FEDMG: MODEL GROWTH FOR FEDERATED LEARNING WITH SYSTEM HETEROGENEITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning often encounters the challenge of system heterogeneity, manifested in diverse local hardware resource configurations and varying client-specific model capacities. To address this, several methods have been proposed: Knowledge distillation (KD)-based methods leverage a public dataset to transfer knowledge. Low-rank training methods utilize low-rank parameters. Partial training methods extract sub-models from the global model. Unlike these methods, we introduce a novel model-growth-based federated learning method FedMG to address the challenges of system heterogeneity. We first train an atomic model that can be trained on all participating clients. Subsequently, we introduce a new model growth strategy that focuses on newly introduced parameters, then progressively build larger models. During the training process, FedMG facilitates bidirectional knowledge transfer: Small models provide their parameters to the construction of larger models, and these larger models provide feedback to the small models. FedMG successfully assigns each client a model not exceeding its local capacity, and extensive experiments demonstrate that FedMG consistently outperforms existing methods across various scenarios.

1 INTRODUCTION

Unlike traditional centralized machine learning methods, which data is stored in a single location, federated learning (FL) is a learning paradigm in which data are independently collected by multiple clients and retained locally (McMahan et al., 2017). While numerous works have explored various federated learning algorithms (Li et al., 2020b; Karimireddy et al., 2020; Acar et al., 2021; Caldarola et al., 2022), most of them potentially assume that the models share the same structure and size, which enables parameters averaging on the server side to transfer knowledge.

However, in real-world federated learning scenarios, the local hardware configurations of individual clients, including CPU, memory, and communication capabilities, often vary significantly (Luo et al., 2022; Du et al., 2026). This phenomenon is commonly referred to as **system heterogeneity** (Li et al., 2020a; Gao et al., 2022). The hardware resources limit the maximum model size that each client can process, leading to a diverse range of local model capacities. This presents a significant challenge to traditional federated learning approaches.

Several approaches have been proposed to enhance client coordination within federated learning with system heterogeneity, which can be broadly categorized as follows: 1) Knowledge Distillation (KD)-based Methods (Li & Wang, 2019; Itahara et al., 2021) utilize a public dataset to transfer knowledge by leveraging the local models' outputs, while obtaining such a high-quality public dataset is often a significant challenge. 2) Low-Rank Training methods (Yao et al., 2025; Liu et al., 2024b) let clients to update model parameters in a low-rank manner. However, these methods may lead to error accumulation or a degradation in representational capacity. 3) Partial Training (PT) methods (Caldas et al., 2018; Diao et al., 2021; Liao et al., 2025) extract appropriate size sub-models from the global model for client training. However, the parameters in different sized models may have different functions, and simply averaging them may lead to functional conflicts.

Different from existing methods, in this paper, we introduce **FedMG** (**Fed**erated Learning with **M**odel **G**rowth), a novel system heterogeneous federated learning method. FedMG starts with a small atomic model that can be trained on all clients. This allows the parameters averaged on the server are from models of the same size. Subsequently, we progressively grow small models to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

larger models, and provide each client a model that does not exceed its local capacity. In this way, the parameters of the larger model are grown from the parameters of the small model, which eliminates the need for training from scratch, thereby conserving computational resources. Furthermore, the larger models are structurally constructed from the small models, allowing the feedback of parameters from the larger models back to the small models. This process enhances the performance of the small models.

Contributions We summarize our contributions in a three-fold manner as follows:

- Unlike previous methods, we introduce a novel system heterogeneous federated learning method FedMG, which based on model growth. FedMG allows clients to learn models not exceeding their local capacities, and facilitates bidirectional knowledge transfer between models of different sizes in federated learning.
- We analyzed the impact of newly introduced parameters during model growth based on width perspective, and subsequently proposed a model growth method that specifically focuses on the growth of these newly introduced parameters.
- Extensive experiments across multiple datasets, varying client capacities, and diverse data partitioning schemes have demonstrated that FedMG significantly outperforms existing system heterogeneous federated learning methods, demonstrating its effectiveness..

2 Related Work

2.1 System Heterogeneous Federated Learning

For a comprehensive overview of federated learning, please refer to (Kairouz & McMahan, 2021; Liu et al., 2024a). Here we will introduce methods for system heterogeneous federated learning:

Knowledge Distillation (KD)-based Methods (Li & Wang, 2019; Lin et al., 2020; Itahara et al., 2021; Cho et al., 2022; Wang et al., 2025) leverage knowledge transfer through the transmission of outputs on a public dataset. The server of FedMD (Li & Wang, 2019) averages the outputs of each client, and distributes them to each client for tuning. DS-FL (Itahara et al., 2021) employs a temperature parameter to reduce the entropy of the aggregated outputs. Fed-ET (Cho et al., 2022) leverages the variance of local outputs and introduces a diversity term to enhance generalization. These methods rely on a public dataset with specific requirements and require low data heterogeneity between clients. However, achieving these conditions in practical scenarios is often challenging.

Low-Rank Training Methods (Yao et al., 2025; Mei et al., 2022; Liu et al., 2024b). FedHM (Yao et al., 2025) enables clients to train using low-rank parameters obtained through SVD decomposition, followed by server-side parameter reconstruction and averaging. FLANC (Mei et al., 2022) partitions the clients' model capacity into distinct levels, replacing parameter matrix with a shared tensor and a capacity-specific tensor. FedLMT (Liu et al., 2024b) decomposes the parameters of the later layers of the neural network into the product of two low-rank matrices, and all clients train with the same network architecture. However, low-rank training methods either utilize parameter decomposition, which introduces errors compared to the original parameter matrix, leading to error accumulation, or employ low-rank matrices to approximate the original parameter matrix, thereby reducing the representational capacity of neural networks.

Partial Training (PT) Methods (Caldas et al., 2018; Diao et al., 2021; Alam et al., 2022; Zhou et al., 2023; Wu et al., 2024; Liao et al., 2025) are the most extensively researched methods in the field. HeteroFL (Diao et al., 2021) selects a fixed sub-model for each client. FedRolex (Alam et al., 2022) uses a sliding window to select sub-models for clients. FedBRB (Liao et al., 2025) broadcasts the parameters of small models to the entire global model parameter space for averaging. However, PT methods often suffer from the following limitations: 1) Some parameters may be trained on different local capacity clients. These parameters could have distinct functions across models, but are averaged simply. 2) They do not focus on the performance of small models, and therefore, only the global model is appropriate during deployment. However, neural networks involve substantial computational costs during forward propagation (Hobbhahn & Sevilla, 2021; Wang et al., 2024), which restricts the feasibility of local deployment. FIARSE (Wu et al., 2024) addressed this issue by selecting network parameters based on their importance. However, the resulting network has an unstructured, sparse structure, necessitating specialized hardware in clients.

2.2 WIDE NEURAL NETWORKS AND MODEL GROWTH

Recently, Jian & Liu (2025) suggests that increasing network width can mitigate the impact of data heterogeneity in federtaed learning. Furthermore, wide neural networks offer several inherent advantages: First, expanding the width of a neural network requires significantly less computational resources than expanding its depth (Zagoruyko & Komodakis, 2016). Second, increasing the network width leads to a smoother and nearly convex loss surface (Zhou et al., 2020), which enhances stability during training. Growing models from width has also been explored in some studies. Net2Net (Chen et al., 2015) expands models by splitting neurons, introducing the concept of function-preserving. Staged Training (Shen et al., 2022) and Deep Fusion (Mazzawi et al., 2024) leverage this property to expand the network's width, resulting in efficient resource utilization.

3 METHODOLOGY

3.1 PRELIMINARIES

Let $\{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ represent N clients. Each client $\mathcal{C}_i$ has a local dataset S_i, $i \in [N]$. We consider system heterogeneous federated learning and define the capacity of $\mathcal{C}_i$ as Π_i, which represents the maximum model size that $\mathcal{C}_i$ can handle, Π_i varies among different clients. Each client $\mathcal{C}_i$ trains a local model f^i parameterized by θ^i, $|\theta^i| \leq \Pi_i$, $\forall i \in [N]$. Here $|\theta^i|$ denotes the number of parameters in θ^i. Define $F_i = \frac{1}{|S_i|} \sum_{s_i \in S_i} \mathcal{L}(\theta^i; s_i)$ as the local loss for $\mathcal{C}_i$, where $\mathcal{L}(\theta^i; s_i)$ represents the loss on θ^i and sample s_i. The objective of system heterogeneous federated learning is:

$$\min \sum_{i=1}^{N} p_i F_i(\theta^i), \text{ s.t. } |\theta^i| \leq \Pi_i, \forall i \in [N] \tag{1}$$

where $p_i = \frac{|S_i|}{\sum_{j=1}^{N} |S_j|}$ is the corresponding weight for $\mathcal{C}_i$ and $\sum_{i=1}^{N} p_i = 1$.

Consider a fully connected neural network f with l layers ($l > 2$), parameterized by θ. The j-th layer $g_j : \mathbb{R}^{d_{j-1}} \to \mathbb{R}^{d_j}$ of f consists of weight parameters $W^j \in \mathbb{R}^{d_j \times d_{j-1}}$ and no bias terms. To better focus on the weight parameters, we omit the activation function in our analysis. This simplifies the network structure and makes it more similar to deep linear neural networks (Saxe et al., 2013), which are considered a suitable research object for studying deep neural networks (Hardt & Ma, 2017). Given an input $h^{j-1} \in \mathbb{R}^{b \times d_{j-1}}$ with batch size of b, the output of g_j can be expressed as $h^j = g_j(h^{j-1}) = Norm(W^j h^{j-1})$, where $Norm$ represents the normalization layer of g_j[1].

Next, consider a regression task with inputs $X \in \mathbb{R}^{a \times d_0}$ and labels $Y \in \mathbb{R}^{a \times d_l}$, where a is the number of samples, and the columns in X and Y are normalized with a mean of 0 and a variance of 1. Since no specific activation function is considered, with Xavier initialization (Glorot & Bengio, 2010), the hidden layers' weights follow a Gaussian distribution $\mathcal{N}(0, \frac{1}{n})$. We further assume that after training, the mean and variance of each hidden layer parameters remain constant.

3.2 MAKE LARGER MODELS

In system heterogeneity scenarios, we can simply train a atomic network f_1 parameterized by θ_1:

$$\min \sum_{i=1}^{N} p_i F_i(\theta_1), \text{ s.t. } |\theta_1| \leq \min\{\Pi_k \mid \forall k \in [N]\} \tag{2}$$

However, this model is more likely to exhibit limited performance (Jabbar & Khan, 2015). Instead, building upon Eq. 2 , we can start with f_1 and progressively build larger networks using the model growth approach. This ensures that each client has a model not exceeding its local capacity. For simplicity, we suppose that each hidden layer of f_1 has a width of n. We now focus on the first step of model growth, which involves creating a larger model f_2 and its parameters θ_2. After (f_1, θ_1) grow to (f_2, θ_2), all clients with local capacity not less than $|\theta_2|$ will train f_2, while the others will

[1] In our paper, we primarily focus on the most widely used Batch Normalization (Ioffe & Szegedy, 2015) and Layer Normalization (Ba et al., 2016).

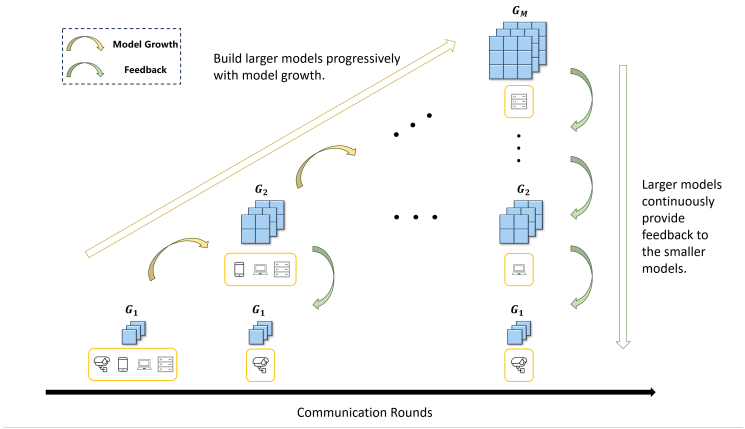


Figure 1: FedMG facilitates bidirectional knowledge transfer between models of different sizes.

continue to train f_1. We denote G_1 and G_2 as the client groups used to train f_1 and f_2, respectively. In this paper, we will use the concept of **function-preserving** (Chen et al., 2015): For model f_1 parameterized by θ_1 and model f_2 parameterized by θ_2, given a sample set $\mathcal{X}$:

$$f_1(\theta_1; x) = f_2(\theta_2; x), \forall x \in \mathcal{X} \tag{3}$$

We will grow f_1 from the perspective of width. To facilitate the discussion, we will expand each hidden layer of f_1 to the same width $n + m$ (where $m > 0$). For weight parameter matrices $W_{11} \in \mathbb{R}^{n\times n}, W_{12} \in \mathbb{R}^{n\times m}, W_{21} \in \mathbb{R}^{m\times n}, W_{22} \in \mathbb{R}^{m\times m}$, we define a growth operator $\mathbb{G}(\cdot,\cdot,\cdot,\cdot)$ that aims to construct a new parameter matrix $\mathbb{G}(W_{11}, W_{12}, W_{21}, W_{22}) \in \mathbb{R}^{(n+m)\times(n+m)}$:

$$\mathbb{G}(W_{11}, W_{12}, W_{21}, W_{22}) = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \tag{4}$$

One straightforward model growth approach involves embedding the parameters of each layer in θ_1 into the corresponding layer in θ_2, and randomly initialize the remaining parameters:

Method A: Given the weight matrix $W^{1,i} \in \mathbb{R}^{n\times n}$ in the i-th layer of f_1, we initialize $W_{11} = W^{1,i}$ and randomly initialize W_{12}, W_{21}, and W_{22}, using values drawn from zero-mean Gaussian distributions. The i-th layer weight matrix $W^{2,i}$ of f_2 is initialized with $W^{2,i} = \mathbb{G}(W_{11}, W_{12}, W_{21}, W_{22})$.

To achieve the function-preserving property of Method A, the new parameters W_{12}, W_{21}, W_{22} must be initialized with 0. However, in practice, this approach is rarely used. In Method A, the functionality provided by f_1 is embedded in W_{11}, and its output is influenced by the other non-zero parameters. This makes it impossible for Method A to completely achieve function-preserving. Considering the parameter positions, W_{22} and W_{11} are symmetric. Can we leverage W_{22} to help maintain the functionality of f_1? This motivates us to propose the following method:

Method B (Base Method in FedMG): When we have a network $f_1^{'}$ that shares the same depth as f_1, and its hidden layer width is m. If $f_1^{'}$ produces the same output as f_1 when parameterized by $\theta_1^{'}$, we can define a new strategy, which we call Method B. Specifically, based on Method A, for the i-th layer of f_2, we can replace the weights W_{22} with the same layer's parameters from $f_1^{'}$.

For Method B, if both W_{12} and W_{21} are non-zero, it is also impossible to achieve function-preserving property. However, we will demonstrate that Method B is closer to function-preserving property than Method A. We firstly propose a metric to measure how closely a model growth method approaches the function-preserving property. Given a sample x, after constructing f_2 and θ_2 using f_1 and θ_1, we define the following relationship:

$$f_2(\theta_2; x) = f_1(\theta_1; x) + \Delta \tag{5}$$

where $\Delta \in \mathbb{R}^{d_l}$ represents the difference in the outputs of f_2 and f_1. We then introduce δ to measure the degree it approaches the function-preserving property, we define:

$$\delta = E[\frac{\sum_{i=1}^{d_l} \Delta_i^2}{\sum_{i=1}^{d_l} f_1(\theta_1; x)^2}] \tag{6}$$

Specifically, δ measures the relative strength of the difference Δ compared to the original network output $f_1(\theta_1; x)$. After training, the output of f_1 should be close to the ground truth label, where each dimension is normalized. And under normal circumstances, we can treat Δ as Gaussian noise with a zero mean. To ensure the stability of neural network training and to allow newly added neurons to have comparable importance to existing neurons, we aim to maintain equal variance across the same layer of neurons before normalization layer. This allows us to achieve the following:

Proposition 1 Given that new parameters W_{12} is sampled from Gaussian distribution $\mathcal{N}(0, p\frac{1}{n})$. In Method A and Method B, we adjust the variance of remaining new parameters to ensure the expected variance of the same layer of neurons before normalization is the same. Denote the term δ produced in Method A as δ_A and the term δ produced in Method B as δ_B. Denote $\alpha = \frac{m}{n}$, then we have:

$$\frac{\delta_A}{\delta_B} = 2 + \frac{2\alpha}{1 - (\frac{1}{1+\alpha p})^{l-1}} \tag{7}$$

Remark: When $p \to 0$, Method B approaches the function-preserving property to a much higher degree than Method A. Furthermore, the lower bound of $\frac{\delta_A}{\delta_B}$ is $2+2\alpha$, which means that the variance of the noise produced by Method A is at least several times that of Method B. Therefore, when we have another network $f_1^{'}$ produces the same output as f_1, we will prioritize Method B.

3.3 Obtain Greater Gradients

In Method B, we denote the portion of parameters initialized using pre-trained parameters as U, i.e., $U = \{vec(W_{11}^{2,2}), vec(W_{22}^{2,2}), ..., vec(W_{11}^{2,l-1}), vec(W_{22}^{2,l-1})\}$. The newly randomly initialized parameters are denoted as V, i.e., $V = \{vec(W_{12}^{2,2}), vec(W_{21}^{2,2}), ..., vec(W_{12}^{2,l-1}), vec(W_{21}^{2,l-1})\}$. Here, $vec(x)$ represents the vectorization of parameter matrix x. Intuitively, U represents the pre-trained parameters, embodying functionalities learned from small model f_1. In contrast, V consists of untrained parameters that require training to acquire specific functionalities within f_2. Our objective is to accelerate the growth of V to perform its intended role within f_2. Therefore, V should ideally receive more substantial updates compared to U.

Gradmax (Evci et al., 2022) maximizes the gradients of newly introduced parameters during model growth, suggesting that increasing the update magnitude for these parameters holds the potential to yield improved performance. The gradient norm could serve as an indicator of the update magnitude. Therefore, we will focus on enhancing the gradient norm of V. To facilitate the discussion, the remaining part of this subsection will omit the subscript in W, and instead use numbers in parentheses to represent different growth methods. Next, we will introduce another growth operator $\mathbb{G}^{(2)}$, and we denote $W^{(1)}$ and $W^{(2)}$ as follows:

$$W^{(1)} = \mathbb{G}(W_{11}, W_{12}, W_{21}, W_{22}) = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \tag{8}$$

$$W^{(2)} = \mathbb{G}^{(2)}(\lambda, W_{11}, W_{12}, W_{21}, W_{22}) = \begin{bmatrix} \lambda W_{11} & W_{12} \\ W_{21} & \lambda W_{22} \end{bmatrix} \tag{9}$$

Denote (U, V) of $W^{(1)}, W^{(2)}$ as $(U^{(1)}, V^{(1)})$, $(U^{(2)}, V^{(2)})$. We now state the theorem below:

Theorem 1 During the training process started from $W^{(1)}$ and $W^{(2)}$, we assume that the gradients of the parameters are bounded, i.e., $\frac{\partial F(W)}{\partial W} \leq G$, $\forall W$. Furthermore, we assume that the loss function F is L-smooth: $\forall \phi$, ψ, there exists $L > 0$: $\|\frac{\partial F(\phi)}{\phi} - \frac{\partial F(\psi)}{\psi}\| \leq L\|\phi - \psi\|$. Under the condition $\lambda < min(\|\frac{\partial F(W^{(1)})}{\partial V^{(1)}}\|/(L\|V^{(1)}\|), 1)$, we have $\|\frac{\partial F(W^{(2)})}{\partial V^{(2)}}\| > \|\frac{\partial F(W^{(1)})}{\partial V^{(1)}}\|$, and this situation will persist for at least $T = \frac{\frac{1-\lambda}{\lambda}\|\frac{\partial F(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(1)}\|}{2\eta LG}$ iterations.

Remark It is crucial to recognize that $\frac{\partial F(W^{(1)})}{\partial V^{(1)}}$ and $V^{(1)}$ are interdependent. In particular, when $V^{(1)} = 0$, we have $\frac{\partial F(W^{(1)})}{\partial V^{(1)}} = \frac{\partial F(W^{(1)})}{\partial 0} = \frac{\partial F(W^{(1)})}{\partial U^{(1)}}$. Furthermore, leveraging the L-smoothness property, we can select an appropriate $V^{(1)}$ to ensure that both λ and T in Theorem 1 achieve their appropriate values. Detailed explanations are provided in Appendix D.1.

3.4 Model Growth in Federated Learning Scenarios

In previous part, we need a network $f_1^{'}$ with the same output as f_1. However, in federated learning, obtaining such another $f_1^{'}$ demands substantial computational resources, let alone when requiring identical outputs. Here, we also need to consider the computational cost of training after each model growth, and the fact that excessive model growth can lead to a smaller number of clients within each client group. Therefore, we should avoid performing frequent model growth in the federated learning. Considering the above factors, we propose directly reusing the network f_1, i.e., $f_1^{'} = f_1$. Each time a model is grown, the hidden size increases by 2x the original size. We now present the model growth method in FedMG:

Method C (Applied Method in FedMG): Based on Method B, we set $m = n$ and initialize $W_{11} = W_{22} = W^{1,i}$. We further multiply the parameters W_{11} and W_{22} with a scaler λ, and the weight matrix $W^{2,i}$ will have a form of Eq. 9.

When $m = n$, there is another model growth method exists:

Method D: We can set $W_{11} = W_{12} = W_{21} = W_{22} = \frac{1}{2}W^{1,i}$. It can be proven that Method D ensures function-preserving. However, we demonstrate that if θ_1 is already at a local minimum, the θ_2 generated by Method D are also at a local minimum. This implies that the newly generated f_2 are unlikely to provide a substantial improvement over f_1. Please refer to Appendix E for more details.

It is worth noting that our model growth method is also applicable to modern architectures such as convolutional neural networks (CNNs) and Transformer architectures, which are widely used in various fields. For a more detailed explanation, please refer to Appendix B.

3.5 Get Feedback from Large Models

After (f_1, θ_1) grow to (f_2, θ_2), G_1 can no longer contribute to the training of f_2. f_1 typically reaches its performance ceiling after a few communication rounds, if G_1 continue to train f_1, their performance will be limited. Recently, Xu et al. (2024) uniformly select parameters from larger pre-trained models to provide initialization to smaller models. In contrast to their approach that use larger models with parameters trained from scratch, in our research, θ_1 is placed in a fixed position in θ_2 during its initialization process. This allows us to propose the following feedback mechanism:

After E_1 communication rounds of training, we preserve the current θ_1 and denote it as $\theta_1^{(1)}$, Simultaneously, we return the $W_{11}^{2,i}$ component of θ_2 to G_1 as feedback, and then utilize this feedback to provide a new initialization for f_1. Since $W^{1,i}$ in θ_1 is scaled with λ to provide initialization for $W^{2,i}$, and its norm may change during the training of $W^{2,i}$, we use $\frac{\|W^{1,i}\|}{\|W_{11}^{2,i}\|}$ instead of $\frac{1}{\lambda}$ to rescale the parameters in $W_{11}^{2,i}$ layer by layer to get a new $\bar{W}^{1,i}$:

$$\bar{W}^{1,i} = \frac{\|W^{1,i}\|}{\|W_{11}^{2,i}\|} W_{11}^{2,i} \tag{10}$$

In this manner, G_1 will train using the new feedback parameters $\bar{W}^{1,i}$. We employ feedback every E_1 communication rounds, and obtain $(\theta_1^{(1)}, \theta_1^{(2)}, \dots)$. Existing research suggests that, averaging the parameters of models may get better performance (Frankle et al., 2020; Wortsman et al., 2022). Inspired by this, we average the parameters of $(\theta_1^{(1)}, \theta_1^{(2)}, \dots)$ to obtain the final parameters $\theta_1^{avg} = Average(\theta_1^{(1)}, \theta_1^{(2)}, \dots)$. By iteratively repeating this process, we can progressively build larger models and apply feedback. It is worth noting that each model with a given capacity requires only a small burden on the server's resources. Further details are provided in Appendix F.

Discussion Besides federated learning, a closely relative approach to our research is model growth. Both Staged Training (Shen et al., 2022) and Deep Fusion (Mazzawi et al., 2024) employ a self-

Table 1: Performance comparison with other methods on CIFAR-10 under 'a1-b1-c1'. FedLMT trains all models in the same capacity, thus, it only presents 'Local', 'Model(0.25)' and 'Global'. FedMD and DS-FL are KD-based methods and do not have global models.

Dataset	CIFAR-10($K=5$)					CIFAR-10($\alpha=0.3$)				
Method	Local	Model(0.25)	Model(0.5)	Model(1.0)	Global	Local	Model(0.25)	Model(0.5)	Model(1.0)	Global
HeteroFL	$73.00_{\pm 0.15}$	$71.51_{\pm 0.58}$	$73.17_{\pm 0.68}$	$74.37_{\pm 0.25}$	$64.17_{\pm 0.57}$	$72.16_{\pm 1.26}$	$68.34_{\pm 3.73}$	$72.37_{\pm 1.60}$	$75.56_{\pm 3.46}$	$71.07_{\pm 2.42}$
FedRolex	$58.29_{\pm 1.86}$	$41.33_{\pm 2.23}$	$55.98_{\pm 3.55}$	$78.55_{\pm 2.13}$	$57.17_{\pm 1.83}$	$53.34_{\pm 0.97}$	$35.05_{\pm 0.77}$	$48.15_{\pm 2.63}$	$78.72_{\pm 1.00}$	$57.20_{\pm 1.09}$
FIARSE	$89.18_{\pm 0.36}$	$88.75_{\pm 1.14}$	$90.04_{\pm 1.87}$	$88.57_{\pm 0.94}$	$72.31_{\pm 0.54}$	$82.82_{\pm 0.62}$	$82.31_{\pm 0.22}$	$84.07_{\pm 1.75}$	$82.44_{\pm 2.02}$	$81.82_{\pm 1.66}$
FedBRB	$75.11_{\pm 1.45}$	$70.34_{\pm 2.16}$	$77.14_{\pm 3.47}$	$77.85_{\pm 2.02}$	$65.94_{\pm 2.38}$	$63.94_{\pm 4.12}$	$57.76_{\pm 6.62}$	$62.02_{\pm 4.49}$	$72.25_{\pm 3.48}$	$68.86_{\pm 4.39}$
FedHM	$65.64_{\pm 1.20}$	$27.00_{\pm 0.76}$	$87.35_{\pm 1.55}$	$83.95_{\pm 2.55}$	$64.13_{\pm 0.76}$	$63.27_{\pm 2.45}$	$29.58_{\pm 1.87}$	$80.31_{\pm 3.30}$	$77.08_{\pm 2.90}$	$64.90_{\pm 2.92}$
FedLMT	$82.37_{\pm 2.21}$	$82.37_{\pm 2.21}$	N/A	N/A	$75.78_{\pm 2.62}$	$76.71_{\pm 0.82}$	$76.71_{\pm 0.82}$	$N/A_{\pm N/A}$	$N/A_{\pm N/A}$	$81.19_{\pm 1.08}$
FedMD	$38.20_{\pm 3.67}$	$34.43_{\pm 3.26}$	$37.13_{\pm 6.20}$	$42.97_{\pm 3.33}$	N/A	$65.63_{\pm 0.67}$	$64.76_{\pm 0.63}$	$65.75_{\pm 0.66}$	$66.34_{\pm 0.73}$	N/A
DS-FL	$48.61_{\pm 0.21}$	$46.68_{\pm 0.57}$	$48.14_{\pm 2.14}$	$50.86_{\pm 1.09}$	N/A	$60.26_{\pm 0.67}$	$59.01_{\pm 1.46}$	$59.02_{\pm 3.49}$	$62.28_{\pm 1.76}$	N/A
Ours	$\mathbf{89.91_{\pm 0.32}}$	$\mathbf{88.88_{\pm 0.62}}$	$\mathbf{90.22_{\pm 2.09}}$	$\mathbf{90.62_{\pm 1.05}}$	$\mathbf{81.38_{\pm 0.63}}$	$\mathbf{85.37_{\pm 0.88}}$	$\mathbf{83.75_{\pm 1.76}}$	$\mathbf{84.49_{\pm 1.58}}$	$\mathbf{87.68_{\pm 2.35}}$	$\mathbf{84.59_{\pm 0.36}}$

Table 2: Performance comparison on CIFAR-100 under 'a1-b1-c1'. Refer to Table 1 for 'N/A'.

Dataset	CIFAR-100($K=50$)					CIFAR-100($\alpha=0.3$)				
Method	Local	Model(0.25)	Model(0.5)	Model(1.0)	Global	Local	Model(0.25)	Model(0.5)	Model(1.0)	Global
HeteroFL	$24.21_{\pm 0.08}$	$19.52_{\pm 0.32}$	$23.65_{\pm 0.25}$	$29.71_{\pm 0.04}$	$23.80_{\pm 0.16}$	$25.03_{\pm 1.29}$	$19.72_{\pm 1.64}$	$24.23_{\pm 1.43}$	$31.23_{\pm 1.31}$	$24.65_{\pm 1.21}$
FedRolex	$18.42_{\pm 0.44}$	$11.05_{\pm 0.31}$	$15.44_{\pm 0.61}$	$28.76_{\pm 1.03}$	$18.46_{\pm 0.43}$	$17.18_{\pm 0.42}$	$8.84_{\pm 0.10}$	$13.90_{\pm 1.13}$	$29.62_{\pm 0.21}$	$17.53_{\pm 0.60}$
FIARSE	$53.25_{\pm 0.46}$	$51.60_{\pm 0.53}$	$55.36_{\pm 1.37}$	$52.86_{\pm 1.00}$	$49.64_{\pm 0.33}$	$51.57_{\pm 1.67}$	$50.08_{\pm 2.31}$	$53.65_{\pm 1.42}$	$50.96_{\pm 1.29}$	$52.93_{\pm 1.41}$
FedBRB	$29.57_{\pm 2.48}$	$27.19_{\pm 2.16}$	$30.10_{\pm 2.56}$	$31.45_{\pm 2.74}$	$29.98_{\pm 2.26}$	$27.01_{\pm 0.43}$	$23.36_{\pm 0.58}$	$26.77_{\pm 0.54}$	$31.21_{\pm 0.26}$	$28.94_{\pm 1.00}$
FedHM	$33.95_{\pm 0.28}$	$4.96_{\pm 0.36}$	$50.38_{\pm 1.12}$	$47.69_{\pm 0.73}$	$35.34_{\pm 0.23}$	$35.08_{\pm 0.58}$	$5.67_{\pm 0.62}$	$51.41_{\pm 0.77}$	$48.98_{\pm 0.03}$	$37.76_{\pm 0.43}$
FedLMT	$48.34_{\pm 0.59}$	$48.34_{\pm 0.59}$	N/A	N/A	$50.18_{\pm 0.39}$	$49.50_{\pm 0.09}$	$49.50_{\pm 0.09}$	$N/A_{\pm N/A}$	$N/A_{\pm N/A}$	$51.80_{\pm 0.36}$
FedMD	$20.37_{\pm 0.30}$	$20.83_{\pm 0.49}$	$19.59_{\pm 0.55}$	$20.66_{\pm 0.33}$	N/A	$20.15_{\pm 0.22}$	$18.60_{\pm 0.20}$	$20.36_{\pm 0.24}$	$21.31_{\pm 0.29}$	N/A
DS-FL	$17.64_{\pm 0.18}$	$16.40_{\pm 0.26}$	$18.18_{\pm 0.24}$	$18.30_{\pm 0.22}$	N/A	$23.14_{\pm 0.33}$	$21.76_{\pm 1.49}$	$23.56_{\pm 0.50}$	$24.03_{\pm 0.18}$	N/A
Ours	$\mathbf{56.27_{\pm 1.17}}$	$\mathbf{52.31_{\pm 0.31}}$	$\mathbf{57.21_{\pm 0.59}}$	$\mathbf{57.50_{\pm 0.34}}$	$\mathbf{56.00_{\pm 0.20}}$	$\mathbf{54.82_{\pm 0.67}}$	$\mathbf{52.57_{\pm 0.81}}$	$\mathbf{55.40_{\pm 1.06}}$	$\mathbf{56.49_{\pm 0.26}}$	$\mathbf{57.53_{\pm 0.50}}$

fusion model growth strategy, but they typically assume that newly added parameters are zero during analysis. In contrast, our approach considers the fusion of models with varying sizes, and we investigate the impact of non-zero initial values for newly added parameters. To accelerate the growth of these parameters, we also scale other parameters with λ and provide a theoretical analysis.

4 EXPERIMENTS

Datasets and Models. We evaluated our method on image classification tasks using the CIFAR-10 and CIFAR-100 datasets (Krizhevsky et al., 2009), and on a masked language modeling task using the WikiText2 dataset (Merity et al., 2016). Following Alam et al. (2022), for CIFAR-10 and CIFAR-100, we employ a PreResNet18 model with static batch normalization and a scaling factor applied after each convolutional layer. For WikiText2, we employ a Transformer with a 15% masking rate.

Data Heterogeneity. For image classification tasks, we employ a pathological and Dirichlet non-IID data partitioning scheme. In the pathological scheme, we restrict the number of labels held by each client to a maximum of K. For CIFAR-10 and CIFAR-100, we set $K = 5$ and $K = 50$, respectively. In the Dirichlet scheme, we set the Dirichlet parameter $\alpha = 0.3$. For Wikitext2, following Diao et al. (2021), we randomly and uniformly distribute the data across all clients.

System Heterogeneity. Following Diao et al. (2021), we use the following notation to represent the different model capacities: 'a': 1.0; 'b': 0.5; 'c': 0.25; 'd': 0.125, where each represents the ratio of the hidden layer width to the global model width. We use a number after each letter to indicate the sample weight of each local capacity. For example, 'a1-c1-d1' means clients with local capacities $1.0, 0.25, 0.125$ have the same weight and no client has 0.5 local capacity.

Baselines. We compare FedMG with other system heterogeneous FL methods, including 1) PT methods: HeteroFL (Diao et al., 2021), FedRolex (Alam et al., 2022), FIARSE (Wu et al., 2024),

Table 3: Communication costs and FLOPs comparison with other system heterogeneous FL methods on CIFAR-10 and CIFAR-100 under 'a1-b1-c1'. FedLMT trains a low-rank global model with the smallest capacity among clients. However, this approach inherently suffers from the performance ceiling imposed by the inherent limitations of a low-rank model.

Dataset	CIFAR-10				CIFAR-100			
Method	PT-based	FedHM	FedLMT	Ours	PT-based	FedHM	FedLMT	Ours
COMM(GB)	1148.4	1098.4	93.8	623.4	1339.8	1281.5	109.4	541.4
FLOPs(1e15)	4.41	7.18	0.42	2.41	5.15	8.38	0.49	2.09

Table 4: Performance comparison with other system heterogeneous FL methods on WikiText2 under 'a1-b1', a lower perplexity indicates better performance. In common settings, KD-based methods do not participate in this task. The perplexity for both FIARSE and FedHM are above several hundred, so they are not displayed. Here, as the data is divided in an IID manner, the first two lines show the perplexity of the model on the global test dataset, and the last row shows their average.

Model capacity	HeteroFL	FedRolex	FedLMT	FedMG(ours)
0.5	$3.48_{\pm 0.09}$	$19.30_{\pm 6.72}$	$3.55_{\pm 0.03}$	$\mathbf{3.29_{\pm 0.02}}$
1.0	$3.04_{\pm 0.02}$	$3.16_{\pm 0.03}$	N/A	$\mathbf{2.99_{\pm 0.01}}$
Global	$3.26_{\pm 0.06}$	$11.23_{\pm 3.35}$	$3.55_{\pm 0.03}$	$\mathbf{3.14_{\pm 0.01}}$

FedBRB (Liao et al., 2025); 2) Low-rank training methods: FedHM (Yao et al., 2025), FedLMT (Liu et al., 2024b); 3) KD-based methods: FedMD (Li & Wang, 2019), DS-FL (Itahara et al., 2021).

Setup. For all tasks, we set up 100 clients and set the client participation rate to 10%. We set 3000, 3500 and 200 communication rounds for CIFAR-10, CIFAR-100 and WikiText2, respectively. In practice, symmetry should be avoided in the initialization of networks parameters (Goodfellow et al., 2016), so we assign weights λ_1 to W_{11} and λ_2 to W_{22} in Eq. 9, where $\lambda_1 \approx \lambda_2$. If no special instructions are given, we use $(\lambda_1, \lambda_2) = (0.4, 0.44)$. See Appendix G.2 for more details.

Evaluation Metrics. For image classification tasks, we evaluate performance using accuracy. For each capacity level, we report the average local test accuracy. 'Local' refers to the average local test accuracy across all clients. Moreover, we create a unified global test dataset by combining the test sets from all clients. 'Global' refers to the average test accuracy achieved by each size global model on the global test dataset. For WikiText2, we use perplexity as our evaluation metric.



(a) $(\lambda_1, \lambda_2) = (0, 4, 0.44)$ (b) $(\lambda_1, \lambda_2) = (0, 5, 0.55)$

Figure 2: The value of $(\|\frac{\partial F(W^{(2)})}{\partial V^{(2)}}\|)/(\|\frac{\partial F(W^{(1)})}{\partial V^{(1)}}\|)$ remained greater than 1 for a period of time.

4.1 Experimental Results

Table 1 and Table 2 present the performance of FedMG compared with other system heterogeneous FL methods under the capacity setting 'a1-b1-c1'. Notably, methods such as FedRolex and FedHM don't exhibit ideal local performance when the model capacity is 'c', while in FedMG, the local performance across different capacities is superior to that of other methods. This outcome validates our objective: each client obtains a model not exceeding its local capacity, thereby achieving local performance that surpasses existing system heterogeneous federated learning methods. Moreover, FedMG achieves better performance than other methods in 'Global' accuracy. In contrast to FedLMT, FedMG's model growth strategy avoids the performance limitations that arise from using a single, small model on all clients. In addition, we present a comparison of communication costs and FLOPs in Table 3. Compared to the most commonly used PT methods, FedMG can save almost half of the communication costs and FLOPs, while FedLMT ensures that all clients use the same size low-rank global model, which inevitably limits the performance to the smallest client capacity.

To verify Theorem 1, we conduct an experiment in $W^{(1)}$, and conduct experiments in $W^{(2)}$ using $(\lambda_1, \lambda_2) = (0.4, 0.44)$ and $(0.5, 0.55)$. We growing a 'c' sized model to a 'b' sized model in CIFAR-10 ($\alpha = 0.3$). The implementation details are provided in Appendix H.3. Figure 2 shows that $(\|\frac{\partial F(W^{(2)})}{\partial V^{(2)}}\|)/(\|\frac{\partial F(W^{(1)})}{\partial V^{(1)}}\|)$ is greater than 1 for a sustained period, thus verifying Theorem 1.

Tabel 4 presents the experimental results on WikiText2 under the capacity setting 'a1-b1'. We implemented both FIARSE and FedHM, but their final perplexity scores exceeded several hundred,

Table 5: 'Global' accuracy comparison with other system heterogeneous FL methods on CIFAR-10 ($\alpha = 0.3$) and various clients model capacity.

Method	HeteroFL	FedRolex	FIARSE	FedBRB	FedHM	FedLMT	Ours
a1-b1-c1-d1	$67.15_{\pm 2.12}$	$48.33_{\pm 0.76}$	$79.76_{\pm 3.67}$	$55.93_{\pm 1.36}$	$49.48_{\pm 0.57}$	$66.47_{\pm 1.93}$	$80.16_{\pm 0.33}$
a1-b1-d1	$67.06_{\pm 1.26}$	$50.95_{\pm 1.91}$	$80.46_{\pm 1.29}$	$60.84_{\pm 2.29}$	$58.55_{\pm 0.42}$	$66.47_{\pm 1.93}$	$81.08_{\pm 0.06}$
a1-c1-d1	$68.46_{\pm 0.69}$	$44.95_{\pm 1.84}$	$77.38_{\pm 1.73}$	$52.30_{\pm 1.24}$	$41.58_{\pm 0.31}$	$66.47_{\pm 1.93}$	$79.27_{\pm 0.09}$
a1-b1	$73.71_{\pm 0.39}$	$72.30_{\pm 1.34}$	$84.82_{\pm 3.92}$	$78.02_{\pm 1.70}$	$84.41_{\pm 1.01}$	$85.52_{\pm 2.03}$	$88.19_{\pm 0.32}$
a1-c1	$73.06_{\pm 1.30}$	$56.37_{\pm 1.21}$	$82.06_{\pm 2.17}$	$67.35_{\pm 4.31}$	$54.60_{\pm 1.87}$	$81.19_{\pm 1.08}$	$84.57_{\pm 0.31}$
a1-d1	$71.32_{\pm 1.56}$	$52.16_{\pm 0.16}$	$79.37_{\pm 1.97}$	$42.92_{\pm 3.60}$	$48.20_{\pm 0.90}$	$66.47_{\pm 1.93}$	$80.15_{\pm 0.23}$

Table 6: Performance comparison with different combinations of λ_1 and λ_2.

(λ_1, λ_2)	Local	Model(0.25)	Model(0.5)	Model(1.0)	Global
$(0.2, 0.22)$	$73.99_{\pm 1.87}$	$53.01_{\pm 4.25}$	$82.22_{\pm 1.96}$	$87.12_{\pm 0.44}$	$62.47_{\pm 2.05}$
$(0.3, 0.33)$	$87.82_{\pm 0.16}$	$85.23_{\pm 0.83}$	$88.73_{\pm 0.36}$	$89.45_{\pm 0.80}$	$77.91_{\pm 0.51}$
$(0.4, 0.44)$	$89.91_{\pm 0.32}$	$88.88_{\pm 0.62}$	$90.22_{\pm 2.09}$	$90.62_{\pm 1.05}$	$81.38_{\pm 0.63}$
$(0.5, 0.55)$	$89.49_{\pm 0.33}$	$88.05_{\pm 1.48}$	$90.58_{\pm 0.41}$	$89.89_{\pm 0.92}$	$80.37_{\pm 1.11}$
$(0.7, 0.77)$	$89.25_{\pm 0.31}$	$89.26_{\pm 0.19}$	$89.75_{\pm 0.55}$	$88.80_{\pm 1.35}$	$80.01_{\pm 0.60}$
$(0.9, 0.99)$	$88.74_{\pm 0.94}$	$88.78_{\pm 1.87}$	$90.02_{\pm 1.13}$	$87.46_{\pm 0.80}$	$78.57_{\pm 0.68}$
$(1.0, 1.0)$	$89.26_{\pm 0.37}$	$90.17_{\pm 0.66}$	$90.26_{\pm 0.91}$	$87.49_{\pm 1.85}$	$78.96_{\pm 0.53}$

so we did not include them in the table. In contrast, FedMG still outperforms other methods. This further demonstrates the generality of FedMG for various model architectures.

We further conduct experiments under various client capacities. Table 5 presents the performance on the CIFAR-10, using a Dirichlet distribution with $\alpha = 0.3$. The results demonstrate that FedMG is capable of effectively handling system heterogeneity arising from varying model capacities. Notably, even when some groups with specific capacities may not have clients training on, they still participate in the FedMG's feedback process without updating their model parameters by training.

4.2 Ablation Study

The Impact of λ We set different values for λ_1 ranging from 0.2 to 0.9, with $\lambda_2 = (1+10\%)\lambda_1$, and we also considered the unscaled λ case where $\lambda_1 = \lambda_2 = 1.0$. The results are presented in Table 6. When the parameters λ_1 and λ_2 are small, the smaller models receive feedback and averaging their trained models does not achieve good performance. Conversely, when λ_1 and λ_2 are close to or equal to 1, the performance of the larger models is limited. This observation validates the importance of focusing on the newly added parameters in FedMG. From the results, it can be observed that FedMG is robust to the selection of parameter λ_1, and an appropriate choice is $\lambda_1 \in [0.4, 0.7]$.

The Effectiveness of Feedback To validate the effectiveness of feedback in FedMG, we conducted an ablation study comparing performance with and without feedback, on the CIFAR-10 dataset with $K = 5$. As shown in Table 7, our approach outperforms the method that without feedback. This demonstrates that feedback from larger models can effectively transfer knowledge to smaller models.

5 Conclusion

In this paper, we propose FedMG, a novel approach for system heterogeneous federated learning. FedMG first analyzes the model growth process and introduces a more efficient growth strategy, progressively constructing large models from small models while facilitating bidirectional parameter transfer between them. To the best of our knowledge, this is the first time that a model growth method has been proposed to address the problem of system heterogeneity in federated learning.

Table 7: The Effectiveness of Feedback.

Method	Model(0.25)	Model(0.5)	Global
without feedback	$84.43_{\pm 2.77}$	$88.06_{\pm 1.10}$	$76.73_{\pm 1.11}$
with feedback	$\mathbf{88.88_{\pm 0.62}}$	$\mathbf{90.22_{\pm 2.09}}$	$\mathbf{81.38_{\pm 0.63}}$

AI USE STATEMENT

This section does not count toward the page limit.

In this work, we used generative AI tools for aiding or polishing writing. We have not used generative AI tools for retrieval or discovery; We have not used generative AI tools for research ideation or execution; We have not used generative AI tools for drafting sections of the paper; Generating synthetic datasets and proving mathematical claims are not applicable to this work. We have reviewed all AI-assisted work. We use LLM to check for spelling errors and clarity in sentence expression. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv:2111.04263*, 2021.

Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. FedRolex: Model-heterogeneous federated learning with rolling sub-model extraction. In *Advances in Neural Information Processing Systems 35 (NeurIPS'22)*, pp. 29677–29690, 2022.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

Debora Caldarola, Barbara Caputo, and Marco Ciccone. Improving generalization in federated learning by seeking flat minima. In *Proceedings of the 17th European Conference on Computer Vision (ECCV'22)*, pp. 654–672, 2022.

Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv:1812.07210*, 2018.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv:1511.05641*, 2015.

Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI'22)*, pp. 2881–2887, 2022.

Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *Proceedings of the 9th International Conference on Learning Representations (ICLR'21)*, 2021.

Haizhou Du, Lixin Huang, Zonghan Wu, Nitin Bisht, Huan Huo, and Xiufeng Liu. FedEMoE: Improving personalization on heterogeneous federated learning via elastic mixture of experts architecture. In *Proceedings of the 43rd International Conference on Machine Learning (ICML'26)*, 2026.

Utku Evci, Bart van Merrienboer, Thomas Unterthiner, Max Vladymyrov, and Fabian Pedregosa. Gradmax: Growing neural networks using gradient information. In *Proceedings of the 10th International Conference on Learning Representations (ICLR'22)*, 2022.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, pp. 3259–3269, 2020.

Dashan Gao, Xin Yao, and Qiang Yang. A survey on heterogeneous federated learning. *arXiv:2210.04505*, 2022.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Moritz Hardt and Tengyu Ma. Identity matters in deep learning. 2017.

Marius Hobbhahn and Jaime Sevilla. What's the backward-forward flop ratio for neural networks?, 2021. URL https://epochai.org/blog/backward-forward-FLOP-ratio.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456, 2015.

Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22(1):191–205, 2021.

Haider Khalaf Jabbar and Dr Rafiqul Zaman Khan. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). 2015.

Like Jian and Dong Liu. Widening the network mitigates the impact of data heterogeneity on fedavg. In *Proceedings of the 42nd International Conference on Machine Learning (ICML'25)*, 2025.

Peter Kairouz and H Brendan McMahan. Advances and open problems in federated learning. *Foundations and trends in machine learning*, 14(1-2):1–210, 2021.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, pp. 5132–5143, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4), 2009.

Daliang Li and Junpu Wang. FedMD: Heterogenous federated learning via model distillation. *arXiv:1910.03581*, 2019.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys'20)*, pp. 429–450, 2020b.

Tianchi Liao, Ziyue Xu, Qing Hu, Hong-Ning Dai, Huaiwei Huang, Zibin Zheng, and Chuan Chen. FedBRB: A solution to the small-to-large scenario in device-heterogeneity federated learning. *IEEE Transactions on Mobile Computing*, 2025.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, pp. 2351–2363, 2020.

Bingyan Liu, Nuoyan Lv, Yuanchun Guo, and Yawen Li. Recent advances on federated learning: A systematic survey. *Neurocomputing*, 597:128019, 2024a.

Jiahao Liu, Yipeng Zhou, Di Wu, Miao Hu, Mohsen Guizani, and Quan Z Sheng. FedLMT: Tackling system heterogeneity of federated learning via low-rank model training with theoretical guarantees. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, 2024b.

Bing Luo, Wenli Xiao, Shiqiang Wang, Jianwei Huang, and Leandros Tassiulas. Tackling system and statistical heterogeneity for federated learning with adaptive client sampling. In *IEEE INFOCOM 2022-IEEE conference on computer communications*, pp. 1739–1748, 2022.

Hanna Mazzawi, Javier Gonzalvo, Michael Wunder, Sammy Jerome, and Benoit Dherin. Deep fusion: Efficient network training via pre-trained initializations. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*, pp. 35225–35239, 2024.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS'17)*, pp. 1273–1282, 2017.

Yiqun Mei, Pengfei Guo, Mo Zhou, and Vishal Patel. Resource-adaptive federated learning with all-in-one neural composition. In *Advances in Neural Information Processing Systems 35 (NeurIPS'22)*, pp. 4270–4284, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv:1609.07843*, 2016.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. Staged training for transformer language models. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*, pp. 19893–19908, 2022.

Haolin Wang, Xuefeng Liu, Jianwei Niu, Wenkai Guo, and Shaojie Tang. Why go full? elevating federated learning through partial network updates. In *Advances in Neural Information Processing Systems 38 (NeurIPS'24)*, pp. 99773–99799, 2024.

Zichen Wang, Feng Yan, Tianyi Wang, Cong Wang, Yuanchao Shu, Peng Cheng, and Jiming Chen. Fed-DFA: Federated distillation for heterogeneous model fusion through the adversarial lens. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI'25)*, pp. 21429–21437, 2025.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*, pp. 23965–23998, 2022.

Feijie Wu, Xingchen Wang, Yaqing Wang, Tianci Liu, Lu Su, and Jing Gao. FIARSE: Model-heterogeneous federated learning via importance-aware submodel extraction. In *Advances in Neural Information Processing Systems 38 (NeurIPS'24)*, pp. 115615–115651, 2024.

Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Lingjie Liu, Zhuang Liu, et al. Initializing models with larger ones. In *International Conference on Learning Representations*, volume 2024, pp. 14336–14351, 2024.

Dezhong Yao, Wanning Pan, Yuexin Shi, Michael J O'Neill, Yutong Dai, Yao Wan, Peilin Zhao, Hai Jin, and Lichao Sun. FedHM: Efficient federated learning for heterogeneous models via low-rank factorization. *Artificial Intelligence*, 344:104333, 2025.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv:1605.07146*, 2016.

Denny Zhou, Mao Ye, Chen Chen, Tianjian Meng, Mingxing Tan, Xiaodan Song, Quoc Le, Qiang Liu, and Dale Schuurmans. Go wide, then narrow: Efficient training of deep thin networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, pp. 11546–11555, 2020.

Hanhan Zhou, Tian Lan, Guru Prasadh Venkataramani, and Wenbo Ding. Every parameter matters: Ensuring the convergence of federated learning with dynamic heterogeneous models reduction. In *Advances in Neural Information Processing Systems 36 (NeurIPS'23)*, pp. 25991–26002, 2023.

A FEDMG ALGORITHM

We provide the pseudo code for algorithm FedMG here:

Algorithm 1 FedMG: Federated Learning with **M**odel **G**rowth

Input: total rounds T, max model capacity level M, model growth rounds $\mathcal{T} = \{T_1, T_2, \ldots, T_{M-1}\}$, clients $\{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ with local data $\{S_1, \ldots, S_N\}$, local epoch E, feedback interval rounds $E_1, E_2, \ldots, E_{M-1}$.
Output: model $f_1, \ldots, f_M$ with parameters $\theta_1^{avg}, \ldots, \theta_{M-1}^{avg}, \theta_M$.
Randomly initialize model f_1 with parameters θ_1, current model capacity level $Level = 1$, client groups $G_1 = \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}, G_i = \{\}, \forall i \in [2, M]$.
for $t = 1$ **to** T **do**

  for $i = 1$ **to** $Level - 1$ **do**
    if $t - T_i \; mod \; E_i = 0$ **then**
      Reserve current parameters θ_i in server, gets feedback from θ_{i+1}.
    end if
  end for

  if $t \in \mathcal{T}$ **then**
    Construct $(f_{Level+1}, \theta_{Level+1})$ with $(f_{Level}, \theta_{Level})$.
    $G_{Level+1} = \{\mathcal{C}_i \mid \forall \, \mathcal{C}_i \in G_{Level} \; with \; \Pi_i \geq |\theta_{Level+1}|\}$.
    $G_{Level} = G_{Level} \setminus G_{Level+1}$.
    $Level = Level + 1$.
  end if

  Server randomly selects clients $\mathcal{R}_t$ from $\{\mathcal{C}_1, ..., \mathcal{C}_N\}$.
  for $i = 1$ **to** $Level$ **do**
    if $Q_i = G_i \cap \mathcal{R}_t$ is not ϕ **then**
      Server broadcast θ_i to all clients in Q_i.
      All clients in Q_i updates θ_i with their local data for E epochs and send the parameters to the server.
      Server aggregates the models and update θ_i.
    end if
  end for

end for
Generate $\theta_i^{avg}, i \in [M-1]$ from server. Return $f_1, \ldots, f_M$ and $\theta_1^{avg}, \ldots, \theta_{M-1}^{avg}, \theta_M$.

B DETAILS FOR METHOD A, METHOD B AND MODERN NETWORK ARCHITECTURES

For the first layer, upon obtaining the pre-trained parameters W^1, we define the growth matrix $\mathbb{G}_{first-layer}$ as:

$$\mathbb{G}_{first-layer}(W_{11}, W_{12}) = \begin{bmatrix} W_{11} \\ W_{12} \end{bmatrix} \tag{11}$$

In Method A, we set W_{11} with the pre-trained parameters W^1 and W_{12} with randomly initialized parameters. The variance of each row in the first layer of the large model is the same. In Method B, we set both $W_{11} = W_{12} = W^1$. For the pre-trained weights γ^1 and bias β^1 in normalization layers, we employ a similar initialization strategy in the large model, for Method A, we set $\gamma_{12} = 1$ and $\beta_{12} = 0$.

For the last layer, upon obtaining the pre-trained parameters W^l, we define the growth matrix $\mathbb{G}_{last-layer}$ as:

$$\mathbb{G}_{last-layer}(W_{11}, W_{12}) = [W_{11} \quad W_{12}] \tag{12}$$

In Method A, we set W_{11} with the pre-trained parameters W^l and W_{12} with randomly initialized parameters. The variance of each column in the last layer of the large model is the same. In Method B, like previous model growth methods (Shen et al., 2022; Mazzawi et al., 2024), we set both $W_{11} = W_{12} = \frac{W^l}{2}$.

For the convolution kernel $W \in \mathbb{R}^{s\times t\times h\times w}$ in convolution neural networks (CNNs), we still use the similar method for model growth. Specifically, we focus on the growth of the first two dimensions of the kernel, while treating the parameters of the last two dimensions as a unified entity. For the embedding layer and positional embedding layer within Transformer architecture, we use a similar processing method as defined in Eq. 11.

C Proof For Proposition

In this section, we consider a hidden layer, omitting the subscript indicating the layer number. In f_1, we represent the input to this layer as x and the output as a. We assume that the original width n and the newly expanded width $n+\alpha n$ are sufficiently large. Since we are not considering activation functions, for the original small network f_1, we have:

$$\begin{aligned} y_j &= \sum_{k=1}^{n} w_{j,k}x_k, \forall j \in [n] \\ a_j &= \frac{y_j - mean(y)}{\sqrt{Var(y)}}, \forall j \in [n] \end{aligned} \tag{13}$$

For Batch Normalization layers, $mean(y), Var(y)$ denote the mean and variance of the j-th dimension output of all samples in the layer. For Layer Normalization layers, $mean(y), Var(y)$ denote the mean and variance of the output of all neurons in the sample in the layer.

For the newly grown network f_2, we have:

$$\begin{aligned} y_j^{'} &= \sum_{k=1}^{n+\alpha n} w_{j,k}x_k^{'}, \forall j \in [n+\alpha n] \\ a_j^{'} &= \frac{y_j^{'} - mean(y^{'})}{\sqrt{Var(y^{'})}}, \forall j \in [n+\alpha n] \end{aligned} \tag{14}$$

We assume that, following a linear mapping layer and prior to entering a normalization layer, the variances of the outputs from the original neurons and the newly added neurons are equal. This also implies that, after passing through the normalization layer, the variance of each neuron's output is equal. Since $x^{'}$ represents the output of the previous layer, we have $Var(x_k^{'}) = 1$ for $k \in [n+\alpha n]$ (where we previously assumed that the normalization layer's parameters γ and β remain unchanged before and after training).

This implies that, for Method A and Method B, the distribution of the newly introduced parameters must adhere to the following properties, respectively:

For Method A:

Assume that the parameters W_{12}, W_{21}, W_{22} are sampled from independent normal distributions $\mathcal{N}(0, p\frac{1}{n}), \mathcal{N}(0, q\frac{1}{n}), \mathcal{N}(0, r\frac{1}{n})$, respectively.

For the first n neurons, we have:

$$\begin{aligned} Var(y_j^{'}) &= Var(\sum_{k=1}^{n+\alpha n} w_{j,k}x_k^{'}) = Var(\sum_{k=1}^{n} w_{j,k}x_k^{'} + \sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k^{'}) \\ &= n \times \frac{1}{n} \times 1 + \alpha n \times p\frac{1}{n} \times 1 = 1 + \alpha p, \ \forall j \in [n] \end{aligned} \tag{15}$$

For the last αn neurons, we have:

$$\begin{aligned} Var(y_j^{'}) &= Var(\sum_{k=1}^{n+\alpha n} w_{j,k}x_k^{'}) = Var(\sum_{k=1}^{n} w_{j,k}x_k^{'} + \sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k^{'}) \\ &= n \times q\frac{1}{n} \times 1 + \alpha n \times r\frac{1}{n} \times 1 = q + \alpha r,\ \forall j \in [n+1 : n+\alpha n] \end{aligned} \tag{16}$$

Consequently, the following equation should be satisfied:

$$1 + \alpha p = q + \alpha r \tag{17}$$

For Method B:

Assume that the parameters W_{12}, W_{21} are sampled from independent normal distributions $\mathcal{N}(0, p\frac{1}{n}), \mathcal{N}(0, q\frac{1}{n})$, respectively.

For the first n neurons, we have:

$$\begin{aligned} Var(y_j^{'}) &= Var(\sum_{k=1}^{n+\alpha n} w_{j,k}x_k^{'}) = Var(\sum_{k=1}^{n} w_{j,k}x_k^{'} + \sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k^{'}) \\ &= n \times \frac{1}{n} \times 1 + \alpha n \times p\frac{1}{n} \times 1 = 1 + \alpha p,\ \forall j \in [n] \end{aligned} \tag{18}$$

For the last αn neurons, due to $Var(w_{j,k}) = \frac{1}{\alpha n}, \forall k \in [n+1 : n+\alpha n]$, we have:

$$\begin{aligned} Var(y_j^{'}) &= Var(\sum_{k=1}^{n+\alpha n} w_{j,k}x_k^{'}) = Var(\sum_{k=1}^{n} w_{j,k}x_k^{'} + \sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k^{'}) \\ &= n \times q\frac{1}{n} \times 1 + \alpha n \times \frac{1}{\alpha n} \times 1 = q + 1,\ \forall j \in [n+1 : n+\alpha n] \end{aligned} \tag{19}$$

Consequently, the following equation should be satisfied:

$$q = \alpha p \tag{20}$$

In the newly grown network f_2, for the first n dimensions of the input $x^{'}$, we assume that $x_k^{'} = sx_k + tz_k, \forall k \in [n]$, where $s^2 + t^2 = 1$ and z_k is a random value sampled from a standard normal distribution. For the first hidden layer, we have $s = 1, t = 0$.

We use a recursive approach to study the $s^{'}$ and $t^{'}$ that satisfy the first n dimensions of the output $a^{'}$ of the current layer, which is the input of the next layer, while proving that $(s^{'})^2 + (t^{'})^2 = 1$.

For the output $y_j^{'}$ of $x^{'}$ after a linear mapping, where $j \in [n]$, we have:

$$\begin{aligned} y_j^{'} = \sum_{k=1}^{n+\alpha n} w_{j,k}x_k^{'} &= \sum_{k=1}^{n} w_{j,k}x_k^{'} + \sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k^{'} = \sum_{k=1}^{n} w_{j,k}(sx_k + tz_k) + \sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k^{'} \\ &= s\sum_{k=1}^{n} w_{j,k}x_k + t\sum_{k=1}^{n} w_{j,k}z_k + \sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k^{'} \\ &= sy_j + t\sum_{k=1}^{n} w_{j,k}z_k + \sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k^{'} \end{aligned} \tag{21}$$

C.1 THE CASE OF BATCH NORMALIZATION

For the last αn dimensions of $x^{'}$, after passing through the batchnorm layer in the previous layer, we can consider $x_k^{'}, k \in [n+1 : n+\alpha n]$ to be random values sampled from a standard normal distribution.

Therefore:

$$\begin{aligned} E[c_j] &= E[\sum_{k=1}^{n} w_{j,k} z_k] = 0 \\ Var[c_j] &= Var[\sum_{k=1}^{n} w_{j,k} z_k] = \sum_{k=1}^{n} Var[w_{j,k} z_k] \\ &= \sum_{k=1}^{n} Var[w_{j,k}] Var[z_k] = n \times \frac{1}{n} \times 1 = 1 \end{aligned} \tag{22}$$

$$\begin{aligned} E[d_j] &= E[\sum_{k=n+1}^{n+\alpha n} w_{j,k} x_k^{'}] = 0 \\ Var[d_j] &= Var[\sum_{k=n+1}^{n+\alpha n} w_{j,k} x_k^{'}] = \sum_{k=n+1}^{n+\alpha n} Var[w_{j,k} x_k^{'}] \\ &= \sum_{k=n+1}^{n+\alpha n} Var[w_{j,k}] Var[x_k^{'}] = \alpha n \times p \frac{1}{n} \times 1 = \alpha p \end{aligned} \tag{23}$$

Then, $c_j \sim \mathcal{N}(0,1),\ \forall j \in [n]$, $d_j \sim \mathcal{N}(0, \alpha p),\ \forall j \in [n]$.

The random variables $y_j^{'}$, c_j, and d_j are independent, and denote $\sum Var = Var(sy_j) + Var(tc_j) + Var(d_j)$, we have:

$$\begin{aligned} a_j^{'} &= \frac{y_j^{'} - mean(y^{'})}{Var(y^{'})} = \frac{sy_j + tc_j + d_j - mean[sy] - mean[tc] - mean[d]}{\sqrt{\sum Var}} \\ &= \frac{sy_j - mean[sy]}{\sqrt{Var(sy)}} \frac{\sqrt{Var(sy)}}{\sqrt{\sum Var}} + \frac{tc_j - mean[tc]}{\sqrt{Var(tc)}} \frac{\sqrt{Var(tc)}}{\sqrt{\sum Var}} + \frac{d_j - mean[d]}{\sqrt{Var(d)}} \frac{\sqrt{Var(d)}}{\sqrt{\sum Var}} \\ &= \frac{y_j - mean[y]}{\sqrt{Var(y)}} \frac{\sqrt{Var(sy)}}{\sqrt{\sum Var}} + \frac{tc_j - mean[tc]}{\sqrt{Var(tc)}} \frac{\sqrt{Var(tc)}}{\sqrt{\sum Var}} + \frac{d_j - mean[d]}{\sqrt{Var(d)}} \frac{\sqrt{Var(d)}}{\sqrt{\sum Var}} \\ &= \frac{s}{\sqrt{s^2 + t^2 + \alpha p}} a_j + \frac{t}{\sqrt{s^2 + t^2 + \alpha p}} \frac{tc_j - mean[tc_j]}{\sqrt{Var(tc_j)}} + \frac{\sqrt{\alpha p}}{\sqrt{s^2 + t^2 + \alpha p}} \frac{d_j - mean[d_j]}{\sqrt{Var(d_j)}} \\ &= \frac{s}{\sqrt{s^2 + t^2 + \alpha p}} a_j + \frac{\sqrt{t^2 + \alpha p}}{\sqrt{s^2 + t^2 + \alpha p}} z_j \end{aligned} \tag{24}$$

In the above equation, $mean(c)$ and $Var(c)$ represent the mean and variance of all samples of c_j at this layer. Similarly, $mean(d)$ and $Var(d)$ represent the mean and variance of all samples of d_j at this layer. z_j is a random variable following a standard normal distribution. The last equation arises because $\frac{tc_j - mean[tc]}{\sqrt{Var(tc)}}$ and $\frac{d_j - mean[d]}{\sqrt{Var(d)}}$ are two independent random variables following a standard normal distribution. Therefore, the sum of the latter two terms can be interpreted as a random variable sampled from $\mathcal{N}(0, \frac{t^2}{s^2+t^2+\alpha p} + \frac{\alpha p}{s^2+t^2+\alpha p})$. This leads to a recursive relationship, combined

with the constraint $s^2 + t^2 = 1$:

$$\begin{aligned} s^{'} &= \frac{s}{\sqrt{s^2 + t^2 + \alpha p}} = \frac{s}{\sqrt{1 + \alpha p}} \\ t^{'} &= \frac{\sqrt{t^2 + \alpha p}}{\sqrt{s^2 + t^2 + \alpha p}} = \frac{\sqrt{t^2 + \alpha p}}{\sqrt{1 + \alpha p}} \\ (s^{'})^2 &+ (t^{'})^2 = 1 \end{aligned} \tag{25}$$

C.2 The case of Layer Normalization

Considering the scenario of Layernorm, within the small network f_1, we have:

$$mean(x_1, \ldots, x_n) = 0 \tag{26}$$

$$Var(x_1, \ldots, x_n) = \frac{1}{n}\sum_{i=1}^{n}(x_i - mean(x_1, \ldots, x_n))^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i)^2 = 1 \tag{27}$$

In the newly grown network f_2, after the output of the previous layer's Layer Normalization, there is:

$$mean(x_1^{'}, \ldots, x_{n+\alpha n}^{'}) = 0 \tag{28}$$

$$\begin{aligned} 1 = Var(x_1^{'}, \ldots, x_{n+\alpha n}^{'}) &= \frac{1}{n + \alpha n}\sum_{i=1}^{n+\alpha n}(x_i^{'} - mean(x_1^{'}, \ldots, x_{n+\alpha n}^{'}))^2 \\ &= \frac{1}{n + \alpha n}\sum_{i=1}^{n+\alpha n}(x_i^{'})^2 = \frac{1}{n + \alpha n}\sum_{i=1}^{n}(x_i^{'})^2 + \frac{1}{n + \alpha n}\sum_{i=n+1}^{n+\alpha n}(x_i^{'})^2 \\ &= \frac{1}{n + \alpha n}\sum_{i=1}^{n}(sx_k + tz_k)^2 + \frac{1}{n + \alpha n}\sum_{i=n+1}^{n+\alpha n}(x_i^{'})^2 \\ &= \frac{1}{n + \alpha n}\sum_{i=1}^{n}s^2x_k^2 + \frac{1}{n + \alpha n}\sum_{i=1}^{n}2stx_kz_k + \frac{1}{n + \alpha n}\sum_{i=1}^{n}t^2z_k^2 + \frac{1}{n + \alpha n}\sum_{i=n+1}^{n+\alpha n}(x_i^{'})^2 \\ &= \frac{s^2}{n + \alpha n}n + 0 + \frac{1}{n + \alpha n}\sum_{i=1}^{n}t^2Var(z_k) + \frac{1}{n + \alpha n}\sum_{i=n+1}^{n+\alpha n}(x_i^{'})^2 \\ &= \frac{s^2}{1 + \alpha} + \frac{t^2}{1 + \alpha} + \frac{1}{n + \alpha n}\sum_{i=n+1}^{n+\alpha n}(x_i^{'})^2 \\ &= \frac{1}{1 + \alpha} + \frac{1}{n + \alpha n}\sum_{i=n+1}^{n+\alpha n}(x_i^{'})^2 \end{aligned} \tag{29}$$

$z_k^2 = Var(z_k)$ is due to the fact that z_k has a mean of 0. The last equation uses the fact that $s^2 + t^2 = 1$. Therefore, we have:

$$\sum_{i=n+1}^{n+\alpha n}(x_i^{'})^2 = (n + \alpha n)(1 - \frac{1}{1 + \alpha}) = \alpha n \tag{30}$$

For $c_j,\ \forall j \in [n]$:

$$E[c_j] = E[\sum_{k=1}^{n} w_{j,k}z_k] = 0 \tag{31}$$

$$
\begin{aligned}
Var[c_j] &= Var[\sum_{k=1}^{n} w_{j,k} z_k] = \sum_{k=1}^{n} Var[w_{j,k} z_k] \\
&= \sum_{k=1}^{n} Var[w_{j,k}] Var[z_k] = n \times \frac{1}{n} \times 1 = 1
\end{aligned}
\tag{32}
$$

For $d_j,\ \forall j \in [n]$:

$$
E[d_j] = E[\sum_{k=n+1}^{n+\alpha n} w_{j,k} x_k^{'}] = 0 \tag{33}
$$

$$
\begin{aligned}
Var[d_j] &= Var[\sum_{k=n+1}^{n+\alpha n} w_{j,k} x_k^{'}] = \sum_{k=n+1}^{n+\alpha n} Var[w_{j,k} x_k^{'}] \\
&= \sum_{k=n+1}^{n+\alpha n} Var[w_{j,k}] Var[x_k^{'}] = p\frac{1}{n} \sum_{k=n+1}^{n+\alpha n} (x_k^{'})^2 = p\frac{1}{n} \times \alpha n = \alpha p
\end{aligned}
\tag{34}
$$

According to the previous setting, after passing through the linear mapping layer and before entering the normalization layer, the output variance of the original neurons and the newly added neurons is the same. Due to the random variables $y_j^{'}$, c_j, and d_j are independent:

$$
\begin{aligned}
Var(y_1^{'}, \cdots, y_{n+\alpha n}^{'}) &= Var(y_1^{'}, \cdots, y_n^{'}) \\
&= Var(sy_1, \cdots, sy_n) + Var(tc_1, \cdots, tc_n) + Var(d_1, \cdots, d_n)
\end{aligned}
\tag{35}
$$

Denote $\sum Var = Var(sy_1, \cdots, sy_n) + Var(tc_1, \cdots, tc_n) + Var(d_1, \cdots, d_n)$, we have:

$$
\begin{aligned}
a_j^{'} &= \frac{y_j^{'} - mean(y^{'})}{Var(y^{'})} = \frac{sy_j + tc_j + d_j - mean[sy_j] - mean[tc_j] - mean[d_j]}{\sqrt{Var(y_1^{'}, \cdots, y_{n+\alpha n}^{'})}} \\
&= \frac{sy_j - mean[sy_j]}{\sqrt{Var(sy_j)}} \frac{\sqrt{Var(sy_j)}}{\sqrt{\sum Var}} + \frac{tc_j - E[tc_j]}{\sqrt{Var(tc_j)}} \frac{\sqrt{Var(tc_j)}}{\sqrt{\sum Var}} + \frac{d_j - E[d_j]}{\sqrt{Var(d_j)}} \frac{\sqrt{Var(d_j)}}{\sqrt{\sum Var}} \\
&= \frac{y_j - E[y_j]}{\sqrt{Var(y_j)}} \frac{\sqrt{Var(sy_j)}}{\sqrt{\sum Var}} + \frac{tc_j - E[tc_j]}{\sqrt{Var(tc_j)}} \frac{\sqrt{Var(tc_j)}}{\sqrt{\sum Var}} + \frac{d_j - E[d_j]}{\sqrt{Var(d_j)}} \frac{\sqrt{Var(d_j)}}{\sqrt{\sum Var}} \\
&= \frac{s}{\sqrt{s^2 + t^2 + \alpha p}} a_j + \frac{t}{\sqrt{s^2 + t^2 + \alpha p}} \frac{tc_j - mean[tc_j]}{\sqrt{Var(tc_j)}} + \frac{\sqrt{\alpha p}}{\sqrt{s^2 + t^2 + \alpha p}} \frac{d_j - mean[d_j]}{\sqrt{Var(d_j)}} \\
&= \frac{s}{\sqrt{s^2 + t^2 + \alpha p}} a_j + \frac{\sqrt{t^2 + \alpha p}}{\sqrt{s^2 + t^2 + \alpha p}} z_j
\end{aligned}
\tag{36}
$$

The subsequent derivation process is the same as C.1.

C.3 Comparison of Method A and Method B and the result

For Method A:

For the input x_k' of the output layer, where $k \in [n]$, we have

$$
s^{l-1} = (\frac{1}{\sqrt{1 + \alpha p}})^{l-1} \tag{37}
$$

$$
t^{l-1} = \sqrt{1 - (s^{l-1})^2} = \sqrt{1 - (\frac{1}{1 + \alpha p})^{l-1}} \tag{38}
$$

For the last layer's output $y_j^{'},\ \forall j \in d_l$ in f_2:

$$\begin{aligned} y_j^{'} &= \sum_{k=1}^{n+\alpha n} w_{j,k} x_k^{'} \\ &= \sum_{k=1}^{n} w_{j,k}(s^{l-1}x_k + t^{l-1}z_k) + \sum_{k=n+1}^{n+\alpha n} w_{j,k} x_k^{'} \\ &= s^{l-1}\sum_{k=1}^{n} w_{j,k}x_k + t^{l-1}\sum_{k=1}^{n} w_{j,k}z_k + \sum_{k=n+1}^{n+\alpha n} w_{j,k} x_k^{'} \\ &= s^{l-1}y_j + (\sqrt{1-(s^{l-1})^2+\alpha})v_j \end{aligned} \tag{39}$$

where v_j is a random variable drawn from a standard normal distribution.

For Method B:

For the input $x_k^{'}$ of the output layer, where $k \in [n+\alpha n]$, we also have Eq. 37 and Eq. 38.

For the last layer's output $y_j^{'},\ \forall j \in d_l$ in f_2:

$$\begin{aligned} y_j^{'} &= \frac{1}{2}\sum_{k=1}^{n+\alpha n} w_{j,k} x_k^{'} = \frac{1}{2}\sum_{k=1}^{n+\alpha n} w_{j,k}(s^{l-1}x_k + t^{l-1}z_k) \\ &= \frac{1}{2}(s^{l-1}\sum_{k=1}^{n} w_{j,k}x_k + s^{l-1}\sum_{k=n+1}^{n+\alpha n} w_{j,k}x_k + t^{l-1}\sum_{k=1}^{n} w_{j,k}z_k + t^{l-1}\sum_{k=n+1}^{n+\alpha n} w_{j,k}z_k) \\ &= \frac{1}{2}(2s^{l-1}y_j + \sqrt{2}(\sqrt{1-(s^{l-1})^2})v_j) \\ &= s^{l-1}y_j + \frac{\sqrt{2}}{2}(\sqrt{1-(s^{l-1})^2})v_j \end{aligned} \tag{40}$$

where v_j is a random variable drawn from a standard normal distribution. The penultimate equation holds due to $Var(z_k) = 1,\ \forall k \in [n+\alpha n]$, and $Var(w_{j,k}) = \frac{1}{n},\ \forall k \in [n]$, as well as $Var(w_{j,k}) = \frac{1}{\alpha n},\ \forall k \in [n+1, n+\alpha n]$, the variance of $\sum_{k=1}^{n} w_{j,k}z_k$ and $\sum_{k=n+1}^{n+\alpha n} w_{j,k}z_k$ are both equal to 1.

Since the parameters in the expression $s^{l-1} = (\frac{1}{1+\alpha p})^{l-1}$ are predetermined, s^{l-1} can be calculated in advance. Therefore, after appropriate scaling, Method A can yield the following output:

$$\frac{1}{s^{l-1}}y_j^{'} = y_j + \frac{\sqrt{1-(s^{l-1})^2+\alpha}}{s^{l-1}}v_j \tag{41}$$

Method B can yield the following output:

$$\frac{1}{s^{l-1}}y_j^{'} = y_j + \frac{\sqrt{2}}{2}\frac{\sqrt{1-(s^{l-1})^2}}{s^{l-1}}v_j \tag{42}$$

As a result:

$$\frac{\delta_A}{\delta_B} = \frac{1-(s^{l-1})^2+\alpha}{(s^{l-1})^2}\frac{4(s^{l-1})^2}{2(1-(s^{l-1})^2)} = 2\frac{1-(\frac{1}{1+\alpha p})^{l-1}+\alpha}{1-(\frac{1}{1+\alpha p})^{l-1}} = 2 + \frac{2\alpha}{1-(\frac{1}{1+\alpha p})^{l-1}} \tag{43}$$

D Proof For Theorems

D.1 The Relationship between $V^{(1)}$ and $\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|$

The value of $\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|$ is related to $V^{(1)}$, We will first explain the relationship between them.

Considering only the parameters within the hidden layers of the small model, let

$$G_{small} = \|\frac{\partial l(W)}{\partial W}\| \tag{44}$$

When $V^{(1)}$ is 0, the large network with parameters $W^{(1)}$ has function-preserving properties compared to the small network with parameters W. The hidden layer output of the large network is composed of two identical parts concatenated together. The output of an hidden layer in large model that does not pass through a normalization layer can be represented as follows:

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{45}$$

and $X_1 = X_2, Y_1 = Y_2$. Since $V^{(1)} = 0$, the two components do not influence each other before reaching the final output. Moreover, the two components contribute equally to the overall output. So we have $\frac{\partial l}{\partial Y_1} = \frac{\partial l}{\partial Y_2}$.

The derivative of the loss function with respect to W can be expressed as $\frac{\partial l}{\partial W} = \frac{\partial l}{\partial Y} X^T$. Therefore, all four components W_{11}, W_{12}, W_{21}, and W_{22} of W^1 have the same derivative. Since the outputs from the two networks in the output layer are averaged, $\|\{vec(W_{11}^2), \ldots, vec(W_{11}^{l-1})\}\|$ and $\|\{vec(W_{22}^2), \ldots, vec(W_{22}^{l-1})\}\|$ are both half of G_{small}, resulting in:

$$\|\frac{\partial l(W^1)}{\partial W_{new}^1}\| = G_{small},\ s.t.\ W_{new}^1 = 0 \tag{46}$$

In particular, when $V^{(1)} = 0$, we have $P = \frac{\partial F(W^{(1)})}{\partial V^{(1)}} = \frac{\partial F(W^{(1)})}{\partial 0} = \frac{\partial F(W^{(1)})}{\partial U^{(1)}}$. Leveraging the L-smooth property, for non-zero $V^{(1)}$, we obtain: $\|V^{(1)} - 0\| \leq L\|\frac{\partial F(W^{(1)})}{\partial V^{(1)}} - P\|$. This inequality demonstrates that as $V^{(1)}$ approaches 0, the change in $\frac{\partial F(W^{(1)})}{\partial V^{(1)}}$ relative to P is also small. Consequently, we can select an appropriate $V^{(1)}$ to ensure that both λ and T in Theorem 1 achieve their optimal values.

D.2 Proof For Theorem 1

We introduce $\mathbb{G}^{(3)}$, and denote $W^{(3)}$ as follows:

$$W^{(3)} = \mathbb{G}^{(3)}(\lambda, W_{11}, W_{12}, W_{21}, W_{22}) = \begin{bmatrix} \lambda W_{11} & \lambda W_{12} \\ \lambda W_{21} & \lambda W_{22} \end{bmatrix} \tag{47}$$

Let the parameters (U, V) of $W^{(3)}$ be denoted as $(U^{(3)}, V^{(3)})$.

Lemma 1 (Ioffe & Szegedy, 2015) For a weight matrix W, input x and a scaler λ, we have:

$$Norm(Wx) = Norm((\lambda W)x) \tag{48}$$

$$\frac{\partial Norm((\lambda W)x)}{\partial x} = \frac{\partial Norm(Wx)}{\partial x} \tag{49}$$

$$\frac{\partial Norm((\lambda W)x)}{\partial (\lambda W)} = \frac{1}{\lambda}\frac{\partial Norm(Wx)}{\partial W} \tag{50}$$

D.2.1 The First part of Theorem 1

First,

$$\begin{aligned} \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| &= \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}} - \frac{\partial l(W^{(2)})}{\partial V^{(2)}} + \frac{\partial l(W^{(2)})}{\partial V^{(2)}}\| \\ &\geq \|\frac{\partial l(W^{(2)})}{\partial V^{(2)}}\| - \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}} - \frac{\partial l(W^{(2)})}{\partial V^{(2)}}\| \\ &= \frac{1}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}} - \frac{\partial l(W^{(2)})}{\partial V^{(2)}}\| \end{aligned} \tag{51}$$

With L-smoothness, note that $U^{(2)} = U^{(3)}, V^{(2)} = \lambda V^{(3)}$ in $W^{(2)}$ and $W^{(3)}$, we have:

$$\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}} - \frac{\partial l(W^{(2)})}{\partial V^{(2)}}\| \leq \|\frac{\partial l(W^{(3)})}{\partial W^{(3)}} - \frac{\partial l(W^{(2)})}{\partial W^{(2)}}\| \leq L\|W^{(3)} - W^{(2)}\| = L(1-\lambda)\|V^{(3)}\| \tag{52}$$

Therefore,

$$\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| \geq \frac{1}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(3)}\| \tag{53}$$

To ensure that $\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| > \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|$, we need to have:

$$\frac{1-\lambda}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(1)}\| > 0 \tag{54}$$

This is equivalent to:

$$\|V^{(1)}\| = \|V^{(3)}\| < \frac{1}{L\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| \tag{55}$$

Which completes the proof of the first part.

D.2.2 The Second part of Theorem 1

We will explain how many iterations it will take. Assume after performing T iterations of updates, parameters $W^{(1)}$, $W^{(2)}$ and $W^{(3)}$ turns to $\mathcal{W}^{(1)}$, $\mathcal{W}^{(2)}$ and $\mathcal{W}^{(3)}$, respectively. and $(U^{(1)}, V^{(1)}), (U^{(2)}, V^{(2)}), (U^{(3)}, V^{(3)})$ turns to $(\mathcal{U}^{(1)}, \mathcal{V}^{(1)}), (\mathcal{U}^{(2)}, \mathcal{V}^{(2)}), (\mathcal{U}^{(3)}, \mathcal{V}^{(3)})$. First, we have:

$$\begin{aligned}\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}}\| &= \|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}} - \frac{\partial l(W^{(3)})}{\partial V^{(3)}} + \frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| \\ &\geq \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| - \|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}} - \frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|\end{aligned} \tag{56}$$

For $\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}} - \frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|$, with the L-smoothness and the bounded gradient:

$$\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}} - \frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| \leq \|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{W}^{(3)}} - \frac{\partial l(W^{(3)})}{\partial W^{(3)}}\| \leq L\|\mathcal{W}^{(3)} - W^{(3)}\| \leq \eta LTG \tag{57}$$

With Eq. 53, we have:

$$\begin{aligned}\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}}\| &\geq \frac{1}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(3)}\| - \eta LTG \\ &= \frac{1-\lambda}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| + \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(3)}\| - \eta LTG\end{aligned} \tag{58}$$

Then, for $\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|$ and $\|\frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}}\|$:

$$\begin{aligned}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| &= \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}} - \frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}} + \frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}}\| \geq \|\frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}}\| - \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}} - \frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}}\| \\ &\geq \|\frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}}\| - \eta LTG\end{aligned} \tag{59}$$

Substituting the above equation into the second term of Eq. 58,

$$\begin{aligned}\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}}\| &\geq \frac{1-\lambda}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| + (\|\frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}}\| - \eta LTG) - L(1-\lambda)\|V^{(3)}\| - \eta LTG \\ &= \|\frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}}\| + \frac{1-\lambda}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(3)}\| - 2\eta LTG\end{aligned} \tag{60}$$

Therefore, if we require $\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}}\| > \|\frac{\partial l(\mathcal{W}^{(1)})}{\partial \mathcal{V}^{(1)}}\|$, we simply need to ensure that:

$$\begin{aligned}\frac{1-\lambda}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(3)}\| - 2\eta LTG \\ = \frac{1-\lambda}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(1)}\| - 2\eta LTG > 0\end{aligned} \tag{61}$$

This is equivalent to:

$$T < \frac{\frac{1-\lambda}{\lambda}\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| - L(1-\lambda)\|V^{(1)}\|}{2\eta LG} \tag{62}$$

which completes the proof of Theorem 1.

D.3 THEOREM 2

The growth method applied to $W^{(3)}$ can also amplify the gradients of the parameters within the V component. In the subsequent theorem, we will demonstrate another advantage of the $W^{(2)}$ growth method:

Theorem 2 With the bounded gradients and L-smoothness assumption in Theorem 1. Let $\Gamma = \|U^{(1)}\|/\|\frac{\partial F(W^{(1)})}{\partial U^{(1)}}\| + \|V^{(1)}\|/\|\frac{\partial F(W^{(1)})}{\partial V^{(1)}}\|$. When $\lambda < min(\frac{1}{l\Gamma}, 1)$, in the first iteration of training, we have: $\|\frac{\partial F(W^{(2)})}{\partial V^{(2)}}\|/\|\frac{\partial F(W^{(2)})}{\partial U^{(2)}}\| > \|\frac{\partial F(W^{(1)})}{\partial V^{(1)}}\|/\|\frac{\partial F(W^{(1)})}{\partial U^{(1)}}\| = \|\frac{\partial F(W^{(3)})}{\partial V^{(3)}}\|/\|\frac{\partial F(W^{(3)})}{\partial U^{(3)}}\|$. Let $a = \frac{1}{\lambda} - (L(1-\lambda)\|V^{(3)}\|/\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|)$, $b = 1 + (L(1-\lambda)\|U^{(1)}\|/\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|)$, this situation will persist for $T = \frac{1}{\eta LG} min\{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|, \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|, \frac{(a-b)\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}{((b+1)\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|+(a+1)\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|)}\}$ iterations.

Remark Theorem 2 demonstrates that, compared to models employing $W^{(1)}$ and $W^{(3)}$, model employing $W^{(2)}$ exhibit a proportionally larger update rate for V relative to U. Specifically, relative to the proportional gradient improvement in $W^{(3)}$, where both U and V receive same proportional updates, the update rate of V within $W^{(2)}$ is also enhanced. This suggests that $W^{(2)}$ offers a more efficient method for optimizing V compared to the other two models.

D.3.1 THE FIRST PART OF THEOREM 2

First, with L-smoothness

$$\begin{aligned}\|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| &= \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}} - \frac{\partial l(W^{(1)})}{\partial U^{(1)}} + \frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| \le \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}} - \frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| \\ &\le \|\frac{\partial l(W^{(3)})}{\partial W^{(3)}} - \frac{\partial l(W^{(1)})}{\partial W^{(1)}}\| + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| \le L(1-\lambda)\|W^{(3)} - W^{(1)}\| + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| \\ &= L(1-\lambda)\|U^{(1)}\| + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|\end{aligned} \tag{63}$$

Dividing both sides of the above inequality by $\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|$, we obtain:

$$\frac{\|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|}{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|} \le L(1-\lambda)\frac{\|U^{(1)}\|}{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|} + 1 \tag{64}$$

With Eq. 53, we have:

$$\frac{\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} \ge \frac{1}{\lambda} - L(1-\lambda)\frac{\|V^{(3)}\|}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} = \frac{1}{\lambda} - L(1-\lambda)\frac{\|V^{(1)}\|}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} \tag{65}$$

To have $\frac{\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} > \frac{\|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|}{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}$, we simply need to ensure that:

$$\frac{1}{\lambda} - L(1-\lambda)\frac{\|V^{(1)}\|}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} > L(1-\lambda)\frac{\|U^{(1)}\|}{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|} + 1 \tag{66}$$

Let $\Gamma = \frac{\|V^{(1)}\|}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} + \frac{\|U^{(1)}\|}{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}$, then the above inequality is equivalent to:

$$\lambda < \frac{1}{L\Gamma} \tag{67}$$

Which completes the proof of the first part.

D.3.2 The Second part of Theorem 2

We will explain how many iterations it will take, with the L-smoothness and the bounded gradient assumptions:

$$\begin{aligned}
\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{U}^{(3)}}\| &= \|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{U}^{(3)}} - \frac{\partial l(W^{(3)})}{\partial U^{(3)}} + \frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| \leq \|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{U}^{(3)}} - \frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| \\
&\leq \|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{W}^{(3)}} - \frac{\partial l(W^{(3)})}{\partial W^{(3)}}\| + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| \leq L\|\mathcal{W}^{(3)} - W^{(3)}\| + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| \\
&\leq \eta LTG + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|
\end{aligned} \tag{68}$$

For $W^{(2)}$, since multiplying λ will increase its gradient norm to $\frac{1}{\lambda}$ times, we assume that the gradient norm from $W^{(2)}$ to $\mathcal{W}^{(2)}$ is bounded above by $\frac{G}{\lambda}$. Therefore, we have:

$$\begin{aligned}
\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{U}^{(2)}}\| &= \|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{U}^{(2)}} - \frac{\partial l(W^{(2)})}{\partial U^{(2)}} + \frac{\partial l(W^{(2)})}{\partial U^{(2)}}\| \geq -\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{U}^{(2)}} - \frac{\partial l(W^{(2)})}{\partial U^{(2)}}\| + \|\frac{\partial l(W^{(2)})}{\partial U^{(2)}}\| \\
&\geq -\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{W}^{(2)}} - \frac{\partial l(W^{(2)})}{\partial W^{(2)}}\| + \|\frac{\partial l(W^{(2)})}{\partial U^{(2)}}\| \geq -L\|\mathcal{W}^{(2)} - W^{(2)}\| + \|\frac{\partial l(W^{(2)})}{\partial U^{(2)}}\| \\
&\geq -\frac{\eta LTG}{\lambda} + \frac{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}{\lambda}
\end{aligned} \tag{69}$$

Similarly, we have:

$$\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}}\| \geq -\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| \tag{70}$$

$$\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial V^{(2)}}\| \leq \frac{\eta LTG}{\lambda} + \frac{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|}{\lambda} \tag{71}$$

After obtaining the four inequalities above, when $T < \min\{\frac{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}{\eta LG}, \frac{\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|}{\eta LG}\}$:

$$\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{U}^{(3)}}\|/(\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{U}^{(2)}}\| \leq (\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|)/(-\frac{\eta LTG}{\lambda} + \frac{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}{\lambda}) \tag{72}$$

$$\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}}\|/(\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{V}^{(2)}}\| \geq (-\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|)/(\frac{\eta LTG}{\lambda} + \frac{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|}{\lambda}) \tag{73}$$

To have $\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}}\|/(\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{V}^{(2)}}\| > \|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{U}^{(3)}}\|/(\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{U}^{(2)}}\|$, we simply need to ensure that:

$$\begin{aligned}(-\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|)/(\frac{\eta LTG}{\lambda} + \frac{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|}{\lambda}) \\ > (\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|)/(-\frac{\eta LTG}{\lambda} + \frac{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}{\lambda})\end{aligned} \tag{74}$$

which is equivalent to:

$$\begin{aligned}(-\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|)/(\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|) \\ > (\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|)/(-\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|)\end{aligned} \tag{75}$$

Let $a = \frac{1}{\lambda} - L(1-\lambda)\frac{\|V^{(1)}\|}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|}$ and $b = L(1-\lambda)\frac{\|U^{(1)}\|}{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|} + 1$, then Eq. 64 and Eq. 65 could be rewritten as:

$$\frac{\|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|}{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|} \leq b \tag{76}$$

$$\frac{\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} \geq a \tag{77}$$

The condition $a > b$ remains crucial. Therefore, the previously established condition Eq. 67 $\lambda < \frac{1}{L\Gamma}$ should be satisfied.

We further have:

$$\begin{aligned}\|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| \leq b\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| \\ \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| + b\eta LTG \leq b\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| + b\eta LTG\end{aligned} \tag{78}$$

Similarly:

$$\begin{aligned}\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| \geq a\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| \\ \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| + a\eta LTG \geq a\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| + a\eta LTG\end{aligned} \tag{79}$$

So we have:

$$\begin{aligned}\frac{\|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\| + b\eta LTG}{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| + \eta LTG} \leq b \\ \frac{\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\| + a\eta LTG}{\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\| + \eta LTG} \geq a\end{aligned} \tag{80}$$

As a result:

$$\begin{aligned}&(\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|)/(-\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|) \\ &= ((b+1)\eta LTG - b\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial U^{(3)}}\|)/(-\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|) \\ &\leq b + \frac{(b+1)\eta LTG}{-\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}\end{aligned} \tag{81}$$

and

$$\begin{aligned}
&(-\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|)/(\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|) \\
&= (-(a+1)\eta LTG + a\eta LTG + \|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|)/(\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|) \\
&\geq a - \frac{(a+1)\eta LTG}{\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|}
\end{aligned} \tag{82}$$

To meet Eq. 75, we only need:

$$a - \frac{(a+1)\eta LTG}{\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} > b + \frac{(b+1)\eta LTG}{-\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|} \tag{83}$$

By multiplying both sides of the inequality by $(\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|)(-\eta LTG + \|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|)$, we can eliminate the quadratic term in ηLTG, resulting in:

$$((b+1)\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| + (a+1)\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|)\eta LTG < (a-b)\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| \tag{84}$$

which is equivalent to:

$$T < \frac{1}{\eta LG}\frac{(a-b)\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}{(b+1)\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\| + (a+1)\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|} \tag{85}$$

Based on the above conditions, we conclude that: When $\lambda < \frac{1}{L\Gamma}$ and $T < \min\{\frac{\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}{\eta LG}, \frac{\|\frac{\partial l(W^{(3)})}{\partial V^{(3)}}\|}{\eta LG}, \frac{1}{\eta LG}\frac{(a-b)\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|}{(b+1)\|\frac{\partial l(W^{(1)})}{\partial U^{(1)}}\|+(a+1)\|\frac{\partial l(W^{(1)})}{\partial V^{(1)}}\|}\}$, then $\|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{V}^{(3)}}\|/(\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{V}^{(2)}}\| > \|\frac{\partial l(\mathcal{W}^{(3)})}{\partial \mathcal{U}^{(3)}}\|/(\|\frac{\partial l(\mathcal{W}^{(2)})}{\partial \mathcal{U}^{(2)}}\|$, which completes the proof of theorem 2.

E About Method D

Network f_2, grown using Method D, maintains function-preserving properties. However, when f_1 and its parameters θ_1 are located at a local minimum, so are f_2 and θ_2. During the training of f_2, the parameter changes correspond to the parameter changes in the training of f_1, and the output is also identical to the output produced during f_1's training.

Let M^i denote the parameters of the i-th layer of f_2. In the hidden layers, we have:

$$M^i = \mathbb{G}(W_{11} = \frac{W^i}{2}, W_{12} = \frac{W^i}{2}, W_{21} = \frac{W^i}{2}, W_{22} = \frac{W^i}{2}) = \begin{bmatrix} \frac{W^i}{2} & \frac{W^i}{2} \\ \frac{W^i}{2} & \frac{W^i}{2} \end{bmatrix} \tag{86}$$

The input to the i-th layer of f_2 is denoted by $\hbar^{i-1}$, and the output of f_2 is $\hbar^l$. The model's growth pattern at the first layer is $M^1 = \begin{bmatrix} \frac{W^1}{2} \\ \frac{W^1}{2} \end{bmatrix}$. At the last layer, the model's growth pattern is $M^l = [W^l, W^l]$, and then divided by 2 to obtain the final output, which is:

$$\bar{h}^l = \frac{1}{2}M^l\bar{h}^{l-1} \tag{87}$$

Easy to verify, Method D is able to preserve the function-preserving property. We further demonstrate that the parameters θ_2 of f_2, generated using Method D, have a one-to-one correspondence with the gradients of the parameters θ_1 of f_1, which are trained with the same learning rate at each

step (The learning rate for the normalization layer of f_2 is doubled). Specifically, when the parameters of f_1 in a certain hidden layer are updated to W_T^i, the parameters of f_2 in that layer can be expressed as:

$$M_T^i = \begin{bmatrix} \frac{W_T^1}{2} & \frac{W_T^1}{2} \\ \frac{W_T^1}{2} & \frac{W_T^1}{2} \end{bmatrix} \tag{88}$$

The conclusion holds true for neural networks incorporating activation functions. The proof is presented below:

First, the input to the f_1 output layer can be represented as h^{l-1}, and the input $\bar{h}^{l-1}$ to the f_2 output layer can be represented as:

$$\bar{h}^{l-1} = \begin{bmatrix} h^{l-1} \\ h^{l-1} \end{bmatrix} \tag{89}$$

For f_1, its output $h^l = W^l h^{l-1}$. For f_2, its output $\bar{h}^l = \frac{1}{2}(W^l h^{l-1} + W^l h^{l-1}) = W^l h^{l-1} = h^l$. Then:

$$\frac{\partial l}{\partial h^l} = \frac{\partial l}{\partial \bar{h}^l} \tag{90}$$

And we have:

$$\frac{\partial l}{\partial h^{l-1}} = (W^l)^T \frac{\partial l}{\partial h^l} \tag{91}$$

$$\frac{\partial l}{\partial \bar{h}^{l-1}} = \frac{1}{2}[W^l, W^l]^T \frac{\partial l}{\partial \bar{h}^l} = \frac{1}{2} \begin{bmatrix} \frac{\partial l}{\partial h^{l-1}} \\ \frac{\partial l}{\partial h^{l-1}} \end{bmatrix} \tag{92}$$

$$\frac{\partial l}{\partial W^l} = \frac{\partial l}{\partial h^l}(h^{l-1})^T \tag{93}$$

$$\frac{\partial l}{\partial M^l} = \frac{1}{2}\frac{\partial l}{\partial \bar{h}^l}(\bar{h}^{l-1})^T = \frac{1}{2}\frac{\partial l}{\partial h^l} \begin{bmatrix} h^{l-1} \\ h^{l-1} \end{bmatrix}^T = \frac{1}{2}[\frac{\partial l}{\partial W^l}, \frac{\partial l}{\partial W^l}] \tag{94}$$

The backpropagation process through the normalization layer:

$$h^{l-1} = \gamma^{l-1} \frac{y^{l-1} - \mu^{l-1}}{\sigma^{l-1}} + \beta^{l-1} \tag{95}$$

In f_2, the four normalization parameters, $\gamma, \mu, \sigma, \beta$, are derived from the corresponding parameters in f_1 through a self-concatenation process. Therefore, they all exhibit a form similar to Eq. 92. Furthermore, from Eq. 92, we can derive:

$$\frac{\partial l}{\partial \bar{y}^{l-1}} = \frac{1}{2} \begin{bmatrix} \frac{\partial l}{\partial y^{l-1}} \\ \frac{\partial l}{\partial y^{l-1}} \end{bmatrix} \tag{96}$$

Since

$$\frac{\partial l}{\partial \bar{h}^{l-2}} = (M^{l-1})^T \frac{\partial l}{\partial \bar{y}^{l-1}} = \begin{bmatrix} \frac{W^{l-1}}{2} & \frac{W^{l-1}}{2} \\ \frac{W^{l-1}}{2} & \frac{W^{l-1}}{2} \end{bmatrix} \frac{1}{2} \begin{bmatrix} \frac{\partial l}{\partial y^{l-1}} \\ \frac{\partial l}{\partial y^{l-1}} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \frac{\partial l}{\partial h^{l-2}} \\ \frac{\partial l}{\partial h^{l-2}} \end{bmatrix} \tag{97}$$

and

$$\frac{\partial l}{\partial M^{l-1}} = \frac{\partial l}{\partial \bar{y}^{l-1}}(\bar{h}^{l-2})^T = \frac{1}{2} \begin{bmatrix} \frac{\partial l}{\partial y^{l-1}} \\ \frac{\partial l}{\partial y^{l-1}} \end{bmatrix} \begin{bmatrix} h^{l-2} \\ h^{l-2} \end{bmatrix}^T = \frac{1}{2} \begin{bmatrix} \frac{\partial l}{\partial W^{l-1}} & \frac{\partial l}{\partial W^{l-1}} \\ \frac{\partial l}{\partial W^{l-1}} & \frac{\partial l}{\partial W^{l-1}} \end{bmatrix} \tag{98}$$

Employing an inductive approach, we demonstrate that for all hidden layer parameters within the function f_2, the form of the equation presented above holds. The gradients of the input layer are formed by concatenating two identical parts, each multiplied by a weight of $\frac{1}{2}$. Therefore, if both f_1 and its parameters θ_1 are already at a local minimum, then the network f_2 initialized using Method D, along with its parameters θ_2, also resides at a local minimum. When employing the same learning rate as used for f_1, the training process for f_2 mirrors that of f_1. This conclusion extends to networks incorporating activation functions, where the underlying proof follows a similar logic.

F STORAGE COST ON THE SERVER SIDE

In this section, we introduce the server-side storage method employed in FedMG. This method ensures that the storage cost for each capacity model on the server is no more than twice the model's size.

In FedMG, for global model f_i and θ_i, our goal is to get the averaged model $\theta_i^{avg} = Average(\theta_i^{(1)}, \theta_i^{(2)}, \cdots)$. When we obtain the parameters θ_i^1, we set $\theta_i^{avg} = \theta_i^{(1)}$. Then, when we obtain the parameters $\theta_i^{(k)}$ for $k \geq 2$, we set

$$\theta_i^{avg} = \frac{k-1}{k}\theta_i^{avg} + \frac{1}{k}\theta_i^{(k)} \tag{99}$$

Using the incremental mean style method, it is not necessary to save all the models produced in the whole process, and we can save storage cost on the server side.

G DETAILS OF EXPERIMENT

G.1 DETAILS FOR KD-BASED METHODS

Compared to other methods, KD-based methods typically require a larger number of local epochs and converge much faster (Itahara et al., 2021; Cho et al., 2022), requiring significantly fewer communication rounds. In our experiments, we set the number of local epochs to 10 and the communication rounds to 30. Following the setup in Li & Wang (2019), we conducted experiments with a 100% client participation rate. We ensured that these methods had already converged within the specified communication rounds. For the CIFAR-10 dataset, we selected 5000 samples from the CIFAR-100 dataset as the public shared dataset in each communication round. For the CIFAR-100 dataset, we selected 5000 samples from the CIFAR-10 dataset as the public shared dataset in each communication round.

G.2 MORE EXPERIMENTAL SETUP DETAILS

Data Preprocessing For CIFAR-10 and CIFAR-100, in pre-processing, we standardize the images by utilizing channel means and standard deviations. Data augmentation is carried out through a 4x4 random translation operation, which is then followed by a random horizontal flip.

Experiments Compute Resources We run our codes on a server with 8 NVIDIA A6000 GPUs and an Intel(R) Xeon(R) Gold 6242R CPU, using Python 3.11, PyTorch 2.0, and Ray 2.43.

Experimental Set up Details for Table 1, Table 2, Table 4 and Table 5 Table 8 presents the detailed hyperparameter settings used in Table 1, Table 2 and Table 4. In addition, Table 5 utilizes the same settings as those described in Table 8 for the CIFAR-10 dataset, except for the model growth rounds and feedback interval rounds. The specific values for model growth rounds and feedback interval rounds used in Table 5 are detailed in Table 9.

H ADDITIONAL EXPERIMENTS

H.1 ABLATION STUDY: CLIENT PARTICIPATION RATE

In the prior experiments, we set the client participation rate to 10%. As a part of the ablation study, we now investigate the impact of varying client participation rates. Table 10 and Table 11 present the results of experiments conducted on the CIFAR-10 dataset ($\alpha = 0.3$), under 'a1-b1-c1', respectively. The results consistently demonstrate that FedMG outperforms other methods under both participation rates, further validating its effectiveness.

H.2 MODEL GROWTH AND TRAINING FROM SCRATCH

This section compares the training rates of FedMG's model growth and training from scratch. The goal is to demonstrate that model growth effectively conserves communication and computational

Table 8: Experimental setup details, including hyperparameter settings and model architecture in Table 1, Table 2 and Table 4.

Dataset	CIFAR-10	CIFAR-100	WikiText2
Model	PreResNet18	PreResNet18	Transformer
Total rounds	3000	3500	200
Local epochs	1	1	1
Model Growth Rounds	[1200, 1800]	[1900, 2700]	[110]
Feedback Interval Rounds	[300, 300]	[400, 400]	[30]
Batch size	10	24	100
Optimizer	SGD	SGD	SGD
Learning rate	2e-3	2e-3	1e-2
Momentum	0.9	0.9	0.9
Embedding Size	N/A		128
Number of head	N/A		8
Dropout	N/A		0.1
Sequence length	N/A		64

Table 9: Model growth rounds and Feedback interval rounds in Table 5 for different clients' capacities on CIFAR-10($\alpha = 0.3$).

Capacity	Model Growth Rounds	Feedback Interval Rounds
a1-b1-c1-d1	[600, 1200, 1800]	[300, 300, 300]
a1-b1-d1 (a1-b1-c0-d1)	[600, 1200, 1800]	[300, 300, 300]
a1-c1-d1 (a1-b0-c1-d1)	[600, 1200, 1800]	[300, 300, 300]
a1-b1	[1800]	[300]
a1-c1 (a1-b0-c1)	[1200, 1800]	[300, 300]
a1-d1 (a1-b0-c0-d1)	[600, 1200, 1800]	[300, 300, 300]

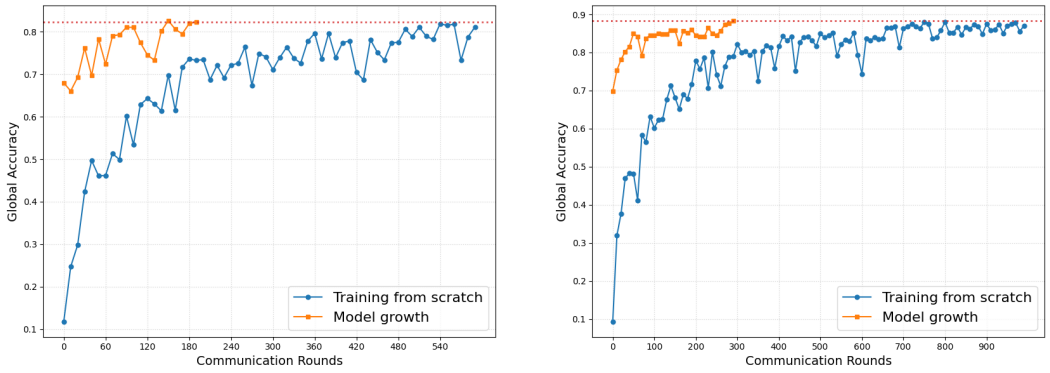
Table 10: Performance comparison with other system heterogeneous FL methods on CIFAR-10 ($\alpha = 0.3$) under 'a1-b1-c1'. Client participation rate is set to 5%.

Method	Local	Model(0.25)	Model(0.5)	Model(1.0)	Global
HeteroFL	$68.41_{\pm 0.71}$	$63.10_{\pm 2.39}$	$72.33_{\pm 2.31}$	$69.62_{\pm 2.20}$	$66.78_{\pm 2.11}$
FedRolex	$52.56_{\pm 1.53}$	$34.17_{\pm 2.28}$	$48.38_{\pm 3.88}$	$77.14_{\pm 2.58}$	$54.76_{\pm 4.24}$
FIARSE	$80.62_{\pm 1.00}$	$81.28_{\pm 0.20}$	$80.21_{\pm 2.26}$	$79.70_{\pm 1.70}$	$80.67_{\pm 1.21}$
FedBRB	$62.58_{\pm 2.69}$	$56.22_{\pm 4.21}$	$62.01_{\pm 4.33}$	$70.33_{\pm 1.18}$	$66.26_{\pm 3.27}$
FedHM	$62.58_{\pm 2.69}$	$56.22_{\pm 4.21}$	$62.01_{\pm 4.33}$	$70.33_{\pm 1.18}$	$66.26_{\pm 3.27}$
FedLMT	$75.25_{\pm 1.07}$	$75.25_{\pm 1.07}$	$N/A_{\pm N/A}$	$N/A_{\pm N/A}$	$75.28_{\pm 2.34}$
Ours	$\mathbf{83.16_{\pm 0.42}}$	$\mathbf{81.52_{\pm 0.88}}$	$\mathbf{83.58_{\pm 0.29}}$	$\mathbf{84.32_{\pm 2.30}}$	$\mathbf{82.79_{\pm 0.93}}$

Table 11: Performance comparison with other system heterogeneous FL methods on CIFAR-10 ($\alpha = 0.3$) under 'a1-b1-c1'. Client participation rate is set to 20%.

Method	Local	Model(0.25)	Model(0.5)	Model(1.0)	Global
HeteroFL	$70.32_{\pm 1.15}$	$66.64_{\pm 2.34}$	$70.21_{\pm 1.57}$	$74.14_{\pm 2.50}$	$71.41_{\pm 0.43}$
FedRolex	$54.02_{\pm 2.00}$	$35.03_{\pm 0.84}$	$50.01_{\pm 3.91}$	$78.79_{\pm 1.55}$	$58.93_{\pm 1.88}$
FIARSE	$83.03_{\pm 0.34}$	$81.76_{\pm 0.94}$	$85.20_{\pm 1.63}$	$81.85_{\pm 2.02}$	$84.06_{\pm 1.31}$
FedBRB	$72.22_{\pm 2.47}$	$67.48_{\pm 4.35}$	$71.42_{\pm 2.11}$	$78.08_{\pm 1.94}$	$75.08_{\pm 2.42}$
FedHM	$65.60_{\pm 1.25}$	$32.21_{\pm 2.95}$	$83.55_{\pm 1.23}$	$80.52_{\pm 2.20}$	$68.97_{\pm 0.78}$
FedLMT	$80.88_{\pm 1.57}$	$80.88_{\pm 1.57}$	$N/A_{\pm N/A}$	$N/A_{\pm N/A}$	$81.70_{\pm 1.58}$
Ours	$\mathbf{87.08_{\pm 0.44}}$	$\mathbf{84.58_{\pm 1.15}}$	$\mathbf{87.56_{\pm 0.78}}$	$\mathbf{89.19_{\pm 0.18}}$	$\mathbf{85.28_{\pm 0.17}}$

resources. We conducted experiments following the setup detailed in Table 8. We first pretrain two models of size 'b' and size 'c' as pre-trained models. Then, we compared the training rates of: (1) training a model of size 'b' from scratch, versus growing a model of size 'b' with the pre-trained model of size 'c', and (2) training a model of size 'a' from scratch, versus growing a model of size 'a' with the pre-trained model of size 'b'. The results of Figure 3 illustrate the efficiency gains achieved through FedMG's model growth.



(a) comparison with size 'b'

(b) comparison with size 'a'

Figure 3: Comparison between FedMG's model growth and training from scratch. Employing FedMG's model growth method with pre-trained small models indeed reduces communication and computational costs.

H.3 Experimental setup for Theorem 1 and validation of Theorem 2

To validate Theorem 1, we conducted experiments with 100 clients, setting the client participation rate to 100%, and configuring the local batch size on each client to match the size of their local dataset. We first trained a 'c' sized model on the CIFAR-10 dataset ($\alpha = 0.3$) until it achieved satisfactory performance. Subsequently, we grow the model using $W^{(1)}$ and $W^{(2)}$ with $(\lambda_1, \lambda_2) = (0.4, 0.44)$ and $(0.5, 0.55)$, respectively. After obtaining a 'b' sized model, we continued with the same settings: 100% client participation, local batch size equal to the local dataset size, and a learning rate of 0.002 using SGD. After each communication round, we select 5000 samples from the CIFAR-10 training dataset to compute $(\|\frac{\partial F(W^{(2)})}{\partial V^{(2)}}\|)/(\|\frac{\partial F(W^{(1)})}{\partial V^{(1)}}\|)$.

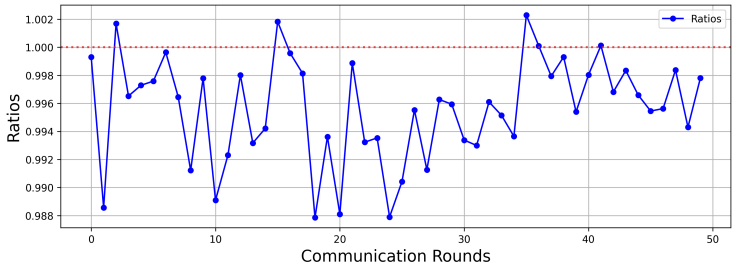


Figure 4: Verification of Theorem 2 with $(\lambda_1, \lambda_2) = (0.4, 0.44)$

Here, we present an experimental validation of Theorem 2 under the same settings. Figure 4 and Figure 5 show the experimental results. Since the stochastic gradient can never be exactly equal to the true gradient of the underlying data distribution, there may be some noise that causes the values to fluctuate and exceed 1. We believe this is reasonable. As can be seen, the values $(\|\frac{\partial F(W^{(3)})}{\partial V^{(3)}}\|/\|\frac{\partial F(W^{(3)})}{\partial U^{(3)}}\|)/(\|\frac{\partial F(W^{(2)})}{\partial V^{(2)}}\|/\|\frac{\partial F(W^{(2)})}{\partial U^{(2)}}\|)$ are mostly less than 1, which validates Theorem 2.

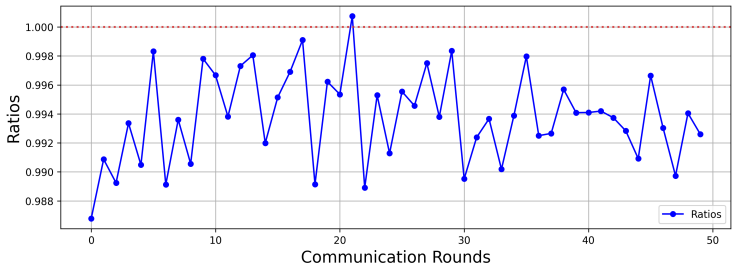


Figure 5: Verification of Theorem 2 with $(\lambda_1, \lambda_2) = (0.5, 0.55)$