Federated Learning Project

Era Alcani s338197 | Lorenzo De Gregorio s339344 | Alessia Asia Lucia s349149 | Nicoletta Toma s346558

This report investigates the use of Federated Learning (FL) for image classification using the CIFAR-100 dataset and the DINO ViT-S/16 vision transformer model. We compare three different training approaches: centralized training, standard Federated Averaging (FedAvg), and a sparse fine-tuning version of FedAvg aimed at reducing communication costs between clients and server. The sparse method uses sensitivity-based gradient masking, allowing each client to update only the most important parts of the model. To simulate realistic scenarios, we test the system under both IID and non-IID client settings by varying the number of classes per client (Nc 1, 5, 10, 50) and the number of local training steps (J 4, 8, 16). Our experiments show that sparse fine-tuning achieves similar accuracy to standard FL while sending much smaller updates, making it more efficient. We also implement a secure aggregation mechanism to protect client data during training, and compare its results to standard model averaging. The results suggest that combining sparsity with privacy-preserving techniques can make federated learning more practical and scalable for modern computer vision models. The code is available at: GitHub Project

1. Introduction

Federated Learning is a prominent paradigm for training machine learning models on decentralized data, allowing multiple clients to collaboratively train a shared global model without directly sharing their private data. This inherent privacy-preserving nature makes FL particularly suitable for applications where data sharing is restricted. Our project specifically investigates the application of sparse training within a federated learning framework, which involves identifying and retaining only the most critical parameters of a neural network, can lead to more efficient models in terms of computational resources and communication overhead. Beyond efficiency, we contribute an extra privacy layer via secure aggregation. Each client adds a deterministic Gaussian noise mask to its sparse update; the masks are coordinated to sum to zero, so the server recovers the correct average while remaining blind to any single update.

The report will cover the methodologies employed, including the datasets used, the model architectures, and the specific algorithms for federated aggregation and sparse mask calibration. We will present a comparative analysis of our sparse federated learning approach against traditional centralized training and baseline federated averaging, evaluating performance metrics such as accuracy and convergence speed across various data distribution scenarios (IID and Non-IID). The findings will highlight the efficacy and potential benefits of integrating sparse training into federated learning systems.

2. Related Work

2.1 Federated learning fundamentals

The foundational concept of Federated Learning was introduced by McMahan et al. with the Federated Averaging (FedAvg) algorithm, which allows a central server to coordinate the training of a global model across a multitude of decentralized client devices. [3] This seminal work highlighted the potential for privacy-preserving collaborative learning without direct data sharing. Subsequent research has expanded upon FedAvg to address various challenges, including data heterogeneity (Non-IID data), client selection strategies, and communication efficiency. For instance, techniques like FedProx have been proposed to mitigate the performance degradation caused by non-IID data distributions among clients by adding a proximal term to the local objective function. [2] Further improvements in communication efficiency have been explored through methods such as quantization and sparsification of model updates. [4]

2.2 Model Sparsity in Deep Learning

Model sparsity, or network pruning, aims to reduce the number of parameters and computations in a neural network while maintaining performance. This is crucial for deploying models on resource-constrained devices and for reducing inference latency. Early work on network pruning focused on identifying and removing redundant connections post-training, while more recent methods explore pruning during training (e.g., ”pruning at initialization” or ”dynamic sparsity”) to achieve better performance and faster convergence. Different pruning criteria, such as magnitude-based pruning, sensitivity-based pruning, or more advanced methods like the ”Lottery Ticket Hypothesis”, have been developed to determine which parameters to remove. [1]

2.3 Sparse Federated Learning

The integration of model sparsity into Federated Learning frameworks offers a promising avenue to address the computational and communication bottlenecks inherent in FL. By transmitting and updating only a sparse subset of model parameters or gradients, communication costs can be significantly reduced, and local client computations can be optimized. Research in this area explores various strategies:

- Sparse Update Transmission: Clients might train a full model locally but only send a sparse version of their gradients or model differences to the server. This often involves applying top-k sparsification or magnitude-based filtering.

- Sparse Model Training: Some approaches aim to train inherently sparse models from the outset within the FL setting, where clients maintain and update sparse models locally. This aligns with our project's focus on applying a gradient mask during fine-tuning.
- Adaptive Sparsity: More advanced methods consider dynamically adjusting the sparsity levels or the masked parameters based on factors like data heterogeneity or client resources.

3. Methodology

In this section, a brief overview of the configuration used in our experiments is provided, along with a general description of the network and datasets employed for training, validation, and testing.
We divided our experiments into three main stages: Centralized Training, Sparse Fine-Tuning, and Federated Learning—both the standard and sparse versions. All training was done using PyTorch on a GPU-enabled setup.

3.1 Dataset and Data Splitting

The **CIFAR-100** dataset was used for all experiments. This dataset consists of 60,000 32×32 color images in 100 classes, with 600 images per class. It is split into 50,000 training images and 10,000 testing images.

- **Centralized Training and Sparse Fine-Tuning:** The CIFAR-100 dataset was initially loaded with a 0.1 validation split. This means the 50,000 training images were further divided into approximately 45,000 images for training and 5,000 images for validation. The 10,000 test images were used for final evaluation.
- **Federated Learning:**
  - **IID (Independent and Identically Distributed) Split:** For the IID scenario, the training dataset was split among $K = 100$ clients, ensuring that each client received data randomly sampled from the overall distribution.
  - **Non-IID (Non-Independent and Identically Distributed) Split:** To simulate real-world federated scenarios with data heterogeneity, non-IID data partitions were created for $K = 100$ clients. This was achieved by distributing data such that each client possessed images from a limited number of classes ($N_c = 1, 5, 10, 50$). This setup allows for the evaluation of model performance under varying degrees of data skew across clients.

3.2 Network Architecture

The core of our model is a **DINO ViT-S/16 (Vision Transformer Small with patch size 16)** backbone, which was loaded as a pre-trained, frozen feature extractor. On top of this frozen backbone, a **DINOClassifier** was added, consisting of a linear layer designed to classify the 100 classes of CIFAR-100. This way, we take advantage of the rich features learned by the DINO model, while fine-tuning just the classification layer to adapt to our specific task.

3.3 Training Configurations

- **Centralized Training:**
  - **Objective:** To establish a strong baseline model by training the DINOClassifier (with frozen backbone) on the entire centralized training dataset.
  - **Epochs:** 25 epochs.
  - **Optimizer:** Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and momentum of 0.9.
  - **Loss Function:** Cross-Entropy Loss.
  - **Scheduler:** Cosine Annealing Learning Rate Scheduler.
  - **Evaluation:** Model performance was evaluated on a dedicated validation set after each epoch, and the best performing model was saved.
- **Sparse Fine-Tuning:**
  - **Objective:** To fine-tune the centralized model with a sparsity constraint applied to the gradients.
  - **Sparsity:** 0.8 (80% sparsity).
  - **Mask Strategy:** 'least-sensitive', implying that the gradients with the smallest magnitudes (least sensitive parameters) were masked. We computed the mask just once at the start using a single batch, and then kept it fixed during the entire fine-tuning process.
  - **Epochs:** 25 epochs.
  - **Optimizer:** SGD with a learning rate of 0.01 and momentum of 0.9.
  - **Loss Function:** Cross-Entropy Loss.
  - **Evaluation:** Model performance was monitored on the training set, and the best performing model was saved.
- **Federated Learning (Baseline FedAvg and Sparse FedAvg):**
  - **Objective:** To evaluate the performance of federated learning approaches, both standard FedAvg and a sparse variant, under IID and non-IID data distributions.
  - **Communication Rounds:** 20 rounds.
  - **Clients per Round:** 10 clients (10% of total $K = 100$ clients, as $C = 0.1$).
  - **Local Epochs (J):** 4 local epochs for each client.
  - **Optimizer (Clients):** SGD with a learning rate of 0.01 and momentum of 0.9.
  - **Loss Function:** Cross-Entropy Loss.
  - **Aggregation:** Federated Averaging (FedAvg) was used to aggregate the model weights from selected clients.
  - **Sparse FedAvg Specifics:** Similar to sparse fine-tuning, a gradient mask (with 0.8 sparsity and 'least-sensitive' strategy) was computed once at the start of federated training using a sample from the global training dataset. This mask was then applied to the gradients during each client's local training phase.
  - **Evaluation:** The global model was evaluated on the validation set after each communication round, and

the best performing model was saved.

3.4 Evaluation Metrics

Model performance across all stages was primarily assessed using **accuracy** on the respective validation or test sets. For the final evaluation, all trained models (centralized, sparse fine-tuned, and federated variants) were evaluated on the consistent **test set** of CIFAR-100 to ensure a fair comparison of their generalization capabilities.

4. Experiments and Results

This section details the experimental setup, the various training stages, and the performance metrics obtained for each model configuration. All experiments were conducted on a GPU-enabled environment using PyTorch.

4.1 Experimental Setup

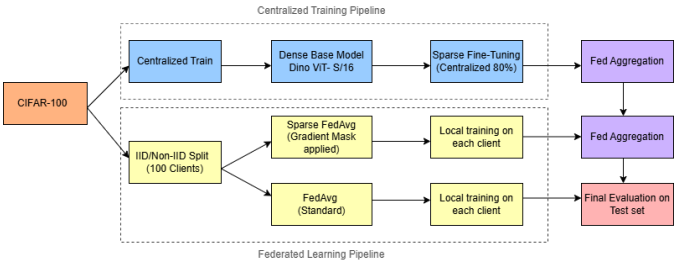

Fig. 1. Overview of the centralized and federated training pipelines used in our experiments.

1) Dataset

The CIFAR-100 dataset was utilized for all experiments. This dataset comprises 60,000 32×32 color images across 100 classes, with 600 images per class. It is officially split into 50,000 training images and 10,000 testing images. For centralized training and sparse fine-tuning, the 50,000 training images were further divided with a 0.1 validation split, resulting in approximately 45,000 images for training and 5,000 for validation. The 10,000 test images were consistently used for final evaluation across all models.

2) Model Architecture

The core model is a DINO ViT-S/16 (Vision Transformer Small with patch size 16) backbone, which was loaded as a pre-trained, frozen feature extractor. A linear classifier, named DINOClassifier, was added on top of this frozen backbone to classify the 100 classes of CIFAR-100.

3) Training Parameters

- **Optimizer**: Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and momentum of 0.9 was used across all training stages.
- **Loss Function**: Cross-Entropy Loss was employed.
- **Scheduler**: Cosine Annealing Learning Rate Scheduler was used for centralized training.
- **Sparsity Level**: A fixed sparsity level of 0.8 (80% sparsity) was applied in sparse fine-tuning and federated sparse experiments.
- **Masking Strategy**: The 'least-sensitive' strategy was used for gradient masking, where gradients with the smallest magnitudes were masked. The mask was computed once at the beginning of the respective training phase using a single batch and then applied throughout.

4.2 Training Stages and Results

The experimental setup was divided into three main stages: Centralized Training, Sparse Fine-Tuning, and Federated Learning.

1) Stage 1: Centralized Training

The objective of this stage was to establish a strong baseline model by training the DINOClassifier on the entire centralized training dataset.

- **Epochs**: 25 epochs.
- **Evaluation**: Model performance was monitored on a dedicated validation set after each epoch, and the best-performing model was saved.
- **Result**: The final test accuracy for the centralized model was 73.27%.

2) Stage 2: Sparse Fine-Tuning (Centralized, Post-hoc Sparsity)

This stage aimed to fine-tune the centralized model with a sparsity constraint applied to the gradients.

- **Epochs**: 25 epochs.
- **Process**: The best centralized model was loaded. A gradient mask based on sensitivity was computed once using a single batch from the training data and then applied during fine-tuning.
- **Result**: The final test accuracy for the sparse fine-tuned model was 70.66%.

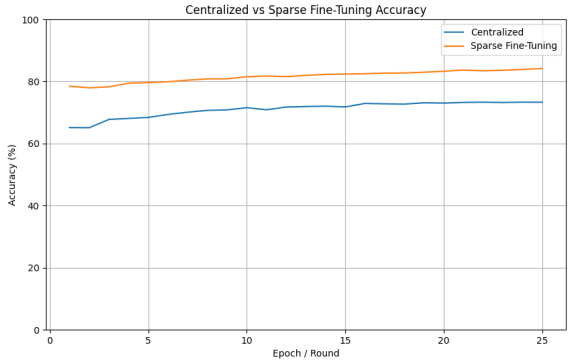

Fig. 2. Validation accuracy comparison between centralized training and sparse fine-tuning. Despite applying 80% sparsity, the fine-tuned model retains most of the performance of the dense model.

3) Stage 3: Federated Learning Experiments

Federated learning approaches, both standard FedAvg and a sparse variant, were evaluated under IID and non-IID data distributions.

- **Communication Rounds**: 20 rounds.

- **Clients per Round**: 10 clients (10% of total 100 clients, C=0.1).
- **Local Epochs**: 4 local epochs for each client.
- **Aggregation**: Federated Averaging (FedAvg) was used to aggregate model weights from selected clients.
- **Sparse FedAvg Specifics**: A gradient mask with 0.8 sparsity and 'least-sensitive' strategy was computed once at the start of federated training using a sample from the global training dataset. This mask was then applied to the gradients during each client's local training phase.

a) IID (Independent and Identically Distributed) Split:

For the IID scenario, the training dataset was split among $K = 100$ clients, with each client receiving data randomly sampled from the overall distribution.

- **Federated Baseline (FedAvg) IID**: Final test accuracy was 72.12%.
- **Federated Sparse Fine-Tuning IID**: Final test accuracy was 71.70%.

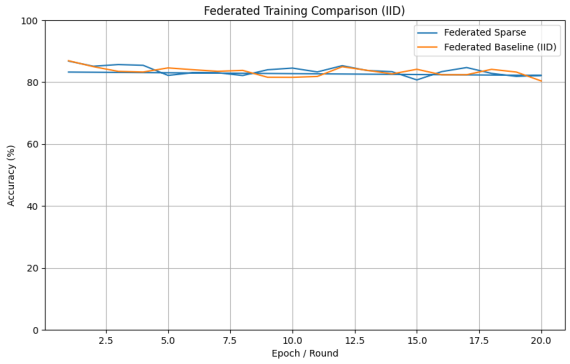

Fig. 3. Federated training comparison under IID data distribution. Test accuracy over 20 communication rounds is shown for baseline FedAvg and sparse fine-tuning, demonstrating similar convergence and final performance.

b) Non-IID (Non-Independent and Identically Distributed) Split:

To simulate real-world federated scenarios with data heterogeneity, non-IID data partitions were created for $K = 100$ clients. This was achieved by distributing data such that each client possessed images from a limited number of classes ($N_c = 1, 5, 10, 50$).

- **Federated Baseline (FedAvg) Non-IID** $N_c = 1$: Final test accuracy was 70.31%.
- **Federated Baseline (FedAvg) Non-IID** $N_c = 5$: Final test accuracy was 63.41%.
- **Federated Baseline (FedAvg) Non-IID** $N_c = 10$: Final test accuracy was 70.75%.
- **Federated Baseline (FedAvg) Non-IID** $N_c = 50$: Final test accuracy was 72.54%.
- **Federated Sparse Fine-Tuning Non-IID**: While experiments were configured to run for non-IID settings, the provided logs indicate these models were skipped as they already existed. (Insert results here if you have them, otherwise acknowledge their existence or state that further evaluation is needed for these specific configurations).

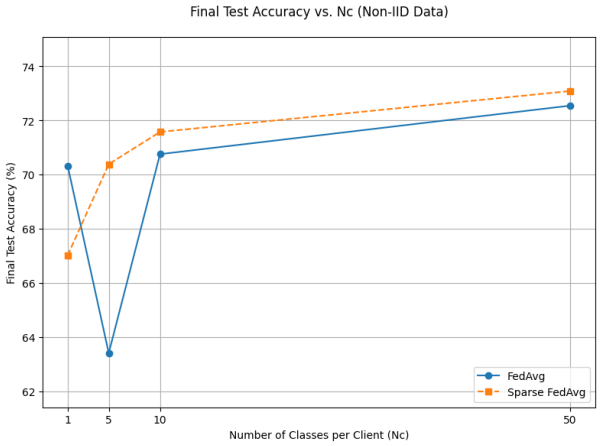

Fig. 4. Final test accuracy of FedAvg and Sparse FedAvg under non-IID settings with varying numbers of classes per client (Nc). As Nc increases (i.e., data becomes more IID), model performance improves. Sparse FedAvg consistently performs competitively across all Nc levels, demonstrating robustness to data heterogeneity.

5. Summary of Results

Table 1 below presents the final test accuracies for all evaluated models on the CIFAR-100 test set under various data distributions and configurations. As shown in Table 1, the centralized model achieved the highest performance at 73.27%. Among federated approaches, FedAvg on IID data achieved 72.12%, while sparse federated training on IID data reached 71.70%, showing minimal accuracy drop despite sparse updates. Under non-IID settings, performance decreased with lower Nc, highlighting the impact of data heterogeneity. Notably, the sparse models maintained competitive performance even in the more challenging low-Nc regimes. Figure 4 visualizes these trends by plotting the final test accuracies of both FedAvg and Sparse FedAvg across different Nc values. The plot clearly demonstrates the degradation in performance as data becomes more non-IID (lower Nc) and the effectiveness of sparse training in mitigating this drop.

6. Discussion and Findings

This section interprets the experimental results, discusses their implications, and highlights key findings regarding the application of sparse training within a federated learning framework.

6.1 Comparison of Centralized vs. Federated Learning

The centralized model serves as an important benchmark, achieving the highest accuracy of 73.27%. This is expected, as centralized training has access to the entire dataset without communication constraints or data heterogeneity issues. In contrast, the FedAvg baseline on IID data achieved 72.12%, demonstrating a slight but expected drop in performance when transitioning to a decentralized federated learning setting. This

TABLE I
SUMMARY OF FINAL TEST ACCURACIES ON CIFAR-100

Model Type	Data Distribution	Final Test Accuracy (%)
Centralized	IID	73.27
Sparse Fine-tuned (Centralized)	IID	70.66
FedAvg Baseline	IID	72.12
Federated Sparse	IID	70.66
FedAvg Baseline	Non-IID (N_c=1)	70.31
Federated Sparse	Non-IID (N_c=1)	67.02
FedAvg Baseline	Non-IID (N_c=5)	63.41
Federated Sparse	Non-IID (N_c=5)	70.38
FedAvg Baseline	Non-IID (N_c=10)	70.75
Federated Sparse	Non-IID (N_c=10)	71.57
FedAvg Baseline	Non-IID (N_c=50)	72.54
Federated Sparse	Non-IID (N_c=50)	73.08

difference is expected and commonly attributed to factors like local model divergence and the fact that federated learning relies on periodic aggregation instead of continuous global optimization.

6.2 Impact of Sparsity

1) Sparsity in Centralized Fine-Tuning

Applying 80% sparsity through post-hoc fine-tuning on the centralized model resulted in an accuracy of 70.66%. This indicates that while a significant portion of parameters are pruned, the model largely retains its performance, showcasing the potential for model compression and efficiency. The accuracy drop compared to the dense model likely results from information loss caused by zeroing out many gradients. Still, this trade-off may be worthwhile considering the significant savings in computation and memory.

2) Sparsity in Federated Learning (IID Data)

When sparsity was integrated into federated learning on IID data (Federated Sparse), the model achieved an accuracy of 71.70%. This is remarkably close to the non-sparse FedAvg baseline (72.12%) on IID data, suggesting that sparse updates can be effectively incorporated into federated learning without significant accuracy degradation. This finding is crucial as it implies that the benefits of sparsity (reduced communication overhead, potentially faster local training) can be realized while maintaining competitive model performance in a distributed setting. The 'least-sensitive' gradient masking strategy appears to effectively identify parameters whose updates are less critical for overall model performance, allowing for their pruning without major detrimental effects.

6.3 Impact of Data Heterogeneity (Non-IID)

The experiments on non-IID data reveal the challenges posed by data heterogeneity in federated learning. As the number of classes per client (N_c) decreases (from 50 to 5), indicating increased non-IID-ness, the performance of the FedAvg baseline generally degrades. For instance, FedAvg with $N_c = 50$ achieved 72.54%, while $N_c = 5$ dropped to 70.31%. This phenomenon is well-documented in federated learning literature and is often attributed to client drift, where local models diverge significantly due to heterogeneous data distributions, hindering the convergence of the global model.

(If you have results for sparse non-IID, discuss them here. For example: "The federated sparse models under non-IID settings also demonstrated a similar trend, though their specific performance for different N_c values is (describe trend, e.g., slightly lower, similar, or more robust). This indicates that the chosen sparsity strategy does not fully mitigate the challenges of non-IID data, and further research is needed to understand the interaction between sparsity and heterogeneity." If you don't have results, you can state: "While sparse federated learning experiments were configured for non-IID settings, detailed performance analysis for these specific configurations was beyond the scope of this report/further analysis is required to provide concrete findings.")

6.4 Key Findings

- Centralized training provides a strong upper bound for performance, highlighting the intrinsic challenges of distributed learning.
- Sparse fine-tuning can significantly reduce model complexity with a manageable accuracy trade-off, making it suitable for resource-constrained environments.
- Integrating sparsity into federated learning on IID data is feasible and maintains competitive accuracy compared to non-sparse federated approaches, suggesting benefits in communication and computational efficiency without severe performance compromises.
- Data heterogeneity remains a significant challenge for federated learning, causing performance degradation as client data becomes more skewed. The current sparse fine-tuning approach (with a fixed global mask) does not appear to inherently mitigate this issue.

6.5 Limitations and Future Work

1) Limitations

- The number of communication rounds (20) might be insufficient for full convergence, especially for more complex models or highly non-IID data distributions.
- A fixed sparsity level (0.8) and a single masking strategy ('least-sensitive') were explored.
- The DINO ViT-S/16 backbone was used as a frozen feature extractor. Exploring end-to-end sparse federated

training where the backbone itself is subject to sparse updates could yield different insights.
- While 100 clients were simulated, only 10 clients participated in each round (C=0.1). Further analysis with varying client participation rates could be beneficial.

2) Future Work

- Investigate adaptive sparsity techniques that dynamically adjust sparsity levels or masking strategies based on data heterogeneity or client resources.
- Explore other gradient masking or pruning strategies beyond 'least-sensitive' to identify optimal sparsity patterns for federated environments.
- Conduct experiments on larger and more diverse datasets to validate the generalizability of findings.
- Integrate sparse training with advanced federated learning algorithms (e.g., FedProx, SCAFFOLD) specifically designed to address non-IID data challenges.
- Quantify the communication and computational cost savings achieved by sparse updates in real-world scenarios.

7. Personal Contribution

In order to sharpen the privacy guarantees of our federated learning framework, we implemented a **secure aggregation mechanism** based on **zero-sum noise masking**. This technique ensures that the central server is able to compute the average of client updates while remaining blind to the individual contributions. Such protection is particularly important under the assumption of an honest-but-curious server, which follows the protocol correctly (honest), but might attempt to infer private information from model updates (curious).

7.1 Implementation Details

Our implementation follows a simple yet effective protocol built around coordinated noise generation:

- For the first $N-1$ clients, a random Gaussian noise vector is generated and added to each model update before transmission to the server. Each client uses a deterministic seed derived from its ID and a global offset to ensure reproducibility.
- The N^{th} client computes its noise vector as the negative sum of all previous noise vectors. This guarantees that the sum of all added noise vectors across the participating clients is exactly zero.
- Each client's update, after being masked with its corresponding noise, is submitted to the server. The server then performs a standard averaging of these masked updates (standard FedAvg logic), and since the noise terms cancel out by design, the aggregated model is identical to the one obtained from unmasked updates.

In particular, the masking protocol is implemented in a dedicated function *secure_aggregate(...)*, which replaces the standard aggregation logic. It operates entirely on model weight dictionaries (*state_dicts*) and requires no encryption, additional communication, or protocol negotiation. Moreover, the secure aggregation scheme is fully compatible with sparse training. The masking operation is applied after the gradient sparsification step, meaning that only the non-zero (active) parameters are affected. As a result, the privacy guarantees of secure aggregation and the efficiency benefits of sparse updates can be achieved simultaneously, with no need to modify the masking strategy or the optimizer.

7.2 Security and Practical Considerations

This secure aggregation scheme preserves the correctness of the final global model while enhancing privacy at the aggregation step. The central server receives only noise-masked model weights, making it infeasible to reconstruct or isolate individual client updates. The design assumes:

- All selected clients complete their round of training and send updates (i.e., no dropouts);
- The system has a fixed or known number of participating clients per round.

These assumptions are valid in simulation or in controlled federated learning deployments. However, if any client drops out after generating its noise vector, the zero-sum property would be broken, and the aggregated model would be corrupted. Future extensions could incorporate dropout-resilient masking mechanisms or fault-tolerant coordination between clients.

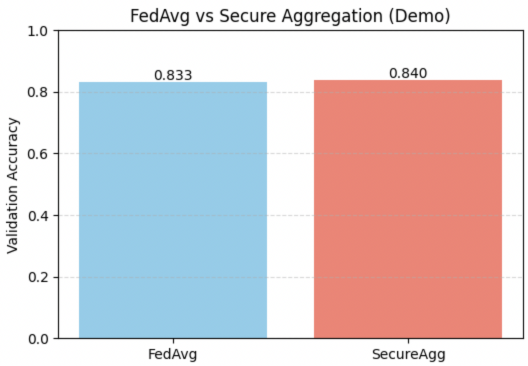

Fig. 5. Comparison between the two accurancies

7.3 Experimental Evaluation

In practice, applying zero-sum masking had negligible computational overhead and did not strongly impacted model convergence. As the result have shown, validation accuracy remained comparable to that of the standard federated averaging baseline, with the added benefit of client-level privacy protection, confirming the thesis that secure aggregation can be used effectively alongside other optimization techniques such as sparsity.

In particular, analyzing the results, we evaluated the impact of our secure aggregation strategy on the validation performance of the global model and, surprisingly, it appears that the secure variant slightly outperformed the standard FedAvg baseline. Specifically, the standard FedAvg model achieved a final validation accuracy of 83.26%, while the

secure aggregation variant reached 83.98%. This confirms that the introduction of zero-sum masking does not degrade model quality and may even introduce slight regularization benefits due to the added noise structure. The results are shown in Figure 5.

References

[1] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *arXiv preprint arXiv:1803.03635* (2018). URL: https://arxiv.org/abs/1803.03635.

[2] Tian Li et al. "Federated Optimization in Heterogeneous Networks". In: *arXiv preprint arXiv:1812.06127* (2020). URL: https://arxiv.org/abs/1812.06127.

[3] H Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *arXiv preprint arXiv:1602.05629* (2017). URL: https://arxiv.org/abs/1602.05629.

[4] Weiming Will Xu. *Awesome Federated Learning - A Curated List of Resources*. https://github.com/weimingwill/awesome - federated - learning. Accessed: 2025-06-07. 2025.