Where to Begin? On the Impact of Pre-Training and Initialization in Federated Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

An oft-cited challenge of federated learning is the presence of heterogeneity. *Data heterogeneity* refers to the fact that data from different clients may follow very different distributions. *System heterogeneity* refers to the fact that client devices have different system capabilities. A considerable number of federated optimization methods address this challenge. In the literature, empirical evaluations usually start federated training from random initialization. However, in many practical applications of federated learning, the server has access to proxy data for the training task that can be used to pre-train a model before starting federated training. We empirically study the impact of starting from a pre-trained model in federated learning using four standard federated learning benchmark datasets. Unsurprisingly, starting from a pre-trained model reduces the training time required to reach a target error rate and enables the training of more accurate models (up to 40%) than is possible when starting from random initialization. Surprisingly, we also find that starting federated learning from a pre-trained initialization reduces the effect of both data and system heterogeneity. We recommend that future work proposing and evaluating federated optimization methods evaluate the performance when starting from random and pre-trained initializations. We also believe this study raises several questions for further work on understanding the role of heterogeneity in federated optimization.

1 Introduction

Federated learning (FL) has emerged as a popular distributed machine learning paradigm for privately training a shared model across many participants while the training data never leaves the participant devices. This paper empirically investigates the impact of model initialization on federated optimization methods. Previous empirical evaluations of FL methods nearly always start federated training from a randomly initialized model. Transfer learning from pre-trained models has become common practice in natural language processing [31, 6] and computer vision [12, 7], yielding state-of-the-art results on many tasks and enabling faster model convergence in the centralized setting. Although public proxy data is available at the server in many applications, few prior works studied the impact of starting federated training from a pre-trained model.

In cross-device FL [18], the primary setting considered in this paper, a central server coordinates a large number of client devices (possibly on the order of hundreds of millions). Each device possesses a local dataset, and the data at different devices follow different distributions, leading to the *data heterogeneity* challenge [18]. Moreover, devices communicate with the server over low-bandwidth, high-latency links, making communication overhead a significant performance factor.

The predominant approach to federated training builds on local update methods such as FEDAVG [26], where a device performs several local updates (e.g., one epoch of SGD on their local training set)

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

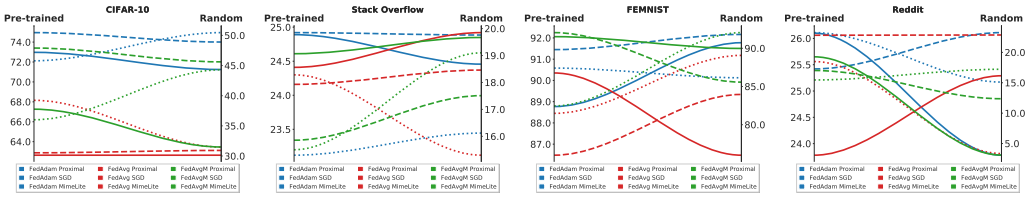

Figure 1: The test set accuracy on four datasets with random and pre-trained weights. We represent SGD by solid lines, PROXIMAL by dashed lines, and MIMELITE by dotted lines. See Section 3.3 and Table 3 in Appendix C for comparison characteristics of each method.

before transmitting an update to the server. Although this reduces communication overhead, it can also exacerbate data heterogeneity. Several approaches have been proposed to address this challenge [24, 14, 32, 36, 19, 20, 41, 1].

Contributions. In this work, we consider the question: *How does model initialization (random or pre-trained) impact the behavior of federated optimization methods?* Towards answering this question, we perform an extensive empirical study, comparing 15 variations of federated optimization methods on four commonly-used FL benchmark datasets. Our study reveals three key findings:

1. Although optimizers designed to address heterogeneity typically lead to better performance when starting from a random initialization, when starting from pre-trained model we observe that: (i) there is not as big of a difference between optimizers in terms of accuracy after a fixed number of rounds, and (ii) using an adaptive optimizer at the server, such as FEDADAM, is more important than using any method for addressing heterogeneity.
2. Starting from a pre-trained model significantly reduces the difference between having non-IID vs IID data at clients. Furthermore, when starting from a pre-trained model, the number of local epochs per round can be significantly increased without degrading the final accuracy.
3. The initial loss is not always lower when starting from a pre-trained model. However, the largest Hessian eigenvalue (i.e., local Lipshitz constant, or smoothness) is consistently smaller at initialization when starting from a pre-trained model, compared to when starting from a random initialization.

Some of our empirical observations are consistent with existing FL theoretical convergence guarantees. Our findings also highlight that some aspects of FL are not captured with the existing theory, suggesting vital directions for future work.

Initializing FL with a pre-trained model can increase final model accuracy and reduce the number of rounds required to achieve a target accuracy. Pre-training leads to communication savings and reduces the overall training time. Figure 4 in Appendix D.1) demonstrates the benefit of pre-training across several datasets (hyperparameters were tuned separately for each dataset–initialization pair; see Section 3 for details of the experimental setup).

Our findings are reproducible using the open-source federated learning framework FLSim [8]. Informed by these findings, we present several recommendations for future research on federated optimization.

2 Problem Formulation and the FEDOPT framework

We consider the standard optimization formulation of federated training. We seek to find model parameters w that solve the problem,

$$\min_{w \in \mathbb{R}^d} f(w) := \sum_{i=1}^{m} p_i F_i(w) \tag{1}$$

where m is the total number of clients, the function F_i measures the average loss of a model with parameters w on the ith client's training data, and $p_i > 0$ is the weight given to client i. Usually p_i is taken to be proportional to the number of samples at client i so that the optimization problem gives

equal weight to all training samples. The goal is to find a model that fits all clients' data well on (weighted) average. In FL, F_i is only accessible by client i.

All of the methods we consider in this study can be expressed in the general FEDOPT framework introduced in Reddi et al. [32], which encompasses both client and server optimization updates; see Algorithm 1 in the Appendix for pseudo-code. At round t, the server sends its model x^{t-1} to a cohort of clients. Each client in the cohort performs E epochs of training using CLIENTOPT with client learning rate η_c, producing a local model y_i^E. Each client then communicates the difference between the client's local model and the server model, Δ_i where $\Delta_i^t := y_i^E - x^{t-1}$ (*client update*). The server computes a weighted average Δ^t of the client updates and updates its own model via

$$x_t = \text{SERVEROPT}(x^{t-1}, \Delta^t, \eta_s, t)$$

where $\text{SERVEROPT}(x_t, -\Delta_t, \eta_s, t)$ is a first-order optimizer, η_s is the server learning rate, and t is the round number.

3 Experimental Setup

Next, we describe the experimental setup used to evaluate how initialization impacts the performance of federated optimization methods.

3.1 Datasets, Models, and Tasks.

To facilitate comparing with other work in the literature, we experiment on four standard FL benchmark datasets: CIFAR-10 [22], Federated EMNIST-62 (FEMNIST) [5, 3], Stack Overflow [2] and Reddit [3]. All datasets have a natural non-IID partitioning of the data except for CIFAR-10, for which we use the Dirichlet allocation approach of [14] with parameter 0.1 to partition data across 50 users. For CIFAR-10 and FEMNIST, we train Squeezenet [17] and ResNet18 [11] models, replacing batch norm with group norm [32, 13]. For Stackoverflow and Reddit, we use the DistilGPT2 [16] and CharLM [21] models. For additional information about the datasets and models, see Appendix B.1.

3.2 Initialization Strategies.

We consider two initialization strategies: random initialization and supervised pre-training from public data.

Random initialization. Most prior federated optimization works use random weights to initialize the model. We can use the same random initialization strategies used in the standard (centralized) training of deep networks for each model [17, 16, 11, 21].

Supervised pre-training. In many FL applications, pre-training can be done on a large non-private proxy dataset available at the server. To quickly reproduce our results and compare them with other methods in the literature, we use publicly available pre-trained models in this study. For tasks using Squeezenet and ResNet18, we use the version of the model pre-trained on ImageNet, available in the PyTorch Torchvision library.[1] For tasks using DistilGPT2, we use the model weights provided in the HuggingFace library that has been distilled from a pre-trained GPT2[2], and for tasks using CharLM, we pre-train the model on WikiText-103 [28] for 100 epochs.

Supervised pre-training is just one possibility, and we leave investigating other pre-training strategies (e.g., self-supervised pre-training or meta-learning) as future work; see Section 7.

3.3 Algorithms

We compare federated training with five different CLIENTOPT strategies: (C1) **SGD** where clients perform standard stochastic gradient updates; (C2) **FEDPROX**-style local updates [24]; (C3) **FEDNOVA**-style local updates and aggregation [36]; (C4) **MIMELITE**-style local updates, where clients use auxiliary optimizer state (e.g., momentum buffers) from the server [20]; and (C5) full-batch gradient updates (**GD**), i.e., the update Δ_i^t sent to the server is a full-batch gradient on client i's local training set.

[1] `https://github.com/pytorch/vision`

[2] `https://huggingface.co/distilgpt2`

Recall that the server treats the aggregated update Δ^t as a stochastic gradient when updating its model x^t. We consider three strategies for SERVEROPT. (S1) In the **SGD** strategy, the server updates the global model using stochastic gradient descent; when CLIENTOPT is also SGD< this is equivalent to FEDAVG [26]. (S2) In **SGD with momentum**, the server uses SGD with heavy-ball momentum; when CLIENTOPT is SGD, this is equivalent to FEDAVGM [14]. (S3) In **Adam**, SERVEROPT corresponds to an Adam update; when CLIENTOPT is SGD, this is equivalent to FEDADAM [32].

The method commonly referred to as FEDSGD [27] is obtained when CLIENTOPT is full-batch gradient descent (GD) and SERVEROPT is SGD, with $\eta_c = 1$ and $E = 1$.

We focus on the above choices for CLIENTOPT and SERVEROPT because they reflect the most widely-cited federated optimization methods. They also represent a diverse set of possible choices available to the practitioner seeking to deploy cross-device federated training at scale.

3.4 Implementation and Tuning

We repeat each experiment with three different seeds and report the average. For all algorithms, we run hyperparameter sweeps to tune client and server learning rates η_ℓ and η_g, and the proximal penalty parameter μ for FEDPROX. We use momentum parameter 0 for FEDAVG and 0.9 for FEDAVGM. For FEDADAM, we fix $\beta_1 = 0.9$ and $\beta_2 = 0.99$ for Adam; see Appendix B for details. Unless otherwise specified, each client update entails running one local epoch with a fixed batch size per task. We perform 1050 rounds of training for Stackoverflow, 942 rounds of training for Reddit, 1000 rounds of training for CIFAR-10, and 1082 training rounds for FEMNIST. See Appendix B.2 for additional implementation details. We ran our experiments using PyTorch [29] and the open-source federated learning simulation framework FLSim [8].

4 The Impact of Pre-Training in FL

Pre-training changes the ranking of federated optimization algorithms. If one sorts federated optimization methods based on their performance when starting from a random initialization, the order is substantially different from when using a pre-trained initialization. We focus on nine combinations of SERVEROPT and CLIENTOPT, only using local update methods for CLIENTOPT and excluding full-batch gradient descent. Figure 1 presents the changes between pre-trained and random weight initializations.

First, observe that the span of final accuracies is much smaller when starting from a pre-trained model. Second, all methods starting with pre-trained models achieve a better accuracy after the same number of steps compared to random models. Lastly, observe that the order of methods changes depending on the initialization. Although no particular method dominates across all workloads in Figure 1, FEDADAM with SGD for CLIENTOPT performs consistently well when starting from a pre-trained model, especially on the two larger language modeling workloads, Stack Overflow and Reddit, and so we focus on studying FEDADAM-SGD below.

Pre-training closes the accuracy gap between non-IID and IID. We study how pre-training and data heterogeneity affect convergence without system heterogeneity by fixing $E_i = 1$. We compare FEDADAM with SGD locally under IID, and Non-IID data splits. In Figure 3, we report the average accuracy for FedAdam [32] on four datasets. As expected, randomly initialized models perform much worse than their pre-trained counterparts, and IID partitions yield better quality than non-IID. Surprisingly, the gap between models trained on IID data and models trained on non-IID data is significantly smaller when starting with pre-trained weights. Moreover, pre-training reduces the negative effects of data heterogeneity (i.e., client drift). As a result, we observe that (see Figure 2) when training from a pre-trained model, increasing the number of local updates does not degrade the final accuracy, in contrast to training from a random model.

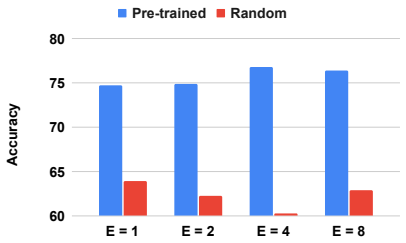

Figure 2: The accuracy for CIFAR-10 using ResNet-18 with increasing number of local epochs.

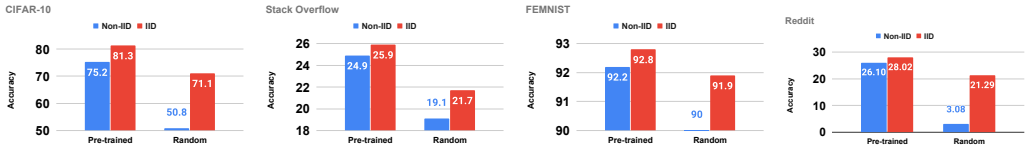

Figure 3: The average accuracy on 3 different seeds for FEDADAM trained on IID and non-IID data. For CIFAR-10 Non-IID, we generate 100 non-IID clients using a Dirichlet(0.1). For other three datasets, we use the natural non-IID client partitions.

	CIFAR-10	FEMNIST	Stack Overflow	Reddit
Pre-trained	661.99	26.29	151.05	647.19
Random	4843.13	355.51	185.02	1309.68

Table 1: The top eigenvalue of the Hessian matrix for each dataset between the pre-trained and random initialized models.

Pre-training reduces the impact of system heterogeneity. To study the impact of pre-training on system heterogeneity, we follow the setup described in [24, 36].

We sample 30% of clients uniformly at random, and client i performs E_i local epochs where $E_i \sim U(1, 5)$ to model the setting where clients have different processing capabilities where clients perform as much work as they can within a given time limit. Detailed results are shown in Figure 5 in the Appendix.

When starting from pre-trained weights, SGD locally outperforms other CLIENTOPT methods across most SERVEROPT options, except for FEDAVG on CIFAR-10 (Figure 5 left). We postulate that adding momentum to the server is enough to correct the negative effect of system heterogeneity when starting from pre-trained weights. On the other hand, when starting from random weights, optimizers specifically designed for system heterogeneity outperform SGD. (Figure 5 right). Moreover, the accuracy gap between algorithms is more pronounced in the random initialization setting, whereas in the pre-trained setting, all algorithms converge to similar accuracies. Our results suggest that pre-training reduces the need for algorithms that try to correct system heterogeneity. See Appendix D.3 for additional supporting results.

5 Understanding Why Pre-training Helps Federated Optimization

While pre-training unsurprisingly speeds up convergence, the reason for the speedup is less apparent. In this section, we examine why pre-training is beneficial to federated learning.

Connection to Existing Theory. Here, we present the existing optimization theory for FEDAVG and discuss how pre-training helps to improve the model convergence.

Following the formulation in Section 3, suppose that there are total m clients, jointly optimizing a global objective function $f(w) = \sum_{i=1}^{m} p_i F_i(w)$. Each client's local loss function $F_i(w)$ is L-Lipschitz smooth. For ease of presentation, we assume that all clients participate in training and perform K local SGD updates at each round. Then, under standard assumptions, one can show that after R communication rounds, the expected gradient norm satisfies (see Theorem V in [19]):

$$\mathbb{E}\left\|\nabla f(\bar{x}^R)\right\|^2 \leq \mathcal{O}\left(\frac{\sqrt{F}}{\sqrt{RKm}} + \frac{F^{2/3}\zeta^{2/3}}{R^{2/3}}\right), \tag{2}$$

where $\bar{w}^R$ represents a weighted sampled model from all previous rounds, ζ is a measure of data heterogeneity, and $F = f(x^0) - f^*$ denotes the gap between the initial loss value and the optimal loss value. When starting from a pre-trained model, the initial gap F becomes lower, as observed in Figure 7 (top left) on CIFAR-10. As a result, the optimization error upper bound (2) will be smaller, i.e., we get better worst-case performance. We need to mention that a lower initial training loss is only one possible reason for the significantly improved training performance. We also observe that the initial training loss can be higher when using a pre-trained model on some training tasks (details

are in the Appendix D.2). However, the pre-trained model still converges faster. This phenomenon calls for a new optimization theory to explain this phenomenon.

In addition, in order for FEDAVG to achieve the $\mathcal{O}(1/\sqrt{RKm})$ asymptotic convergence rate, previous works [37, 39, 19, 35] showed that the number of local updates K should be upper bounded as follows:

$$K \leq \mathcal{O}\left(\frac{R^{1/3}}{F^{1/3}\zeta^{4/3}m}\right). \tag{3}$$

Clearly, if F becomes smaller starting from a pre-trained model, one can use a larger number of local updates. This corroborates our empirical observations in Figure 2.

6 Recommendations

In this work, we study the effects of pre-training on federated optimization methods. Our results inform a series of the following recommendations:

1. When evaluating FL algorithms, the researchers need to experiment with both pre-trained (if available) and random weights, as they might have different behaviors.

2. When deploying FL to a production environment, using adaptive server optimizers such as FEDADAM and SGD at the client is a simple competitive approach before moving on to more complex methods.

3. When there is public data to pre-train a model, the impact of heterogeneity can be reduced. Thus, when focusing on heterogeneity, it might be more productive for the research community to focus on problems that pre-trained models cannot improve (e.g., FL-powered recommendation systems or semi-supervised learning).

7 Conclusion and Limitations

Limitations. Depending on the application, it may not be possible to get public data, in which case random initialization may be the only option. Nevertheless, we believe there is sufficient prevalence and importance of applications where public data is available for this study to be of broad interest. When public data is available, it may not necessarily reflect the distribution of all users in the population. Consequently, pre-training using public data may introduce bias, which warrants further study, including methods to detect and mitigate such bias. Moreover, we only consider one warm-start initialization strategy type, supervised pre-training. Several other possibilities may be natural, including meta-learning the warm-start initialization and self-supervised pre-training (e.g., if public data does not come with labels).

Conclusion. In this paper, we present a thorough empirical analysis of initialization on federated learning by evaluating it on twelve federated learning algorithms across four vision and text tasks. We find that pre-training on public data can recover most of the accuracy drop from heterogeneity. We show that client updates starting from pre-trained weights have higher cosine similarity, which explains why initialized with pre-trained weights can speed up convergence and achieve high accuracy even in heterogeneous settings. We further show that using simple SGD locally can be as good as other local optimizers.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regret. In *International Conference on Learning Representations*, 2021.

[2] The TensorFlow Federated Authors. Tensorflow federated stack overflow dataset. 2019. URL `https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data`.

[3] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[4] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han-Wei Shen, and Wei-Lun Chao. On pre-training for federated learning. *arXiv preprint arXiv:2206.11488*, 2022.

[5] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] FLSim Authors. Federated learning simulator (flsim). 2022. URL `https://github.com/facebookresearch/FLSim`.

[9] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. *CoRR*, abs/1912.05671, 2019. URL `http://arxiv.org/abs/1912.05671`.

[10] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. *CoRR*, abs/2002.10365, 2020. URL `https://arxiv.org/abs/2002.10365`.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking ImageNet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[13] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[14] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[15] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *European Conference on Computer Vision*, pages 76–92. Springer, 2020.

[16] HuggingFace. Distilgpt2. 2019. URL `https://huggingface.co/distilgpt2`.

[17] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[18] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[19] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[20] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Shashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. 2021.

[21] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *CoRR*, abs/1712.09913, 2017. URL `http://arxiv.org/abs/1712.09913`.

[24] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[25] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: Benchmarking federated learning methods for natural language processing tasks. *arXiv preprint arXiv:2104.08815*, 2021.

[26] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2016.

[27] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2017.

[28] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[30] Krishna Pillutla, Kshitiz Malik, Abdelrahman Mohamed, Michael Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. *arXiv preprint arXiv:2204.03809*, 2022.

[31] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[32] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[33] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25:2951–2959, 2012.

[34] Joel Stremmel and Arjun Singh. Pretraining federated text models for next word prediction. In *Future of Information and Communication Conference*, pages 477–488. Springer, 2021.

[35] Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of local-update sgd algorithms. *Journal of Machine Learning Research*, 22, 2021.

[36] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[37] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[38] Orion Weller, Marc Marone, Vladimir Braverman, Dawn Lawrie, and Benjamin Van Durme. Pretrained models for multilingual federated learning. *arXiv preprint arXiv:2206.02291*, 2022.

[39] Blake E Woodworth, Kumar Kshitij Patel, and Nati Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33:6281–6292, 2020.

[40] Dong Yin, Ashwin Pananjady, Maximilian Lam, Dimitris S. Papailiopoulos, Kannan Ramchandran, and Peter L. Bartlett. Gradient diversity empowers distributed learning. *CoRR*, abs/1706.05699, 2017. URL `http://arxiv.org/abs/1706.05699`.

[41] Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. FedPD: A federated learning framework with adaptivity to non-iid data. *IEEE Transactions on Signal Processing*, 69:6055–6070, 2021.

[42] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Table 2: Dataset Statistics

Dataset	Train Clients	Eval Clients	Test Clients	Samples/Client Mean	Std
CIFAR-10	100	10	10	500	63
Stack Overflow	10,815	378	1,115	5,821	34,229
FEMNIST	3,150	350	350	272	67
Reddit	9,403	4,352	4,352	34	63

A Additional Discussion of Related Work

Transfer learning. Model initialization can significantly impact training and final performance. Previous work studying the loss landscape of deep networks observed significant differences between the landscape around a random initialization and the landscape later in training. In particular, later in training, the loss can be much more "well-behaved" [23, 10, 9]. Fine-tuning from pre-trained models is common practice in natural language processing and computer vision, yielding strong performance on many tasks [31, 7, 6, 12].

Federated Optimization. While a significant amount of research focused on various aspects of FL, including communication-efficiency [26], data and systems heterogeneity [24, 36], and faster convergence rate []. Nearly all previous work in this field neglect the importance of initialization. In our work, we study the impact of initialization on federated optimization in the cross-device setting. We defer the interested reader to surveys of Kairouz et al. [18] and Wang et al. [37] for additional background.

Pre-training in Federated Learning. Very few works have studied pre-trained models in federated learning [30, 15, 42, 25, 34]. Zhao et al. [42] studied pre-training as a mechanism to remedy the adverse effect of heterogeneity in FL. However, Zhao et al. [42] found that pre-trained initialization does not alleviate the effect of heterogeneity. In this work, we find that pre-training can alleviate both data and system heterogeneity. Pillutla et al. [30], Hsu et al. [15], Lin et al. [25], Stremmel and Singh [34] experimented with pre-trained models but did not study the difference between random initialization and pre-training, which is the focus of this work. In concurrent and independent works, Chen et al. [4], Weller et al. [38] find that pre-training closes the accuracy gap between FEDAVG and centralized learning under non-IID data. Weller et al. [38] studied multilingual language tasks using large transformer models on synthetically partitioned data, not a realistic setup for large-scale cross-device federated settings. Chen et al. [4] studied the impact of pre-training on data heterogeneity using synthetically partitioned image data. Furthermore, Chen et al. [4] proposed a method to pre-train with synthetic data. In our work, we systematically study both forms of heterogeneity, data-induced and system-induced, across both visual and language tasks on 15 SOTA federated optimization algorithms. We offer theoretical and empirical explanations for why pre-training is beneficial to FL.

B Experiment Details

B.1 Datasets and Models

CIFAR-10. We evaluate a multi-class image classification problem on CIFAR-10 [22] using a SqueezeNet [17]. We normalize the images by the dataset mean and standard deviation. Following [14], we partition the dataset using a Dirichlet distribution with parameter 0.1. The statistics on the number of clients and examples in both the training and test splits of the datasets are in Table 2.

Stack Overflow. Stack Overflow consists of questions and answers from Stack Overflow. We experiment a next-word-prediction task using a DistilGPT-2 model with a casual LM head. We perform padding and truncation to ensure that each sentences have 25 words. We then use a GPT-2 tokenizer to encode the tokens.

FEMNIST. Federated EMNIST-62 (FEMNIST) consists of digits and English characters, totaling 62 classes. We evaluate a multi-class image classification problem on the federated version [3] which partitions the digits by the writer and filter out clients that less than one example. As for the model,

Algorithm 1 FedOpt framework

Input: initial global model x^0, server and client step sizes η_s, η_c, local epochs E, rounds T
for each round $t = 1, \ldots, T$ **do**
  Server sends x^{t-1} to all clients $i \in \mathcal{S}^t$.
  for each client $i \in \mathcal{S}^t$ in parallel **do**
    Initialize local model $y_i^0 \leftarrow x^{t-1}$.
    Each client performs E epochs of local updates via $y_i^{k+1} = \textsc{ClientOpt}(y_i^k, F_i, \eta_c)$. Let y_i^E denote the result after performing E epochs of local updates.
    After local training, client i sends $\Delta_i^t = y_i^E - x^{t-1}$ to the server.
  end for
  Server computes aggregate update $\Delta^t = \frac{1}{|\mathcal{S}^t|} \sum_{i \in \mathcal{S}^t} p_i \Delta_i^t$.
  Server updates global model $x^t = \textsc{ServerOpt}(x^{t-1}, -\Delta^t, \eta_s, t)$.
end for

we use a SqueezeNet 1.0 [17]. Since FEMNIST contains grayscale images, we replicated the one channel value into three channels with the same values.

Reddit. Reddit contains preprocessed comments posted on the social network on December 2017. The dataset consists of 1,660,820 users totaling 56,587,343 tokens. Due to limited compute capabilities, we sub-sampled 9,403 users for training, 4352 for evaluation and 4352 for test. We evaluate a next-word-prediction task using a CharLM [21] model.

B.2 Implementation Details

We implemented all algorithms in Pytorch [29] and evaluated them on a cluster of machines, each with eight NVidia V100 GPUs. We evaluate our experiments in FLSim [3]. For all experiments, we tune hyperparameters using Bayesian optimization [33]. We select the best hyperparameters based the final accuracy after a fixed number of rounds for each dataset.

C Algorithms Summary

The FEDOPT framework is shown in Algorithm 1.

D Additional Results

D.1 Faster Convergence to Better Accuracy when starting from a Pre-trained Model

Figure 4 shows that, as one would hope, when starting from a pre-trained model, it is possible to achieve much better accuracy after a fixed number of steps than when starting federated training from a random initialization. Note that the initial accuracy is not always substantially higher than a random initialization, e.g., for CIFAR.

D.2 Initial Loss for Pre-trained versus Random Models

Table D.2 shows that pre-training does not always lead to lower initial loss. For Squeezenet 1.0 on CIFAR-10 and ResNet-18 on FEMNIST, the initial loss of the randomly initialized models are lower pre-trained models. However, pre-trained model still converges faster as illustrated in Figure **??**.

D.3 System Heterogeneity

In this section, we show the system heterogeneity results for all three datasets: CIFAR-10, FEMNIST, and Stack Overflow. With the exception of Stack Overflow, FEDADAM SGD consistently outperforms

[3] Our code is available at https://github.com/facebookresearch/FLSim

Table 3: Comparison of characteristics considered in previous work and the methods analyzed in this paper. Notation: *NA* = Normalized Averaging, *LS* = Non-identical Local Steps, *GM* = Global Momentum, *AS* = Adaptive Server Learning Rate, *GS* = Apply Server Optimizer State Locally.

	NA	LS	GM	AS	GS
FEDAVG NOVA	✓	✓	✗	✗	✗
FEDAVG PROXIMAL	✗	✓	✗	✗	✗
FEDAVG SGD	✗	✓	✗	✗	✗
FEDAVG GD	✗	✗	✗	✗	✗
FEDAVG MIMELITE	✗	✓	✗	✗	✓
FEDAVGM NOVA	✓	✓	✓	✗	✗
FEDAVGM PROXIMAL	✗	✓	✓	✗	✗
FEDAVGM SGD	✗	✓	✓	✗	✗
FEDAVGM GD	✗	✗	✓	✗	✗
FEDAVGM MIMELITE	✗	✓	✓	✗	✓
FEDADAM NOVA	✓	✓	✓	✓	✗
FEDADAM PROXIMAL	✗	✓	✓	✓	✗
FEDADAM SGD	✗	✓	✓	✓	✗
FEDADAM GD	✗	✗	✓	✓	✗
FEDADAM MIMELITE	✗	✓	✓	✓	✓

	CIFAR-10		FEMNIST	
	Random	Pre-trained	Random	Pre-trained
Squeezenet 1.0	1.90	2.71	4.31	3.99
ResNet-18	1.11	1.07	4.17	6.58
	Stack Overflow		**Reddit**	
	Random	Pre-trained	Random	Pre-trained
CharLM	7.71	6.78	8.61	5.15
DistilGPT2	9.82	4.93	9.99	6.34

Table 4: Loss at beginning of training for various model architecture and datasets. While pre-trained model converges faster in all cases, the initial loss of the pretrained model is not always lower than random initialization.

other methods specifically designed for system heterogeneity (NOVA, PROXIMAL) in the the pre-trained setting.

D.4 FEDADAM GD is as effective as FEDADAM SGD with pre-training.

The seminal work of McMahan et al. [26] shows that taking local SGD steps before server averaging reduces communication by 10-100$\times$ compared to taking a full batch gradient step. To understand how pre-training impacts this comparison, we compare FEDADAM with SGD and FEDADAM with GD. While local SGD can reduce communication, the saving is much less when the models are initialized with pre-trained weights compared to random weights. Figure 8 shows that with pre-trained initialization, using GD at the client can yield almost the same result as taking local SGD steps.

D.5 Fine-tuning only the last layer

In this section, we present the results for fine-tuning only the last linear layer rather in the model as commonly done in practice. Figure 6 shows that fine-tuning only the last layer might not yield optimal model quality and should be consider carefully. While fine-tuning only the last layer can achieve close to full fine-tuning on Stack Overflow, the performance is much worse on CIFAR-10.

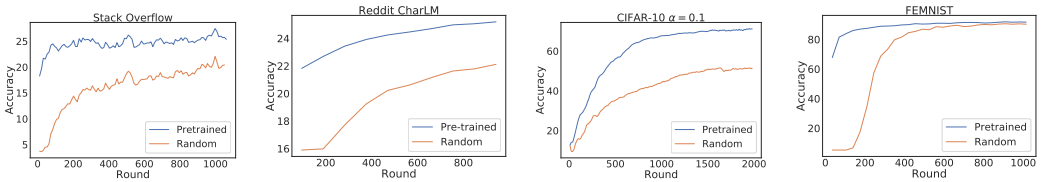

Figure 4: While prior works ignore the importance of initialization, using pre-trained models should be the first step for any practical deployment to save on communication bandwidth and achieve high model accuracy. This figure shows the advantage of using a pre-trained model for four tasks. For Stack Overflow and Reddit, we use DistilGPT2. For CIFAR-10 and FEMNIST, we use SqueezeNet.

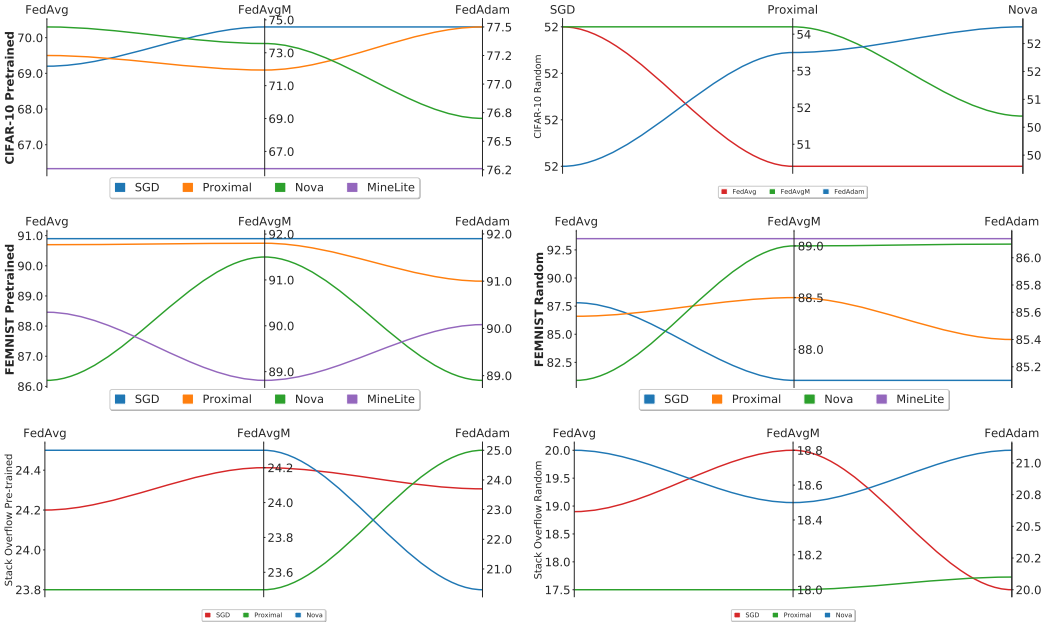

Figure 5: Average accuracy under system heterogeneity for FEDAVG, FEDAVGM and FEDADAM with NOVA, SGD and PROXIMAL as CLIENTOPT, for various tasks.

D.6 Pre-training helps align client updates.

To better understand why pre-training alleviates the heterogeneity challenge, we first investigate the gradient diversity of the updates received from different clients. We adopt the notion introduced in Yin et al. [40], adapted here to apply to client updates:

$$\text{GradientDiversity}(\{\Delta_i : i \in \mathcal{S}^t\}) = \frac{\sum_{i \in \mathcal{S}^t} ||\Delta_i||^2}{||\sum_{i \in \mathcal{S}^t} \Delta_i^2||}.$$

In Figure 7, we plot the gradient diversity of client updates Δ_i at each round for FEDADAM. In the pre-trained setting, client updates have significantly lower gradient diversity (see the bottom left plot in Figure 7). This suggests that when starting from a pre-trained model, the client local model changes are more similar to each other. On the other hand, clients local model changes from randomly initialized weights are almost orthogonal, suffering more from the client drift problem. In addition, when looking at the cosine similarity of consecutive aggregated update vectors in time (bottom middle), we see that consecutive updates point more consistently in a similar direction at the beginning of training when starting from a pre-trained model.

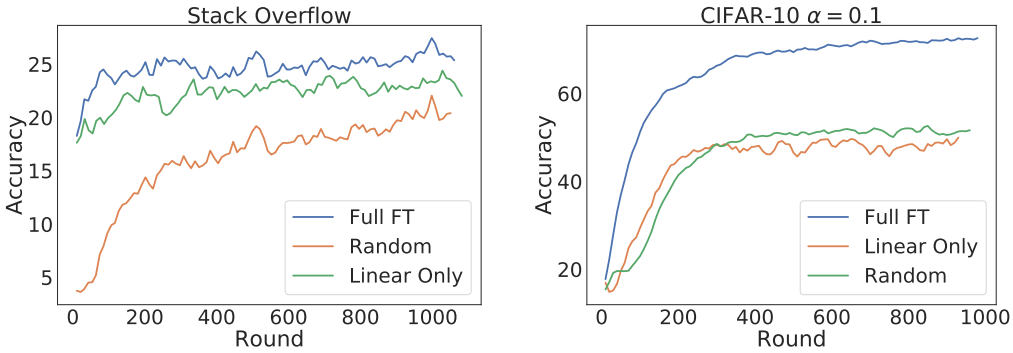

Figure 6: Average accuracy for full fine-tuning, random, and last layer only on Stack Overflow and CIFAR-10.

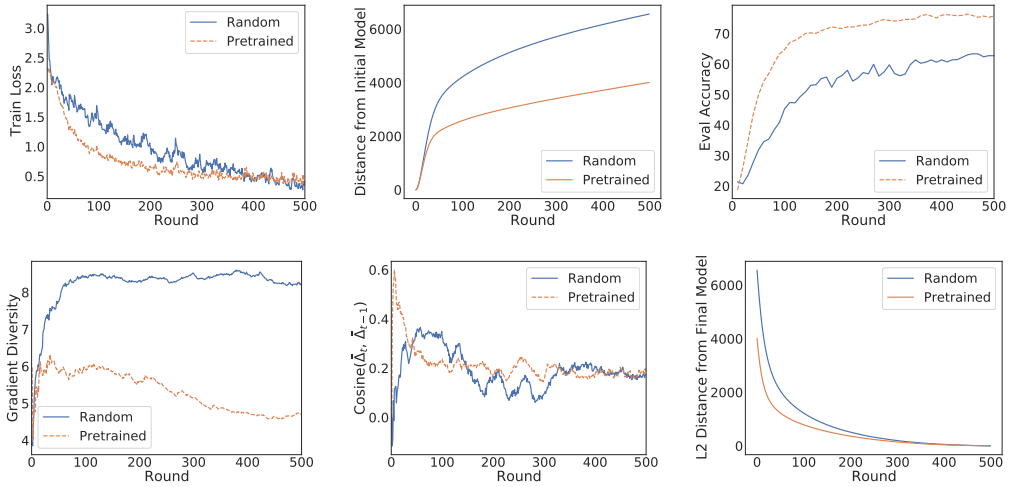

Figure 7: Training and gradient statistics of a Resnet18 on CIFAR-10 with Dirichlet distribution with parameter 0.1. Top row: Train loss of global model; train accuracy of global model; evaluation accuracy of global model; evaluation loss of global model. Bottom row: Gradient diversity of client updates; cosine similarity between client updates; L2 distance of server weights from their final values at the end of training.

In addition, from the top middle and bottom right plots in Figure 7, we see that the pre-trained model starts closer to the final result. We also examine the largest eigenvalue of the Hessian matrix (i.e., local Lipshitz constant or smoothness) at the beginning of training, a larger value of which suggests a harder-to-optimizer loss surface. In Table 1, one can observe that pre-trained models always lead to smaller eigenvalues on different datasets.

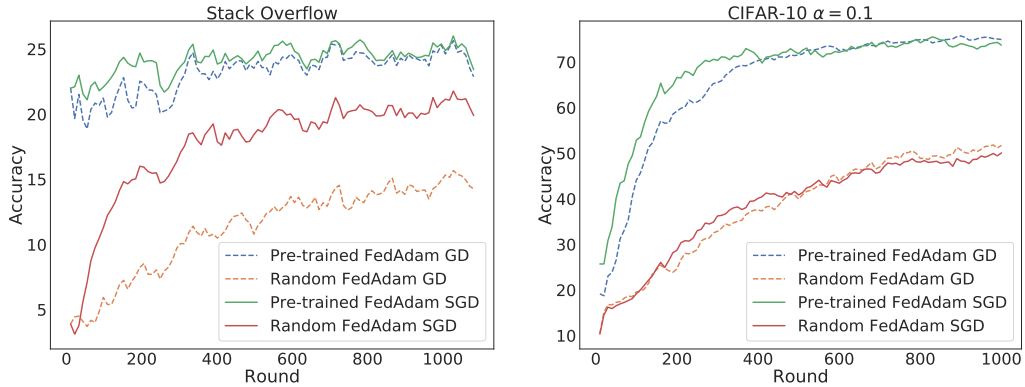

Figure 8: The average accuracy for Stack Overflow and CIFAR-10 comparing FEDADAM with SGD and FEDADAM with full-batch gradient descent (GD).