NTK-DFL: Enhancing Decentralized Federated Learning in Heterogeneous Settings via Neural Tangent Kernel

Gabriel Thompson [1] **Kai Yue** [1] **Chau-Wai Wong** [1 2] **Huaiyu Dai** [1]

Abstract

Decentralized federated learning (DFL) is a collaborative machine learning framework for training a model across participants without a central server or raw data exchange. DFL faces challenges due to statistical heterogeneity, as participants often possess data of different distributions reflecting local environments and user behaviors. Recent work has shown that the neural tangent kernel (NTK) approach, when applied to federated learning in a centralized framework, can lead to improved performance. We propose an approach leveraging the NTK to train client models in the decentralized setting, while introducing a synergy between NTK-based evolution and model averaging. This synergy exploits inter-client model deviation and improves both accuracy and convergence in heterogeneous settings. Empirical results demonstrate that our approach consistently achieves higher accuracy than baselines in highly heterogeneous settings, where other approaches often underperform. Additionally, it reaches target performance in 4.6 times fewer communication rounds. We validate our approach across multiple datasets, network topologies, and heterogeneity settings to ensure robustness and generalization. Source code for NTK-DFL is available at https://github.com/Gabe-Thomp/ntk-dfl.

1. Introduction

Federated learning (FL) is a machine learning paradigm in which multiple clients train a global model without the explicit communication of training data. In most FL scenarios, clients communicate with a central server that performs model aggregation. In the popular federated averaging (FedAvg) algorithm (McMahan et al., 2017), clients perform multiple rounds of stochastic gradient descent (SGD) on their own local data, then send this new weight vector to a central server for aggregation. As FL gains popularity in both theoretical studies and real-world applications, numerous improvements have been made to address challenges, including communication efficiency, heterogeneous data distributions, and security concerns (Sattler et al., 2020; Li et al., 2020b; Zhu et al., 2019). To handle the performance degradation caused by data heterogeneity, many works have proposed mitigation strategies for FedAvg (Karimireddy et al., 2020; Li et al., 2020b). Notably, some researchers have introduced the neural tangent kernel (NTK), replacing the commonly-used SGD in order to improve the model convergence (Yu et al., 2022; Yue et al., 2022).

Despite these advancements, the centralized nature of traditional FL schemes introduces the possibility for client data leakage, computational bottlenecks at the server, and high communication bandwidth demand (Kairouz et al., 2021). Decentralized federated learning (DFL) has been proposed as a solution to these issues (Martínez Beltrán et al., 2023). In DFL, clients may communicate with each other along an undirected graph, where each node represents a client and each edge represents a communication channel between clients. While DFL addresses some of the issues inherent to centralized FL, both frameworks grapple with the challenge of statistical heterogeneity across clients. Although mixing data on a central server could readily resolve this issue, transmitting raw, private training data from clients introduces privacy concerns, making FL and DFL approaches good candidates to address this challenge (Yuan et al., 2024). This paper focuses on the following research question: How can we design a DFL approach that effectively addresses statistical heterogeneity?

We propose a method that exploits the NTK to evolve weights. We denote this paradigm NTK-DFL. Our approach combines the advantages of NTK-based optimization with the decentralized structure of DFL. The NTK-DFL weight evolution scheme makes use of the communication of client Jacobians, allowing for more expressive updates than traditional weight vector transmissions and improving per-

[1]Electrical and Computer Engineering, NC State University [2]Secure Computing Institute, NC State University, Raleigh, USA. Correspondence to: Chau-Wai Wong <chauwai.wong@ncsu.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

formance under heterogeneity. Complementing this NTK-based evolution, we utilize a model averaging step that exploits inter-client model deviation, creating a global model with much better generalization than any local model. We demonstrate that NTK-DFL maintains high performance even under aggressive compression measures. Through reconstruction attack studies, we also analyze how this compression affects data privacy. The contributions of this paper are threefold.

1. The proposed NTK-DFL method achieves convergence with 4.6 times fewer communication rounds than existing approaches in heterogeneous settings. To the best of our knowledge, this is the first work leveraging NTK-based weight evolution for decentralized federated training.

2. The effective synergy between NTK-based evolution and DFL demonstrates superior resilience to data heterogeneity with model averaging.

3. The NTK-DFL aggregated model exhibits robust performance across various network topologies, datasets, data distributions, and compression measures. This performance is further supported by theoretical bounds demonstrating improved convergence rates.

2. Related Work

Federated Learning (FL) FL was introduced by McMahan et al. (2017) as a machine learning approach that enables training a model on distributed datasets without sharing raw data. It attempts to address key issues such as data privacy, training on decentralized data, and data compliance for more heavily regulated data (e.g., medical imaging) (Zhang et al., 2021). Despite its advantages, the centralized topology of FL introduces several challenges. These include potential privacy risks at the central server, scalability issues due to computational bottlenecks, and high communication overhead from frequent model updates between clients and the server (Mothukuri et al., 2021).

Decentralized Federated Learning (DFL) DFL aims to eliminate the need for a central server by connecting clients in a fully decentralized topology. Sun et al. (2021) adapted the FedAvg approach of multiple local SGD iterations to the decentralized setting. Our NTK-DFL method may be viewed as building on this foundation, using the neural tangent kernel for more effective weight updates. Dai et al. (2022) proposed a method of DFL where each client possesses their own sparse mask personalized to their specific data distribution. Shi et al. (2023) employed the sharpness-aware minimization optimizer to reduce the inconsistency of local models, whereas we tackle this issue through per-round averaging and final model aggregation. DFL approaches can aim to train one global model, such as the case of many hospitals training a model for tumor classification with local, confidential images (Shiri et al., 2022). They may also aim to train a personalized model for each client to perform better on the local data distribution. For example, different groups of smartphone users may use different emojis and would benefit from a personalized model (Tan et al., 2023). Our method focuses on training a high-performing global model that generalizes well across all clients, offering improved convergence and resilience to data heterogeneity compared to existing DFL methods.

Neural Tangent Kernel (NTK) NTK has primarily been used for the analysis of neural networks (Golikov et al., 2022), though it has recently seen use in the training of neural networks for FL (Yue et al., 2022). Introduced by Jacot et al. (2018), it shows that the evolution of an infinitely wide neural network converges to a kernelized model. This approach has enabled the analytical study of models that are well approximated by this infinite width limit (Liu et al., 2020). NTK has also been extended to other model types, such as the recurrent neural network (Alemohammad et al., 2021) and convolutional neural network (Arora et al., 2019). We instead use the linearized model of the NTK approximation as a tool for weight evolution. Some studies have explored the integration of NTKs with FL. For instance, Huang et al. (2021) applied the NTK analysis framework to study the convergence properties of FedAvg, while Yu et al. (2022) extended NTK applications beyond theoretical analysis by training a convex neural network. Moreover, Yue et al. (2022) replaced traditional SGD-based optimization with NTK-based evolution in a federated setting, where clients transmit Jacobian matrices to a central server that performs weight updates using NTK.

3. Proposed Method: NTK-Based Decentralized Federated Learning

3.1. Problem Statement

We begin with a brief overview of centralized FL. The goal of centralized FL is to train a global model $\boldsymbol{w}$ across M clients with their private, local data $\mathcal{D}_i = \{(\boldsymbol{x}_{i,j}, \boldsymbol{y}_{i,j})\}_{j=1}^{N_i}$, where N_i is the number of training examples of the ith client. FL algorithms aim to numerically solve the sample-wise optimization problem of $\min_{\boldsymbol{w}} F(\boldsymbol{w})$, where $F(\boldsymbol{w}) = \frac{1}{M}\sum_{i=1}^{M} N_i F_i(\boldsymbol{w})$ and $F_i(\boldsymbol{w}) = \frac{1}{N_i}\sum_{j=1}^{N_i} \ell(\boldsymbol{w}; \boldsymbol{x}_{i,j}, \boldsymbol{y}_{i,j})$.

In the decentralized setting, an omnipresent global weight $\boldsymbol{w}$ is not available to clients in each communication round. Rather, each client possesses their own model $\boldsymbol{w}_i$ that is trained in the update process. Following related DFL work (Shi et al., 2023; Sun et al., 2021), we seek a global model $\boldsymbol{w}$ that benefits from the heterogeneous data stored locally across clients and generalizes better than any indi-

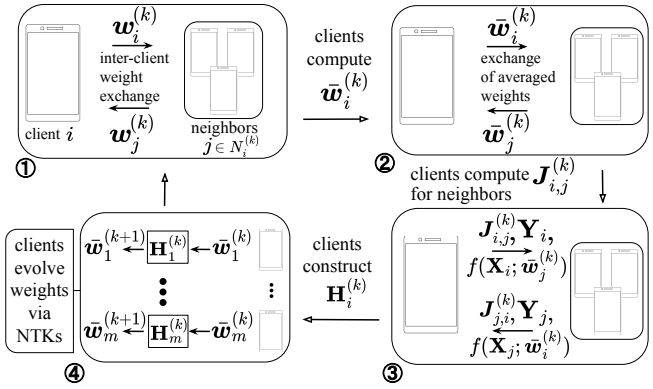


Figure 1. NTK-DFL process: ① Clients exchange weights, ② Average weights with neighbors, ③ Compute and exchange Jacobians, labels, and function evaluations, ④ Construct local NTK and evolve weights [Eq. (5a)]. This decentralized approach enables direct client collaboration and NTK-driven model evolution without a central server.

vidual client model $\boldsymbol{w}_i$. A global or aggregated model may take the form $\boldsymbol{w} = \frac{1}{N}\sum_{i=1}^{M} N_i \boldsymbol{w}_i$, where $N = \sum_{i=1}^{M} N_i$.

Notation Formally, we have a set of clients $\mathcal{C} = \{1, \ldots, i, \ldots, M\}$. Each client is initialized with its weight $\boldsymbol{w}_i^{(0)} \in \mathbb{R}^d$, where d is the size of the parameter vector and the superscript in $\boldsymbol{w}_i^{(0)}$ denotes the initial communication round. Model training is done in a series of communication rounds denoted $k \in \{1, 2, ..., K\}$. Let the graph at round k be $\mathcal{G}^{(k)} = (\mathcal{C}, E^{(k)})$, where $E^{(k)}$ is the set of edges representing connections between clients. Furthermore, the neighborhood of client i at round k is denoted $\mathcal{N}_i^{(k)} = \{j \mid (i,j) \in E^{(k)}\}$. This graph is specified before each communication round and can take an arbitrary form.

3.2. Proposed NTK-DFL

Figure 1 illustrates the proposed NTK-DFL method. We describe its key components below.

Per-round Parameter Averaging At the beginning of each communication round k, each client i both sends and receives weights—client i sends its model $\boldsymbol{w}_i^{(k)}$ to all neighbors $j \in \mathcal{N}_i^{(k)}$ and receives $\boldsymbol{w}_j^{(k)}$ from all neighbors. Each client then aggregates its own weights with its neighbors' weights to form a new averaged weight as follows:

$$\bar{\boldsymbol{w}}_i^{(k)} = \frac{1}{N_i + \sum_{j \in \mathcal{N}_i^{(k)}} N_j} \Big(N_i \boldsymbol{w}_i^{(k)} + \sum_{j \in \mathcal{N}_i^{(k)}} N_j \boldsymbol{w}_j^{(k)} \Big). \quad (1)$$

The client must then send this aggregated weight $\bar{\boldsymbol{w}}_i^{(k)}$ back to all neighbors. This step enables each client to construct a local NTK, comprised of inner products of Jacobians from both neighboring clients and its own Jacobian. See Algorithm 2 in Appendix A for details.

Local Jacobian Computation At this point, each client possesses its own aggregated weight $\bar{\boldsymbol{w}}_i^{(k)}$ as well as a set of aggregated weights $\bar{\boldsymbol{w}}_j^{(k)}$ from each of their neighbors $j \in \mathcal{N}_i^{(k)}$. The ith client computes the Jacobian of $f(\mathbf{X}_i; \bar{\boldsymbol{w}}_j^{(k)})$ with respect to the neighboring model parameters $\bar{\boldsymbol{w}}_j{}^{(k)}$ using its local data $\mathbf{X}_i$. We denote this neighbor-specific Jacobian as

$$\boldsymbol{J}_{i,j}^{(k)} \equiv [\nabla_{\boldsymbol{w}} f(\mathbf{X}_i; \bar{\boldsymbol{w}}_j^{(k)})]^\top. \quad (2)$$

Each client sends every neighbor their respective Jacobian $\boldsymbol{J}_{i,j}^{(k)}$, true label $\mathbf{Y}_i$, and function evaluation $f(\mathbf{X}_i; \bar{\boldsymbol{w}}_j^{(k)})$. Note the order of the indices in the Jacobian: the ith client sends $\boldsymbol{J}_{i,j}^{(k)}$, an evaluation on the ith client's data and its neighbors' weights. In contrast, the ith client receives $\boldsymbol{J}_{j,i}^{(k)}$ from each of its neighbors, an evaluation on its neighbors' data and the ith client's weights. Algorithm 3 in Appendix A describes this process.

Weight Evolution After all inter-client communication is completed, the clients begin the weight evolution phase of the round (see Algorithm 4 in Appendix A). Here, all clients act in parallel as computational nodes. Each client possesses their own Jacobian tensor $\boldsymbol{J}_{i,i}^{(k)}$ as well as their neighboring Jacobian tensors $\boldsymbol{J}_{j,i}^{(k)}$ for each $j \in \mathcal{N}_i^{(k)}$.

We denote the tensor of all Jacobian matrices possessed by the ith client at round k as $\boldsymbol{\mathcal{J}}_i^{(k)}$, which is composed of matrices from the set $\{\boldsymbol{J}_{i,i}^{(k)}\} \cup \{\boldsymbol{J}_{j,i}^{(k)} \mid j \in \mathcal{N}_i^{(k)}\}$ stacked along the third dimension. We denote the matrix of true labels and function evaluations stacked in the same manner as $\boldsymbol{\mathcal{Y}}_i$ and $\boldsymbol{f}(\boldsymbol{\mathcal{X}}_i)$, respectively. Explicitly, we have $\boldsymbol{\mathcal{J}}_i^{(k)} \in \mathbb{R}^{\tilde{N}_i \times d_2 \times d}$, $\boldsymbol{\mathcal{Y}}_i^{(k)} \in \mathbb{R}^{\tilde{N}_i \times d_2}$, and $\boldsymbol{f}(\boldsymbol{\mathcal{X}}_i) \in \mathbb{R}^{\tilde{N}_i \times d_2}$. Additionally, $\tilde{N}_i = N_i + \sum_{j \in \mathcal{N}_i^{(k)}} N_j$ represents the total number of data points between client i and its neighbors, and d_2 is the output dimension.

From here, each client performs the following operations to evolve its weights. First, compute the local NTK $\mathbf{H}_i^{(k)}$ from the Jacobian tensor $\boldsymbol{\mathcal{J}}_i^{(k)}$ using the definition of NTK:

$$[\mathbf{H}_i^{(k)}]_{m,n} = \frac{1}{d_2} \langle \boldsymbol{\mathcal{J}}_i^{(k)}(\boldsymbol{x}_m), \boldsymbol{\mathcal{J}}_i^{(k)}(\boldsymbol{x}_n) \rangle_{\mathrm{F}}. \quad (3)$$

Each element of the NTK is a pairwise Frobenius inner product between Jacobian matrices, where the indices m and n correspond to the mth and nth data points, respectively. Second, using $\mathbf{H}_i^{(k)}$, the client evolves their weights as follows (see Appendix B for more details):

$$\begin{aligned} \boldsymbol{f}^{(k,t)}(\boldsymbol{\mathcal{X}}_i) = \big(\mathbf{I} - e^{-\frac{\eta t}{\tilde{N}_i}\mathbf{H}_i^{(k)}}\big)\boldsymbol{\mathcal{Y}}_i^{(k)} \\ + \; e^{-\frac{\eta t}{\tilde{N}_i}\mathbf{H}_i^{(k)}} \; \boldsymbol{f}^{(k)}(\boldsymbol{\mathcal{X}}_i). \end{aligned} \quad (4)$$

We unroll gradient steps to find the weight $\boldsymbol{w}_i^{(k,t)}$ as follows:

$$\boldsymbol{w}_i^{(k,t)} = \sum_{j=1}^{d_2} (\boldsymbol{\mathcal{J}}_{i,:j:}^{(k)})^\top \mathbf{R}_{i,:j}^{(k,t)} + \bar{\boldsymbol{w}}_i^{(k)}, \tag{5a}$$

$$\mathbf{R}_{i,:j}^{(k,t)} \equiv \frac{\eta}{\tilde{N}_i d_2} \sum_{u=0}^{t-1} [\boldsymbol{\mathcal{Y}}_i^{(k)} - \boldsymbol{f}^{(k,u)}(\boldsymbol{\mathcal{X}}_i)]. \tag{5b}$$

Third, the client selects the weight $\boldsymbol{w}_i^{(k,t)}$ for a timestep t with the lowest loss according to the evolved residual $\boldsymbol{f}^{(k,t)}(\boldsymbol{\mathcal{X}}_i) - \boldsymbol{\mathcal{Y}}_i$. This is used as the new weight $\boldsymbol{w}_i^{(k+1,0)}$ for the next communication round.

Final Model Averaging Throughout the paper, we study the convergence of the final averaged model $\bar{\boldsymbol{w}}^{(k)} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{w}_i^{(k)}$. In the decentralized setting, clients would average all K client models after all training is completed. This may be done through a fully-connected topology, sequential averaging on a ring topology, or in a secure, centralized manner. The average may also be carried out over a subset of clients. In practice, we observe that the aggregated model $\bar{\boldsymbol{w}}^{(k)}$ generally performs better than any individual client model $\boldsymbol{w}_i^{(k)}$. We study the impact of the order of client averaging on model performance with a client selection algorithm and show the results in Figure 6. Each client that opts in to model averaging contributes a portion of its data to a global validation set before training begins. Our client selection algorithm selects clients in the order of their accuracy on the validation set. We will demonstrate that in the practical setting, with a proper selection of clients, not all nodes must opt into final model averaging in order for the aggregated model to benefit from improved convergence. We note a difference between *model consensus*, often discussed in the DFL literature (Savazzi et al., 2020; Liu et al., 2022), and the proposed final model averaging approach. Model consensus refers to the eventual convergence of all client models to a single, unified model over numerous communication rounds. In contrast, our approach implements final model averaging as a distinct step performed after the completion of the training process.

Lastly, while communication overhead and memory efficiency are not the primary focus of this paper, we briefly note a technique to address potential memory constraints in NTK-DFL implementations. For scenarios involving dense networks or large datasets, we introduce Jacobian batching. This approach allows clients to process their local datasets in smaller batches, reducing memory complexity from $O(N_i d_2 d)$ to $O(N_i d_2 d/m_1)$, where m_1 is the number of batches. We also study communication efficiency, where NTK-DFL is resilient to compression measures such as top-k sparsification and random projections. This enables significant reductions in communication costs without significantly compromising convergence. A thorough discussion of overhead can be found in Appendices D and E.

We also provide results on a reconstruction attack performed using client Jacobians in Appendix F.

4. Theoretical Analysis

In this section, we derive a convergence bound for NTK-DFL by analyzing the behavior of the average client weight $\bar{\boldsymbol{w}}^{(k)} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{w}_i^{(k)}$, similar to Sun et al. (2021); Shi et al. (2023). Unlike DFedAvg, the NTK-DFL bound includes a key additional dependence on the number of local iterations T in its main term. The bound captures the unique advantage that NTK-DFL gains by using much larger T values compared to other methods, improving convergence. The analysis also captures key factors such as the relationship between the spectral gap and model convergence, impact of data heterogeneity, and NTK approximation error.

Definition 4.1. (DFL objective). We are interested in minimizing the global loss $\mathcal{L}(\boldsymbol{w})$ across all clients, defined as

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{M} \frac{N_i}{N_{\text{total}}} \mathcal{L}_i(\boldsymbol{w}), \;\; \mathcal{L}_i(\boldsymbol{w}) = \frac{1}{N_i} \sum_{j=1}^{N_i} \ell(\boldsymbol{w}; \boldsymbol{x}_{i,j}, \mathbf{y}_{i,j}), \tag{6}$$

where $\ell(\cdot)$ is a sample-wise loss applied to client data.

Definition 4.2. (The gossip/mixing matrix). [Definition 1, (Sun et al., 2021)] The matrix $\mathbf{M} = [m_{i,j}] \in [0,1]^{m \times m}$ is assumed to satisfy the following properties: (i) (Graph) If $i \neq j$ and $(i,j) \notin \mathcal{V}$, then $m_{i,j} = 0$; otherwise, $m_{i,j} > 0$. (ii) (Symmetry) The matrix is symmetric, i.e., $\mathbf{M} = \mathbf{M}^\top$. (iii) (Null space property) The null space of $\mathbf{I} - \mathbf{M}$ is spanned by the all-ones vector, $\text{null}(\mathbf{I} - \mathbf{M}) = \text{span}\{\mathbb{1}\}$. (iv) (Spectral property) The matrix satisfies $\mathbf{I} \succeq \mathbf{M} \succ -\mathbf{I}$. The eigenvalues of $\mathbf{M}$ satisfy $1 = |\lambda_1(\mathbf{M})| > |\lambda_2(\mathbf{M})| \geq \cdots \geq |\lambda_m(\mathbf{M})|$, and the spectral gap is defined as $(1-\lambda)^2 \in (0,1]$, where $\lambda := \max\{|\lambda_2(\mathbf{M})|, |\lambda_m(\mathbf{M})|\}$.

Assumption 4.3. (Standard DFL assumptions). We assume that $\nabla \mathcal{L}_i$ is Lipschitz continuous. We bound the variance of client gradients with σ_g^2 and the client gradient norm with B. We note that these assumptions are relatively mild and common in the DFL literature. Additional details can be found in Appendix G.

Assumption 4.4. (Approximation error of NTK gradient). For any client $i \in \{1, 2, \ldots, M\}$, the difference between the NTK gradient and the true gradient of the loss is bounded by δ_{NTK}, i.e.,

$$\left\| \nabla \mathcal{L}_i^{\text{NTK}}(\boldsymbol{w}) - \nabla \mathcal{L}_i(\boldsymbol{w}) \right\|^2 \leq \delta_{\text{NTK}}^2, \tag{7}$$

for all $\boldsymbol{w} \in \mathbb{R}^d$. We note that an explicit formulation of δ_{NTK} is possible given assumptions of a simplified model (Huang et al., 2021; Yue et al., 2022) (see Lemma G.9 in Appendix G). In our analysis, we treat this as a constant in order to remain model-agnostic.

Theorem 4.5. *Consider the average weight over M clients $\bar{\boldsymbol{w}}^{(k)} := \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{w}_i^{(k)}$ for round $k \in \{1, \dots, K\}$ and suppose the previously stated assumptions hold. Assuming that the learning rate satisfies $0 < \eta \leq \frac{1}{8LT}$, we have*

$$\min_{1\leq k\leq K} \left\|\nabla \mathcal{L}\left(\bar{\boldsymbol{w}}^{(k)}\right)\right\|^2 \leq \frac{2\left[\mathcal{L}(\bar{\boldsymbol{w}}^{(1)}) - \mathcal{L}^*\right]}{K\gamma(T,\eta)} + \alpha(\eta, T, \sigma_g, \delta_{NTK}) + \beta(\eta, T, \sigma_g, \delta_{NTK}, \lambda), \quad (8)$$

where the constants are defined as

$$\gamma(T,\eta) := \eta T - 32\eta^2 T^2 L(\eta T L + 1), \quad (9a)$$

$$\alpha(\eta, T, \sigma_g, \delta_{NTK}) := \tfrac{1}{\gamma(T,\eta)} 2\eta T\big[L(\eta L T + 1) \cdot 8\eta T(\delta_{NTK}^2 + \sigma_g^2) + \delta_{NTK} B\big], \quad (9b)$$

$$\beta(\eta, T, \sigma_g, \delta_{NTK}, \lambda) := \tfrac{1}{\gamma(T,\eta)} 512\eta^4 T^4 L^3(\eta T L + 1) \cdot (1-\lambda)^{-2}(\delta_{NTK}^2 + \sigma_g^2 + B^2). \quad (9c)$$

Corollary 4.6. *Let the learning rate satisfy $O(1/L\sqrt{KT})$. Using similar assumptions as Theorem 4.5 in Appendix G, we have the following convergence rate for NTK-DFL:*

$$\min_{1\leq k\leq K} \left\|\nabla \mathcal{L}\left(\bar{\boldsymbol{w}}^{(k)}\right)\right\|^2 \leq O\Bigg(\frac{\mathcal{L}(\bar{\boldsymbol{w}}^{(1)}) - \mathcal{L}^*}{\sqrt{KT}} + \frac{\sqrt{T}(\delta_{NTK} B + \delta_{NTK}^2 + \sigma_g^2)}{\sqrt{K}} + \frac{T(\delta_{NTK}^2 + \sigma_g^2)}{K} + \frac{T(\delta_{NTK}^2 + \sigma_g^2 + B^2)}{(1-\lambda)^2 K}\Bigg) + O(\delta_{NTK}). \quad (10)$$

From the bound above, we can see that the main convergence term can be improved when we increase the number of local iterations, T. However, as expected, we cannot arbitrarily increase T without inducing a greater error in the terms δ_{NTK}, B, and σ_g. Compared to bounds in Sun et al. (2021), $O\Big(\frac{1}{\sqrt{K}} + \frac{\sigma_g^2}{\sqrt{K}} + \frac{\sigma_g^2 + B^2}{(1-\lambda)^2 K^{3/2}}\Big)$, the first term to the right of the inequality highlights improved convergence derived from the ability to select a value of T that is 1 to 2 orders of magnitude larger than that of other DFL methods. Intuitively, the bound tightens with an increasing spectral gap $(1-\lambda)^2$, which is associated with mixing speed. We note that there is an irreducible convergence floor from the NTK approximation term δ_{NTK}. Obtaining an explicit bound on this term can be done through specific assumptions about the loss function and the model itself. For instance, as detailed in Lemma G.9, assumptions about the smoothness of f_i and a bounded per-client residual norm lead to $\delta_{\text{NTK}} = O\big(K^{-1}T(\sigma_g^2 + B^2)\big)$, thus becoming reducible.

5. Experiments

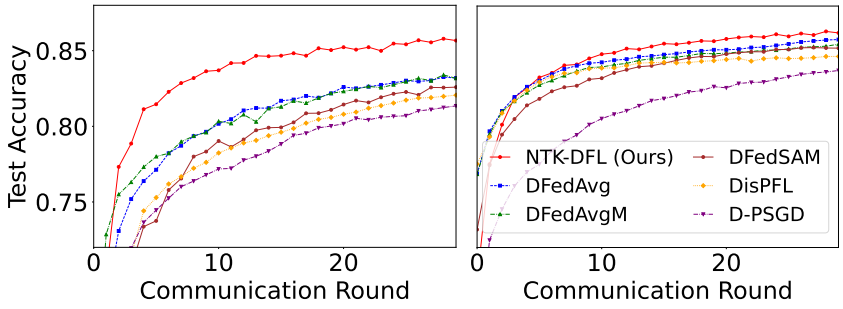
5.1. Experimental Setup

Datasets and Model Specifications Following Yue et al. (2022), we experiment on three datasets: Fashion-MNIST (Xiao et al., 2017), FEMNIST (Caldas et al., 2019), and MNIST (Lecun et al., 1998). Each dataset contains $C = 10$ output classes. For Fashion-MNIST and MNIST, data heterogeneity has been introduced in the form of non-IID partitions created by the symmetric Dirichlet distribution (Good, 1976). For each client, a vector $\boldsymbol{q}_i \sim \text{Dir}(\alpha)$ is sampled, where $\boldsymbol{q}_i \in \mathbb{R}^C$ is confined to the $(C-1)$-standard simplex such that $\sum_{j=1}^{C} q_{ij} = 1$. This assigns a probability distribution over labels to each client, creating heterogeneity in the form of label-skewness. For smaller values of α, a client possesses a distribution concentrated in fewer classes. We test over a range of α values in order to simulate different degrees of heterogeneity. In FEMNIST, data is split into shards based on the writer of each digit, introducing heterogeneity in the form of feature-skewness. For the model, we use a two-layer multilayer perceptron with a hidden width of 100 neurons for all trials.

Network Topologies A sparse, time-variant κ-regular graph with $\kappa = 5$ was used as the standard topology for experimentation, where for each communication round k, a new random graph $\mathcal{G}^{(k)}$ with the same parameter κ is created. Various values of κ were tested to observe the effect of network density on model convergence. We also experimented with various topologies to ensure robustness to different connection settings. We used a network of 300 clients throughout our experiments.

Baseline Methods We compare our approach to various state-of-the-art baselines in the DFL setting. These include D-PSGD (Lian et al., 2017), DFedAvg, DFedAvgM (Sun et al., 2021), DFedSAM (Shi et al., 2023), and DisPFL (Dai et al., 2022). We also compare with the centralized baseline NTK-FL (Yue et al., 2022). The upper bound NTK-FL would consist of a client fraction of 1.0 where the server constructs an NTK from all client data each round, which is infeasible due to memory constraints. Instead, we conducted a comparison following Dai et al. (2022), which we include in Appendix C.1. We also include details on baseline hyperparameters in Appendix C.2.

Performance Metrics We evaluate the performance of the various DFL approaches by studying the aggregate model accuracy on a global, holdout test set. This ensures that we are measuring the generalization of the aggregate model from individual, heterogeneous local data to a more representative data sample. Our approach is in line with the goal of training a global model capable of improved generalization over any single local model (Section 3.1), unlike personalized federated learning where the goal is to fine-tune a global model to each local dataset (Tan et al., 2023). When evaluating the selection algorithm in Figure 6, we split the global test set in a 50:50 ratio of validation to test data. We use the validation data to sort the models based on their accuracy, and report the test accuracy in the figure.



Method	IID	$\alpha = 0.5$	$\alpha = 0.1$
DFedAvg	18	31	83
DFedAvgM	23	43	86
DFedSAM	24	45	200+
DisPFL	43	87	200+
D-PSGD	53	79	125
NTK-DFL	**12**	**17**	**18**

Figure 2. Convergence of different methods on Fashion-MNIST for (left) highly non-IID with $\alpha = 0.1$ and (middle) IID settings. (Right) The table displays the communication rounds required to reach 85% test accuracy on Fashion-MNIST. We observe increased improvement in NTK-DFL convergence over baselines for more heterogeneous settings.

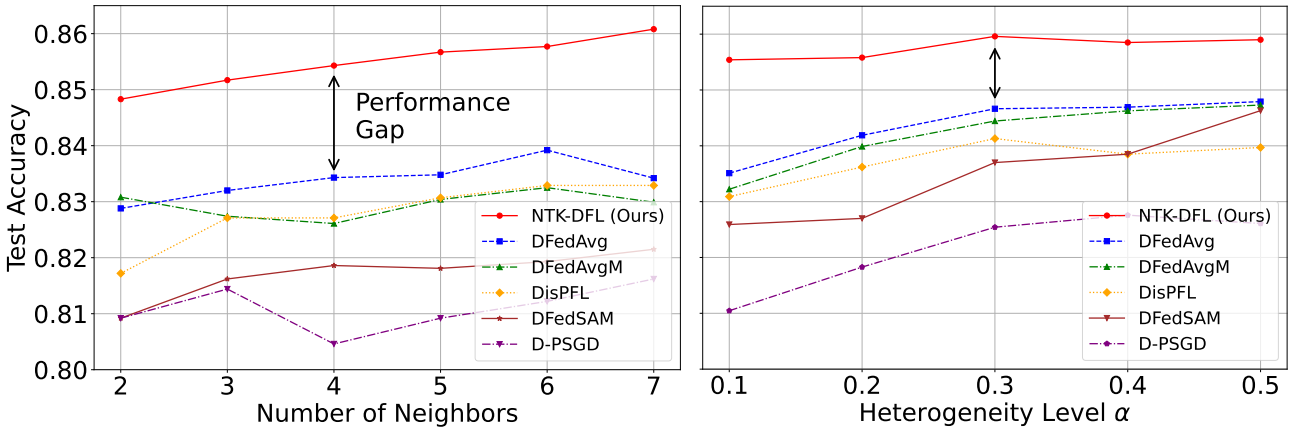


Figure 3. Performance of NTK-DFL vs. (left) sparsity level and (right) heterogeneity level (smaller $\alpha \rightarrow$ more heterogeneous). NTK-DFL outperforms the baselines and the gains are stable as the factors vary.

5.2. Experimental Results

Test Accuracy & Convergence Our experiments demonstrate the superior convergence properties of NTK-DFL compared to baselines. Figure 2 illustrates the convergence trajectories of NTK-DFL and other baselines on Fashion-MNIST. We see that NTK-DFL convergence benefits are enhanced under increased heterogeneity. Under high heterogeneity with $\alpha = 0.1$, NTK-DFL establishes a 3–4% accuracy lead over the best-performing baseline within just five communication rounds and maintains this advantage throughout the training process. Additionally presented are the number of communication rounds necessary for convergence to 85% test accuracy, where NTK-DFL consistently outperforms all baselines. For the $\alpha = 0.1$ setting, NTK-DFL achieves convergence in 4.6 times fewer communication rounds than DFedAvg, the next best performing baseline. Figure 10 in Appendix C demonstrates a similar convergence advantage for NTK-DFL on both feature-skewed FEMNIST and label-skewed MNIST datasets.

Factor Analyses for NTK-DFL We evaluate NTK-DFL's performance over various factors, including the sparsity and heterogeneity levels, the choices of the topology, and weight initialization scheme. Figure 3 illustrates the test accuracy of NTK-DFL and other baselines as functions of the sparsity and heterogeneity levels, respectively. We observe a mild increase in convergence accuracy with decreasing sparsity. NTK-DFL experiences stable convergence across heterogeneity values α ranging from 0.1 to 0.5. The left plot reveals that NTK-DFL consistently outperforms baselines by 2–3% across all sparsity levels. The right plot demonstrates NTK-DFL's resilience to data heterogeneity—while baseline methods' performance deteriorates with decreasing α, NTK-DFL maintains stable performance. In Figure 8 of Appendix C, we evaluate NTK-DFL across a range of network topologies and find that it performs consistently well across different connection structures with the same sparsity level. Additionally, Figure 11 illustrates the impact of a dynamic network topology on NTK-DFL convergence. The dynamic topology accelerates convergence compared to the static topology, likely due to improved information flow among clients. Figure 12 demonstrates the effect of weight initialization on NTK-DFL performance. While random per-client initialization slightly slows convergence compared

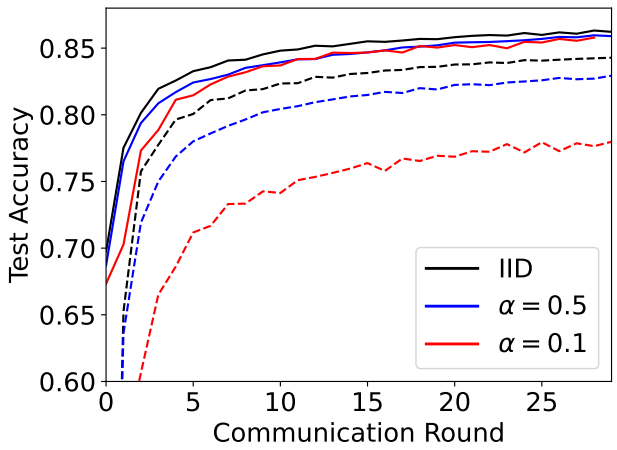


Figure 4. Performance gains of model averaging on convergence, trained on Fashion-MNIST. Solid lines correspond to the accuracy of the aggregated global model, whereas dotted lines correspond to the mean accuracy across client models. NTK-DFL's aggregated model maintains high performance, whereas mean client accuracy declines significantly with increased heterogeneity.

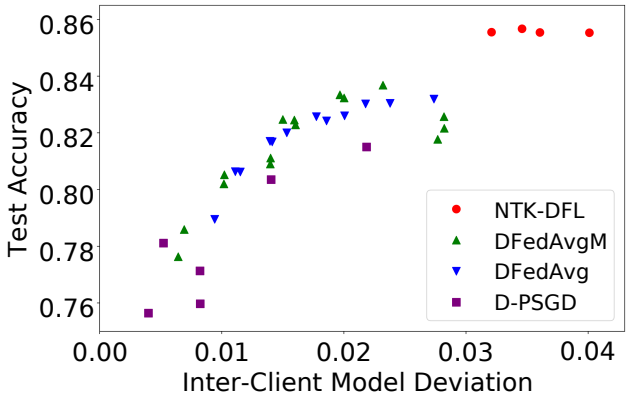


Figure 5. Relationship between inter-client model deviation and final test accuracy on Fashion-MNIST. Each point represents a trial with distinct hyperparameters. The plot reveals a positive correlation between model deviation and accuracy, suggesting that higher deviation may benefit model averaging in DFL to a certain extent. Notably, the NTK-DFL approach demonstrates both higher accuracy and greater model deviation compared to other methods.

to uniform initialization, NTK-DFL exhibits robustness to these initialization differences.

Gains Due to Final Model Aggregation Figure 4 demonstrates the dramatic effect of final model aggregation on final test accuracy. Though the individual client models decrease in accuracy as the level of heterogeneity increases, the final aggregated model remains consistent across all levels of heterogeneity (as seen in Figures 2 and 3). In the most heterogeneous setting $\alpha = 0.1$ that we tested, the difference between the mean accuracy of each client and the aggregated model accuracy is nearly 10%, as shown in Figure 4. A similar phenomenon is observed in Figure 9 of Appendix C as the client topology becomes more sparse. For the same heterogeneity setting with a sparser topology of $\kappa = 2$, the difference between these accuracies is nearly 15%. Though the individual performance of local client models may suffer under extreme conditions, the inter-client model deviation (see detailed definition in Appendix C.3) created by such unfavorable settings is exploited by model averaging to recuperate much of that lost performance. Figure 5 suggests that inter-client model deviation enhances the performance of model averaging in DFL. While extreme dissimilarity in model weights would likely result in poor performance of the averaged model, we observe that a moderate degree of deviation can be beneficial. For example, this is seen in Li et al. (2020a) with the tuning of the "proximal term" μ that regulates this degree of client diversity in model updates. We posit that the NTK-based update steps generate a more advantageous level of deviation compared to baseline approaches, contributing to improved overall performance.

Selection Algorithm Figure 6 demonstrates the results of the selection algorithm for the final model aggregation. The "best-to-worst" selection algorithm is highly effective in a highly heterogeneous setting with $\alpha = 0.1$. It significantly outperforms a random averaging order and the lower-bound averaging order, which requires the fewest clients to be averaged to achieve the same level of accuracy. In practical deployments, final model aggregation has implications in a fully decentralized setting. For example, when training starts to converge, clients may need to connect in a denser topology to shortlist neighbors with higher validation accuracies for final model aggregation.

Per-round Averaging Ablation Study In Figure 7, we perform an ablation study in which we remove the per-round parameter averaging that is a part of the NTK-DFL process. Here, clients forego the step of averaging their weight vectors with their neighbors during each communication round. Instead, clients compute Jacobians with respect to their original weight vector and send these to each of their neighbors (see Algorithm 3 in Appendix A). A massive distribution shift can be seen in the figure, where the distribution in the ablated setting is clearly skewed into lower accuracies. In contrast, NTK-DFL with per-round averaging demonstrates a much tighter distribution around a higher mean accuracy, effectively eliminating the long tail of low-performing models. Per-round averaging in NTK-DFL serves as a stabilizing mechanism against local model drift, safeguarding clients against convergence to suboptimal solutions early in the training process. In other words, client collaboration in the form of per-round averaging with neighbors ensures that no client lags behind in convergence. This is a particularly valuable feature in decentralized federated learning scenarios, where maintaining uniformity across a diverse

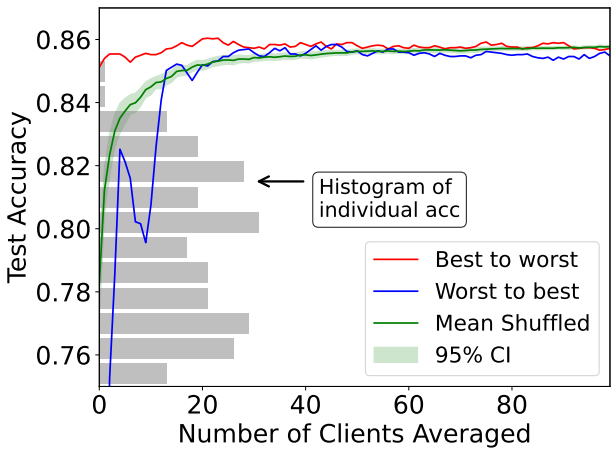


Figure 6. Final model test accuracy on Fashion-MNIST vs. the number of clients averaged for a highly heterogeneous setting with $\alpha = 0.1$. The histogram shows the distribution of individual client model accuracies. Three client selection criteria are tested: proposed high-to-low (red), random (green), and low-to-high (blue).

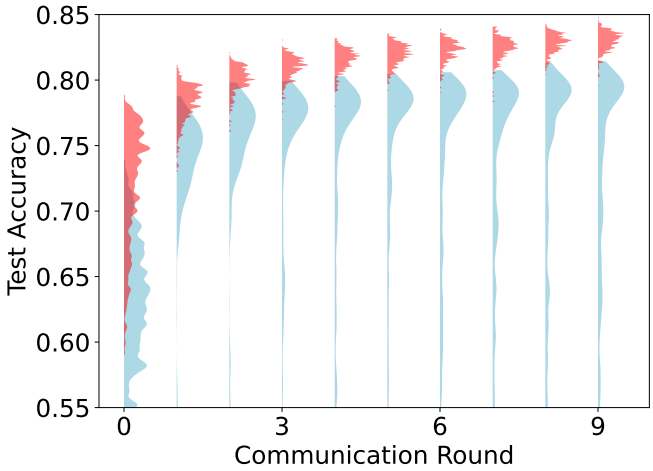


Figure 7. Distributions of individual client model accuracy vs. the communication round for Fashion-MNIST. The proposed scheme (red) conducts per-round averaging among neighbors, whereas the ablated setup (blue) does not. Per-round averaging reduces the skewness of the model performance distribution.

set of clients with heterogeneous data is a major challenge (Martínez Beltrán et al., 2023).

Communication and Memory Overhead While overhead is not the primary focus of our work, we propose several strategies to allow for control over these factors. To mitigate memory overhead, we experiment with Jacobian batching, where clients divide their data into batches each round and perform an NTK-DFL update per batch. As shown in Figure 13 of Appendix E, test accuracy per communication round improves with increasing batch size. To address communication overhead, we explore datapoint subsampling, where clients compute Jacobians with respect to only a fraction of their data each round. Figure 14 of Appendix E illustrates the expected accuracy drop from this approach—a natural trade-off for reduced communication. Finally, we evaluate compression techniques to reduce NTK-DFL's communication volume further. Figure 15 of Appendix E compares the convergence of NTK-DFL with different compression methods, while Figure 16 presents convergence in terms of bits communicated. We leave readers to Appendix E for detailed discussions.

6. Conclusion and Future Work

In this paper, we have introduced NTK-DFL, a novel approach to decentralized federated learning that leverages the neural tangent kernel to address the challenges of statistical heterogeneity in decentralized learning settings. Our work extends NTK-based training to the decentralized setting, while discovering a unique synergy between NTK evolution and decentralized model averaging that improves final model accuracy. Our method combines the expressiveness of NTK-based weight evolution with a decentralized architecture, allowing for efficient, collaborative learning without a central server. We reduce the number of communication rounds needed for convergence, which may prove advantageous for high-latency settings or those with heavy encoding/decoding costs.

There are promising unexplored directions for NTK-DFL. For instance, extending the algorithm to training models such as CNNs, ResNets (He et al., 2016), and transformers (Vaswani et al., 2017), possibly making use of new NTK methods suited for modern architectures (Arora et al., 2019; Tirer et al., 2022; Yang, 2019). Additionally, future research could explore the application of NTK-DFL to cross-silo federated learning scenarios, particularly in domains such as healthcare, where data privacy concerns and regulatory requirements often necessitate decentralized approaches (Guo et al., 2025). Lastly, NTK-DFL may serve as a useful paradigm for transfer learning applications in scenarios where a single, centralized source of both compute and data is not available.

Impact Statement

This paper presents work aimed at advancing the field of machine learning. Our research has several potential societal consequences, none of which we believe need to be specifically highlighted in this context.

Acknowledgments

This work was supported in part by the US National Science Foundation under grants SaTC-2340856 and ECCS-2203214, and the ECE Undergraduate Research Program and the Caldwell Fellows Program at the NC State University. The views expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Alemohammad, S., Wang, Z., Balestriero, R., and Baraniuk, R. The recurrent neural tangent kernel. In *International Conference on Learning Representations*, 2021.

Alistarh, D., Hoefler, T., Johansson, M., Konstantinov, N., Khirirat, S., and Renggli, C. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, 2019.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., and Talwalkar, A. LEAF: A benchmark for federated settings. In *Advances in Neural Information Processing Systems*, 2019.

Dai, R., Shen, L., He, F., Tian, X., and Tao, D. DisPFL: Towards communication-efficient personalized federated learning via decentralized sparse training. In *International Conference on Machine Learning*, 2022.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Golikov, E., Pokonechnyy, E., and Korviakov, V. Neural tangent kernel: A survey, 2022.

Good, I. J. On the application of symmetric Dirichlet distributions and their mixtures to contingency tables. *The Annals of Statistics*, 4(6), 1976.

Guo, P., Wang, P., Zhou, J., Jiang, S., and Patel, V. M. Enhancing MRI reconstruction with cross-silo federated learning. In Li, X., Xu, Z., and Fu, H. (eds.), *Federated Learning for Medical Imaging*, The MICCAI Society book Series, pp. 155–171. Academic Press, 2025.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Huang, B., Li, X., Song, Z., and Yang, X. FL-NTK: A neural tangent kernel-based framework for federated learning analysis. In *International Conference on Machine Learning*, pp. 4423–4434. PMLR, 2021.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pp. 8580–8589, Red Hook, NY, USA, 2018.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14:1–210, 2021.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *3rd MLSys Conference*, 2020a.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020b.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Liu, C., Zhu, L., and Belkin, M. On the linearity of large non-linear models: When and why the tangent kernel is constant. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15954–15964, 2020.

Liu, W., Chen, L., and Zhang, W. Decentralized federated learning: Balancing communication and computing costs. *IEEE Transactions on Signal and Information Processing over Networks*, 8:131–143, 2022.

Martínez Beltrán, E. T., Pérez, M. Q., Sánchez, P. M. S., Bernal, S. L., Bovet, G., Pérez, M. G., Pérez, G. M., and Celdrán, A. H. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys; Tutorials*, 25 (4):2983–3013, 2023.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3400–3413, 2020.

Savazzi, S., Nicoli, M., and Rampa, V. Federated learning with cooperating devices: A consensus approach for massive IoT networks. *IEEE Internet of Things Journal*, 7(5): 4641–4654, 2020.

Shi, Y., Shen, L., Wei, K., Sun, Y., Yuan, B., Wang, X., and Tao, D. Improving the model consistency of decentralized federated learning. In *International Conference on Machine Learning*, 2023.

Shiri, I., Vafaei Sadr, A., Amini, M., Salimi, Y., Sanaat, A., Akhavanallaf, A., Razeghi, B., Ferdowsi, S., Saberi, A., Arabi, H., Becker, M., Voloshynovskiy, S., z, D., Rahmim, A., and Zaidi, H. Decentralized distributed multi-institutional PET image segmentation using a federated deep learning framework. *Clin Nucl Med*, 47(7): 606–617, Jul 2022.

Sun, T., Li, D., and Wang, B. Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4289–4301, 2021.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):9587–9603, 2023.

Tirer, T., Bruna, J., and Giryes, R. Kernel-based smoothness analysis of residual networks. In *2nd Mathematical and Scientific Machine Learning Conference*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms, 2017.

Yang, G. Wide feedforward or recurrent neural networks of any architecture are Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yu, Y., Wei, A., Karimireddy, S. P., Ma, Y., and Jordan, M. TCT: Convexifying federated learning using bootstrapped neural tangent kernels. In *Advances in Neural Information Processing Systems*, 2022.

Yuan, L., Wang, Z., Sun, L., Yu, P. S., and Brinton, C. G. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 2024.

Yue, K., Jin, R., Pilgrim, R., Wong, C.-W., Baron, D., and Dai, H. Neural tangent kernel empowered federated learning. In *International Conference on Machine Learning*, 2022.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., and Gao, Y. A survey on federated learning. *Knowledge-Based Systems*, pp. 106775, 2021.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

A. NTK-DFL Algorithms

Algorithm 1 Consolidated Federated Learning Process

Require: A set of clients $\mathcal{C}$

Initialize weights $\boldsymbol{w}_i^{(0)}$ for each client i.
for each communication round $k = 1$ to K **do**
  Initialize graph structure $G^{(k)} = (\mathcal{C}, E^{(k)})$, specifying the neighbors $\mathcal{N}_i^{(k)}$ for each client i.
  Execute Algorithm 2 for Per-Round Parameter Averaging
  Execute Algorithm 3 for Local Jacobian Computation and Sending
  Execute Algorithm 4 for Weight Evolution
end for

Algorithm 2 Per-Round Parameter Averaging

Require: For each client i, a set of neighbors $\mathcal{N}_i^{(k)}$ and initial weights $\boldsymbol{w}_i^{(k)}$

for each client $i \in \mathcal{C}$ **in parallel do**
  Send $\boldsymbol{w}_i^{(k)}$ to all neighbors $j \in \mathcal{N}_i^{(k)}$
  Receive $\boldsymbol{w}_j^{(k)}$ from all neighbors $j \in \mathcal{N}_i^{(k)}$
  $\bar{\boldsymbol{w}}_i^{(k)} \leftarrow \frac{1}{N_i + \sum_{j \in \mathcal{N}_i^{(k)}} N_j} \left(N_i \boldsymbol{w}_i^{(k)} + \sum_{j \in \mathcal{N}_i^{(k)}} N_j \boldsymbol{w}_j^{(k)} \right)$
  Send aggregated weight $\bar{\boldsymbol{w}}_i^{(k)}$ back to all neighbors $j \in \mathcal{N}_i^{(k)}$
end for

Algorithm 3 Local Jacobian Computation and Sending Jacobians

Require: Each client i knows its neighbors $\mathcal{N}_i^{(k)}$ and has access to local data $\mathbf{X}_i$ and the aggregated weights $\bar{\boldsymbol{w}}_j^{(k)}$ from each neighbor $j \in \mathcal{N}_i^{(k)}$.

for each client $i \in \mathcal{C}$ **in parallel do**
  Compute the Jacobian $\boldsymbol{J}_{i,i}^{(k)} \equiv \nabla_{\boldsymbol{w}} f(\mathbf{X}_i; \bar{\boldsymbol{w}}_i^{(k)})$ using the client's own aggregated weight $\bar{\boldsymbol{w}}_i^{(k)}$ and local data $\mathbf{X}_i$.
  for each neighbor $j \in \mathcal{N}_i^{(k)}$ **do**
    Compute the Jacobian $\boldsymbol{J}_{i,j}^{(k)} \equiv \nabla_{\boldsymbol{w}} f(\mathbf{X}_i; \bar{\boldsymbol{w}}_j^{(k)})$ using the neighbor's aggregated weight $\bar{\boldsymbol{w}}_j^{(k)}$ and client's local data $\mathbf{X}_i$.
    Send $\boldsymbol{J}_{i,j}^{(k)}$, true label $\mathbf{Y}_i$, and function evaluation $f(\mathbf{X}_i; \bar{\boldsymbol{w}}_j^{(k)})$ to neighbor j.
  end for
end for

B. Additional Details on Weight Evolution

In implementation, computing the matrix exponential $e^{-\frac{\eta t}{N_i}\mathbf{H}_i^{(k)}}$ in (4) to evolve weights can be computationally expensive. In practice, the weights are evolved according to the more general differential equation from which (4) is derived, reliant upon the linearized model approximation $\boldsymbol{f}(\boldsymbol{\mathcal{X}}_i; \bar{\boldsymbol{w}}_j^{(k,t)}) \approx \boldsymbol{f}(\boldsymbol{\mathcal{X}}_i; \bar{\boldsymbol{w}}_j^{(k,0)}) + \nabla_{\boldsymbol{w}} \boldsymbol{f}(\boldsymbol{\mathcal{X}}_i; \bar{\boldsymbol{w}}_j^{(k,0)})^\top (\bar{\boldsymbol{w}}_j^{(k,t)} - \bar{\boldsymbol{w}}_j^{(k,0)})$. The differential equation is as follows

$$\frac{d}{dt} \boldsymbol{f}(\boldsymbol{\mathcal{X}}_i; \bar{\boldsymbol{w}}_j^{(k,t)}) = -\eta \mathbf{H}_j^{(k)} \nabla_{\boldsymbol{f}} \mathcal{L}. \tag{11}$$

Here, $\mathcal{L}$ is the loss function. For example, for a half mean-squared error (MSE) loss, term on the right becomes the residual matrix $\nabla_{\boldsymbol{f}} \mathcal{L} = \boldsymbol{f}(\boldsymbol{\mathcal{X}}_i; \bar{\boldsymbol{w}}_j^{(k,t)}) - \boldsymbol{\mathcal{Y}}_i$. During weight evolution, a client j evolves their neighboring function evaluation from the initial condition $\boldsymbol{f}(\boldsymbol{\mathcal{X}}_i; \bar{\boldsymbol{w}}_j^{(k,0)})$ to the time-evolved $\boldsymbol{f}(\boldsymbol{\mathcal{X}}_i; \bar{\boldsymbol{w}}_j^{(k,t)})$ using a differential equation solver and the differential equation above. To implement (5a), we use a process similar to Yue et al. (2022) where the initial client residual is evolved over a series of timesteps specified by the user. For user-specified timesteps, the loss at that time is found using the evolved residual. Then, the best-performing weights are evolved using the left side of (5a) and selected for the next communication round.

Algorithm 4 Weight Evolution

Require: Each client i has access to local data $\mathbf{X}_i$ and initial weights $\bar{\boldsymbol{w}}_i^{(k)}$, and knows its neighbors $\mathcal{N}_i^{(k)}$

for each client $i \in \mathcal{C}$ after intra-client communication **do**
  Compute local Jacobian tensor $\boldsymbol{J}_{i,i}^{(k)}$ and receive $\boldsymbol{J}_{j,i}^{(k)}$ from each neighbor j
  Construct tensor $\boldsymbol{\mathcal{J}}_i^{(k)}$ from $\{\boldsymbol{J}_{ii}^{(k)}\} \cup \{\boldsymbol{J}_{ji}^{(k)} \mid j \in \mathcal{N}_i^{(k)}\}$
  Compute local NTK $\mathbf{H}_i^{(k)}$ using $\boldsymbol{\mathcal{J}}_i^{(k)}$:
  for each data point pair (x_m, x_n) **do**
    $[\mathbf{H}_i^{(k)}]_{m,n} \leftarrow \frac{1}{d_2}\langle \boldsymbol{\mathcal{J}}_i^{(k)}(x_m), \boldsymbol{\mathcal{J}}_i^{(k)}(x_n)\rangle_F$
  end for
  for each timestep $t = 1$ to T **do**
    $\boldsymbol{f}^{(k,t)}(\boldsymbol{\mathcal{X}}_i) \leftarrow (\mathbf{I} - e^{-\frac{\eta t}{N_i}\mathbf{H}_i^{(k)}})\boldsymbol{\mathcal{Y}}_i^{(k)} + e^{-\frac{\eta t}{N_i}\mathbf{H}_i^{(k)}}\boldsymbol{f}^{(k)}(\boldsymbol{\mathcal{X}}_i)$
    $\bar{\boldsymbol{w}}_i^{(k,t)} \leftarrow \sum_{j=1}^{d_2}(\boldsymbol{\mathcal{J}}_{i,:j:}^{(k)})^T \boldsymbol{R}_{i,:j}^{(k,t)} + \bar{\boldsymbol{w}}_i^{(k)}$
  end for
  Select $\boldsymbol{w}_i^{(k+1,0)} \leftarrow \bar{\boldsymbol{w}}_i^{(k,t)}$ with the lowest loss given the residual $\boldsymbol{f}^{(k,t)}(\boldsymbol{\mathcal{X}}_i) - \boldsymbol{\mathcal{Y}}_i$
end for

C. Additional Experimental Details

C.1. Baselines

NTK-FL is the only centralized baseline that we compare with. We choose a per-round client fraction that ensures that the busiest node in the centralized setting is no busier than the busiest decentralized setting. By busier, we mean the degree of the node or the number of clients communicating with it. We note that NTK-FL is not an upper bound in this case due to the comparison being founded on node busyness, which disadvantages a centralized approach where all communication happens through a single, centralized node. Evaluating NTK-FL in the same setting as the table in Figure 2, NTK-FL converges to threshold accuracy in 73, 85, and 180 communication rounds for heterogeneity settings IID, $\alpha = 0.5$, and $\alpha = 0.1$, respectively. D-PSGD (Lian et al., 2017) is one of the first decentralized, parallel algorithms for distributed machine learning that allows nodes to only communicate with neighbors. DFedAvg (Sun et al., 2021) adapts FedAvg to the decentralized setting, and DFedAvgM makes the use of SGD-based momentum and extends DFedAvg. Both use multiple local epochs between communication rounds, like vanilla FedAvg. DFedSAM (Shi et al., 2023) incorporates the SAM algorithm (Foret et al., 2021) into the DFL process. DisPFL (Dai et al., 2022) is a personalized federated learning approach that aims to train a global model and personalize it to each client with a local mask. In order to make the comparision fair, we report the accuracy of the global model on our test set.

C.2. Hyperparameters

We perform a hyperparameter search over each baseline and select the hyperparameters corresponding to the best validation accuracy. We use the $\alpha = 0.1$ Fashion-MNIST test accuracy at communication round 30 as the metric for selection. This is done because the majority of comparisons take place on Fashion-MNIST in the non-IID setting. For D-PSGD, we use a learning rate of 0.1, and a batch size of 10 (local epoch count is defined to be one in this approach). For DFedAvg, we use a learning rate of 0.1, a batch size of 25, and 20 local epochs. For DFedAvgM, we use a learning rate of 0.01, a batch size of 50, 20 local epochs, and a momentum of 0.9. For DisPFL, we use a learning rate of 0.1, a batch size of 10, and 10 local epochs. Following Dai et al. (2022), we use the sparsity rate of 0.5 for DisPFL. For DFedSAM, we began with the parameters suggested in Shi et al. (2023). After a hyperparameter search, we found that a radius $\rho = 0.01$, $\eta = 0.01$, momentum of 0.99, learning rate decay of 0.95, weight decay of 5×10^{-4}, 5 local epochs, and a batch size of 32 yielded the best performance. Note that we used a single gossip step per round for all approaches in order to maintain a fair comparison. As for the NTK-DFL, we use a learning rate of 0.01 and search over values $t \in \{100, 200, \ldots, 800\}$ during the weight evolution process. For the baseline compression methods, we experimented with quantization as well as top-k sparsification (see Appendix E). We found that the baseline methods, which were gradient-based, cannot withstand the aggressive sparsification applied to NTK-DFL. Therefore, we used only quantization [such as in Sun et al. (2021)] and selected the most effective value to facilitate a fair comparison.

C.3. Inter-Client Model Deviation

We plot the relationship between the accuracy and the inter-client model deviation among NTK-DFL clients in Figure 5, where the inter-client model deviation is defined for a set of weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M \in \mathbb{R}^d$ as follows:

$$V = \frac{1}{d}\sum_{j=1}^{d}\sqrt{\sum_{i=1}^{M}(w_{i,j} - \bar{w}_j)^2}, \quad \bar{\boldsymbol{w}} = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{w}_i, \tag{12}$$

where it captures per-parameter variation in an averaged sense.

C.4. Additional Experimental Results

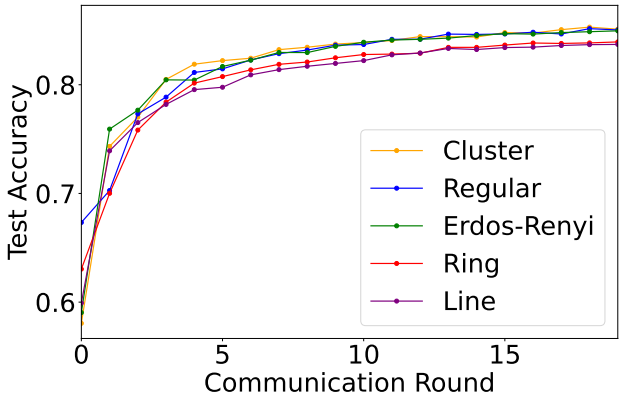


Figure 8. Convergence of NTK-DFL across different dynamic topologies, trained on Fashion-MNIST. NTK-DFL is evaluated with a clustered graph (in yellow) with 5 neighbors per client, a $\kappa = 5$ regular graph (in blue), an Erdos-Renyi random graph with five mean neighbors (in green), a ring topology (in red), and a line topology (in purple). We observe that NTK-DFL demonstrates steady convergence across different topology classes. For the ring and line topologies, convergence is a bit slower due to a sparser graph of 2 rather than 5 neighbors per client.

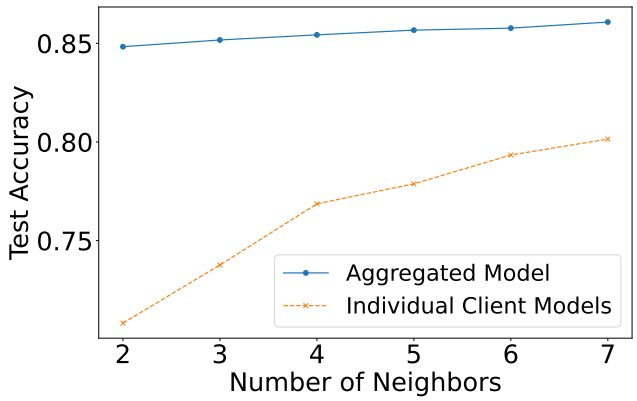
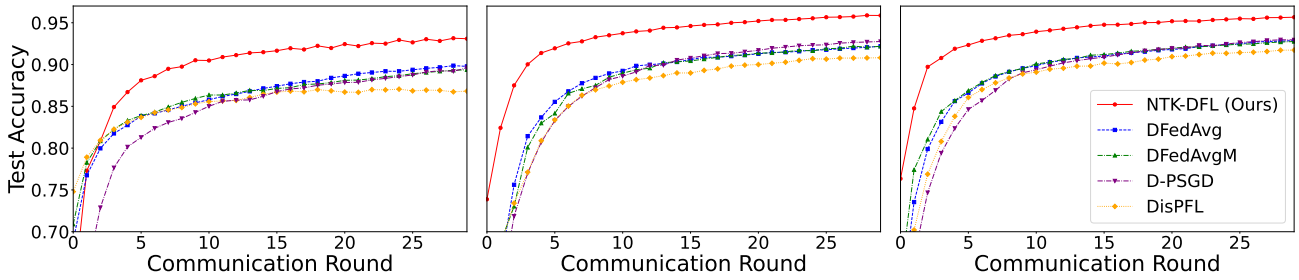


Figure 9. NTK-DFL model accuracy as a function of neighbor count κ, trained on Fashion-MNIST. Notably, the aggregated model accuracy across NTK-DFL clients (in blue) remains consistent, even as network sparsity varies. This stability persists despite a significant decline in mean individual client test accuracy (in yellow) as the number of neighbors decreases.



(a) FEMNIST (Feature-skewed) (b) MNIST ($\alpha = 0.05$) (c) MNIST ($\alpha = 0.1$)

Figure 10. Convergence of various methods on heterogeneous datasets: (a) FEMNIST, (b) Non-IID MNIST ($\alpha = 0.05$), and (c) Non-IID MNIST ($\alpha = 0.1$). NTK-DFL consistently outperforms all baselines.

D. Discussion of Communication Trade-offs

Federated learning methods balance two key communication dimensions: the volume of data transmitted per round and the total number of communication rounds required. NTK-DFL transmits more data per round than traditional approaches such as DFedAvg by sharing Jacobian matrices rather than gradients or weights, but requires fewer total rounds. We emphasize that the number of communication rounds is a key practical consideration. Below, we highlight scenarios where minimizing it is especially beneficial:

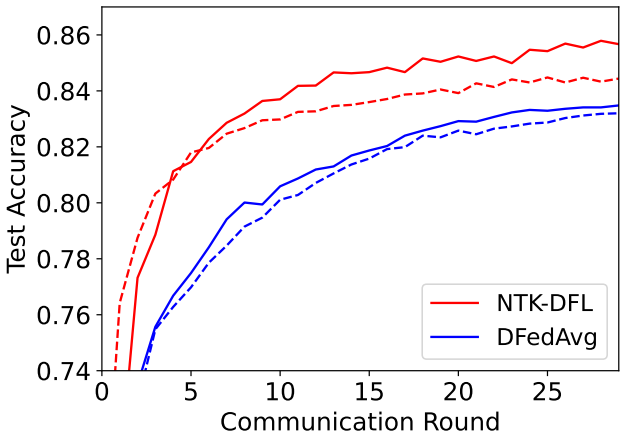


Figure 11. The effect of static vs. dynamic topology on NTK-DFL (Fashion-MNIST, $\alpha = 0.1$). Solid lines correspond to a dynamic topology, whereas dotted lines correspond to a static topology. Both methods benefit from the dynamic topology and NTK-DFL outperforms DFedAvg under both topologies. Other baselines are not drawn but perform similarly to DFedAvg.

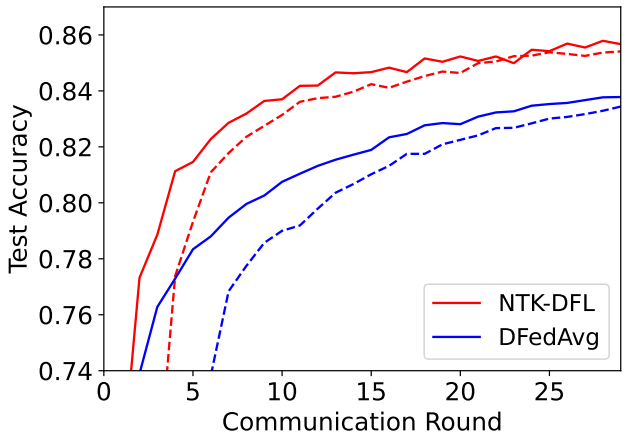


Figure 12. The effect of different vs. identical weight initialization (Fashion-MNIST, $\alpha = 0.1$). Solid lines correspond to the same weight initialization for all clients, whereas dotted lines correspond to different initialization. The convergence of NTK-DFL is affected less than that of DFedAvg. Other baselines are not drawn but perform similarly to DFedAvg.

- **Large per-round communication latency**, where fewer rounds can significantly reduce overall training time (e.g., compression or encryption of weights/gradients, heavy preprocessing of input data).
- **Limited device availability**, in which fewer rounds allow more efficient training when devices are intermittently available.
- **High bandwidth applications**, where ample network bandwidth (e.g., gigabit home internet) can accommodate a large data volume for each communication round, making the number of communication rounds the dominant factor in training efficiency.
- **Synchronization delays**, where each round must wait for all devices to complete computation, with the slowest device bottlenecking progress, thus making the number of communication rounds an important factor.

E. Overhead Mitigation Strategies

While analysis of memory and communication overhead are not a central theme of this paper, we include strategies to mitigate both forms of overhead for practical deployment. A thorough analysis of optimization and parallelization is out of the scope of this work and we leave it to future research.

E.1. Jacobian Batching

We introduce Jacobian batching to address potential memory constraints in NTK-DFL implementations. For scenarios involving dense networks or large datasets, clients can process their local datasets in smaller batches, reducing memory complexity from $O(N_i d_2 d)$ to $O(N_i d_2 d/m_1)$, where m_1 is the number of batches. Clients compute and transmit Jacobians for each batch separately, evolving their weights multiple times per communication round. This approach effectively trades a single large NTK $\mathbf{H} \in \mathbb{R}^{N\times N}$ for m_1 smaller NTKs $\mathbf{H}_{m_1} \in \mathbb{R}^{N/m_1 \times N/m_1}$ that form block diagonals of $\mathbf{H}$, where N represents the total number of data points between client i and its neighbors $\mathcal{N}_i$. While some information is lost in the uncomputed off-diagonal entries of $\mathbf{H}$, this is mitigated by the increased frequency of NTK evolution steps. Figure 13 demonstrates this phenomenon, where an increasing batch number m_1 actually leads to improved convergence. This complexity reduction enables clients to connect in a denser network for the same memory cost.

E.2. Use of the Clustered Topology

In the traditional NTK-DFL approach, each client must compute Jacobians on their own data with respect to the weight vector of every neighbor, as NTK construction requires all Jacobians to be evaluated at the same point in weight space. This

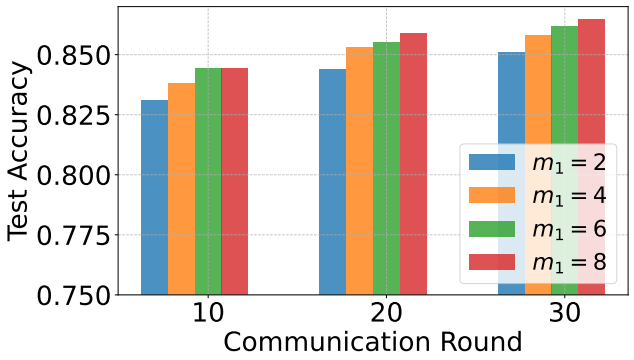


Figure 13. Test accuracy of NTK-DFL vs. communication round for various Jacobian batch numbers m_1, with higher m_1 values denoting more batches (Fashion-MNIST, $\alpha = 0.1$). We observe a general, counterintuitive increase in test accuracy with an increased number of batches.

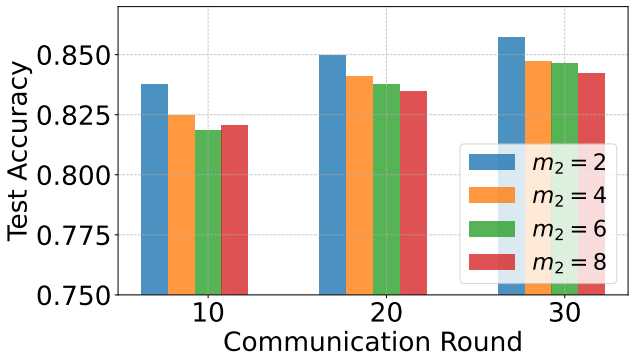


Figure 14. Test accuracy of NTK-DFL vs. communication round for sampling divisors m_2 (Fashion-MNIST, $\alpha = 0.1$). Different from Jacobian batching, only a $1/m_2$ fraction of client data is selected each communication round. We observe a slight decrease in test accuracy with increased m_2.

implies that Jacobian computation scales linearly with the number of neighbors. However, a (dynamic) *clustered topology* reduces this burden: after the initial weight synchronization step, all clients within a cluster share the same (aggregated) weight, so each client only needs to compute a single set of Jacobians. These Jacobians can then be reused for all neighbor interactions within the cluster. Furthermore, the weight evolution step can be offloaded to a single designated client, or distributed across clients by sharing intermediate weights during the weight unrolling process (5b). As a result, **both communication and computation overheads become independent of the number of neighbors**, assuming each round's evolution is handled by a single client per cluster. We illustrate the performance of NTK-DFL under this clustered topology in Figure 8 of Appendix C, which shows convergence comparable to that of a regular graph with the same average degree.

E.3. Communication Cost

Compared to traditional weight-based approaches that communicate a client's parameters $\boldsymbol{w}_i$ each round, NTK-DFL utilizes Jacobian matrices to enhance convergence speed and heterogeneity resilience. This tensor has memory complexity $O(N_i d_2 d)$, where N_i denotes the number of data points between client i and its neighbors $\mathcal{N}_i$, d is the model parameter dimension, and d_2 is the output dimension. We propose the following strategies to improve the communication efficiency of NTK-DFL while maintaining convergence properties in heterogeneous settings.

Data Subsampling We introduce an approach where clients sample a $1/m_2$ fraction of their data each round for NTK evolution. Clients follow the protocol described in Section 3.2, but exchange Jacobian matrices of reduced size. As demonstrated in Figure 14, moderate values of m yield light performance degradation, validating this communication reduction strategy.

Jacobian Compression We employ several techniques to reduce Jacobian tensor dimensionality. First, we apply top-k sparsification, zeroing out elements with the smallest magnitude (Alistarh et al., 2018). The remaining nonzero values are quantized to b bits. Additionally, we introduce a shared random projection matrix $\mathbf{P} \in \mathbb{R}^{d_1 \times d_1'}$ generated from a common seed, creating projections $\mathbf{Z}_i = \mathbf{X}_i \mathbf{P}$ that reduce input dimension from d_1 to d_1'. This combination of techniques maintains convergence properties while significantly reducing communication costs. Note that similar compression schemes applied to weight-based approaches lead to significant degradation in performance (Yue et al., 2022). Figure 15 illustrates the relative differences in communication load for a different combinations of the techniques above, with a sparsification of 0.5, quantization to 6 bits, a sampling of $m_2 = 5$, and a projection to $d_1' = 200$ for the full optimization curve.

Figure 16 compares the convergence of NTK-DFL with the baseline methods in terms of communication volume. NTK-DFL uses the compression methods detailed above, as well as a clustered topology (Section E.2) to reduce communication overhead. The communication-optimized NTK-DFL converges in fewer rounds than the baselines. However, with more expressive updates, it uses greater communication volume. This enforces the idea that NTK-DFL is especially useful in scenarios where convergence in fewer rounds is important, such as those outlined in Appendix E.

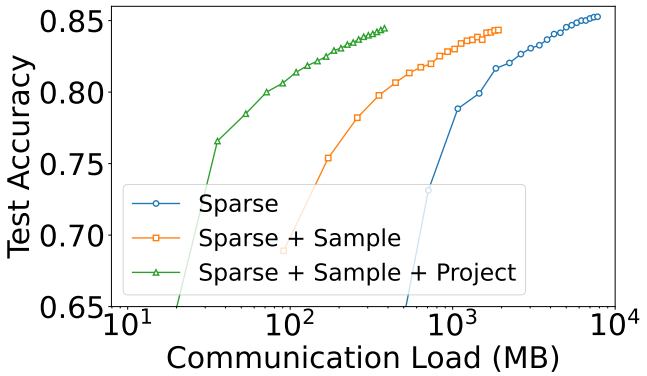


Figure 15. Comparison of NTK-DFL variants with progressive communication optimizations (Fashion-MNIST, $\alpha = 0.1$). Data sampling and projection technique provides compounding reductions in communication load compared to sparsification alone, while the fully optimized variant demonstrates significantly lower communication requirements at a comparable test accuracy.

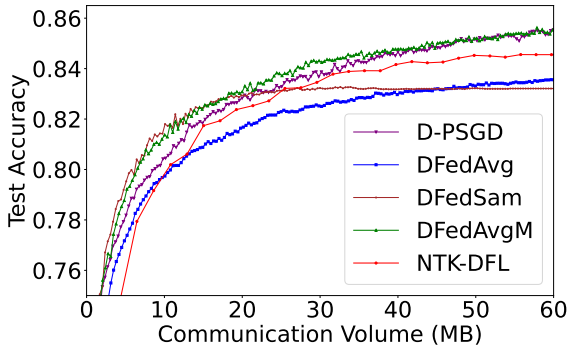


Figure 16. Test accuracy vs. communication volume for NTK-DFL and baselines (Fashion-MNIST, $\alpha = 0.1$). With the use of compression techniques and a clustered topology, NTK-DFL performs similarly in term of communication volume. It is slightly outperformed by more communication efficient algorithms, though they need about 5 times as many communication rounds to converge.

F. Reconstruction Attack

While privacy preservation is not the primary focus of this work, we conduct a brief analysis of data privacy in NTK-DFL. Following the reconstruction attack method of Zhu et al. (2019), we evaluate the feasibility of reconstructing client data from transmitted Jacobian matrices under varying compression levels. Our experiments range from basic top-k sparsification with sparsity 0.25 to combined sparsification with random projection to dimension $d_1' = 200$. Figure 17 illustrates that client data reconstruction becomes increasingly difficult when a random projection is additionally applied to the Jacobian matrices.

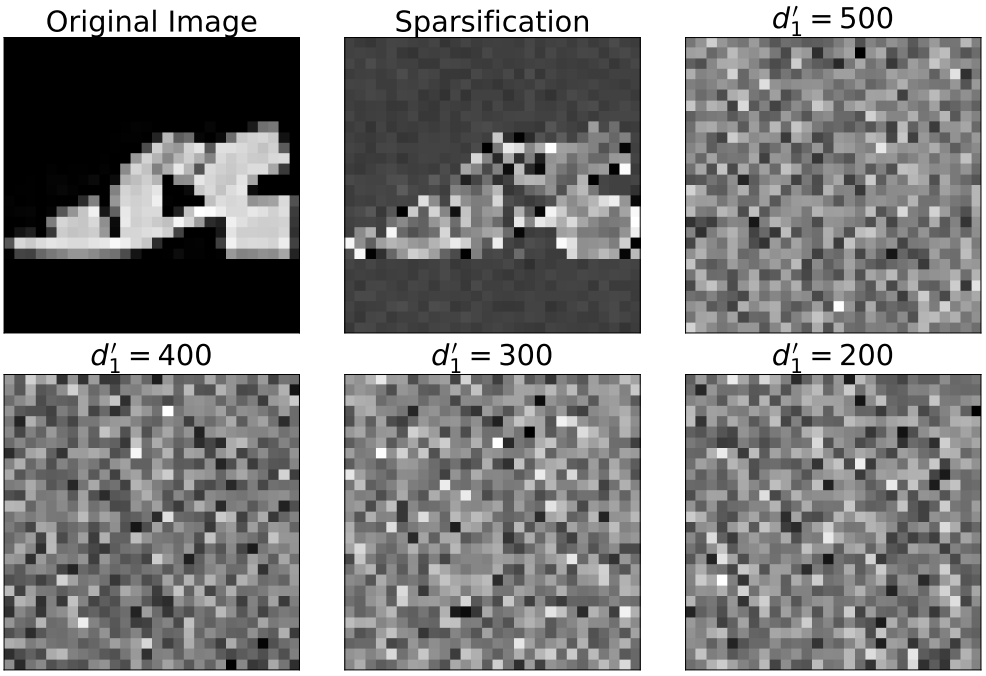


Figure 17. Reconstruction attack of client data from Jacobian matrices for various levels of compression. For the image corresponding to sparsified matrices (the middle image of the first row), no random projection is done. We observe the ability to reconstruct a very noisy version of client data. For the other images, we use sparsification and a random projection to dimension d_1'. We observe an inability to reconstruct client data when the random projection is additionally applied.

G. Mathematical Analysis

Below, we include more detailed assumptions that are used throughout the proof of the lemmas presented below and the subsequent proof of Theorem 4.5. We note that the assumptions listed below are relatively mild and common in DFL settings (Sun et al., 2021; Shi et al., 2023).

Assumption G.1. (Lipschitz smoothness). The function $\mathcal{L}_i$ is differentiable, and its gradient $\nabla\mathcal{L}_i$ is L-Lipschitz continuous for all $i \in \{1, 2, \ldots, m\}$, i.e.,

$$\|\nabla\mathcal{L}_i(\boldsymbol{w}) - \nabla\mathcal{L}_i(\boldsymbol{v})\| \leq L\|\boldsymbol{w} - \boldsymbol{v}\|, \tag{13}$$

for all $\boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^d$.

Assumption G.2. (Bounded variance). The global variance of the gradient for functions $\{\mathcal{L}_i\}_{i=1}^m$ is jointly bounded, i.e.,

$$\frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}_i(\boldsymbol{w}) - \nabla\mathcal{L}(\boldsymbol{w})\|^2 \leq \sigma_g^2, \tag{14}$$

for all $\boldsymbol{w} \in \mathbb{R}^d$. This quantity is directly related to data heterogeneity among clients.

Assumption G.3. (Bounded gradients). For any $i \in \{1, 2, \ldots, m\}$ and $\boldsymbol{w} \in \mathbb{R}^d$, the gradient of the local loss function is bounded as $\|\nabla\mathcal{L}_i(\boldsymbol{w})\| \leq B$, for some constant $B > 0$.

We present the following lemmas to aid in the analysis. First, we present Lemma G.4 that is common in DFL literature relating to the spectral properties of the mixing matrix. The following lemmas bound the mean divergence of the client weights over local update steps t (Lemma G.5), the variance of client weights (Lemma G.7), and the mean gradient norm over clients (Lemmma G.8). Lastly, we bound the error term δ_{NTK} (Lemma G.9).

Lemma G.4. *[Following Lian et al. (2017)] For any $k \in \mathbb{Z}^+$, the mixing matrix $\mathbf{M} \in \mathbb{R}^{m\times m}$ satisfies the inequality*

$$\|\mathbf{M}^k - \mathbf{P}\|_{op} \leq \lambda^k, \tag{15}$$

where the parameter λ is defined as

$$\lambda := \max\{|\lambda_2(\mathbf{M})|, |\lambda_m(\mathbf{M})|\}. \tag{16}$$

Here, the spectral norm of a matrix $\mathbf{A}$ is denoted by $\|\mathbf{A}\|_{op}$, and the matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \frac{\mathbb{1}\mathbb{1}^\top}{m} \in \mathbb{R}^{m\times m}, \tag{17}$$

where $\mathbb{1}$ is the all-one vector $[1, 1, \ldots, 1]^\top \in \mathbb{R}^m$.

Lemma G.5. *Assume that Assumptions G.3 and 4.4 hold. Let $\mathbf{w}^{(k,t)}$ denote the model weights at round k and timestep t, and let the weight evolution be governed by the NTK-DFL approach. Then, it follows that*

$$\|\boldsymbol{w}^{(k,t+1)} - \boldsymbol{w}^{(k,t)}\|^2 \leq \eta^2(B + \delta_{NTK})^2, \tag{18}$$

for $1 \leq k \leq K$.

Proof. [Following Sun et al. (2021)]
We begin with the NTK update step

$$\boldsymbol{w}_i^{(k,t+1)} - \boldsymbol{w}_i^{(k,t)} = \frac{\eta}{N_i^{(k)}}\nabla f_i(\mathbf{X}_i; \boldsymbol{w}_i^{(k,0)})^\top \mathbf{r}_i(\mathbf{X}_i, \mathbf{Y}_i, \boldsymbol{w}_i^{(k,t)}), \tag{19}$$

where $\mathbf{r}_i(\mathbf{X}_i, \mathbf{Y}_i, \boldsymbol{w}) = \nabla_f \ell(f_i(\mathbf{X}_i; \boldsymbol{w}), \mathbf{Y}_i)$ is the (possibly transformed) residual over the data between client i and its neighbors $j \in \mathcal{N}_i$. The specific $\mathbf{r}_i$ depends on the loss of choice, and we provide general derivation that is loss-agnostic. Define the NTK-approximated gradient of the local loss function $\mathcal{L}_i$ with respect to the model weights $\boldsymbol{w}_i$, using the Jacobian evaluated at $t = 0$, we obtain

$$\boldsymbol{w}_i^{(k,t+1)} - \boldsymbol{w}_i^{(k,t)} = \eta\left(\nabla\mathcal{L}_i(\boldsymbol{w}_i^{(k,t)}) + \Delta_i^{\text{NTK}}(\boldsymbol{w}_i^{(k,t)})\right), \tag{20}$$

where

$$\nabla_{\boldsymbol{w}}\mathcal{L}_i^{\text{NTK}}(\boldsymbol{w}_i^{(k,t)}) = \frac{1}{\tilde{N}_i^{(k)}}\nabla_{\boldsymbol{w}} f(\tilde{\mathbf{X}}_i, \boldsymbol{w}_i^{(k,0)})^\top \mathbf{r}_i(\tilde{\mathbf{X}}_i, \tilde{\mathbf{y}}_i, \boldsymbol{w}_i^{(k,t)}), \tag{21a}$$

$$\Delta_i^{\text{NTK}} = \nabla_{\boldsymbol{w}}\mathcal{L}_i(\boldsymbol{w}^{(k,t)}) - \nabla_{\boldsymbol{w}}\mathcal{L}_i^{\text{NTK}}(\boldsymbol{w}^{(k,t)}). \tag{21b}$$

Using Assumptions G.3 and 4.4 and taking the norm:

$$\|\boldsymbol{w}_i^{(k,t+1)} - \boldsymbol{w}_i^{(k,t)}\| \leq \eta(B + \delta_{\text{NTK}}), \tag{22}$$

therefore completing the proof.

Lemma G.6. *Given the stepsize $0 < \eta \leq \frac{1}{8LT}$, assume $\boldsymbol{w}_i^{(k,t)}$ and $\boldsymbol{w}_i^{(k)}$ are generated by the decentralized NTK-FL algorithm for all $i \in \{1, 2, \ldots, m\}$. If Assumptions G.1, G.2, and 4.4 hold, it follows that*

$$\frac{1}{m}\sum_{i=1}^{m}\|\boldsymbol{w}_i^{(k,t)} - \boldsymbol{w}_i^{(k,0)}\|^2 \leq 16\eta^2T^2\left[\delta_{NTK}^2 + \sigma_g^2 + \tfrac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2\right], \tag{23}$$

for $1 \leq t \leq K$.

Proof. [Following Sun et al. (2021)]
Decomposing the difference into global variance, the NTK difference, and the difference between $t = 0$ and t,

$$\begin{aligned}\boldsymbol{w}_i^{(k,t+1)} - \boldsymbol{w}_i^{(k,0)} = \boldsymbol{w}_i^{(k,t)} - \boldsymbol{w}_i^{(k,0)} - \eta\Big[\Delta_i^{\text{NTK}}(\boldsymbol{w}_i^{(k,t)}) + \Big(\nabla\mathcal{L}_i(\boldsymbol{w}_i^{(k,t)}) - \nabla\mathcal{L}_i(\boldsymbol{w}_i^{(k,0)})\Big) \\ + \Big(\nabla\mathcal{L}_i(\boldsymbol{w}_i^{(k,0)}) - \nabla\mathcal{L}(\boldsymbol{w}^{(k,0)})\Big) + \nabla\mathcal{L}(\boldsymbol{w}_i^{(k,0)})\Big].\end{aligned} \tag{24}$$

We consider two terms such that $\|\boldsymbol{w}_i^{(k,t+1)} - \boldsymbol{w}_i^{(k,0)}\|^2 \leq \text{I} + \text{II}$, using the Cauchy inequality and Assumptions G.2 and 4.4

$$\text{I} \leq (1 + \tfrac{1}{2T-1})\left\|\boldsymbol{w}_i^{(k,t)} - \boldsymbol{w}_i^{(k,0)}\right\|^2,\ \text{II} \leq 8T\eta^2\Big(\delta_{\text{NTK}}^2 + \|\nabla\mathcal{L}_i(\boldsymbol{w}_i^{(k,t)}) - \nabla\mathcal{L}_i(\boldsymbol{w}_i^{(k,0)})\|^2 + \sigma_g^2 + \|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k,0)})\|^2\Big). \tag{25}$$

From here, rewriting term II using Assumption G.1, we have

$$\left\|\boldsymbol{w}_i^{(k,t+1)} - \boldsymbol{w}_i^{(k,0)}\right\|^2 \leq (1 + \tfrac{1}{2T-1} + 8T\eta^2L^2)\left\|\boldsymbol{w}_i^{(k,t)} - \boldsymbol{w}_i^{(k,0)}\right\|^2 + 8T\eta^2(\delta_{\text{NTK}}^2 + \sigma_g^2) + 8T\eta^2\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k,0)})\|^2. \tag{26}$$

Given an appropriately small step size, we simplify $1 + \frac{1}{2T-1} + 8T\eta^2L^2 \leq 1 + \frac{1}{T-1}$. Solving the recursive relationship yields

$$\frac{1}{m}\sum_{i=1}^{m}\left\|\boldsymbol{w}_i^{(k,t)} - \boldsymbol{w}_i^{(k,0)}\right\|^2 \leq \frac{1}{m}\sum_{i=1}^{m}\sum_{u=1}^{T}(1 + \tfrac{1}{T-1})^u(8T\eta^2)\big[(\delta_{\text{NTK}}^2 + \sigma_g^2) + \|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k,0)})\|^2\big] \tag{27a}$$

$$\leq 16T^2\eta^2\Big[\delta_{\text{NTK}}^2 + \sigma_g^2 + \tfrac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k,0)})\|^2\Big], \tag{27b}$$

with the recursive sum $\sum_{u=1}^{T}(1 + \frac{1}{T-1})^u \leq 2T$ for $T \geq 10$. Thus, we have completed the proof.

Lemma G.7. *Suppose the stepsize satisfies $0 < \eta \leq \frac{1}{8LT}$, and let $\boldsymbol{w}_i^{(k,0)}$ be the model weights produced by the NTK-DFL algorithm for all clients $i \in \{1, 2, \ldots, m\}$. Under Assumptions G.2 and 4.4 the following bound holds:*

$$\frac{1}{m}\sum_{i=1}^{m}\|\boldsymbol{w}_i^{(k,0)} - \bar{\boldsymbol{w}}^{(k,0)}\|^2 \leq \frac{16T^2\eta^2\left[\delta_{NTK} + \sigma_g^2 + \frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k,0)})\|^2\right]}{(1-\lambda)^2}, \tag{28}$$

for all $1 \leq k \leq K$.

Proof. [Following Sun et al. (2021) and Shi et al. (2023)]
With a slight abuse of notation, $\mathbf{Y}^k \coloneqq [\boldsymbol{w}_1^{(k,T)}, \boldsymbol{w}_2^{(k,T)}, \ldots, \boldsymbol{w}_m^{(k,T)}]$ and $\mathbf{X}^k \coloneqq [\boldsymbol{w}_1^{(k,0)}, \boldsymbol{w}_2^{(k,0)}, \ldots, \boldsymbol{w}_m^{(k,0)}]$, we may write the mixing process as

$$\mathbf{X}^{k+1} = \mathbf{M}\mathbf{Y}^k = \mathbf{M}\mathbf{X}^k - \boldsymbol{\zeta}^k, \tag{29}$$

where

$$\boldsymbol{\zeta}^k := \mathbf{M}(\mathbf{X}^k - \mathbf{Y}^k). \tag{30}$$

We note that the order of averaging and updating weights does not affect this proof, as it relies on the spectral gap of the (possibly time-varying) mixing matrix, which does not change over time. We can write recursively

$$\mathbf{X}^k = \mathbf{M}^k\mathbf{X}^0 - \sum_{j=0}^{k-1} \mathbf{M}^{k-1-j}\boldsymbol{\zeta}^j. \tag{31}$$

We note the following property of the mixing operation:

$$\mathbf{P}\mathbf{M} = \mathbf{M}\mathbf{P} = \mathbf{P}. \tag{32}$$

From Lemma G.4 we have that $\|\mathbf{M}^k - \mathbf{P}\|_{\text{op}} \leq \lambda^k$. Left-multiplying both sides of (31) and using the mixing property (32), we obtain

$$\mathbf{P}\mathbf{X}^k = \mathbf{P}\mathbf{X}^0 - \sum_{j=0}^{k-1} \mathbf{P}\boldsymbol{\zeta}^j = -\sum_{j=0}^{k-1} \mathbf{P}\boldsymbol{\zeta}^j, \tag{33}$$

with the initialization $\mathbf{X}^0 = \mathbf{0}$. Then, we are led to

$$\|\mathbf{X}^k - \mathbf{P}\mathbf{X}^k\| = \left\|\sum_{j=0}^{k-1}(\mathbf{P} - \mathbf{M}^{k-1-j})\boldsymbol{\zeta}^j\right\| \leq \sum_{j=0}^{k-1}\|\mathbf{P} - \mathbf{M}^{k-1-j}\|_{\text{op}}\|\boldsymbol{\zeta}^j\| \leq \sum_{j=0}^{k-1}\lambda^{k-1-j}\|\boldsymbol{\zeta}^j\|. \tag{34}$$

With the Cauchy inequality, we obtain

$$\|\mathbf{X}^k - \mathbf{P}\mathbf{X}^k\|^2 \leq \left(\sum_{j=0}^{k-1}\lambda^{\frac{k-1-j}{2}}\cdot\lambda^{\frac{k-1-j}{2}}\|\boldsymbol{\zeta}^j\|\right)^2 \leq \left(\sum_{j=0}^{k-1}\lambda^{k-1-j}\right)\left(\sum_{j=0}^{k-1}\lambda^{k-1-j}\|\boldsymbol{\zeta}^j\|^2\right), \tag{35}$$

and we can compute that

$$\|\boldsymbol{\zeta}^j\|^2 \leq \|\mathbf{M}\|^2\cdot\|\mathbf{X}^j - \mathbf{Y}^j\|^2 \leq \|\mathbf{X}^j - \mathbf{Y}^j\|^2. \tag{36}$$

Applying Lemma G.6, for all j,

$$\frac{1}{m}\|\mathbf{X}^j - \mathbf{Y}^j\|^2 \leq 16T^2\eta^2\left[\delta_{\text{NTK}}^2 + \sigma_g^2 + \frac{1}{m}\sum_{i=1}^m\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2\right]. \tag{37}$$

Finally, we complete the proof with

$$\frac{1}{m}\|\mathbf{X}^t - \mathbf{P}\mathbf{X}^t\|^2 \leq \frac{16T^2\eta^2\left[\delta_{\text{NTK}}^2 + \sigma_g^2 + \frac{1}{m}\sum_{i=1}^m\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2\right]}{(1-\lambda)^2}. \tag{38}$$

Lemma G.8. *Using the result from Lemma G.7, we obtain the following bound for the average squared gradient norm:*

$$\frac{1}{m}\sum_{i=1}^m\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2 \leq \frac{2L^2C_1\eta^2}{(1-\lambda)^2} + 2\|\nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2, \tag{39}$$

where $C_1 = 16T^2(\delta_{NTK}^2 + \sigma_g^2 + B^2)$.

Proof.
We expand the expression below by adding a difference term:

$$\frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2 \leq \frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)}) - \nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)}) + \nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2. \tag{40}$$

Using $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, we have

$$\frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2 \leq \frac{1}{m}\sum_{i=1}^{m}2\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)}) - \nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + 2\|\nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2. \tag{41}$$

With Assumption G.1, we can bound the gradient difference as

$$\frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2 \leq 2L^2\frac{1}{m}\sum_{i=1}^{m}\|\boldsymbol{w}_i^{(k)} - \bar{\boldsymbol{w}}^{(k)}\|^2 + 2\|\nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2. \tag{42}$$

Finally, we substitute the result from Lemma G.7 and use Assumption G.3 to finish the proof:

$$\frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2 \leq \frac{32L^2T^2\eta^2\left(\delta_{\text{NTK}}^2 + \sigma_g^2 + B^2\right)}{(1-\lambda)^2} + 2\|\nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2. \tag{43}$$

Lemma G.9. *Assuming an appropriately small step size (such as is chosen in Theorem 4.5), Lipschitz continuity for $\nabla f_i(\mathbf{X}_i; \boldsymbol{w}^{(k,t)})$ and a bounded mean residual norm across all clients, i.e. $\frac{1}{N_i}\sum\|\mathbf{r_i}(\mathbf{X}_i, \mathbf{Y}_i; \boldsymbol{w})\| \leq r$, we can bound the error term in Assumption 4.4 as follows:*

$$\delta_{NTK}^2 \leq 16\eta^2T^2L^2r^2(\sigma_g^2 + B^2). \tag{44}$$

Proof.
We can express Δ_{NTK} as follows

$$\Delta_i^{\text{NTK}} = \nabla_{\boldsymbol{w}}\mathcal{L}_i(\boldsymbol{w}_i^{(k,t)}) - \nabla_{\boldsymbol{w}}\mathcal{L}_i^{\text{NTK}}(\boldsymbol{w}_i^{(k,t)}) \tag{45a}$$

$$= \frac{1}{N_i^{(k)}}\left(\nabla_{\boldsymbol{w}}f(\mathbf{X}_i, \boldsymbol{w}_i^{(k,t)})^\top - \nabla_{\boldsymbol{w}}f(\mathbf{X}_i, \boldsymbol{w}_i^{(k,0)})^\top\right)\mathbf{r}_i(\mathbf{X}_i, \mathbf{y}_i, \boldsymbol{w}_i^{(k,t)}). \tag{45b}$$

Using the Lipschitz assumption listed above, where the Lipschitz constant for f_i is L, we have

$$\delta_{\text{NTK}}^2 \leq L^2\|\boldsymbol{w}_i^{(k,t)} - \boldsymbol{w}_i^{(k,0)}\|^2r^2. \tag{46}$$

Now, substituting the result from G.6

$$\delta_{\text{NTK}}^2 \leq L^2\left[16\eta^2T^2(\delta_{\text{NTK}}^2 + \sigma_g^2) + 16\eta^2T^2B^2\right]r^2. \tag{47}$$

Solving for δ_{NTK}^2

$$\delta_{\text{NTK}}^2 \leq \frac{16\eta^2T^2r^2L^2(\sigma_g^2 + B^2)}{1 - 16\eta^2T^2r^2} \approx 16\eta^2T^2r^2L^2(\sigma_g^2 + B^2). \tag{48}$$

where the second inequality comes from assuming a step size η such that, for a large communication round K, $16\eta^2T^2r^2 \ll 1$. Thus, the proof is complete.

Proof of Theorem 4.5.

We begin with

$$\bar{\boldsymbol{w}}^{(k+1)} - \bar{\boldsymbol{w}}^{(k)} = \bar{\boldsymbol{w}}^{(k,T)} - \bar{\boldsymbol{w}}^{(k)}, \tag{49}$$

due to the fact that the application of the mixing matrix does not change $\bar{w}$. Using the NTK-DFL weight update step, noting that the average weight at round k and time step $t=0$ is written as $\bar{\boldsymbol{w}}^{(k)} := \bar{\boldsymbol{w}}^{(k,0)}$,

$$\bar{\boldsymbol{w}}^{(k,T)} - \bar{\boldsymbol{w}}^{(k)} = -\frac{\eta}{m}\sum_{i=1}^{m}\sum_{u=1}^{T}\nabla\mathcal{L}^{\text{NTK}}(\boldsymbol{w}_i^{(k,u)}) = -\frac{\eta}{m}\sum_{i=1}^{m}\sum_{u=1}^{T}\left(\nabla_{\boldsymbol{w}}\mathcal{L}_i(\boldsymbol{w}_i^{(k,u)}) + \Delta_{i,\text{NTK}}^{(k,u)}\right). \tag{50}$$

Assumption G.1 gives us

$$\mathcal{L}(\bar{\mathbf{w}}^{(k+1)}) \leq \mathcal{L}(\bar{\mathbf{w}}^{(k)}) + \langle \nabla \mathcal{L}(\bar{\mathbf{w}}^{(k)}), \bar{\mathbf{w}}^{(k+1)} - \bar{\mathbf{w}}^{(k)} \rangle + \frac{L}{2} \|\bar{\mathbf{w}}^{(k+1)} - \bar{\mathbf{w}}^{(k)}\|^2. \tag{51}$$

For the **first inner product term**, we have

$$\langle \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}), \bar{\boldsymbol{w}}^{(k+1)} - \bar{\boldsymbol{w}}^{(k)} \rangle = \langle T \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}), (\bar{\boldsymbol{w}}^{(k,T)} - \bar{\boldsymbol{w}}^{(k)})/T + \eta \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}) - \eta \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}) \rangle \tag{52a}$$

$$= -\eta T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \left\langle T \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}), \eta \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}) + \frac{\bar{\boldsymbol{w}}^{(k,T)} - \bar{\boldsymbol{w}}^{(k)}}{T} \right\rangle \tag{52b}$$

$$= -\eta T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \left\langle T \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}), \frac{\eta}{T} \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \nabla \mathcal{L}_i(\bar{\boldsymbol{w}}^{(k)}) - \frac{\eta}{T} \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \big(\nabla \mathcal{L}_i(\boldsymbol{w}_i^{(k,t)}) + \Delta_i^{\text{NTK}}(\boldsymbol{w}_i^{(k,t)}) \big) \right\rangle \tag{52c}$$

$$= -\eta T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \left\langle T \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}), \frac{\eta}{T} \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \Big(\nabla \mathcal{L}_i(\bar{\boldsymbol{w}}^{(k)}) - \nabla \mathcal{L}_i(\boldsymbol{w}_i^{(k,t)}) - \Delta_i^{\text{NTK}}(\boldsymbol{w}_i^{(k,t)}) \Big) \right\rangle \tag{52d}$$

$$\leq -\eta T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \eta \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\| \cdot \left\| \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \Big(\nabla \mathcal{L}_i(\bar{\boldsymbol{w}}^{(k)}) - \nabla \mathcal{L}_i(\boldsymbol{w}_i^{(k,t)}) \Big) \right\| + \eta T \delta_{\text{NTK}} B. \tag{52e}$$

Using Assumption G.1, we have

$$\langle \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}), \bar{\boldsymbol{w}}^{(k+1)} - \bar{\boldsymbol{w}}^{(k)} \rangle \leq -\eta T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \eta L T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\| \cdot \frac{1}{m} \sum_{i=1}^{m} \|\bar{\boldsymbol{w}}^{(k)} - \boldsymbol{w}_i^{(k,t)}\| + \eta T \delta_{\text{NTK}} B. \tag{53}$$

Using Young's Inequality $a \leq \frac{a^2}{2\alpha} + \frac{\alpha}{2}$, we can write the norm in terms of the squared norm where we set:

$$a = \|\bar{\boldsymbol{w}}^{(k)} - \boldsymbol{w}_i^{(k,t)}\|, \quad \alpha = 1/L, \tag{54}$$

which yields the bound:

$$\eta L T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\| \cdot \frac{1}{m} \sum_{i=1}^{m} \|\bar{\boldsymbol{w}}^{(k)} - \boldsymbol{w}_i^{(k,t)}\| \leq \frac{\eta T}{2} \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \frac{\eta L^2 T}{2} \frac{1}{m} \sum_{i=1}^{m} \|\bar{\boldsymbol{w}}^{(k)} - \boldsymbol{w}_i^{(k,t)}\|^2. \tag{55}$$

With Jensen's inequality and Lemma $G.6$:

$$\eta L T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\| \cdot \frac{1}{m} \sum_{i=1}^{m} \|\bar{\boldsymbol{w}}^{(k)} - \boldsymbol{w}_i^{(k,t)}\| \leq \frac{\eta T}{2} \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \frac{\eta L^2 T}{2} \frac{1}{m} \sum_{i=1}^{m} \frac{1}{m} \sum_{i=1}^{m} \|\boldsymbol{w}_i^{(k,t)} - \boldsymbol{w}_i^{(k)}\|^2, \tag{56a}$$

$$\leq \frac{\eta T}{2} \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \frac{\eta L^2 T}{2} \left[16 \eta^2 T^2 (\delta_{\text{NTK}}^2 + \sigma_g^2 + \frac{1}{m} \sum_{i=1}^{m} \|\nabla \mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2) \right]. \tag{56b}$$

Substituting this bound above, the inner product term is bounded by:

$$\langle \nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)}), \bar{\boldsymbol{w}}^{(k+1)} - \bar{\boldsymbol{w}}^{(k)} \rangle \leq -\eta T \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + \frac{\eta T}{2} \|\nabla \mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2 + 8 \eta^3 L^2 T^3 \left[\delta_{\text{NTK}}^2 + \sigma_g^2 + \frac{1}{m} \sum_{i=1}^{m} \|\nabla \mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2 \right] + \eta T \delta_{\text{NTK}} B. \tag{57}$$

For the **second term**, we have

$$\frac{L}{2} \|\bar{\boldsymbol{w}}^{(k+1)} - \bar{\boldsymbol{w}}^{(k)}\|^2 = \frac{L}{2} \|\bar{\boldsymbol{w}}^{(k,T)} - \bar{\boldsymbol{w}}^{(k)}\|^2 \leq \frac{L}{2} \cdot \frac{1}{m} \sum_{i=1}^{m} \|\boldsymbol{w}_i^{(k,T)} - \boldsymbol{w}_i^{(k)}\|^2. \tag{58}$$

Using results from Lemma G.6

$$\frac{L}{2}\|\bar{\boldsymbol{w}}^{(k+1)}-\bar{\boldsymbol{w}}^{(k)}\|^2 \leq 8\eta^2T^2L\left[\delta_{\text{NTK}}^2+\sigma_g^2+\frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2\right]. \tag{59}$$

From here, we can write (51) as

$$\mathcal{L}(\bar{\boldsymbol{w}}^{(k+1)})-\mathcal{L}(\bar{\boldsymbol{w}}^{(k)}) \leq -\frac{\eta T}{2}\|\nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2+8\eta^2T^2L(\eta LT+1)(\delta_{\text{NTK}}^2+\sigma_g^2)+\eta T\delta_{\text{NTK}}B \\ +8\eta^2T^2L(\eta TL+1)\frac{1}{m}\sum_{i=1}^{m}\|\nabla\mathcal{L}(\boldsymbol{w}_i^{(k)})\|^2. \tag{60}$$

Substituting Lemma G.8, we obtain

$$\mathcal{L}(\bar{\boldsymbol{w}}^{(k+1)})-\mathcal{L}(\bar{\boldsymbol{w}}^{(k)}) \leq -\left[\tfrac{\eta T}{2}-16\eta^2T^2L(\eta TL+1)\right]\|\nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2+8\eta^2T^2L(\eta LT+1)(\delta_{\text{NTK}}^2+\sigma_g^2)+\eta T\delta_{\text{NTK}}B \\ +8\eta^2T^2L(\eta TL+1)\frac{2L^2C_1\eta^2}{(1-\lambda)^2}. \tag{61}$$

Taking the sum over K rounds, we obtain

$$\mathcal{L}(\bar{\boldsymbol{w}}^{(K+1)})-\mathcal{L}(\bar{\boldsymbol{w}}^{(1)}) \leq -K\left[\tfrac{\eta T}{2}-16\eta^2T^2L(\eta TL+1)\right]\min_{1\leq k\leq K}\|\nabla\mathcal{L}(\bar{\boldsymbol{w}}^{(k)})\|^2+K(8\eta^2T^2L)(\eta LT+1)(\delta_{\text{NTK}}^2+\sigma_g^2) \\ +K\eta T\delta_{\text{NTK}}B+K(8\eta^2T^2L)(\eta TL+1)\frac{2L^2C_1\eta^2}{(1-\lambda)^2}, \tag{62}$$

By moving terms around, we can finally prove the theorem below:

$$\min_{1\leq k\leq K}\left\|\nabla\mathcal{L}\left(\bar{\boldsymbol{w}}^{(k)}\right)\right\|^2 \leq \frac{\mathcal{L}\left(\bar{\boldsymbol{w}}^{(1)}\right)-\mathcal{L}\left(\bar{\boldsymbol{w}}^{(K+1)}\right)}{K\left[\frac{\eta T}{2}-16\eta^2T^2L(\eta TL+1)\right]} \\ +\frac{8\eta^2T^2L\left(\eta LT+1\right)\left(\delta_{\text{NTK}}^2+\sigma_g^2\right)+\eta T\delta_{\text{NTK}}B+8\eta^2T^2L(\eta TL+1)\cdot\frac{2L^2C_1\eta^2}{(1-\lambda)^2}}{\frac{\eta T}{2}-16\eta^2T^2L(\eta TL+1)}. \tag{63}$$

Wrapping this up in constants, we can rewrite the above inequality as

$$\min_{1\leq k\leq K}\left\|\nabla\mathcal{L}\left(\bar{\boldsymbol{w}}^{(k)}\right)\right\|^2 \leq \frac{2\left[\mathcal{L}(\bar{\boldsymbol{w}}^{(1)})-\mathcal{L}^*\right]}{K\gamma(T,\eta)}+\alpha(\eta,T,\delta_{\text{NTK}})+\beta(\eta,T,\delta_{\text{NTK}},\lambda), \tag{64}$$

where

$$\gamma(T,\eta)=\eta T-32\eta^2T^2L(\eta TL+1), \tag{65a}$$

$$\alpha(\eta,T,\sigma_g,\delta_{\text{NTK}})=\frac{16\eta^2T^2L\left(\eta LT+1\right)\left(\delta_{\text{NTK}}^2+\sigma_g^2\right)+2\eta T\delta_{\text{NTK}}B}{\gamma(T,\eta)}, \tag{65b}$$

$$\beta(\eta,T,\sigma_g,\delta_{\text{NTK}},\lambda)=\frac{512\eta^4T^4L^3(\eta TL+1)(\delta_{\text{NTK}}^2+\sigma_g^2+B^2)}{(1-\lambda)^2\gamma(T,\eta)}. \tag{65c}$$