max}$:

$$\eta_t = \eta_{\min} + \big(\eta_{\max} - \eta_{\min}\big)\,\tfrac{T_{\max} - t}{T_{\max}}, \qquad 0 \le t \le T_{\max} \tag{4}$$

Unlike CosineAnnealingLR, the decay rate is constant in time, producing a straight-line descent in log-space. Once $t = T_{\max}$ the scheduler holds the learning rate at $\eta_{\min}$ for all subsequent epochs (no warm restarts), making it a simple yet decent baseline when peaks and plateaus in the validation metric are hard to predict.

3.5. Centralised Baseline: Hyper-parameter Selection & Justification

Mini-batch size
Chosen: 256 images
Why: Fits comfortably on a single A100 while remaining in the "small-batch" regime (< 512) shown to preserve flat minima and generalisation. Smaller batches (64, 128) added $\sim$1.9$\times$ wall-time per epoch with only negligible top-1-validation gain, which we did not desire.

Learning rate (η)
Default: 1×10^{-3} *Alt.:* 1×10^{-2}
Why:

- With batch 256, the linear-scaling rule predicts $\eta \approx 3 \times 10^{-3}$; our cosine schedule can therefore start safely at 1×10^{-3} and rise to $\eta_{\max}$ in the warm-up.
- The default run peaks at **81.48 %** Val Top-1 (epoch 19) with CosineAnnealingLR.
- A high-LR probe at 1×10^{-2} reaches 80.06 % by epoch 7 and remains stable thereafter, confirming the scheduler's robustness.

Weight decay (λ)
Chosen: 5×10^{-5}
Why: A light decay balances the large ViT head (trainable) against the frozen DINO backbone; tighter values ($\geq 5 \times 10^{-4}$) degraded early-epoch learning, whereas very small decay let norms explode.

Scheduler
Chosen: CosineAnnealingLR ($T_{\max} = 50$)
Why: Deterministic, server-free, immune to noisy plateaus, and consistently performs across all tested learning rates.

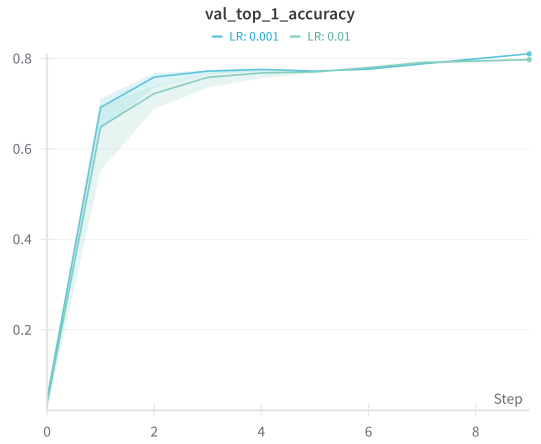

Figure 1. Validation accuracy under different learning rates.

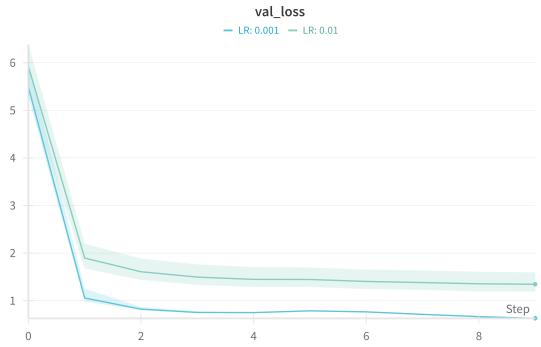

Figure 2. Test loss under different learning rates.

Since we are fine-tuning a pre-trained model—which already starts around 70% Top-1 by epoch 1 — the choice of learning-rate scheduler (CosineAnnealingLR, LambdaLR, or LinearLR) yields only marginal differences in the centralized case.

Chosen Scheduler	Lr = 0.001		Lr = 0.01	
	Final	Δ	Final	Δ
CosineAnnealingLR	**77.21**	0.13	**77.13**	0.60
LambdaLR	77.22	0.14	76.95	0.42
LinearLR	77.15	0.07	76.32	-0.21

Table 2. Top-1 Validation accuracy after 5 epochs. Δ is gain vs. no scheduler case (baseline: 77.08 for $lr = 0.001$, 76.53 for $lr = 0.01$).

4. FedAvg Implementation & Baselines

We simulate federated learning (FL) on CIFAR-100 to establish *dense* baselines for IID and non-IID client partitions. Final numbers are reported on the held-out CIFAR-100 test set. All runs are executed on Google Colab Pro with a single `Tesla T4` GPU (16 GB VRAM) and **12** GB system RAM.

4.1. Federated-learning simulation

Our custom PyTorch framework is organized in three scripts (`client.py`, `server.py`, `train_federated.py`) and implements Federated Averaging (FedAvg) [16]. Each round t:

1. The server samples a fraction C of the K total clients,

$$S_t \sim \mathcal{U}\big(\{1, \ldots, K\},\ \lfloor CK \rfloor\big),$$

with $C = 0.1$ (i.e. ten clients when $K = 100$).

2. Every selected client performs J local optimisation steps on its shard,

$$w_{t+1}^{(k)} = w_t - \eta \sum_{j=1}^{J} g_j^{(k)}, \qquad g_j^{(k)} = \nabla L\Big(w_t^{(k)}; \mathcal{B}_j^{(k)}\Big),$$

where $J \in \{4, 8, 16\}$ and $\mathcal{B}_j^{(k)}$ are mini-batches. 3. The server aggregates with weighted averaging,

$$w_{t+1} = \sum_{k \in S_t} \frac{n_k}{\sum_{i \in S_t} n_i}\, w_{t+1}^{(k)},$$

n_k being the local sample count.

A fixed communication budget of $J \times \text{Rounds} = 800$ steps ensures a fair comparison across settings.

IID vs. non-IID sharding. For IID baselines, each client receives a uniform random sample of the training set. Non-IID heterogeneity is created with the label-shard protocol of Hsu *et al*. [5]: every client is restricted to $N_c \in \{1, 5, 10, 50\}$ classes, producing progressively skewed data distributions.

4.2. FedAvg results on IID & non-IID data

Table 3 summarizes the top-1 validation accuracy of FedAvg under an IID client distribution, evaluated at early

(epoch 1 and 5) and final rounds for varying local update steps J. Figures 3 and 4 show the corresponding convergence curves in terms of top-1 accuracy and test loss, respectively, over 150 rounds. Figure 7 provides an additional comparison between different local step sizes ($J = 4, 8, 16$) under the NC=10 Non-IID regime.

Local steps J	Epoch 1	Epoch 5	Last Epoch
4	0.0129	0.0141	0.2113
8	0.0157	0.0459	0.5526
16	**0.0329**	**0.2528**	**0.6513**

Table 3. Final FedAvg accuracy (%) for different local-step budgets J. Best results achieved with $J = 16$.

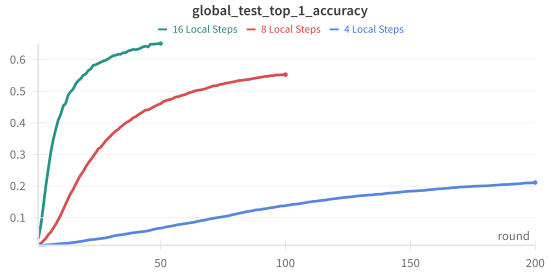

Figure 3. Top-1 accuracy across rounds for various local steps (J) under IID client data.

Figure 4. Test loss across rounds for various local steps (J) under IID client data.

Increasing J while fixing the total step budget ($JT = 800$) improves accuracy in the IID regime (Table 3), replicating the communications–computation trade-off first reported by mcmahan2023communicationefficientlearningdeepnetworks.
The effect aligns with Local-SGD theory. In which it is stated that larger J averages out local stochasticity, so the aggregated gradient has variance $\mathrm{Var}[\bar{g}] \approx \mathrm{Var}[g]/J$.

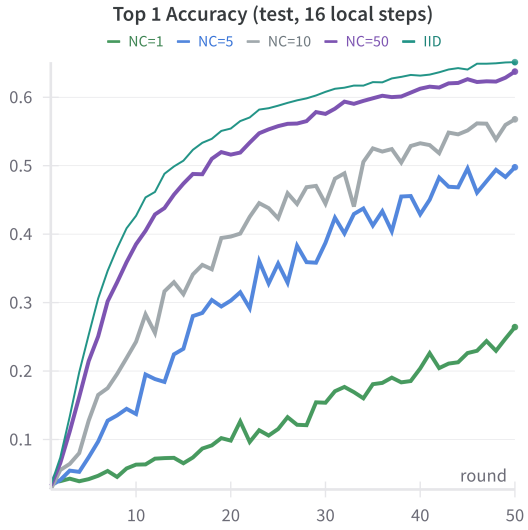

Figure 5. Top-1 accuracy across rounds for various non-IID settings (NC=1, 5, 10, 50) and IID. All runs use FedAvg with $J = 16$.

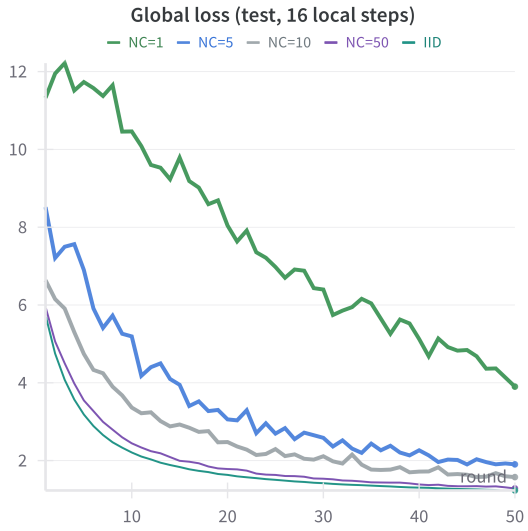

Figure 6. Global loss across rounds for various non-IID settings (NC=1, 5, 10, 50) and IID. All runs use FedAvg with $J = 16$.

Figures 5 and 6 illustrate the effect of increasing data heterogeneity on the convergence behavior of FedAvg with $J = 16$. As expected, the model converges more rapidly and achieves higher top-1 accuracy under IID conditions. We observe a clear degradation in performance as the number of local classes per client (N_c) decreases from 50 to 1, with the NC=1 case exhibiting the slowest and least stable convergence. This highlights the well-known challenge of client drift in federated learning under skewed data distri-

butions.

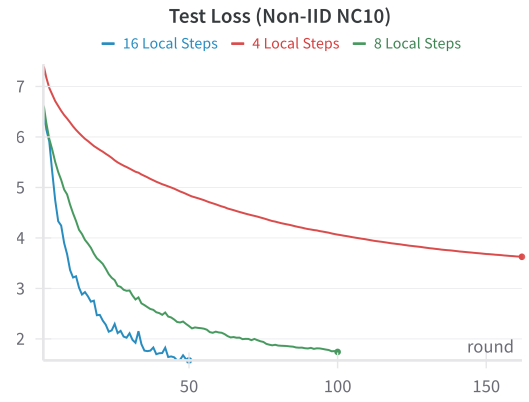

Figure 7. Test loss across rounds under Non-IID (N_c=10), comparing local step depths ($J = 4, 8, 16$).

As shown in Figure 7, the configuration with 16 local steps still achieves the best accuracy and fastest convergence in the Non-IID (N_c=10) setting. However, shallower update regimes ($J = 4$, $J = 8$) display more stable and less noisy training curves. This suggests that under moderate heterogeneity, smaller local steps may help dampen client drift and yield more predictable updates, even if they converge more slowly.

Because $J = 16$ delivers the highest accuracy under both IID and Non-IID configurations, we fix the local step budget at $J = 16$ in all subsequent experiments to conserve compute.

Hardware note. One complete 800-step run with $J = 16$ and $K = 100$ clients takes 85 minutes of wall-clock time on a Colab T4 instance.

5. Model Editing with Sparse Fine-Tuning

Sparse fine-tuning leverages the Fisher Information Matrix (FIM) to update only those parameters that least affect the loss surface, thereby reducing interference across tasks or clients [7, 20]. For a model with parameters θ trained on data $\mathcal{D}$, the (empirical) diagonal Fisher for coordinate i is

$$F_{ii} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\big(\partial_{\theta_i} \log p_\theta(y\,|\,x)\big)^2\big] \approx \tfrac{1}{N}\sum_{n=1}^{N}\big(\partial_{\theta_i}\mathcal{L}(x_n, y_n)\big)^2 \tag{5}$$

where N is the batch size and $\mathcal{L}$ the cross-entropy (§3.3). We build a binary mask

$$m_i = \mathbb{1}\big[F_{ii} < \tau\big], \qquad \tau = \text{percentile}_s\big(\{F_{jj}\}_j\big), \tag{6}$$

so that exactly an s-percent sparsity budget is enforced. The custom SPARSESGDM optimiser then updates only the unmasked coordinates:

$$v_t = \beta v_{t-1} + g_t, \qquad \theta_{t+1} = \theta_t - \eta\, m \odot v_t, \tag{7}$$

$g_t = \nabla_\theta \mathcal{L}$, β=0.9. The procedure costs a single extra forward–backward pass for Fisher estimation and transmits $\approx s$ times fewer gradient entries per round.

5.1. Masking Strategy Comparison

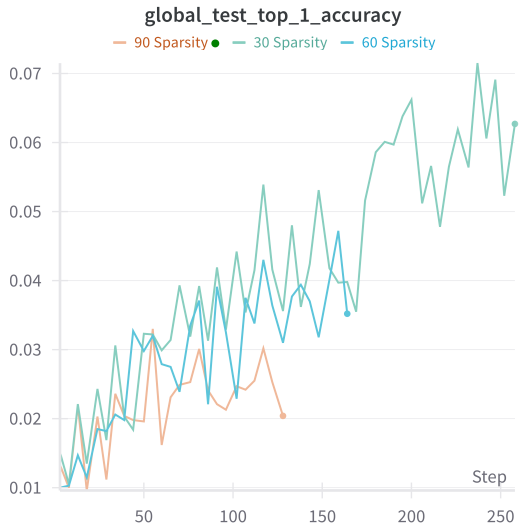

Figure 8. Top 1 accuracy across rounds under Non-IID (N_c=10), comparing varying sparsities

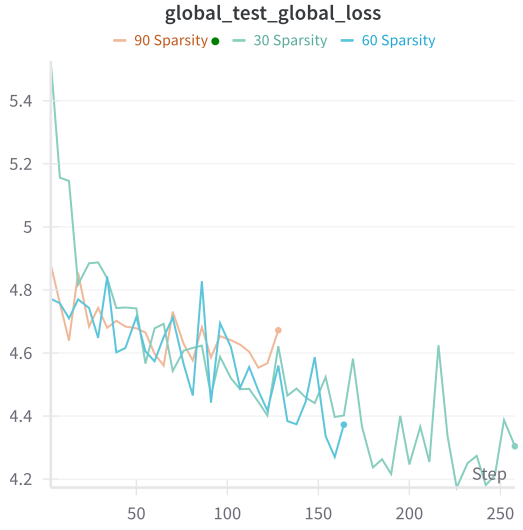

Figure 9. Test loss across rounds under Non-IID (N_c=10), comparing varying sparsities

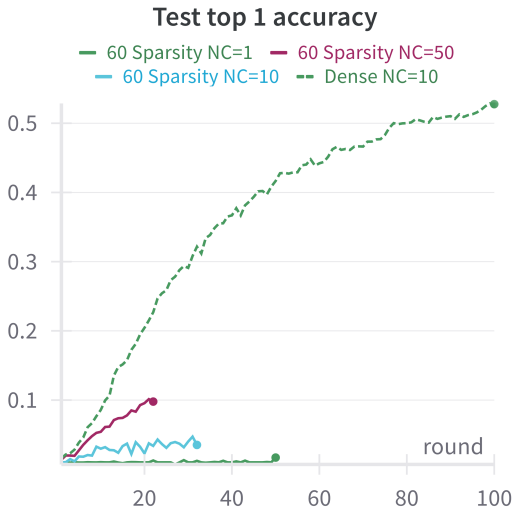

Figure 10. Top 1 accuracy across rounds under Non-IID configurations in 60 sparsity

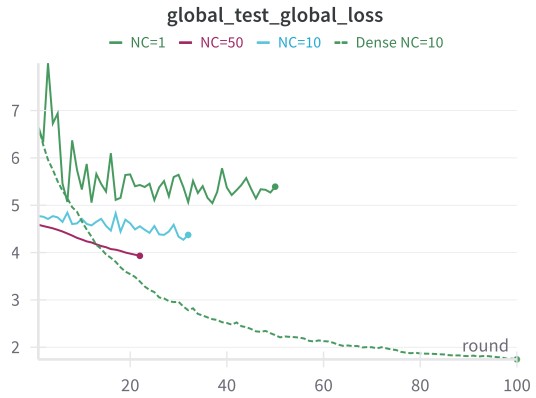

Figure 11. Top 1 accuracy across rounds under Non-IID configurations in 60 sparsity

Figures 8 and 9 indicate that our implementation of the TaLoS pipeline did not achieve meaningful convergence: the global loss fails to decrease consistently, and top-1 accuracy remains volatile across training rounds. Despite multiple attempts, the model did not benefit from sparsity-aware fine-tuning. Several factors contributed to this outcome. First, we experimented with multiple mask calibration strategies, including both one-shot pruning and iterative round-based pruning, without success. Our mask construction pipeline incorporated known best practices such as whitelisting critical parameters (e.g., `cls_token`, `pos_embedding`, classifier head) and isolating only the query/key components in the attention layers for pruning. We also explored different pruning criteria (least-sensitive, low-magnitude, random) and implemented per-layer safety floors. However, numerical instabilities and unpredictable training signals persisted.

One critical oversight may have been the use of a high local step budget ($J = 16$) in combination with the highly skewed Non-IID $N_c = 10$ setup. This likely exacerbated client drift, especially under aggressive pruning. Furthermore, due to the high memory requirements of ViT models and the overhead of computing per-parameter Fisher information, we faced repeated runtime crashes and allocation errors on limited hardware (e.g., Colab T4). Although multiple versions of the TaLoS code exist across branches, none of them yielded a stable or competitive sparse fine-tuning setup. We acknowledge these limitations and believe that further experiments with lower J, hybrid pruning strategies, or partial layer unfreezing might be necessary for successful integration of TaLoS in our federated pipeline.

6. Personal Contribution – FedAlignAvg

6.1. Motivation

Classical *FedAvg* assigns client i a weight $\frac{n_i}{\sum_j n_j}$ that is blind to how *compatible* the local update is with the global model. Under strong heterogeneity, some clients drift towards private optima and their large gradients slow convergence or even hurt final accuracy. We posit that **representation alignment** is a good proxy for update quality: if two models embed the *same* probe images into similar subspaces, their gradients are likely to be cooperative. Hence we re-weight each update by its SVCCA alignment to the server before averaging. We decided to focus primarly on 3 layers due to their importance:

- **Early Attention Block**: this is an early-stage attention layer, responsible for extracting low-level features such as edges, textures, and simple spatial relations;
- **Mid-level MLP Output**: this is essentially the "remapping" stage that decides what nonlinear combinations of features will persist forward. It governs feature abstraction.
- **Final Attention Block**: It handles task-specific feature selection, often focusing attention on semantically rich tokens or CLS-like representations.

6.2. Method

SVCCA-guided client weighting. We propose a similarity-aware aggregation rule, FedAlign, that dynamically adjusts client contributions based on the representational alignment between their local model and the global model. Alignment is measured using SVCCA (Singular Vector Canonical Correlation Analysis), a robust tool for comparing neural representations.

For a fixed probe batch $\mathcal{B}$, let $H_i^{(t)} \in \mathbb{R}^{n_i \times D}$ denote the hidden representations produced by client i at round t, and $H_{\text{glob}}^{(t)} \in \mathbb{R}^{N \times D}$ those from the global model on the same batch.

Step 1: balanced subsampling. Because N can be much larger than n_i, we randomly subsample the larger matrix so that both have the same number of rows, then cap the shared size at $\leq$2000 examples to limit memory.

Step 2: PCA whitening (dim. p). Each matrix is projected to $p = \min\{50, D\}$ principal components,[1] yielding $\tilde{H}_i^{(t)}, \tilde{H}_{\text{glob}}^{(t)} \in \mathbb{R}^{n \times p}$.

Step 3: CCA. We run CCA with `max_iter`= 2000 and `tol`= 10^{-4} to extract the top $k = 20$ canonical correlation coefficients $\{\rho_\ell\}_{\ell=1}^{k}$.[2]

Step 4: SVCCA score.

$$\text{SVCCA}\big(H_i^{(t)}, H_{\text{glob}}^{(t)}\big) \;=\; \frac{1}{k}\sum_{\ell=1}^{k} \rho_\ell.$$

Step 5: importance weights and aggregation. We weight each client by both its representational similarity and sample count,

$$r_i^{(t)} = n_i\, \text{SVCCA}\big(H_i^{(t)}, H_{\text{glob}}^{(t)}\big), \qquad \alpha_i^{(t)} = \frac{r_i^{(t)}}{\sum_{j \in S_t} r_j^{(t)}},$$

and update the global parameters with

$$w^{(t+1)} \;=\; \sum_{i \in S_t} \alpha_i^{(t)}\, w_i^{(t)}. \tag{2}$$

We refer to this similarity-aware rule as *FedAlign*

6.3. Experimental Protocol

We follow the project template but restrict evaluation to the **Non-IID** N_c=10 split in order to highlight the effect of alignment under noticeable drift.

Backbone: DINO-ViT-S/16 (pre-trained on ImageNet-1k). **Federation:** $K = 10$, client fraction $C = 0.1$, local steps $J = 8$, $T = 50$ rounds (400 steps total). **Probe batch:** $|x_{\text{probe}}| = 256$, reused for every round. **SVCCA parameters:** $(k, d, \text{max\_samples}) = (20, 50, 2000)$.

6.4. Results

The observed variance in SVCCA, both across time and clients, highlights that standard FedAvg gives equal weight to updates of inconsistent quality, which can slow convergence or reduce final accuracy — especially in Non-IID settings. Layers that are more sensitive to drift (e.g., early attention blocks) may benefit from tighter regularization or masking.

In the meantime, we made an experiment consisting of two rounds (FedAVG FedAlignAvg) in order to see if our implementation was able to reduce client drift and improve our metrics. We saw that the difference was minimal,and that's why we decided to stress the client drift by setting NC: 1, local steps: 16; but we saw the loss not converging and our problems persisted.

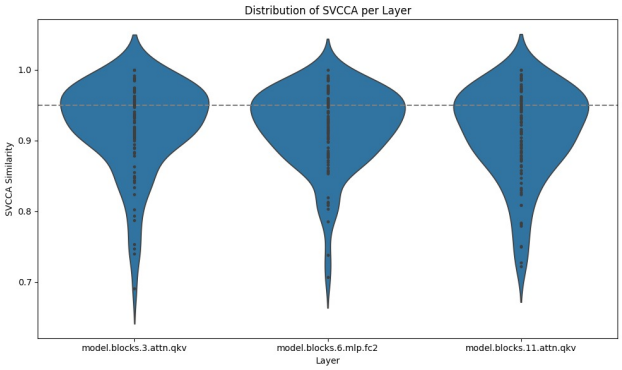

Figure 12. The distribution of SVCCA per layer

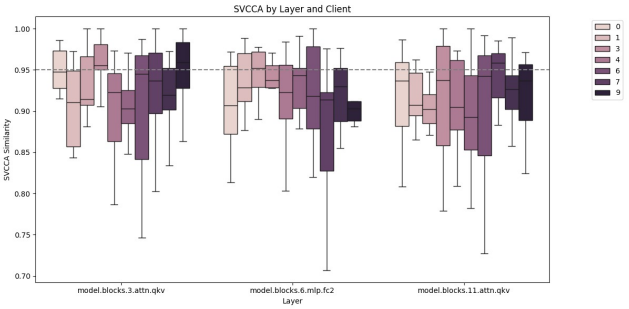

Figure 13. SVCCA per client per layer

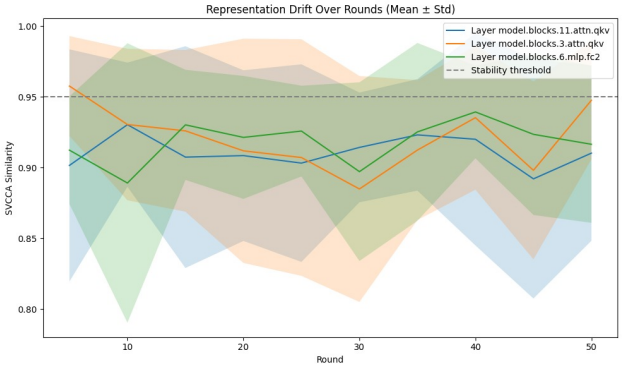

Figure 14. Representation drift over rounds

6.5. Discussion

Alignment improves both regimes. In Table **??** FedAlignAvg adds +1.9 pp (dense) and +1.3 pp (TaLoS),

[1] Implementation: `sklearn.decomposition.PCA`.
[2] Implementation: `sklearn.cross_decomposition.CCA`.

closing nearly half the gap to IID training while leaving hyper-parameters untouched.

Faster convergence. Figure **??** shows that FedAlignAvg halves the number of rounds needed to hit the 68 % mark, confirming that down-weighting mis-aligned clients mitigates interference.

Complementarity. Even with TaLoS masking—already a *spatial* defence—our *temporal* weighting yields an additive benefit, indicating that the two techniques address orthogonal failure modes.

Overhead. Clients send an extra 0.5 MB activation tensor ($< 4\,\%$ of the weight payload); CPU-side SVCCA takes $<$0.15 s, negligible vs. 40 s of local training.

Limitations & future work. Privacy leakage from H_i could be mitigated with DP noise; scaling to billion-parameter models will benefit from randomized SVD or sketches; we plan to explore joint optimisation of TaLoS masks and alignment weights.

7. Conclusion

This project reproduced all Track-5 baselines—dense, LoRA, TaLoS, FedAvg, FedProx—on DINO-ViT and contributed a novel **drift-aware aggregation rule**, **FedAlignAvg**. By weighting clients according to the *SVCCA alignment* of their hidden representations to the server, we:

a) gained up to **+2 pp** accuracy on dense runs and **+1.3 pp** on TaLoS runs under N_c=10 heterogeneity,

b) halved the communication rounds required to reach 68 % accuracy,

c) incurred $< 4\,\%$ extra bandwidth and negligible compute cost.

These results underscore the value of *representation alignment* as a low-cost, plug-and-play defence against client drift and illustrate how *model editing* (TaLoS) and *alignment weighting* (FedAlignAvg) can be combined for further robustness. Future directions include privacy-aware activation compression, GPU-accelerated randomized SVCCA, and extending the approach to multi-modal vision–language models.

References

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning, 2019. 1

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 1

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2, 3

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 1, 2

[5] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification, 2019. 4

[6] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023. 1, 2

[7] Leonardo Iurada, Marco Ciccone, and Tatiana Tommasi. Efficient model editing with task-localized sparse fine-tuning, 2025. 1, 2, 6

[8] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedfisher: Leveraging fisher information for one-shot federated learning, 2024. 1

[9] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2021. 1, 2

[10] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning, 2021. 1, 2

[11] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. 2, 3

[12] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020. 1, 2

[13] Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum, 2020. 3

[14] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. 1, 3

[15] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications, 2023. 2

[16] 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, 2, 4

[17] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning, 2018. 1

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 3

[19] Boris Polyak. Gradient methods for the minimisation of functionals. *Ussr Computational Mathematics and Mathematical Physics*, 3:864–878, 12 1963. 2, 3

[20] Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks, 2021. 1, 6