Centralized and Federated Learning with Sparse Fine-Tuning & Communication and Client Drift Analysis

Gabriele Pirilli s308809
Simone Vigna s349139
Enrico Ferrero s346061
Vida Gallo s308810

Abstract

This work explores centralized and federated training approaches for image classification on the CIFAR-100 dataset using the pre-trained DINO ViT-S/16 model. A centralized baseline was established before simulating federated learning across 100 *clients with both i.i.d. and non-i.i.d. data distributions. Sparse fine-tuning guided by parameter sensitivity (Fisher Information) was applied in both settings, following a two-step strategy: first, only the classifier head was trained, and then the backbone was fine-tuned sparsely. Finally, communication and client drift in the federated approach were analyzed.*

1. Introduction

Federated learning (FL) is a machine learning setting where many clients (e.g. mobile devices or organizations) collaboratively train a model under the orchestration of a central server (e.g. service provider), while keeping the training data decentralized[1]. This method helps protect privacy, but it also creates challenges as clients can have different types of data or hardware.

Additionally, model editing has emerged as a technique for efficiently modifying or merging fine-tuned models without retraining them from scratch. It is possible to adopt a parameter-level strategy based on fine-tuning, where updates can be guided by parameter sensitivity (e.g., Fisher information). This approach can be naturally integrated into federated learning, potentially helping to mitigate conflicts between updates from heterogeneous clients[2].

This work explores centralized and federated training approaches, investigating the effects of heterogeneous data distributions and sparse fine-tuning guided by parameter sensitivity. Experiments were performed on the CIFAR-100 dataset[1] with the pre-trained self-supervised vision transformer DINO ViT-S/16 [2].

The report is structured as follows: Section 2 reviews the relevant literature, Section 3 introduces the methodological framework, Section 4 presents the experimental results, and Section 5 draws the conclusions.

2. Related work

Federated learning (FL) is a distributed machine learning paradigm in which a large number of clients coordinate with a central server to learn a model without sharing their own training data[3]. The term *federated learning* was introduced in 2016 by McMahan et al.[4] and has seen growing interest, both from research and applied perspectives[1].

Federated Learning poses several well-known challenges that are extensively studied in the literature. Among the most critical is *statistical heterogeneity*[5, 6], where client data is typically non-IID and unbalanced. This leads to degraded performance[7] and unstable or slow convergence as the discrepancy between local data distributions increases[3, 8]. This is further compounded by *systems heterogeneity*[5], arising from the varying computational capabilities of participating devices. Additionally, Federated Learning faces *privacy, security threats and fairness*[9, 10], such as gradient leakage and poisoning attacks, as well as concerns regarding training effectiveness and communication efficiency[10].

A foundational algorithm in Federated Learning (FL) is `FedAvg` (short for Federated Averaging), proposed by McMahan et al.[4]. Since then, numerous variants have been introduced to address challenges such as convergence instability and client drift[3]. A more general form of `FedAvg`, known as `FedOpt`, was later developed to provide greater flexibility in optimization strategies[1]. Several improvements to `FedAvg` have been proposed, incorporating momentum into SGD to accelerate convergence and reduce oscillations[7], or leveraging adaptive optimization methods such as `ADAGRAD`, `ADAM`, and `YOGI`[3]. Other strategies to address data heterogeneity and instability include `FedProx`[5], `SCAFFOLD`[8], `FedVC`[11], `FedIR`[11], `FedBN`[12], and several others[6, 10].

Fine-tuning large pre-trained models on specific tasks is a

[1]https://www.cs.toronto.edu/~kriz/cifar.html
[2]https://github.com/facebookresearch/dino

common approach for adapting their capabilities efficiently, and recent works have explored sparse fine-tuning and task arithmetic as scalable strategies for model editing[2, 13]. Sparse fine-tuning updates only a subset of model parameters and this parameter-level editing approach is particularly relevant in federated learning, where it enables efficient local adaptation while preserving privacy and reducing communication, combining multiple specialized versions of a base model into a single unified model [13].

3. Methods

3.1. Centralized Training

Centralized training was performed by minimizing the empirical loss over the entire dataset:

$$\min_{w\in\mathbb{R}^d} f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i; w), \tag{1}$$

where ℓ denotes the loss function and w the model parameters. Optimization was carried out using stochastic gradient descent with momentum (`SGDM`), combined with a cosine annealing learning rate schedule.

3.2. Federated Learning

In the federated setting, data is distributed across K clients, each holding a private dataset $\mathcal{P}_k$ of size n_k. The global objective function is:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w), \tag{2}$$

$$\text{with} \quad F_k(w) = \frac{1}{n_k}\sum_{i\in\mathcal{P}_k} \ell(x_i, y_i; w). \tag{3}$$

If training data is distributed uniformly at random across clients, then $\mathbb{E}_{\mathcal{P}_k}[F_k(w)] = f(w)$, defining the **i.i.d. setting**. When this does not hold, local objectives $F_k(w)$ may differ substantially from the global objective $f(w)$, defining the **non-i.i.d. setting**, which introduces challenges due to heterogeneous client data (*statistical heterogeneity*).

A further challenge in Federated Learning is the significant variability in system characteristics across the devices in the network (*system heterogeneity*). In this work, only the statistical heterogeneity of the data was considered, while all clients were assumed to be identical in terms of system characteristics.

Two strategies were used to partition the dataset among clients:

- In the *i.i.d. split*, the dataset indices are randomly shuffled and evenly divided into K disjoint subsets.
- In the *non-i.i.d. split*, each client is pre-assigned N_c random classes. Samples are shuffled within each class and then iteratively assigned to the clients associated with that class, ensuring that every example belongs to exactly one client.

To enable decentralized optimization, the Federated Averaging (`FedAvg`) algorithm was adopted. In each communication round, a subset of clients performs local `SGDM` updates on their data and the server aggregates the resulting models through weighted averaging.

3.3. Model Editing and Sparse Fine-Tuning

The sparse fine-tuning strategy was applied in both centralized and federated settings. Following Iurada et al. [2], model updates are restricted to a subset of parameters identified as least sensitive, namely those whose modification minimally affects the model's predictive behavior. Sensitivity scores were computed using the diagonal of the Fisher Information Matrix and used to construct a binary gradient mask that selects only the low-sensitivity parameters for updating. In the federated setting, each client computes and calibrates its own gradient mask using local Fisher Information scores.

A modified version of the `SGDM` optimizer, referred to as `SparseSGDM`, was implemented to update only the selected parameters. Starting from a pre-trained model, the classifier head was first trained while keeping the rest of the network frozen. Afterwards, the backbone was updated selectively according to the *sparsity ratio*, which indicates the fraction of parameters that are frozen (i.e., not updated) relative to the total number of trainable parameters.

As noted by Iurada et al. [2], there is a risk of *layer collapse*, where freezing all parameters in a layer could disrupt information flow. To prevent this, all parameters start fully trainable and less sensitive weights are gradually frozen over multiple calibration rounds. Additionally, the mask was designed so that at least one parameter in each layer remains active.

4. Experiments

Experiments were conducted on the CIFAR-100 dataset using `Torchvision` for dataset loading and preprocessing and `PyTorch` for neural network modeling and training. All experiments started from the pretrained DINO ViT-S/16 backbone weights.

The dataset includes 60.000 images across 100 balanced classes, with 50.000 for training and 10.000 for testing. To tune the hyperparameters, a portion of the training data (20%) was reserved as a validation set. In the federated setting, only the training portion was distributed among clients, while the centralized validation set was used exclusively to evaluate the aggregated global model (i.e. the model obtained after averaging client updates). The test set was never used for model selection. After identifying the optimal hyperparameters, the model was trained on the entire training set and then evaluated on the test set.

A centralized training baseline was first established, then extended to simulate a federated learning setup via sequential client updates. Hyperparameters, such as learning rate, momentum, weight decay, number of epochs, number of local steps (for FL), were tuned via Bayesian Optimization or via Grid Search using `Weights & Biases (W&B)`[3]. Experiments were conducted on the free-tier `Google Colab`[4] using an NVIDIA T4 GPU.

4.1. Data Exploration

An initial analysis of the CIFAR-100 dataset was performed to better understand its structure and characteristics. First, the basic properties of the training and test sets were inspected. Each sample in the dataset consists of a color image tensor of shape $(3, 32, 32)$ and an associated label. To verify the balance of the dataset, the class distribution in the training and test sets was analyzed. It was confirmed that each of the 100 classes contains exactly 600 samples (500 for training and 100 for testing), indicating that the class distribution is uniform and matches the expected structure of CIFAR-100. This uniformity prevents the model from being biased toward specific classes, thereby enabling more balanced training and more reliable evaluation.

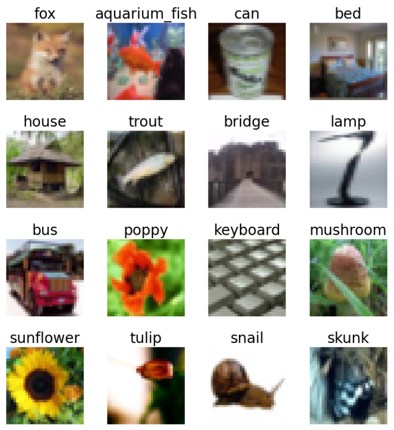


Figure 1. A random sample of 16 training images with their corresponding class labels.

To gain a better intuition about the dataset, a visual inspection of sample images was also conducted. For example, Figure 1 shows a selection of 16 randomly chosen training images with their associated class labels.

To further explore dataset variability, a similarity analysis was performed by flattening image tensors and computing Euclidean distances between them, allowing us to identify the most similar and dissimilar samples with respect to a given reference. Intra-class variability was then assessed by calculating the average distance of each image to the centroid of its class, offering insight into how compact or dispersed each class is. These analyses were purely exploratory and not directly employed in the training experiments.

Finally, the analysis of outliers was conducted, revealing that many of these instances exhibit unusual object colors or cropped fragments of larger images. Outliers were identified via the Mahalanobis distance between each image's mean RGB and its class distribution, using a threshold derived from the chi-squared distribution at 97.5% confidence. Although atypical, such samples provide valuable information for classification, and since CIFAR-100 is already a high-quality, balanced dataset, no instances were removed.

4.2. Data Transformation

Before training, all images were upscaled from 32×32 to 64×64 using bilinear interpolation, increasing the amount of visual information available in each image. The images from both the training and test sets were then converted to tensors and normalized per channel using the mean and standard deviation computed from the training set. The preprocessed datasets were then divided into mini-batches of size 64, allowing efficient training, lower memory usage and more stable gradient estimates.

4.3. Centralized Approach

In order to train the centralized model, first linear probing was performed using the pre-trained weights, freezing the backbone and tuning only the head. Subsequently, sparse fine-tuning was applied, keeping the classifier weights fixed and selectively training the backbone. The effect of sparsity and the selection strategy was analyzed (Figure 2), considering the following options:

- Least-sensitive parameters (lowest Fisher scores)
- Most-sensitive parameters (highest Fisher scores)
- Random selection

The lower the sparsity, the more the three selection methods are equivalent. The outcomes and the used hyperparameters are reported in Table 1. Considering the computational times, the results are similar across different sparsity levels due to the way `SparseSGD` was implemented. Even when using sparsity masks, the network performs dense matrix multiplications during both forward and backward passes, and only afterward are the masked weights zeroed out, so sparsity does not automatically reduce training time. Nonetheless, higher sparsity slightly reduces the computational time required for mask calibration.

As mentioned by Iurada et al. [2], sparse fine-tuning could be applied only to *Linear*, *Attention*, *LayerNorm*, and

[3]https://wandb.ai/site/
[4]https://colab.google/

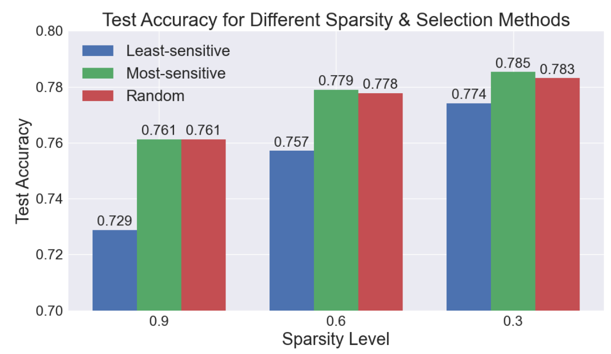


Figure 2. Test accuracy for different Fisher score selection methods at varying sparsity levels.

Convolutional layers, keeping all other layers frozen. In the centralized setting, this led to a slight decrease in accuracy (-0.1%) and increase in the loss ($+0.4\%$) at parity of hyperparameters and epochs, and to a small reduction in training time (-1.7%). The backward pass cost is lowered for the frozen layers, but the forward pass remains dense, so the overall speed-up was limited.

Finally, three different training strategies (Figure 3) were compared, highlighting the performance of (1) Head-only training + Sparse backbone fine-tuning, (2) Full model training, (3) Head-only training + Sparse full model fine-tuning. Scenarios 1 and 3 are almost equivalent and ultimately outperform Scenario 2. The results and the corresponding hyperparameters are again reported in Table 1.

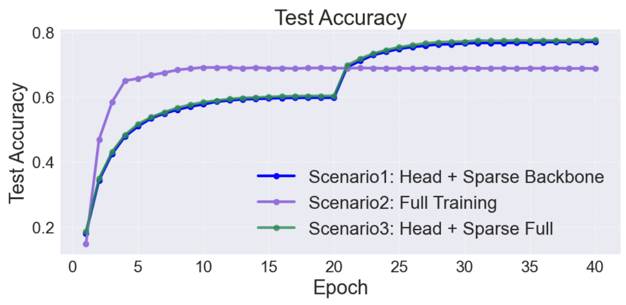


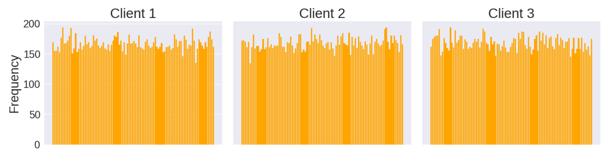
Figure 3. Results for three training strategies.

4.4. Federated Learning Approach

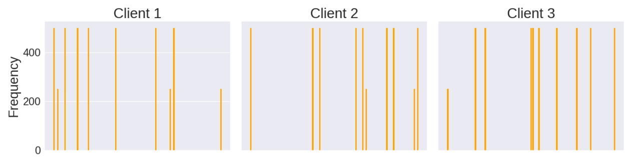
As a first step, the training set was partitioned into $K = 100$ clients:

- *i.i.d. sharding*: each client receives a balanced and representative subset spanning all classes (Figure 4a).
- *non-i.i.d. sharding*: each client receives samples from N_c classes only (Figure 4b).

In the federated setting, each client computes and calibrates its own gradient mask using local Fisher Information scores. Since storing full binary masks for all trainable parameters and for each client can became memory-intensive, we adopted a compression strategy. Each mask tensor was first compressed into a compact bit representation and later decompressed back to its original form during training.



(a) Class distribution for 3 i.i.d. clients.



(b) Class distribution for 3 non-i.i.d. clients ($N_c = 10$)

Figure 4. Example of training data allocation among clients.

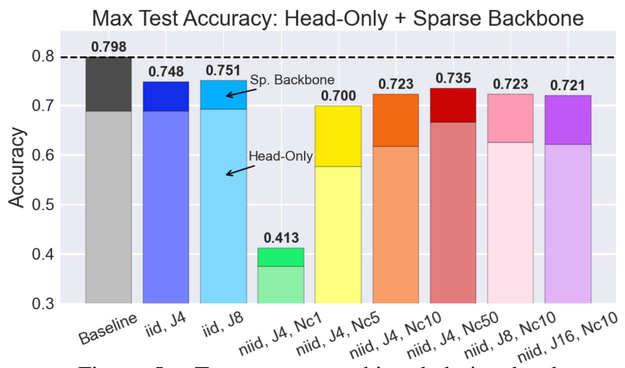


Figure 5. Test accuracy achieved during head training and subsequent sparse backbone fine-tuning across different FL configurations, compared with the centralized baseline, using the same selection method $=$ *least*.

The federated experiments, which are reported in Table 2, followed a procedure similar to the centralized ones. The model was initialized with pre-trained DINO ViT-S/16 weights. Firstly, only the classifier head was trained while keeping the backbone frozen; afterwards, the classifier was kept fixed and the backbone was fine-tuned (Figure 5). It was observed that in the non-i.i.d. setting a higher number of classes per client (N_c) resulted in a behavior more similar to the i.i.d. one. Moreover, increasing the number of local training rounds per client (J) improved performance in both i.i.d. and non-i.i.d. setups, but only up to a certain point. A too-high number of local epochs (J= 16) in the non-i.i.d. case worsened the final overall results due to client drift.

Furthermore, different methods for selecting which parameters to mask were evaluated (Figure 6), and additional experiments investigated training the backbone under vary-

Centralized setting results and hyperparameters

Name	Sparsity	Epochs	Mom.	Learning Rate	Calib. Rounds	Test Acc.$^{(*)}$	Test Loss$^{(*)}$	Epoch Best$^{(*)}$	Time (in s) (Head+Back)
CentrHead	-	40	0.8	0.001	-	0.6967	1.1012	34	1213.8
Baseline	0.5	50+100	0.9	0.0005	2	0.7984	0.9675	50+65	1245.6 + 6511.8
Figure 2:									
CentrSpBack	0.9	40+30	0.9	5×10^{-5}	1	0.7288	0.9945	40+27	1213.8 + 2024.4
CentrSpBack	0.6	40+30	0.9	5×10^{-5}	1	0.7578	0.8962	40+18	1213.8 + 1970.1
CentrSpBack	0.3	40+30	0.9	5×10^{-5}	1	0.7741	0.8941	40+27	1213.8 + 1966.3
Figure 3:									
Scenario_1	0.5	20+30	0.9	5×10^{-5}	2	0.7755	0.8203	20+28	516.1 + 2011.2
Scenario_2	-	50	0.9	5×10^{-5}	2	0.6910	1.3363	10	3246.3
Scenario_3	0.5	20+30	0.9	5×10^{-5}	2	0.7758	0.8163	20+22	518.5 + 2009.3

Table 1. Hyperparameters used for the training and performances of the centralized models. For all runs, a weight decay of 5×10^{-5} was applied ($L2$ regularization). Selection method for sparse fine tuning = *least*.
(*)The reported results are the best obtained, which do not always correspond to the values at the final epoch due to minor fluctuations during training or the onset of overfitting. The reported loss is the value corresponding to the epoch where the highest accuracy was achieved and it may not be the lowest loss observed.

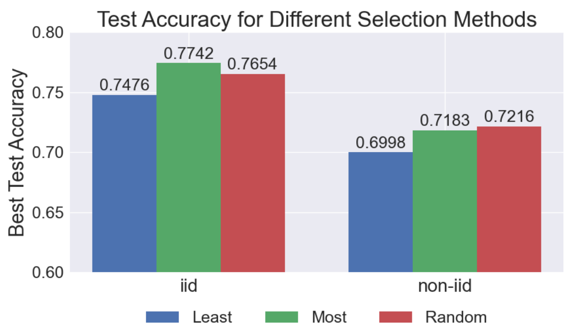


Figure 6. Test accuracy after initial head training, followed by sparse backbone training using three different Fisher score selection methods in both i.i.d. and non-i.i.d. ($N_c = 5$) FL scenarios.

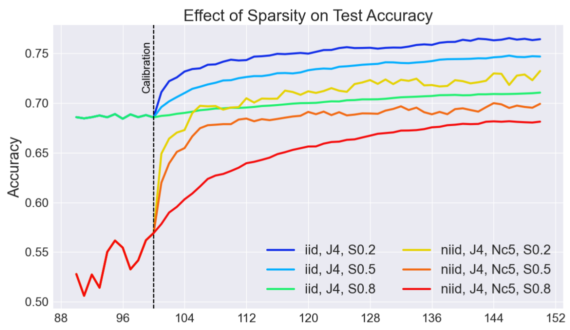


Figure 7. Test accuracy after initial head training, followed by sparse fine-tuning of the backbone for three different sparsity levels in both i.i.d. and non-i.i.d. ($N_c = 5$) federated learning scenarios.

ing sparsity levels (Figure 7). Increasing sparsity generally led to slightly faster training, due to quicker mask calibration, but resulted in lower accuracy.

The effect of mask calibration was also investigated (Figure 8). It was observed that, in general, increasing the number of calibration rounds slightly reduced accuracy and increased loss. For the non-i.i.d. case, it was also noted, by examining the behavior of the loss, that overfitting begins around 90 epochs, where the loss starts to increase slightly.

Finally, two different strategies were compared (Figures 9a and 9b) for the non-i.i.d. case ($N_c = 10$), highlighting the performance of (1) Full model training and (2) Head-only training + Sparse backbone fine-tuning. Scenario 2 shows a slower increase in accuracy, but it is much more stable. In particular, looking at the loss, one can observe a smoother decrease and lower final values when training first the head and then the backbone sparsely, leading to a more steady and controlled convergence. The results and the corresponding hyperparameters are again reported in Table 2, where it is also possible to notice that Scenario 2 achieves a notable saving in training time.

4.5. Further Analysis and Contributions

Communication is a key aspect of Federated Learning. The utilized DINO ViT-S/16 model has more than 21 million parameters, making communication a critical factor. Each global round involves a two-way exchange of model parameters between the server and the clients (server→clients and clients→server). While the full global model needs to be distributed from the server to all clients at the beginning of each round, sparse fine-tuning allows clients to transmit only the modified parameters instead of

Federated Learning setting results and hyperparameters

Figure 5: Sparsity $= 0.5$, Calibration Rounds $= 2$

Name	Global Epochs	Local Epochs	Test Acc.$^{(*)}$ Head $\rightarrow$ Back	Test Loss$^{(*)}$ Head $\rightarrow$ Back	Global$^{(*)}$ Ep.Best	Time (in s) (Back only)
FLSpBack_iid	50	4	0.6888 $\rightarrow$ 0.7476	1.1254 $\rightarrow$ 0.9006	46	2571.35
FLSpBack_iid	50	8	0.6925 $\rightarrow$ 0.7506	1.1492 $\rightarrow$ 0.9454	50	3767.47
FLSpBack_iid	50	16	0.6957 $\rightarrow$ 0.7512	1.1724 $\rightarrow$ 1.0195	39	6560.05
FLSpBack_niid_Nc1	50	4	0.3748 $\rightarrow$ 0.4125	16.5694 $\rightarrow$ 6.8017	43	1867.93
FLSpBack_niid_Nc5	50	4	0.5768 $\rightarrow$ 0.6998	3.0737 $\rightarrow$ 1.3067	44	2420.16
FLSpBack_niid_Nc10	50	4	0.6175 $\rightarrow$ 0.7229	2.2186 $\rightarrow$ 0.9747	47	2448.94
FLSpBack_niid_Nc50	50	4	0.6667 $\rightarrow$ 0.7350	1.5495 $\rightarrow$ 1.1505	49	2484.80
FLSpBack_niid_Nc10	50	8	0.6261 $\rightarrow$ 0.7233	1.4979 $\rightarrow$ 1.0216	42	3850.16
FLSpBack_niid_Nc10	50	16	0.6215 $\rightarrow$ 0.7209	1.5580 $\rightarrow$ 1.0708	50	6932.60

Figure 7: Local epochs $= 4$, Calibration Rounds $= 2$

Name	Sparsity	Global Epochs	Test Acc.$^{(*)}$ Head $\rightarrow$ Back	Test Loss$^{(*)}$ Head $\rightarrow$ Back	Global$^{(*)}$ Ep.Best	Time (in s) (Back only)
FLSpBack_iid	0.2	50	0.6888 $\rightarrow$ 0.7652	1.1254 $\rightarrow$ 0.8551	46	2637.40
FLSpBack_iid	0.5	50	0.6888 $\rightarrow$ 0.7476	1.1254 $\rightarrow$ 0.9006	46	2483.00
FLSpBack_iid	0.8	50	0.6888 $\rightarrow$ 0.7104	1.1254 $\rightarrow$ 1.0271	50	2243.21
FLSpBack_niid_Nc5	0.2	50	0.5768 $\rightarrow$ 0.7321	3.0737 $\rightarrow$ 1.2722	50	2555.71
FLSpBack_niid_Nc5	0.5	50	0.5768 $\rightarrow$ 0.6998	3.0737 $\rightarrow$ 1.3067	44	2420.16
FLSpBack_niid_Nc5	0.8	50	0.5768 $\rightarrow$ 0.6816	3.0737 $\rightarrow$ 1.7675	46	2266.05

Figure 8: Sparsity $= 0.5$, Local epochs $= 4$

Name	Global Epochs	Calib. Rounds	Test Acc.$^{(*)}$ Head $\rightarrow$ Back	Test Loss$^{(*)}$ Head $\rightarrow$ Back	Global$^{(*)}$ Ep.Best	Time (in s) (Back only)
FLSpBack_iid	100	1	0.6888 $\rightarrow$ 0.7596	1.1254 $\rightarrow$ 0.8636	100	3671.02
FLSpBack_iid	100	2	0.6888 $\rightarrow$ 0.7577	1.1254 $\rightarrow$ 0.8734	88	4167.24
FLSpBack_iid	100	5	0.6888 $\rightarrow$ 0.7549	1.1254 $\rightarrow$ 0.8789	100	5460.28
FLSpBack_niid_Nc5	100	1	0.5768 $\rightarrow$ 0.7202	3.0737 $\rightarrow$ 1.1873	95	3747.47
FLSpBack_niid_Nc5	100	2	0.5768 $\rightarrow$ 0.7174	3.0737 $\rightarrow$ 1.1956	94	4138.54
FLSpBack_niid_Nc5	100	5	0.5768 $\rightarrow$ 0.7141	3.0737 $\rightarrow$ 1.1869	95	5700.35

Figure 9a and 9b: Sparsity $= 0.5$ for *Scen2*, Local epochs $= 4$

Name	Global Epochs	Calib. Rounds	Test Acc.$^{(*)}$	Test Loss$^{(*)}$	Global$^{(*)}$ Ep.Best	Time (in s) (Total)
Scen1_niid_Nc10	150	-	0.7211	1.0916	150	6154.90
Scen2_niid_Nc10	50 + 100	2	0.5082 $\rightarrow$ 0.7234	1.8991 $\rightarrow$ 0.9889	145	5205.06

Table 2. Hyperparameters used for the training and performances of the federated learning models. A weight decay of 5×10^{-5} (L2 regularization) was applied in all runs, a learning rate of 1×10^{-4}, momentum 0.9, client fraction $C = 0.1$, and number of total clients $K = 100$. Selection method for sparse fine tuning = *least*.
For all scenarios involving sparse fine-tuning, the head was first trained separately keeping the backbone frozen, with learning rate of 0.003, momentum 0.9, a weight decay of 1×10^{-4} and number of global epochs 100.
(*)The reported results are the best obtained, which do not always correspond to the values at the final epoch due to minor fluctuations during training or the onset of overfitting. The reported loss is the value corresponding to the epoch where the highest accuracy was achieved and it may not be the lowest loss observed.

the entire set of weights. A variant of the original training function was developed, where updates are sent in a sparse format, containing only modified parameter values and their indices. This approach reduces communication only when most parameters remain unchanged (e.g., a sparsity ratio of 80%), since sending both the values and their corresponding indices - where each index must specify all dimensions for multidimensional tensors - can outweigh the benefits if a large portion of the model is updated (Figure 10).

Another problem that often arises in Federated Learning is **client drift**, especially when data is non-i.i.d. distributed across clients. In order to asses it, it is possible to measure how much local models diverge from the global model after each local training. The mean client drift at global round t

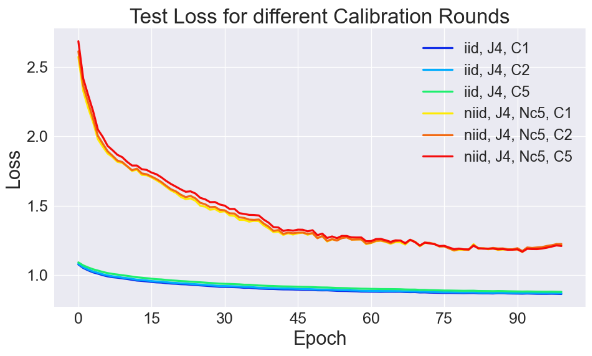


Figure 8. Test loss for different numbers of calibration rounds, for both i.i.d. and non-i.i.d. ($N_c = 5$) federated learning scenarios.

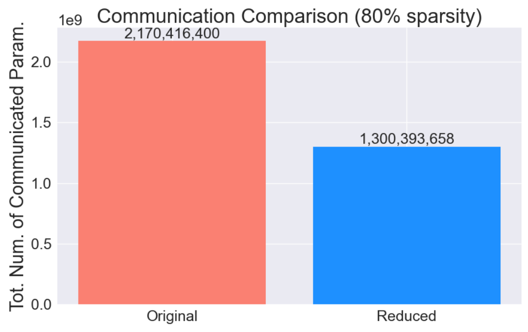
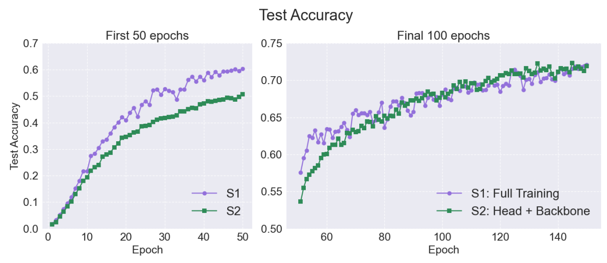
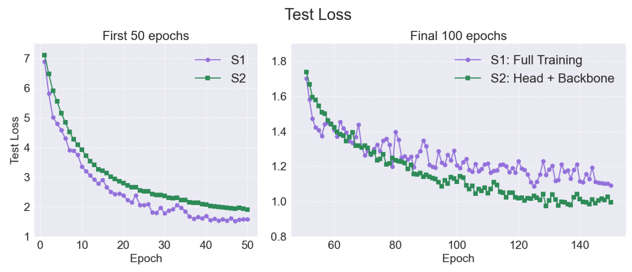


Figure 10. Number of parameters communicated (clients→server) for a FL implementation with original and reduced communication over 10 global rounds (100 clients, 10 selected per round).



(a)



(b)

Figure 9. Test performances for two learning scenarios for a non-i.i.d. setting ($N_c = 10$).

can be defined as:

$$\text{Mean Drift}^{(t)} = \frac{1}{K}\sum_{k=1}^{K}\left\|\mathbf{w}_{\mathbf{k}}^{(\mathbf{t})} - \mathbf{w}^{(\mathbf{t})}\right\|, \tag{4}$$

where $\mathbf{w}_{\mathbf{k}}^{(\mathbf{t})}$ and $\mathbf{w}^{(\mathbf{t})}$ are the tensors of model parameters for client k and the global model, respectively, and $\|\cdot\|$ denotes the ℓ_2 norm. As expected, i.i.d. setting with fewer local steps shows minimal drift, while non-i.i.d. clients and larger number of local steps (J) lead to substantially higher drift (Figure 11).

Finally, an additional method for mask creation was also implemented, although it was then not used in the experiments. One of the main assumptions of task arithmetic-based methods is that there is independence between the parameters of different tasks. When each client has its own

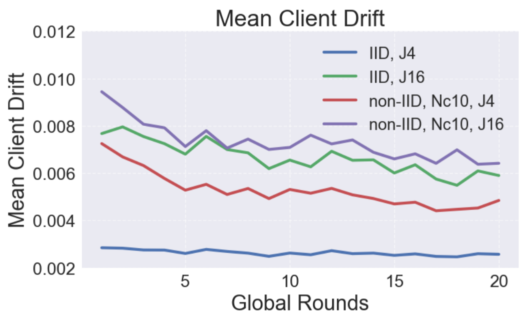


Figure 11. Mean client drift per round for different federated learning scenarios.

mask, this assumption cannot be fully respected: a client may freeze some parameters, but if they are not frozen also in other clients, the global model will update these parameters during the FedAvg step. To evaluate the extent of this issue, we tested also an alternative masking technique in which, upon receiving parameters from the global model, a client computes the following convex linear combination with its previous local parameters:

$$\theta_{\text{new}} = \beta\,\theta_{\text{old}} + (1-\beta)\,\theta_{\text{global}} \tag{5}$$

where $\beta = \frac{F_i}{\sum_j F_j}$, with F_i denoting the Fisher score of the client for that parameter, and $\sum_j F_j$ the sum of Fisher scores across all clients for the same parameter. If no client modifies the frozen parameter, it remains unchanged. If it is modified but its Fisher score is low, the influence of the previous local parameters is minimal; conversely, if the parameter is important, the opposite holds. This method preserves privacy since each client only receives the aggregate Fisher scores for the parameter, but it considerably increases the memory requirements. Due to limited RAM and GPU memory, we evaluated this approach using float16 precision and only 50 clients. The results showed greater training stability, but slower accuracy growth.

5. Conclusion

In this work, we investigated both centralized and federated learning approaches for image classification on the CIFAR-100 dataset using the pre-trained DINO ViT-S/16 model. A centralized baseline was established, and then federated learning experiments were conducted with both i.i.d. and non-i.i.d. data distributions across 100 clients. Sparse fine-tuning guided by parameter sensitivity was applied, using a two-step strategy of head-only training followed by sparse backbone updates.

Our experiments showed that sparse fine-tuning enables efficient model adaptation while maintaining high performance. In particular, training the classifier head first, followed by sparse fine-tuning of the backbone, achieved the most stable convergence.

The centralized baseline with sparse fine-tuning reached a test accuracy of 0.7976. In the federated learning experiments, the i.i.d. setting reached a test accuracy of 0.7512 for the same sparsity ratio of 0.5. For non-i.i.d data distributions, performance was lower overall, especially when clients had very few classes ($N_c = 1 - 5$). However, increasing the number of classes per client ($N_c = 10 - 50$) significantly improved results, approaching i.i.d. performance. Increasing the number of local epochs improved accuracy up to a point, but excessive local training (e.g., 16 epochs) slightly degraded performance in the non-i.i.d. due to client drift.

Further analysis highlighted the trade-offs between sparsity, communication efficiency, and performance. Higher sparsity levels reduced accuracy but allowed faster calibration and lower communication costs when updates were transmitted in a compressed sparse format. Moreover, our drift analysis confirmed that client heterogeneity and longer local training amplify divergence from the global model, reinforcing the importance of carefully balancing local computation with global synchronization.

Overall, these results indicate that sparse fine-tuning combined with a head-first training strategy is robust in both centralized and federated scenarios. Future work could explore alternative parameter selection strategies (e.g., magnitude-based), different mask calibration techniques, or the application of the proposed methods to larger-scale datasets such as ImageNet to further assess scalability.

References

[1] Peter Kairouz, H. Brendan McMahan, Brendan Avent, et al. Advances and open problems in federated learning. Foundations and Trends in Machine Learning, 4, 2021. 1

[2] Leonardo Iurada, Marco Ciccone, and Tatiana Tommasi. Efficient model editing with task-localized sparse fine-tuning. In Proceedings of the International Conference on Learning Representations (ICLR), 2025. 1, 2, 3

[3] Sashank J. Reddi, Zachary Charles, Zachary Garrett Manzil Zaheer, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. In International Conference on Learning Representations (ICLR), 2021. 1

[4] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), volume 54, pages 1273–1282. PMLR, 2017. 1

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

[6] Kilian Pfeiffer, Martin Rapp, Ramin Khalili, and Jörg Henkel. Federated learning for computationally-constrained heterogeneous devices: A survey. ACM Computing Surveys, 55(14s):334:1–334:27, 2023. doi: 10.1145/3596907. 1

[7] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification, 2019. 1

[8] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In International Conference on Machine Learning (ICML), 2020. 1

[9] Huiqiang Chen, Tianqing Zhu, Tao Zhang, Wanlei Zhou, and Philip S. Yu. Privacy and fairness in federated learning: on the perspective of trade-off. ACM Computing Surveys, 1(1): 1–38, 2023. doi: 10.1145/nnnnnnn.nnnnnnn. 1

[10] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. IEEE Transactions on Knowledge and Data Engineering, 2021. 1

[11] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution, 2020. 1

[12] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. FedBN: Federated learning on non-iid features via local batch normalization. In International Conference on Learning Representations (ICLR), 2021. 1

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