ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE (EPFL)

Master Semester Research Project

Effects of Pre-Training on One-Shot Federated Learning

Author:
Alexis LIMOZIN
Email: alexis.limozin@epfl.ch

Supervisors:
Akash DHASADE, Mathis RANDL

Professor:
Anne-Marie KERMARREC

Laboratory:
Scalable Computing Systems (SaCS) laboratory

January 2025

***Abstract*—This report presents research findings highlighting the particularities and effects of utilizing pre-trained models in conjunction with one-shot federated learning (OFL) for fine-tuning tasks. The primary objective was to analyze and quantify the differences that pre-training introduces to the Federated Learning process when constrained to a single round of communication and to identify potential advantages that could bridge the performance gap with traditional Federated Learning (FL). Experiments with the ResNet-18 model were conducted, either randomly initialized or pre-trained on the ImageNet dataset, and then fine-tuned for a visual recognition task on the CIFAR-10 dataset. The impact of data heterogeneity is demonstrated, highlighting key differences between OFL and FL. Unlike FL, pre-training does not bridge the gap between non-IID and IID data. The effect of model size is explored, showcasing the potential of pre-training to enable the training of more complex models. Insights into the effects of various weight regularization methods and the determination of optimal model checkpoints are presented. These findings enhance our understanding of the behavior of local models during fine-tuning within a one-shot federated learning framework.**

***Index Terms*—One-Shot Federated Learning; Pre-Training; non-IID data; ResNet; Ensemble Learning; Deep Learning**

I. Introduction

Federated Learning (FL) is a framework for fine-tuning models across multiple clients, where each client utilizes its private data for training without sharing it directly. In the standard FL process, clients perform several local updates on their models, which are then shared with a central server for aggregation and redistribution. This cycle, constituting a single communication round, is repeated iteratively until the model converges. This framework for model fine-tuning is particularly relevant in sensitive domains like healthcare and finance, where privacy is crucial.

On the other hand, One-shot Federated Learning (OFL) [1] seeks to minimize communication costs by reducing the process to a single communication round. OFL also mitigates privacy risks associated with the sharing of model parameters or gradients, a known vulnerability of FL where private data can be reconstructed from publicly shared gradients [2]. In OFL, clients train their models to convergence locally and transmit the final models to the server. However, this approach often results in lower accuracy compared to traditional FL, particularly when client data are highly non-IID. The lack of iterative communication exacerbates client drift, as each model becomes more specialized to its local data.

The use of pre-trained models for initialization has been explored in the context of FL and has demonstrated significant benefits. It has been shown to mitigate the effects of non-IID data on the final model, allowing for an increase in the number of local epochs per round without negatively impacting the final accuracy [3]. Building on these insights, we hypothesized that pre-training could yield similar benefits in OFL, potentially offsetting its limitations, such as exacerbated client drift.

Our research investigates whether pre-trained models can bridge the performance gap in OFL, as seen in FL. Specifically, we examine their effectiveness in handling non-IID data, their role in enabling more complex model architectures, and their interaction with regularization techniques for mitigating drift.

Contributions Our work presents the first comprehensive investigation into the effects of pre-training on One-shot Federated Learning (OFL). Contrary to traditional Federated Learning (FL), where pre-training reduces the performance gap between IID and non-IID data distributions, we demonstrate that this benefit does not extend to OFL. Specifically, while pre-training in FL yields a substantial 4.79 percentage point reduction in the IID/non-IID performance gap, the corresponding reduction in OFL is minimal at 0.45 percentage points.

We also explore the effectiveness of model complexity in the pre-trained setting, showing that deeper architectures like ResNet-50 can outperform simpler ones such as ResNet-18 when initialized with pre-trained weights. With pre-training, ResNet-50 achieves a 0.66 percentage point improvement over ResNet-18, whereas with random initialization, it performs 2.33 percentage points worse.

To maximize the benefits of pre-trained models, we investigate various regularization techniques designed to preserve learned representations. Among these, weight decay demonstrates particular promise, yielding a 0.31 percentage point improvement over the baseline model.

Our experimental framework encompasses the CIFAR-10 dataset, varying levels of data heterogeneity, and different ResNet architectures. Unlike conventional OFL approaches that rely on model distillation, we directly evaluate ensemble performance using various aggregation methods. Additionally, we observe that pre-trained models demonstrate superior convergence properties, achieving lower local losses and higher accuracies in early epochs. These models consistently require fewer epochs to reach 95% of their peak accuracy, particularly in homogeneous data settings.

The source code[1] of this work is largely adapted from Allouah et al.'s work [4] on FENS[2].

II. Related work

Federated learning (FL) Federated Learning (FL), introduced through FedAVG by McMahan et al. [5], enables distributed model training while preserving data privacy. In this framework, clients share model parameters with a central server after local training, and the server aggregates these parameters through averaging. Various improvements to FedAVG have been proposed, particularly focusing on non-IID data challenges. These include enhanced aggregation schemes [6],

[1]https://gitlab.epfl.ch/sacs/semester-projects/fall-2024/ofl-pretraining
[2]https://github.com/sacs-epfl/fens

[7] and modifications to local training through proximal terms [8] or control variations [9]. However, the iterative sharing of model parameters remains vulnerable to privacy attacks, as demonstrated by [2].

One-shot federated learning (OFL) One-shot Federated Learning (OFL) was introduced by Guha et al. [1] to address the communication overhead and privacy concerns of iterative FL. Their approach uses ensemble methods and knowledge distillation for model aggregation. Subsequent work has focused on two main directions: knowledge distillation-based methods requiring public datasets [10], and parameter-based aggregation methods [11]. While various improvements have been proposed, including synthetic data generation [12] and cluster-based methods [13], OFL still generally underperforms compared to traditional FL, particularly with non-IID data. Recently, FENS [4] demonstrated promising results by combining OFL's efficiency with a lightweight FL phase for prediction aggregation, significantly reducing the performance gap with traditional FL.

Pre-training Pre-training has demonstrated significant benefits across various domains, improving both convergence speed and accuracy in downstream tasks [14]. Research has shown that pre-training can enhance model robustness [15] and reduce catastrophic forgetting [16]. However, despite its widespread success, the impact of pre-training in federated learning contexts, particularly in OFL, remains largely unexplored.

III. Problem formulation

A. Classic OFL

Consider a one-shot federated learning system with maximum M clients indexed by $i \in [M]$. Each client i has a local dataset $\mathcal{D}_i = \{(\mathbf{x}_{i,j}, y_{i,j})\}_{j=1}^{n_i}$, where $\mathbf{x}_{i,j}$ represents the feature vector and $y_{i,j}$ is the corresponding label.

Let $\theta \in \mathbb{R}^d$ denote the model parameters. Each client i aims to solve the following optimization problem:

$$\pi_i = \underset{\theta}{\operatorname{argmin}}\, L_i(\theta), \tag{1}$$

where

$$L_i(\theta) = \frac{1}{n_i} \sum_{k=1}^{n_i} \ell(\theta; \mathbf{x}_{i,k}, y_{i,k}), \tag{2}$$

with L_i the empirical risk of client i and ℓ the loss function.

In OFL, the final ensemble model is formulated as:

$$\pi(x) = \sum_{i \in [M]} p_i \pi_i(x), \tag{3}$$

The weight p_i assigned to client i is typically set as $p_i = \frac{n_i}{n}$, where $n = \sum_{i \in [M]} n_i$ in OFL. The ensemble model is generally distilled into a single global model using either proxy data or synthetic data at the server. In our work, we explore alternative aggregation schemes, including learnable approaches, which are detailed in the next section.

B. Aggregation

Following the work of Allouah et al. [4], we generate the ensemble model using different aggregation schemes. This allows us to generate an ensemble model with non-linear trainable models which have been shown to outperform standard aggregators such as weighted averaging.

Let $\lambda \in \mathbb{R}^q$ denote the aggregator model parameters. The ensemble model is defined as:

$$\pi(x) := f_\lambda(\pi_1(x), \dots, \pi_M(x)), \tag{4}$$

with $\pi_1, \dots, \pi_M$ the client models obtained from 1 and 2.

For trainable aggregators, the objective is defined as the following empirical risk minimization problem:

$$\min_{\lambda \in \mathbb{R}^q} \frac{1}{|D|} \sum_{(x,y) \in D} \ell(f_\lambda(\pi_1(x), \dots, \pi_M(x)), y), \tag{5}$$

with ℓ the loss function. The final ensemble model is given by $\pi(x) := f_{\hat{\lambda}}(\pi_1(x), \dots, \pi_M(x))$.

*1) **Averaging**:* A static aggregation defined as:

$$f = \sum_{i=1}^{M} \lambda_i \odot \pi_i(x),$$

where $\lambda_i = [\frac{1}{M}, \dots, \frac{1}{M}]$ and $\odot$ denotes coordinate-wise product.

*2) **Weighted Averaging**:* Following Gong et al.'s FedKD [17], weighted averaging aggregates predictions by assigning class-specific weights to each client based on their local data distribution. For each client i, a weight vector $\lambda_i \in \mathbb{R}^C$ is computed, where C is the number of classes:

$$\lambda_i = \left[\frac{n_i^1}{\sum_{i \in [M]} n_i^1}, \frac{n_i^2}{\sum_{i \in [M]} n_i^2}, \dots, \frac{n_i^C}{\sum_{i \in [M]} n_i^C} \right],$$

where n_i^j represents the number of samples of class j in client i's local dataset, and M is the total number of clients. This weighting scheme gives more importance to clients with larger class-specific sample sizes. The final prediction for an input x is computed in the same way as simple averaging above. This approach ensures that clients with more samples of a particular class have greater influence on predictions for that class.

*3) **Neural Network Aggregation**:* The neural network aggregator learns a nonlinear mapping from the concatenated predictions of all clients to the final prediction. Given an input x, the aggregation is performed through a two-layer neural network:

$$f(x) = W_2^\top \sigma(W_1^\top z)$$

where:

- $z = \text{concat}(\pi_1(x), \ldots, \pi_M(x)) \in \mathbb{R}^{MC}$ is the concatenation of all client predictions, with $\pi_i(x) \in \mathbb{R}^C$ being the prediction vector from client i
- $W_1 \in \mathbb{R}^{MC \times k}$ is the weight matrix of the first layer, where k is a hyperparameter controlling the dimension of the hidden layer
- $W_2 \in \mathbb{R}^{k \times C}$ is the weight matrix of the output layer
- $\sigma(x) = \max x, 0$ is the ReLU activation function

The network is trained on a validation set to minimize the cross-entropy loss between the aggregated predictions and true labels. This non-linear approach allows the model to learn complex relationships between client predictions, potentially capturing complementary information from different clients and adapting to their individual strengths and weaknesses.

IV. Experimental Setup

A. Datasets, Models, Initialization

We conduct our experiments on the CIFAR-10 dataset [18], which consists of 60,000 32x32 color images evenly distributed across 10 classes, focusing on a classification task. The dataset is partitioned among 20 clients using a Dirichlet distribution with a configurable α parameter to simulate varying degrees of non-IID-ness: smaller values of α result in more non-IID distributions, while larger values approach IID distribution.

Our primary model is ResNet-18 [19], evaluated with either randomly initialized weights or pre-trained weights. Additional experiments were conducted using ResNet-50 as well. For both, the pre-trained version provided by the PyTorch TorchVision library[3], is trained on ImageNet [20]. To align with Nguyen et al. [3], we replace batch normalization layers with group normalization layers, as group normalization is better suited for federated settings than batch normalization [21]. It is important to note that this modification leaves the group normalization layers in the pre-trained models untrained, as they are reset during this process.

B. Training Configuration

We execute the training of 20 clients in parallel using the standard PyTorch training workflow within a containerized environment on a cluster. The cluster specifications are detailed in Appendix A.

[3]https://github.com/pytorch/vision

a) Local Training: Each client performs local training over 500 epochs using SGD optimizer with an initial learning rate of 0.0025. The learning rate is adjusted using Cosine Annealing scheduler.

b) Aggregator Training: The Neural network model (ref. III-B3) is trained using the Adam optimizer. The training process uses $k = 80$ with a learning rate of 5×10^{-5}, batch size of 16, and runs for 300 epochs.

V. Results

A. Data heterogeneity effects

One of the interesting findings of the work from Nguyen et al. [3] is the gap between models trained on IID data and models trained on non-IID data getting smaller when starting with pre-trained weights. In addition, they found that they could increase the number of updates to the local clients without degrading the accuracy of the final model. Since OFL represents an extreme case of increasing the number of local client updates, we investigated the impact of using pre-trained weights in this setting.

The results in figure 1 highlight the differences between traditional FL and OFL approaches when comparing pre-trained and randomly initialized models. In the FL setting, pre-trained weights significantly reduce the performance gap between IID and non-IID scenarios to 13.89 percentage points, compared to 18.68 percentage points with random initialization. This observation aligns with the findings of [3], which emphasize that pre-trained weights help mitigate the challenges posed by data heterogeneity.

In contrast, this benefit is not observed in the OFL setting, where the performance gap remains nearly unchanged — 10.04 percentage points for pre-trained weights versus 10.48 percentage points for random initialization. The difference between the gaps for pre-trained and random initialization models is a notable 4.79 percentage points in FL but only 0.45 percentage points in OFL.

We hypothesize that combining client models into a single global model at the end of each FL round acts as a regularizer. With pre-trained weights, this process likely positions the global model within a connected low-loss valley, as described in the work of Chen et al. [22] enabling effective aggregation despite heterogeneous data. In contrast, random initialization with FL may not initially place the global model in a suitable region of the loss landscape, limiting its ability to use highly heterogeneous data. In OFL, the high number of local epochs likely causes models to drift away from the pre-trained position in the loss landscape, often becoming trapped in isolated low-loss valleys, regardless of whether the data is IID or non-IID.

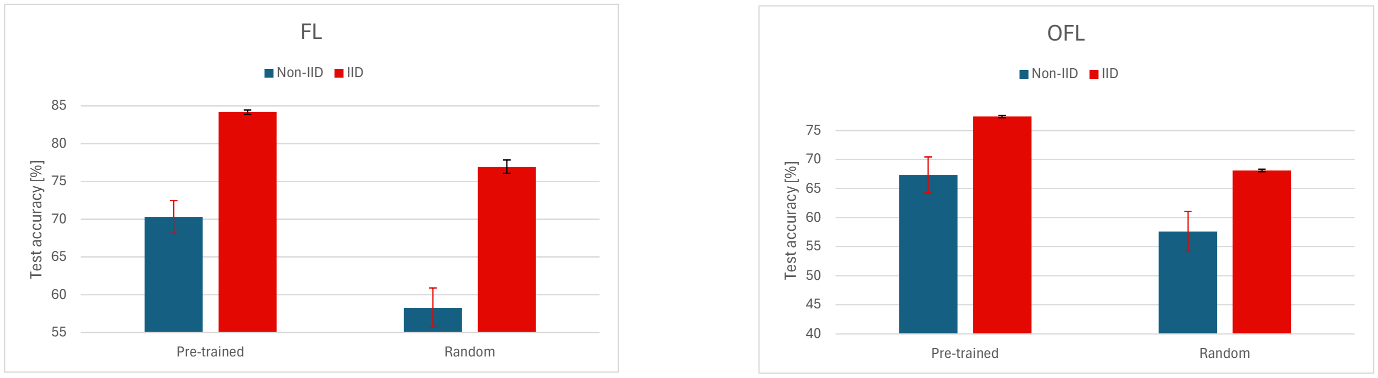

Fig. 1: Average accuracy across 3 seeds of ResNet-18 model trained with FEDADAM FL and OFL ResNet-18 ensemble model with NN aggregation (ref. III-B3) trained on IID ($\alpha = 1$) and non-IID ($\alpha = 0.05$) data.

B. Model size effects

In Table I, we observe the effects of the non-IID data distribution on the performance of ResNet models with random initialization. Notably, as the complexity of the model increases (from ResNet-8 to ResNet-18 to ResNet-50), the accuracy decreases under non-IID settings, evidenced by the lower scores for ResNet-50 compared to ResNet-18 and ResNet-8. This trend suggests that deeper models are more prone to overfitting when exposed to heterogeneous data, potentially due to the increased parameter space and susceptibility to finding suboptimal solutions in the absence of sufficient regularization or training data diversity.

Dataset	α	ResNet-8	ResNet-18	ResNet-50
CIFAR-10	0.05	68.22	57.63	54.10
	0.1	75.61	64.71	63.06

TABLE I: Average accuracy with standard deviation across 3 seeds of different ResNet ensemble models with NN aggregation (ref. III-B3) with random initialization under non-IID settings. ResNet-8 results are obtained from [4].

Figure 2 compares the performance of randomly initialized ResNet-18 and ResNet-50 models under varying levels of data heterogeneity. Similarly to Table I, ResNet-18 outperforms ResNet-50 for the same reasons regardless of the data heterogeneity level.

When examining figure 3, which compares ResNet-18 and ResNet-50 with pre-trained weights, we observe an intriguing trend. ResNet-18 performs better under high data heterogeneity, likely due to its simpler architecture being less prone to overfitting localized patterns. However, as data becomes more homogeneous, ResNet-50 surpasses ResNet-18, leveraging its greater capacity to extract nuanced features. Quantitatively, ResNet-50 performs on average 0.66 percentage points above ResNet-18 across all heterogeneity values when in the pre-trained setting, whereas it performs 2.33 percentage points below ResNet-18 when in the randomly initialized setting. Pre-training appears to stabilize fine-tuning in ResNet-50, by positioning its weights closer to the global optimum in the optimization landscape, reducing the risk of getting stuck in local optima despite the model's complexity.

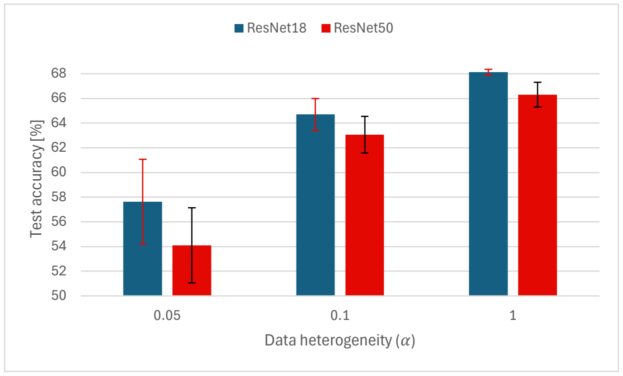

Fig. 2: Average accuracy with standard deviation across 3 seeds of ResNet-18 and ResNet-50 ensemble models with NN aggregation (ref. III-B3) with random initialization under varying data heterogeneity.

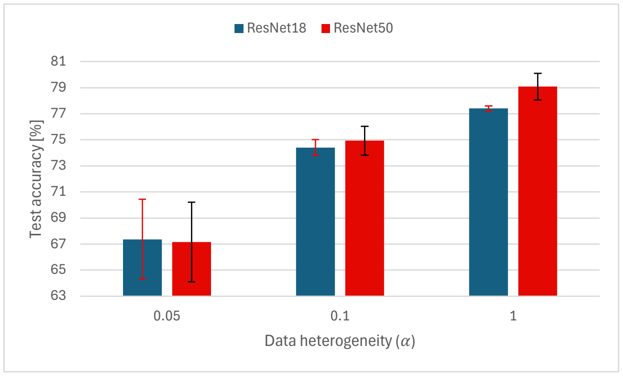

Fig. 3: Average accuracy with standard deviation across 3 seeds of ResNet-18 and ResNet-50 ensemble models with NN aggregation (ref. III-B3) with pre-trained weights under varying data heterogeneity.

These results suggest that pre-training plays a critical role in enabling complex models to perform well under favorable conditions. By positioning the model in advantageous regions of the loss surface, pre-training allows ResNet-50 to fully exploit its capacity and avoid poor initializations.

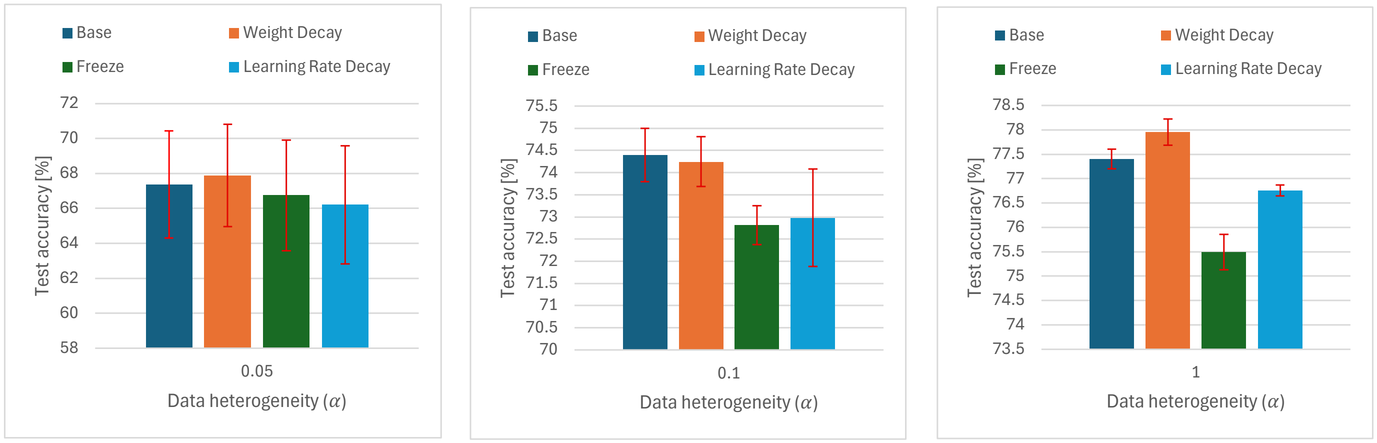

Fig. 4: Average accuracy with standard deviation across 3 seeds of ResNet-18 ensemble models with NN aggregation (ref. III-B3) with various regularization strategies applied to pre-trained weights under varying data heterogeneity. An observation on the influence of random seed on performance in highly heterogeneous data is detailed in Appendix B.

This aligns with findings from [23], which highlight that pre-trained models converge more effectively to shared low-loss basins, even in diverse data settings. Such convergence mitigates the challenges of isolated or unfavorable loss regions, showcasing the potential of pre-training in optimizing complex architectures like ResNet-50.

C. Weights regularization

To maintain stability in the earlier layers of the pre-trained model while allowing the later layers to adapt more freely, we explored the use of gradual layer-wise regularization strategies. The approach involves applying a higher penalty to earlier layers, encouraging them to retain their pre-trained representations, while allowing the later layers to adjust more during fine-tuning. This is hypothesized to contribute to a better global model after aggregation. Building on the work of Tinn et al. [24], we adopt strategies such as layer-wise learning rate decay and layer freezing to enhance the stability of the fine-tuning process.

We explored three approaches to training the model:

- The method proposed by Clark et al. [25], in which a layer-wise learning rate decay factor of 0.9 is applied across the network layers.
- A layer-wise weight decay strategy inspired by the above training [25], where a base weight decay of 0.01 is applied, decreasing with a decay factor of 0.9 for successive layers.
- The method of Grießhaber et al. [26], where 25% of the model's layers were frozen during training, as this configuration was found to be optimal in their procedure.

As shown in figure 4, the results highlight the impact of different regularization methods. Layer-wise weight decay maintains accuracy similar or higher to the base model, while layer-freezing and learning rate decay show a noticeable drop. The weight decay method shows potential, being on average 0.31 percentage points above the base model across different heterogeneity levels, and could be a way to further capitalize on the benefits of pre-trained models.

Intuitively, the methods analyzed by Tinn et al. [24] (learning rate decay and layer freezing) operate on the relative update of the weights, making their effectiveness sensitive to the number of training epochs. While the learning rate is reduced, executing too many epochs may diminish the benefits of these approaches. In contrast, the layer-wise weight decay regularization we propose directly impacts the absolute values of the weights. It allows larger weight magnitudes in the later layers, facilitating task-specific training on CIFAR-10, while penalizing large weights in the earlier layers to preserve the pre-trained representations. This approach is more robust to the number of epochs, which likely explains the observed performance gains.

D. Affinity term regularization

In order to keep the local learned models closer to the initialization point of the pre-trained model, we have used a technique developed by Chen et al. [22]. This involves adding an **affinity regularization term** to the loss function of each client, which penalizes the squared Euclidean distance between the local model and the initial model weights. Equation 2 becomes:

$$L_i(\theta) = \frac{1}{n_i} \sum_{k=1}^{n_i} \left(l(\theta; \mathbf{x}_{i,k}, y_{i,k}) + \lambda_a \|\theta - \theta_0\|^2 \right), \qquad (6)$$

where λ_a is the affinity coefficient. A larger value of λ_a imposes a greater penalty on client models that deviate further from the pre-trained model.

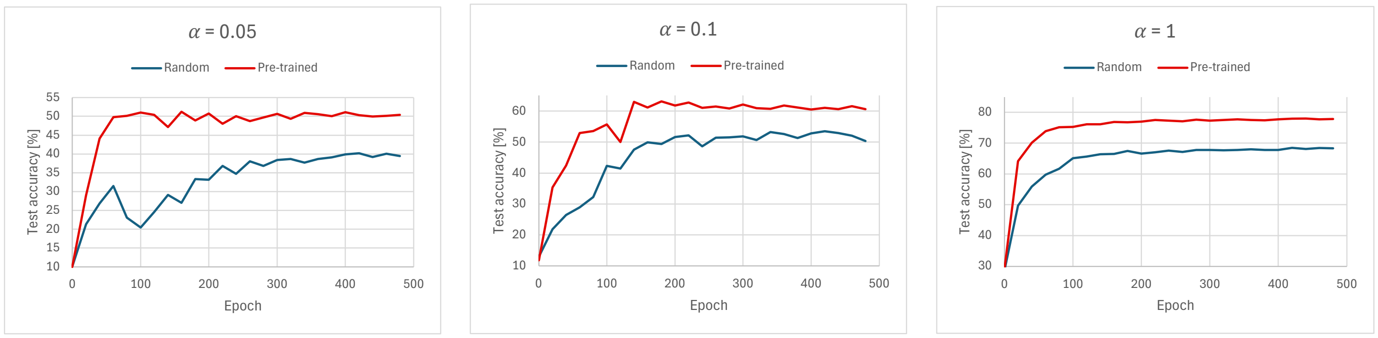

Fig. 5: Accuracy progression of ResNet-18 ensemble models with weighted averaging aggregation under varying data heterogeneity.

α	$\lambda_a = 0$	$\lambda_a = 0.01$	$\lambda_a = 0.1$	$\lambda_a = 1$
0.1	74.40	71.22	67.00	54.04

TABLE II: Accuracy of ResNet-18 ensemble models with NN aggregation (ref. III-B3) with pre-trained weights at $\alpha = 0.1$ heterogeneity with different affinity coefficients.

As shown in Table II, affinity regularization reduced model accuracy, with the baseline model performing best in the NN ensemble setting and accuracy declining as the affinity coefficient increased. Consequently, further exploration of affinity regularization was deemed unnecessary. Chen et al.'s ablation studies [22] offer a possible explanation: the affinity term alone improved accuracy by only 1.5 percentage points, whereas combining it with the diversity term — a loss term summing distances between model pairs to encourage variability — achieved a 4 percentage point increase. This complementary effect was not tested in our experiments, and the affinity term's hyperparameters may have been suboptimal.

E. Optimal Checkpoints

Through our experiments, we observed that selecting the best model for each client based on validation accuracy evaluated on the client's local dataset, which shares the same distribution as the training data, and constructing an ensemble from these locally optimal models resulted in lower test accuracy for the ensemble compared to using the final models from each client. This observation led us to hypothesize the existence of an optimal checkpoint during training at which the ensemble model achieves its best performance.

To investigate this hypothesis, we saved the client models at regular intervals (every 20 epochs) and evaluated the resulting ensemble models on the test dataset at these same intervals. However, as shown in figure 5, the ensemble model does not display a clear optimal checkpoint. Instead, it quickly reaches a plateau where additional training provides no further benefits, and the performance remains stable without declining. This suggests that while training does not yield a specific "best" checkpoint for the ensemble, it is sufficient to train until this plateau is reached.

Figure 5 highlights how pre-trained models accelerate convergence, especially in non-IID scenarios. Table III shows that with highly non-IID data $\alpha = 0.05$), pre-trained models converge in just 60 epochs, compared to 300 epochs for randomly initialized models. This demonstrates the value of pre-training for faster convergence in heterogeneous distributions. As data homogeneity increases (larger α), this advantage diminishes; for $\alpha = 1$, the gap in epochs to reach 95% of final accuracy narrows to less than 20. However, pre-trained models still achieve higher absolute accuracy.

α	Random init epochs	Pre-trained epochs
0.05	300	60
0.1	200	140
1.0	100	80

TABLE III: Epochs required for ResNet-18 ensemble model with weighted averaging aggregation to reach 95% of the best accuracy under varying data heterogeneity.

α	Random init epochs	Pre-trained epochs
0.05	38	17
0.1	72	38
1.0	171	70

TABLE IV: Epochs required for average across 20 clients of local ResNet-18 model accuracies to reach 95% of the best accuracy under varying different data heterogeneity.

Figure 6 confirms the faster convergence rate: the average local loss across the 20 clients is consistently lower with a stronger drop in the early epochs for pre-trained models. Similarly, the average local accuracy across clients is consistently higher, with a sharper incline during the initial epochs for pre-trained models. Table IV further demonstrates that local models consistently require fewer epochs to achieve 95% of their peak accuracy. This difference is most pronounced when training on a homogeneous dataset, likely due to the reduced noise level. However, when accounting for noisy variations in the non-IID accuracy curves, the behavior of local model training appears similar between IID and non-IID settings.

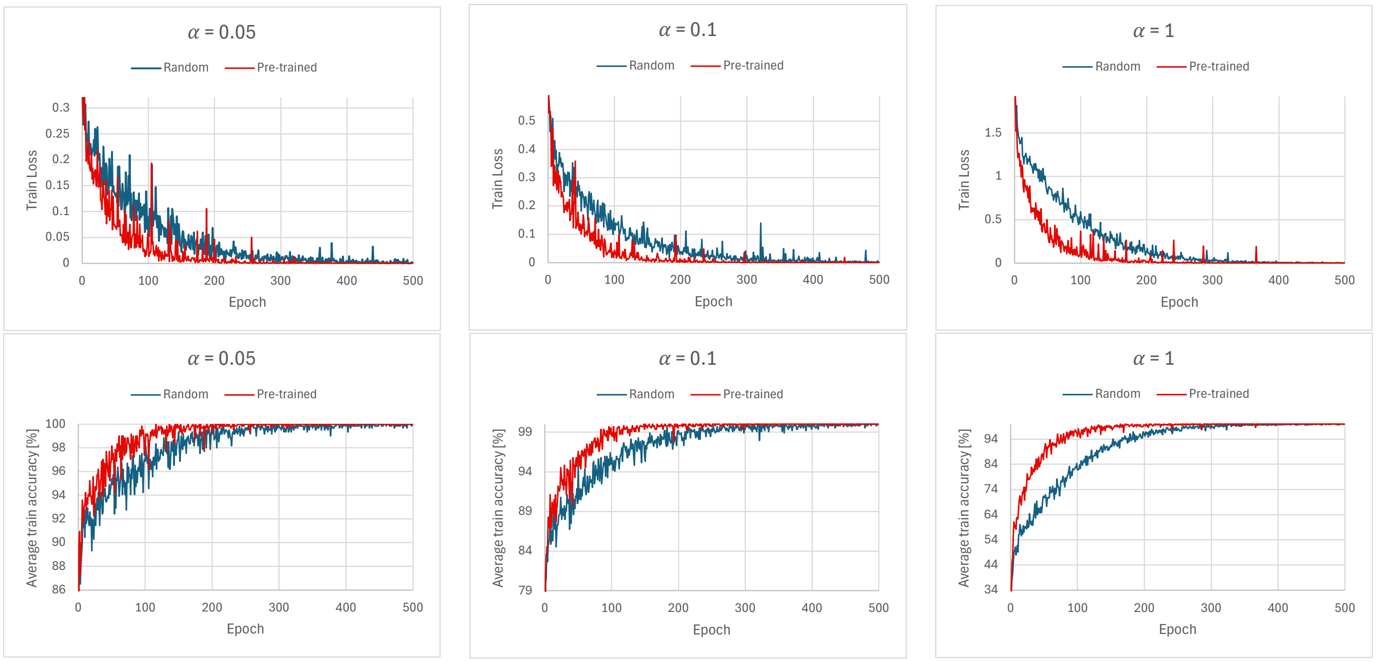

Fig. 6: Average across 20 clients of local training losses and accuracies of ResNet-18 models under varying data heterogeneity.

VI. Discussion

The objective of this project was to conduct a comprehensive investigation into the effects of pre-training on One-shot Federated Learning (OFL), with particular focus on how it compares to traditional Federated Learning (FL) across different aspects of model training and deployment.

A. Key Findings

Our investigation revealed several important insights about the relationship between pre-training and OFL. Most notably, we found that the accuracy gap bridging benefit of pre-training in non-IID vs IID settings observed in traditional FL do not transfer directly to the OFL setting. This finding suggests that OFL may require different approaches to handle data heterogeneity.

The relationship between model complexity and pre-training emerged as another significant finding. Our results demonstrate that deeper architectures like ResNet-50 can effectively leverage pre-trained weights to outperform simpler models such as ResNet-18. This suggests that pre-training can help stabilize the training of more complex models, enabling them to better utilize their capacity for feature extraction.

In our exploration of regularization techniques, weight decay emerged as a promising approach for maximizing the benefits of pre-trained models. However, other techniques such as layer freezing and layer-wise learning rate decay showed decreased performance, while affinity regularization failed to improve model accuracy in the neural network ensemble setting.

B. Limitations

A key limitation of our study is hyperparameter optimization. Due to computational constraints, we couldn't exhaustively tune hyperparameters for each model configuration, which may have limited performance improvements for different architectures.

Another limitation arose when replacing batch normalization with group normalization. The group normalization layers were reinitialized instead of being trained, which may have hindered the model's ability to fully benefit from pre-training, particularly in handling batch statistics across different data distributions.

C. Future Work

Our findings open several promising avenues for future research:

1) Investigation of alternative regularization techniques specifically designed for OFL with pre-trained models. Given the limited success of traditional regularization methods, developing approaches that better account for the continuous nature of OFL could yield significant improvements.
2) Exploring the integration of the diversity term into the loss function, given its complementarity to the affinity term, as demonstrated in prior studies.
3) Exploration of custom pre-training approaches using datasets like Tiny ImageNet, specifically designed to address the initialization issues with normalization layers. This could help ensure that all model components,

including group normalization layers, benefit from pre-training, potentially leading to better overall performance.

4) Development of adaptive checkpoint selection mechanisms. Given that optimal stopping points may vary across clients in heterogeneous settings, future work could explore training a meta-network that considers data distribution characteristics to determine client-specific stopping points. This approach might help achieve the performance gap reduction seen in traditional FL within the OFL setting. Such a network could learn to analyze local data distributions and model behavior patterns to make informed decisions about when to checkpoint each client's model.

These research directions could help bridge the gap between theoretical understanding and practical implementation of pre-training in OFL systems, ultimately leading to more robust and efficient federated learning solutions.

VII. Conclusion

This work presents a thorough investigation of pre-training effects in One-shot Federated Learning, revealing that the benefits observed in traditional Federated Learning do not directly transfer to the one-shot setting. While pre-training proves ineffective at reducing the IID/non-IID performance gap in OFL, it demonstrates value in stabilizing more complex architectures and can potentially be enhanced through appropriate regularization techniques. These findings provide important insights for OFL behavior.

Acknowledgments

The author would like to express sincere gratitude to Akash Dhasade and Mathis Randl for their supervision and guidance throughout this research project. Special thanks to Rishi Sharma, Milos Vujasinovic, and Martijn De Vos for their valuable insights and constructive feedback during discussions.

References

[1] N. Guha, A. Talwalkar, and V. Smith, "One-shot federated learning," *CoRR*, vol. abs/1902.11175, 2019.
[2] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients – how easy is it to break privacy in federated learning?," 2020.
[3] J. Nguyen, J. Wang, K. Malik, M. Sanjabi, and M. Rabbat, "Where to begin? on the impact of pre-training and initialization in federated learning," 2023.
[4] Y. Allouah, A. Dhasade, R. Guerraoui, N. Gupta, A.-M. Kermarrec, R. Pinot, R. Pires, and R. Sharma, "Revisiting ensembling in one-shot federated learning," 2024.
[5] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.
[6] H. Wang, M. Yurochkin, Y. Sun, D. S. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *CoRR*, vol. abs/2002.06440, 2020.
[7] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," *CoRR*, vol. abs/1905.10497, 2019.
[8] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018.
[9] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: stochastic controlled averaging for on-device federated learning," *CoRR*, vol. abs/1910.06378, 2019.
[10] Q. Li, B. He, and D. Song, "Model-agnostic round-optimal federated learning via knowledge transfer," *CoRR*, vol. abs/2010.01017, 2020.
[11] J. Zhang, C. Chen, B. Li, L. Lyu, S. Wu, J. Xu, S. Ding, and C. Wu, "A practical data-free approach to one-shot federated learning with heterogeneity," *CoRR*, vol. abs/2112.12371, 2021.
[12] M. Yang, S. Su, B. Li, and X. Xue, "Exploring one-shot semi-supervised federated learning with a pre-trained diffusion model," 2024.
[13] D. K. Dennis, T. Li, and V. Smith, "Heterogeneity for the win: One-shot federated clustering," *CoRR*, vol. abs/2103.00697, 2021.
[14] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Large scale learning of general visual representations for transfer," *CoRR*, vol. abs/1912.11370, 2019.
[15] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," *CoRR*, vol. abs/1901.09960, 2019.
[16] S. V. Mehta, D. Patil, S. Chandar, and E. Strubell, "An empirical investigation of the role of pre-training in lifelong learning," *CoRR*, vol. abs/2112.09153, 2021.
[17] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. Doermann, and A. Innanje, "Preserving privacy in federated learning with ensemble cross-domain knowledge distillation," 2022.
[18] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," pp. 32–33, 2009.
[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.
[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.
[21] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The non-iid data quagmire of decentralized machine learning," *CoRR*, vol. abs/1910.00189, 2019.
[22] M. Chen, M. Jiang, X. Zhang, Q. Dou, Z. Wang, and X. Li, "Local superior soups: A catalyst for model merging in cross-silo federated learning," 2024.
[23] H.-Y. Chen, C.-H. Tu, Z. Li, H.-W. Shen, and W.-L. Chao, "On the importance and applicability of pre-training for federated learning," 2023.
[24] R. Tinn, H. Cheng, Y. Gu, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Fine-tuning large neural language models for biomedical natural language processing," *CoRR*, vol. abs/2112.07869, 2021.
[25] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," *CoRR*, vol. abs/2003.10555, 2020.
[26] D. Grießhaber, J. Maucher, and N. T. Vu, "Fine-tuning BERT for low-resource natural language understanding via active learning," *CoRR*, vol. abs/2012.02462, 2020.

Appendix

A. Compute Resources

The cluster used for training consists of

- 2 x AMD EPYC 9454 48-Core Processor with 8 x NVIDIA H100 SXM5 80GB HBM3 GPU
- 2 x AMD EPYC 7543 32-Core 2.8GHz Processor with 8 x NVIDIA A100 SXM4 80GB GPU
- 2 x Intel(R) Xeon(R) Gold 6240 18-Core 2.60GHz Processor with 4 x NVIDIA V100 SXM2 32GB GPU.

In wall-clock time, the local models training takes around 2.5 hours for ResNet-18, 4 hours for ResNet-50. Incorporating the affinity loss term caused the ResNet-18 model training to take 16 hours. Aggregation training requires approximately 30 minutes.

B. Seeds on high heterogeneity

An interesting observation was the significant influence of the random seed on the final accuracy of all OFL ensemble models in the highest heterogeneity setting ($\alpha = 0.05$). Specifically, the second seed (91) always underperformed compared to the first and third seeds (90 and 92) across both NN aggregation and weighted averaging. This disparity particularly affected the weight decay regularization results, as summarized in Table V.

Method	**Seed 90**	**Seed 91**	**Seed 92**
Base ResNet-18 Pre-trained	69.18	63.82	69.08
Weight Decay	69.68	64.52	69.46
Layer Freezing	68.58	63.10	68.56
Learning Rate Decay	67.72	62.34	68.54

TABLE V: Comparison of methods across different random seeds (90, 91, 92).

As shown in Table V, the weight decay regularization method consistently achieved the highest accuracy across all seeds, outperforming the other methods. This supports the conclusion regarding the potential of weight decay regularization discussed in Section V-C.