Fed-Grow: Federating to Grow Transformers for Resource-Constrained Users without Model Sharing

Shikun Shen, Yifei Zou, *Member, IEEE*, Yuan Yuan, Hanlin Gu, Peng Li, *Senior Member, IEEE*, Xiuzhen Cheng, *Fellow, IEEE*, Falko Dressler, *Fellow, IEEE*, Dongxiao Yu, *Senior Member, IEEE*

***Abstract*—The growing resource demands of large-scale transformer models pose significant challenges for resource-constrained users, particularly in distributed environments. To address this issue, we propose a federated learning framework called Fed-Grow, which enables multiple participants to collaboratively learn a lightweight scaling operation that transfers knowledge from pretrained small models to a large transformer model. In Fed-Grow, we introduce the Dual-LiGO (Dual Linear Growth Operator) architecture, consisting of Local-LiGO and Global-LiGO components. Local-LiGO addresses model heterogeneity by adapting each participant's pre-trained model to a common intermediate form, while Global-LiGO facilitates knowledge sharing across participants without sharing local models or raw data, ensuring privacy preservation. This federated approach offers a scalable solution for growing large transformers in a distributed manner, where only the Global-LiGO is shared, significantly reducing communication overhead while maintaining comparable model performance under the same communication constraints. Experimental results demonstrate that Fed-Grow outperforms state-of-the-art methods in terms of accuracy and precision, while reducing the number of trainable parameters by 59.25% and communication costs by 73.01%. These improvements allow for higher efficiency in training large models in distributed environments, without sacrificing performance. To the best of our knowledge, Fed-Grow is the first method that enables cooperative transformer scaling in a distributed setting, making it a practical solution for resource-constrained users.**

***Index Terms*—Distributed Algorithm, Transformer, Federated Learning, Model Expansion**

I. Introduction

In recent years, the transformer with billions of parameters [1] has shown its high performance on addressing the various complex tasks in the domains of Natural Language Processing (NLP) [2]–[4] and Computer Vision (CV) [5]–[7]. However, the training of such large-scale transformers has a high requirement on the computing, storage, and data resources [4], [8], [9], and often precludes resource-constrained users from leveraging the advantages of these powerful models [10], [11]. Thus, an efficient and resource-friendly approach to obtain a transformer has been an active research area in recent years.

Shikun Shen, Yifei Zou, Yuan Yuan, Xiuzhen Cheng, and Dongxiao Yu are with the School of Computer Science and Technology, Shandong University, Qingdao, China. (Emails: shikunshen@mail.sdu.edu.cn, {yfzou, yyuan, xzcheng, dxyu}@sdu.edu.cn) Yifei Zou is the corresponding author.

Hanlin Gu is with WeBank AI Lab, WeBank, Shenzhen 518052, China. (E-mail: allengu@webank.com).

Peng Li is with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China (Email: pengli@xjtu.edu.cn).

Falko Dressler is with the School of Electrical Engineering and Computer Science, Technical University Berlin, 10587 Berlin, Germany (Email: dressler@ccs-labs.org).

Due to its significance, some relevant works have been proposed to reduce the resource consumption of training a transformer. Some noteworthy strategies include mixed precision training [12], large batch optimization [13], and dropping layers [14] or tokens [15]. A common ground of the above-mentioned approaches is that they all train transformers from scratch, which is not resource-saving if the targeted transformer is a scaled-up version of a pre-trained one.[1] Compared with the scratch-based methods, a more efficient idea is to reuse the smaller models to initialize a larger transformer, leveraging the implicit knowledge embedded in the pre-trained models. Based on this idea, a series of works have been proposed [16]–[23]. For instance, the Linear Growth Operator (LiGO) in [20] is used to learn a linear mapping from the parameters of a small model to a large one, which reduces the computational cost of scratch-based training by up to 50%, and outperforms several model-based baselines. Fig. 1 (a) and (b) are given to illustrate the scratch-based approach and pre-trained model-based approach, respectively. Even though the existing methods have demonstrated the effectiveness of reusing smaller pre-trained models to train larger ones, all of them are centralized under the single-machine environment. The corresponding problem on the distributed side still requires further investigation.

In this paper, we study the cooperative training of a transformer with several participants, each of which has a heterogeneous pre-trained small model and is resource-constrained to expand a transformer individually. For example, the data of each participant is not sufficient to train a transformer with high performance by itself. To expand a transformer from its own pre-trained model, the participants choose to share some useful knowledge with each other. Compared with traditional federated learning (FL), which relies on aggregating the full model parameters (Fig. 1 (c)), and centralized model-based approaches, cooperatively scaling a transformer with multiple participants has several advantages. First, it allows for the integration of diverse knowledge from the pre-trained models and dataset of different participants, potentially resulting in a comprehensive large model. Second, it enables distributed learning for high efficiency, which can significantly accelerate the training process by utilizing the computational resources of multiple users. Third, it offers an alternative approach for the resource-constrained users to cooperatively train a transformer.

[1]For example, the GPT-2 Large model with 774 million parameters is trained from scratch despite being a scaled-up version of GPT-2 Base with 117 million parameters.

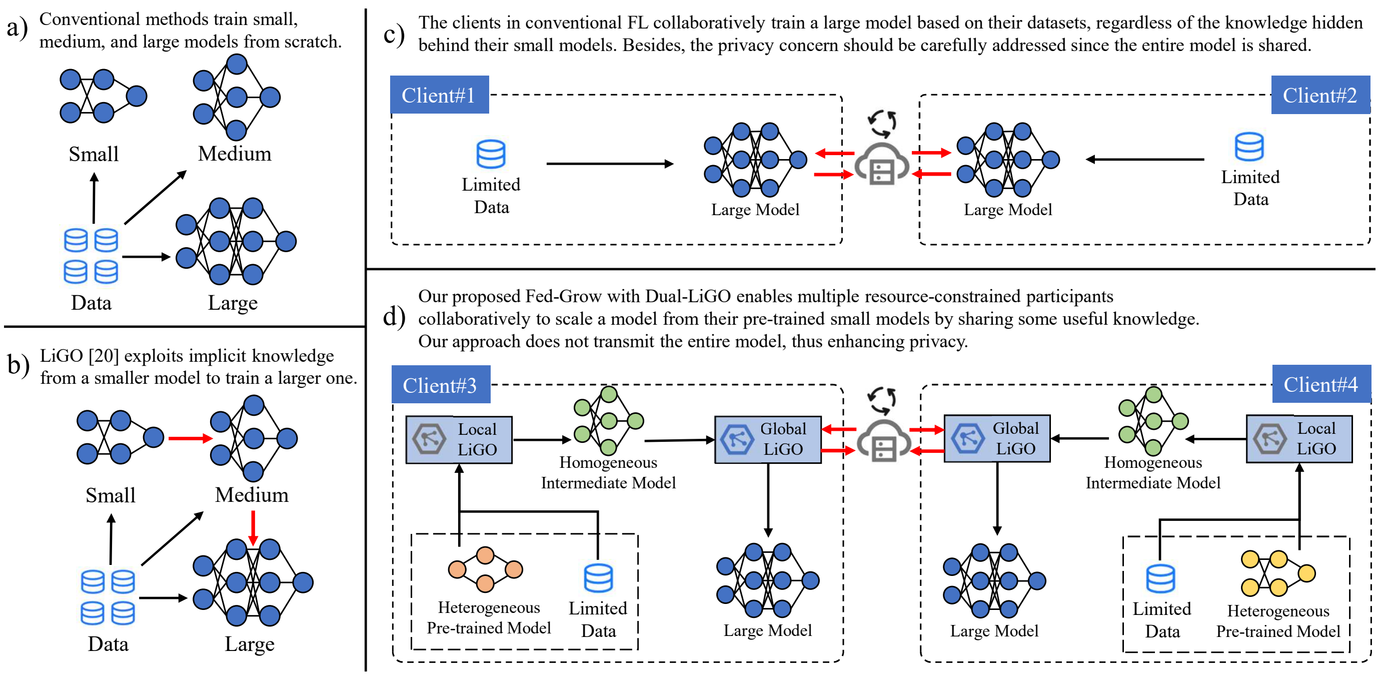

a) Conventional methods train small, medium, and large models from scratch.
Small
Medium
Data
Large
b) LiGO [20] exploits implicit knowledge from a smaller model to train a larger one.
Small
Medium
Data
Large
c) The clients in conventional FL collaboratively train a large model based on their datasets, regardless of the knowledge hidden behind their small models. Besides, the privacy concern should be carefully addressed since the entire model is shared.
Client#1
Limited Data
Large Model
Large Model
Limited Data
Client#2
d) Our proposed Fed-Grow with Dual-LiGO enables multiple resource-constrained participants collaboratively to scale a model from their pre-trained small models by sharing some useful knowledge. Our approach does not transmit the entire model, thus enhancing privacy.
Client#3
Local LiGO
Homogeneous Intermediate Model
Global LiGO
Heterogeneous Pre-trained Model
Limited Data
Large Model
Global LiGO
Homogeneous Intermediate Model
Local LiGO
Client#4
Large Model
Limited Data
Heterogeneous Pre-trained Model

Fig. 1: Illustration of different paradigms for efficient model training. (a) Conventional methods train small, medium, and large models from scratch. (b) Model reusing methods, such as the Linear Growth Operator (LiGO), exploit implicit knowledge from a smaller pre-trained model to initialize a larger one. (c) Conventional Federated Learning (FL) involves clients collaboratively training a large model based on their local datasets. (d) Our proposed Fed-Grow with Dual-LiGO enables resource-constrained participants to cooperatively scale a large model from their heterogeneous pre-trained small models by sharing only the growth operators.

Even though the cooperative scaling of transformers has such advantages, designing a distributed training algorithm is not an easy task. The challenges mainly come from the following aspects. First, determining what to share during the collaborative training process is non-trivial. An intuitive way is to scale their pre-trained small model to a large one and then share the expanded model. However, this approach would lead to a large communication burden due to the increased size of the models being transmitted. Second, heterogeneous pre-trained models pose a significant challenge, as traditional parameter-based aggregation is not applicable when participants have different architectures, sizes, and parameters. Third, participants are often unwilling to share their data or even their pre-trained small models and expanded models, as these models are considered valuable property and may raise significant privacy concerns if disclosed [24], [25]. Solving these challenges, particularly under the constraint of privacy protection, requires careful consideration of the following question: *What should be shared in the collaborative training process for knowledge integration, distributed training acceleration, and privacy protection?*

To answer this question, we introduce **Fed-Grow** framework with **Dual-LiGO** architecture. In Fed-Grow, multiple participants cooperatively grow a transformer from their pre-trained small models. The **Dual-LiGO** architecture enables the extension of heterogeneous small pre-trained models to large models in a federated framework. The key idea of Dual-LiGO is to design two LiGO operators: Local-LiGO and Global-LiGO. In the first, each participant uses the Local-LiGO to expand its heterogeneous pre-trained model to a homogeneous intermediate one. Then, the Global-LiGO is used by each participant to expand its intermediate model to a homogeneous transformer. In each local training process, gradient descent method is used to update the Local-LiGO and Global-LiGO. In each communication round, only the lightweight Global-LiGO is shared for aggregation. This shift from parameter-based to mapping-based aggregation is the core of our efficient federated process, providing a continuous advantage in every training step. Such a local training and federating process will be repeated for sufficient times until a transformer with high performance is obtained. By sharing only the compact Global-LiGO instead of the full model parameters, Fed-Grow dramatically reduces communication costs and enhances privacy, as the abstract mapping reveals far less information about local data than model gradients. An illustration is given in Fig. 1 (d). Note that there are also popular works such as LoRA [26] and prompt tuning [27], which are efficient fine-tuning methods based on existing pre-trained large models. Different from LoRA and prompt tuning, our work focuses on enabling a group of resource-constrained users to cooperatively grow a large model from their pre-trained heterogeneous small models, eliminating the need for a pre-existing large model. Our main contributions are summarized as follows:

1) We introduce Fed-Grow, a federated framework for multiple users to grow a transformer from their heteroge-

neous pre-trained models. To the best of our knowledge, this is the first work tackling the transformer expansion problem under a distributed and heterogeneous setting. Compared with the existing centralized works requiring sufficient data and computing resources on a single machine, Fed-Grow makes full use of the fragmented resources and the pre-trained models of multiple participants to cooperatively grow a transformer. With our Fed-Grow, those resource-constrained users who cannot afford a large transformer individually can also enjoy the bonus taken by transformers.

2) Under our Fed-Grow framework, the Dual Linear Growth Operator (Dual-LiGO) is designed for each participant to expand its pre-trained small model to a transformer. Specifically, the Dual-LiGO consists of two parts: Local-LiGO and Global-LiGO, both of which are trained locally but only the latter one will be shared. With the help of Global-LiGO, the participants share their knowledge in model expansion, which speeds up the training process and improves the final performance of the transformer. The Local-LiGO is designed to address the heterogeneity problem caused by pre-trained models with various architectures and sizes. Since the Local-LiGO is not shared, the adversary lacks the complete knowledge to recover the local data and pre-trained models of participants, which addresses the privacy concerns in federated learning.
3) We conduct extensive experiments to validate the effectiveness of our Fed-Grow framework. The results demonstrate that our collaborative approach significantly outperforms the non-federated LiGO baseline [20], boosting accuracy by an average of 14.76%. More importantly, when benchmarked against established federated learning paradigms like FedAvg, FedProx, and FedKD under an equivalent communication budget, Fed-Grow consistently achieves superior performance. This is accomplished while drastically reducing resource requirements: Fed-Grow cuts the number of trainable parameters by 59.25% and reduces per-round communication costs by 73.01% compared to standard federated approaches. Furthermore, our method enhances privacy protection, reducing Membership Inference Attack (MIA) [28] accuracy by up to 14.98% compared to full model sharing.

Roadmap: Section II reviews the related works. The problem setup is presented in section III and our Fed-Grow framework with Dual-LiGO is presented in section IV. Section V reports the numerical results in simulation. In Section VI, we conclude this paper and discuss the future work.

II. Related Work

The efficient training of transformers has been a topic of interest in recent years. Initial strategies include mixed precision training [12], large batch optimization [13], progressive layer dropping [14], token dropping [15], sparse training [29], and knowledge inheritance [19], which aim to lower the computational cost, memory usage, and communication overhead of training a transformer, while preserving or even improving the performance of transformer on downstream tasks. A common ground of these strategies is that they all train large models from scratch, thereby discarding the valuable knowledge acquired by their smaller counterparts.

Recognizing the inefficiency of training from scratch, a distinct line of research has focused on Model Growth, which reuses small pre-trained models to initialize a larger model. These methods employ a specific growth operator to map the parameters from a small source model to a large target model for better initialization, and have evolved from using fixed operators to learnable ones. Early approaches relied on fixed, rule-based operators. For example, StackBERT [16] and MSLT [17] are depth-only methods that employ a constant identity operator, initializing a deeper model by copying layer parameters from a shallower one. Net2Net [30] supports width expansion by randomly copying neurons. A more advanced method, bert2BERT [18], uses a more flexible fixed mapping based on function-preserving transformations to expand both width and depth. However, a major limitation of all these "fixed operator" approaches is that the mapping itself is fixed and not optimal. This inherently limits the quality of the grown large model, which must then be trained extensively, incurring significant computational costs. To address this limitation, recent state-of-the-art methods propose learnable growth operators. LiGO [20] takes a significant step by learning a linear mapping and factorizing it into width- and depth-growth components. By optimizing only this operator, it produces a superior large model more efficiently. TripLe [22] also fits in this category by partially scaling a model before training and growing the rest of the parameters during the training process. Most recently, Mango [23] further advances this by using multi-linear operators via tensor decomposition to capture global correlations between weights. By leveraging the implicit knowledge of the pre-trained models, these methods improve training efficiency and save significant resources. Crucially, this strategy has found practical application in the industry; for instance, the FLM-101B [31] model explicitly employs a growth strategy, expanding its structure offline and resuming training from a smaller checkpoint to significantly cut training costs.

Despite these advancements, a significant research gap remains. Most of these approaches are centralized and require considerable data and computational resources on a single machine. For example, 64 TPUs and ImageNet-1k dataset are required in TripLe [22], which prevents those resource-constrained users from discovering the capabilities of large models. Furthermore, there is a lack of relevant research in distributed scenarios. Thus, whether it is possible for multiple resource-constrained users to cooperatively expand a transformer from their pre-trained small models is still an open question in the existing research.

Distinction from Existing Distributed Frameworks. To overcome the limitations of centralized training, various distributed learning paradigms have also been developed. However, these frameworks typically address different goals and assumptions than our work. Standard Federated Averaging (FedAvg) [32] and its variants like FedProx [33] require a homogeneous model architecture, a foundational assumption that makes them unsuited for our core challenge of growing from diverse small models. Other approaches like Parameter-Efficient Fine-Tuning (PEFT) [26] or model parallelism (FSL [34], TITANIC [35]) focus on fine-tuning or partitioning a single, pre-existing large model, a prerequisite our work aims to remove. While Federated Knowledge Distillation (FedKD) [36] can handle model heterogeneity by using small models as teachers, its inherent inefficiency (requiring training and communication of the full large model) makes it less suitable for resource-constrained users compared to our direct growth strategy. Furthermore, recent frameworks like TransFed [37] utilize a hypernetwork to learn personalized focal modulation for each client, focusing on domain adaptation and personalization. In contrast, our Fed-Grow aims to aggregate these heterogeneous models into a single, stronger global backbone via direct growth. Finally, Personalized Federated Learning (PFL) [38]–[46] focuses on creating custom models for each client, which is a different objective than our goal of building a single, powerful global model from pretrained small models.

III. System Model and Problem Definition

We consider a setup with n participants (also termed clients in federated learning), and a parameter server to communicate with the clients for knowledge sharing. Each participant is assumed to be computationally capable of training a local small model, but may be resource-constrained when it comes to training large models independently. Importantly, the server has no access to private data or pre-trained small models of the clients, thereby ensuring privacy in federated learning. Each client i has its own data $\mathcal{D}_i = \{x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^{|\mathcal{D}_i|}$ and a pre-trained small model $\Theta_i^{(small)}$. Considering that the participants may come from different institutions, we assume that the data on each client is not independently and identically distributed (non-IID), to reflect the realistic and diverse data distributions across different clients. Critically, we assume the pre-trained small models $\Theta_i^{(small)}$ are architecturally heterogeneous, This means that the models across clients can differ in their number of layers L_i and hidden dimensions D_i, reflecting a realistic scenario where participants developed their models independently. This model heterogeneity is a central challenge, as standard federated aggregation techniques like FedAvg cannot be applied. Therefore, Fed-Grow must introduce a mechanism to bridge these architectural differences and enable meaningful knowledge integration into a unified global model.

The goal is to collaboratively grow a single, unified large model, $\Theta^{(large)}$, by leveraging the collective knowledge embedded in the distributed, heterogeneous small models, $\Theta_i^{(small)}$. Let $\ell(\Theta; x, y)$ denote the loss function for given model Θ and data point (x, y), such as the cross-entropy loss and the mean-squared loss. The underlying optimization goal of our Fed-Grow can be formalized as follows:

$$\min_{\Theta^{(large)}} \left\{ F(\Theta^{(large)}) := \sum_{i=1}^{n} \frac{|\mathcal{D}_i|}{\sum_{j=1}^{n} |\mathcal{D}_j|} F_i(\Theta^{(large)}) \right\} \quad (1)$$

where $F_i(\Theta^{(large)}) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\ell(\Theta^{(large)}; x, y)]$ denotes the average loss of $\Theta^{(large)}$ over local dataset $\mathcal{D}_i$. The objective of the optimization problem is to find $\Theta^{(large)}$ that minimizes the weighted sum of the average losses overall local datasets, which ensures that the large model performs well on all clients' data, taking into account the size of their dataset.

We denote the parameters of a neural network with L layers and D dimensions as $\Theta_{L,D} \in \mathbb{R}^{LD\times D}$. The LiGO in the original work [20] learns a linear mapping $\mathbf{M} : \mathbb{R}^{L_1 D_1 \times D_1} \to \mathbb{R}^{L_2 D_2 \times D_2}$ from the parameters of a smaller pre-trained model Θ_{L_1,D_1} to initialize a larger one Θ_{L_2,D_2}, i.e., $\Theta_{L_2,D_2} = \mathbf{M}(\Theta_{L_1,D_1})$ where $L_1 < L_2$ and $D_1 < D_2$. LiGO effectively reuses the knowledge from the smaller model to accelerate the training of the larger model. In the context of Fed-Grow, the LiGO method is extended to heterogeneous federated settings, where each client scales its own heterogeneous local model and shares this knowledge with the central server. The central server then aggregates these contributions and uses them to scale up the local models into a global larger model. This process allows the system to leverage pre-trained, smaller models while ensuring data privacy and minimizing communication costs. The federated scaling of the transformer model in Fed-Grow is both data- and model-efficient, as it avoids the need to train a large model from scratch on each client, instead allowing each client to contribute knowledge that leads to a large, globally shared transformer model.

The important notations in problem setup and our method are presented in Table I.

TABLE I: Table of Notations

Notation	Description
n	Number of clients
$\mathcal{D}_i$	Dataset of client i
$\Theta_{L,D}$	A neural network with L layers and D dimensions
$\Theta_i^{(small)}$	Pre-trained small model of client i
$\Theta_i^{(inter)}$	Intermediate model of client i
$\Theta_i^{(large)}$	Large model of client i
$\mathbf{M}_i^{(local)}$	Local-LiGO of client i
$\mathbf{M}_i^{(global)}$	Global-LiGO of client i
$\ell(\Theta; x, y)$	Loss function given model Θ and data point (x, y)
$F(\Theta^{(large)})$	Objective function of the large model
$F_i(\Theta^{(large)})$	Average loss of the large model over dataset $\mathcal{D}_i$
η_l	The learning rate of Local-LiGO
η_g	The learning rate of Global-LiGO
E_l	Number of epochs for training the Local-LiGO
E_g	Number of epochs for training the Global-LiGO
R	Number of rounds for aggregation round

IV. Algorithm Design

In this section, we introduce Fed-Grow with Dual-LiGO, a novel approach that enables multiple resource-constrained participants to collaboratively scale a transformer model from their pre-trained heterogeneous small models. This method is designed to enhance privacy by ensuring that neither the data nor the pre-trained models of the participants are shared during the collaborative process. While this privacy-preserving approach offers significant advantages, it also introduces complexities that necessitate careful algorithm design.

A. Overview of Our Method

Building upon the problem setup, we now turn our attention to the challenges posed by the heterogeneous model settings in a multi-user collaborative scenario. Specifically, we need to figure out what to share in the federated learning framework, to guarantee efficiency and privacy simultaneously.

To solve the challenge of collaborative growth from heterogeneous models, we propose the **Dual-LiGO architecture**, a two-stage mechanism that decouples local model adaptation from global knowledge aggregation.

1) **Local Adaptation (via Local-LiGO):** First, each client uses a private Local-LiGO operator. This operator is unique to each client and is trained to map its specific heterogeneous small model, $\Theta_i^{(small)}$, to a common, pre-defined intermediate architecture, $\Theta^{(inter)}$. This crucial step resolves the model heterogeneity problem, creating a standardized foundation for collaboration.
2) **Global Growth (via Global-LiGO):** Second, all clients use a shared Global-LiGO operator to expand their now-homogeneous intermediate models into the final large model, $\Theta^{(large)}$. Crucially, it is only this lightweight Global-LiGO operator that is communicated and aggregated. This represents a fundamental shift from traditional parameter-based aggregation to a novel and highly efficient mapping-based aggregation paradigm.

This dual-operator design allows Fed-Grow to effectively integrate knowledge from diverse sources while preserving the privacy of each client's unique small model and local data. Furthermore, by reusing knowledge from pre-trained models, Dual-LiGO accelerates convergence and improves the performance of the final large model.

B. Dual-LiGO Architecture

The Dual-LiGO architecture consists of two main components: the Local-LiGO and the Global-LiGO.

Local-LiGO. The Local-LiGO is a private, client-specific operator responsible for resolving model heterogeneity. For each client i, its Local-LiGO, denoted $\mathbf{M}_i^{(local)}$, learns a linear mapping to expand its unique pre-trained small model $\Theta_i^{(small)}$ (with architecture L_i, D_i) into a fixed, common intermediate architecture $\Theta^{(inter)}$ (with architecture L_{inter}, D_{inter}).

The unification is achieved because the shape of each $\mathbf{M}_i^{(local)}$ is tailored to bridge the client's specific model to the common target. For example, if Client 1 has a model with a hidden dimension $D_1 = 256$ and Client 2 has a model with $D_2 = 192$, their respective Local-LiGOs will contain mapping matrices of shapes like $256 \times D_{inter}$ and $192 \times D_{inter}$. Although the operators $\mathbf{M}_i^{(local)}$ are different for each client and are never shared, they all produce an intermediate model with the same, unified architecture:

$$\Theta_i^{(inter)} = \mathbf{M}_i^{(local)}(\Theta_i^{(small)}). \tag{2}$$

The parameters of $\mathbf{M}_i^{(local)}$ are trained locally by minimizing the loss on the client's private data $\mathcal{D}_i$, keeping the pre-trained model $\Theta_i^{(small)}$ frozen:

$$\min_{\mathbf{M}_i^{(local)}} J(\mathbf{M}_i^{(local)}) := \big\{\mathbb{E}_{(x,y)\sim\mathcal{D}_i} \\ [\ell(\mathbf{M}_i^{(local)}(\Theta_i^{(small)}); x, y)]\big\} \tag{3}$$

To achieve this goal, each client updates $\mathbf{M}_i^{(local)}$ using the following gradient descent rule:

$$\mathbf{M}_i^{(local)} \leftarrow \mathbf{M}_i^{(local)} - \eta_l \nabla_{\mathbf{M}_i^{(local)}} J(\mathbf{M}_i^{(local)}) \tag{4}$$

where η_l is the learning rate for updating $\mathbf{M}_i^{(local)}$.

Global-LiGO. The Global-LiGO, denoted $\mathbf{M}^{(global)}$, is the shared component responsible for collaborative knowledge aggregation. It is a single, unified operator that learns a linear mapping from the common intermediate model architecture (L_{inter}, D_{inter}) to the final large model architecture (L_{large}, D_{large}). All clients use the same Global-LiGO:

$$\Theta_i^{(large)} = \mathbf{M}^{(global)}(\Theta_i^{(inter)}). \tag{5}$$

During local training, each client i computes updates for its local copy of the Global-LiGO, $\mathbf{M}_i^{(global)}$, based on its data. After local training, only the parameters of this lightweight $\mathbf{M}_i^{(global)}$ are sent to the server for aggregation using standard federated averaging. This aggregated operator is then broadcast back to the clients for the next round. The local optimization objective is:

$$\min_{\mathbf{M}_i^{(global)}} J(\mathbf{M}_i^{(global)}) := \big\{\mathbb{E}_{(x,y)\sim\mathcal{D}_i} \\ [\ell(\mathbf{M}_i^{(global)}(\Theta_i^{(inter)}); x, y)]\big\} \tag{6}$$

Likewise, each client updates its local copy of the Global-LiGO using the following gradient descent rule:

$$\mathbf{M}_i^{(global)} \leftarrow \mathbf{M}_i^{(global)} - \eta_g \nabla_{\mathbf{M}_i^{(global)}} J(\mathbf{M}_i^{(global)}) \tag{7}$$

where η_g is the learning rate for updating $\mathbf{M}_i^{(global)}$.

The workflow of Dual-LiGO is illustrated in Fig. 2, and its pseudocode is given in Algorithm 1 and 2. For each client, it firstly train the $\mathbf{M}_i^{(local)}$ locally for E_l epochs, and get the intermediate homogeneous model via $\mathbf{M}_i^{(local)}$. Next, it trains Global-LiGO locally for E_g epochs and sends the Global-LiGO to the parameter server for aggregation. The aggregated Global-LiGO will be used as the starting point of the next E_g epochs local training. Such a local training and aggregation process will be repeated for R times. Finally, based on the intermediate model and Global-LiGO, each client gets a homogeneous large model.

C. Efficiency and Privacy Preserving of Fed-Grow

Efficiency over Alternative Paradigms. Our approach is architecturally designed to be significantly more resource-efficient than other federated paradigms for the task of collaborative model growth. The standard parameter-based aggregation of FedAvg [32] is often impractical, as it requires clients

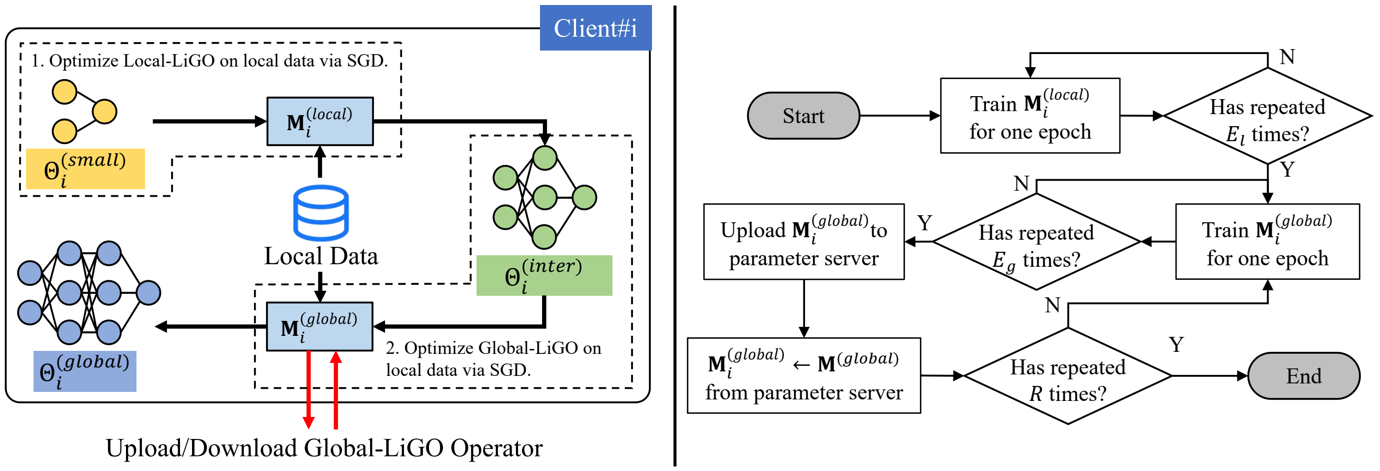

Client#i
1. Optimize Local-LiGO on local data via SGD.
$\Theta_i^{(small)}$
$\mathbf{M}_i^{(local)}$
Local Data
$\Theta_i^{(inter)}$
$\Theta_i^{(global)}$
$\mathbf{M}_i^{(global)}$
2. Optimize Global-LiGO on local data via SGD.
Upload/Download Global-LiGO Operator
Start
Train $\mathbf{M}_i^{(local)}$ for one epoch
Has repeated E_l times?
N
Y
Train $\mathbf{M}_i^{(global)}$ for one epoch
Has repeated E_g times?
Upload $\mathbf{M}_i^{(global)}$ to parameter server
$\mathbf{M}_i^{(global)} \leftarrow \mathbf{M}^{(global)}$ from parameter server
Has repeated R times?
End

Fig. 2: Workflow of the Dual Linear Growth Operator (Dual-LiGO) architecture. The process consists of two stages: (1) Local Adaptation, where the private Local-LiGO ($\mathbf{M}_i^{(local)}$) maps the client's specific heterogeneous small model to a common intermediate representation; and (2) Global Growth, where the shared Global-LiGO ($\mathbf{M}_i^{(global)}$) expands the intermediate model to the final large model. Crucially, only the lightweight Global-LiGO operator is uploaded to the parameter server for aggregation, preserving model and data privacy.

Algorithm 1 Fed-Grow with Dual-LiGO for Client i

```
1: Input: Local dataset $\mathcal{D}_i$; Federated round $R$;
2:        Pre-trained small model $\Theta_i^{(small)}$;
3:        Local-LiGO training epoch $E_l$;
4:        Global-LiGO training epoch $E_g$;
5: Output: The global large model $\Theta_i^{(large)}$;
   // Train the Local-LiGO $\mathrm{M}_i^{(local)}$ for $E_l$ epochs.
6: for $E_l$ epochs do
7:     Train $\mathrm{M}_i^{(local)}$ on local data $\mathcal{D}_i$ with equation 3 as the
       goal and equation 4 as the gradient descent rule;
8: end for
   // Train and upload the Global-LiGO for $R$ rounds.
9: for $r = 1$ to $R$ do
10:    for $E_g$ epochs do
11:        Train $\mathrm{M}_i^{(global)}$ on local data $\mathcal{D}_i$ with equation 6 as
           the goal and equation 7 as the gradient descent rule;
12:    end for
13:    Upload its Global-LiGO $\mathrm{M}_i^{(global)}$ to parameter server;
14:    $\mathrm{M}_i^{(global)} \leftarrow$ aggregated result from parameter server ;
15: end for
16: Obtain global large model $\Theta_i^{(large)}$ via $\mathrm{M}_i^{(global)}$;
```

Algorithm 2 Fed-Grow with Dual-LiGO for Parameter Server

```
1: for $r = 1$ to $R$ do
2:     Receive $M_i^{(global)}$ from all clients;
3:     $\mathrm{M}^{(global)} \leftarrow \sum_{i=1}^{n} \frac{|\mathcal{D}_i|}{\sum_{j=1}^{n} |\mathcal{D}_j|} \mathrm{M}_i^{(global)}$;
4:     Send $\mathrm{M}^{(global)}$ to all clients;
5: end for
```

to train and communicate the entire large model in every round, leading to prohibitive computational and communication costs. A potential alternative, adapting Federated Knowledge Distillation (FedKD) [36] by using clients' small models as "teachers" for the large global "student", proves to be even less efficient. Such a setup would still require communicating the full parameters of the large student model, matching the high communication cost of FedAvg. Moreover, it would increase the client-side computational load, as each client would need to run both its local teacher and the large student model to compute the distillation loss. Fed-Grow avoids these pitfalls. By design, it fundamentally reduces both the number of trainable parameters and the volume of communicated data. Clients only update and exchange the compact Global-LiGO operator, which lowers both the local computational burden and the network bandwidth requirements. This makes Fed-Grow a practical and feasible solution for growing large models in resource-constrained environments. These efficiency gains are empirically quantified in Section V.

Privacy Preservation. Fed-Grow with Dual-LiGO ensures privacy by sharing only the Global-LiGO operator, a learned linear mapping, rather than raw model parameters or gradients. This design significantly reduces the risk of privacy leakage, as the Global-LiGO does not directly reveal the structure or parameters of the local models. Meanwhile, by avoiding the sharing of sensitive information, Fed-Grow minimizes the attack surface and mitigates common privacy attacks, such as model inversion, membership inference, gradient-based attacks, and model extraction. This approach maintains robust privacy preservation while enabling collaborative knowledge sharing among clients.

Continuous Efficiency in Dynamic Scenarios. Furthermore, the advantages of Fed-Grow extend beyond the initial training phase to dynamic scenarios where new data becomes available. In traditional parameter-based approaches, adapting the global large model to new local data requires resuming the heavy training and communication process of the full model. In contrast, Fed-Grow allows clients to efficiently update their lightweight local models and the mapping operator. By fine-tuning the small model and the Dual-LiGO architecture, the system can adapt to new data while maintaining the same low communication and computation profile. This ensures that the efficiency advantage is continuous throughout the entire lifecycle of the model, not just during initialization.

V. Experiments

In this section, we first describe the experimental setup, including datasets, models, and the environment used for evaluation. Then, we evaluate the performance of our proposed Fed-Grow framework with Dual-LiGO in terms of accuracy/precision, stability, resource consumption, and privacy. Finally, we summarize the results to highlight the strengths and contributions of our method.

A. Experimental Setup

1) Datasets and Models:

- *Text Classification*: We use the 20 Newsgroups[2] [47] and AG News datasets[3] [48] for text classification tasks. The 20 Newsgroups dataset comprises approximately 20,000 documents across 20 different newsgroups. The AG News dataset is a subset of AG's corpus of news articles, containing 30,000 training and 1,900 test samples per class, totaling 120,000 training and 7,600 testing samples. For these tasks, we employ the BERT [2] model.
- *Sequence Tagging*: For sequence tagging tasks, we utilize the WikiNER dataset[4] [49], a large collection of sentences from Wikipedia predicting four tags: "PER" (person names), "LOC" (location names), "ORG" (organization names), and "MISC" (miscellaneous names). We also use the BERT [2] model for this task.
- *Image Classification*: We employ CIFAR-10 [50], CIFAR-100[5] [50], and Flowers-102[6] [51] datasets for image classification tasks. CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images dataset, designed for image classification. Flowers-102 consists of 102 flower categories commonly found in the United Kingdom. For these tasks, we use the Vision Transformer (ViT) [5] model.

For each client's small model, denoted as $\Theta^{(small)}$, we design three different sizes to simulate a heterogeneous environment. Details about the sizes of small models $\Theta^{(small)}$, intermediate models $\Theta^{(inter)}$, and large models $\Theta^{(large)}$ are provided in Table II. For both BERT and ViT, we adhere to the configurations shown in Table II. Each client's small model configuration is randomly and evenly selected from the three small-scale options outlined in the table.

[2]Source: http://qwone.com/~jason/20Newsgroups/
[3]Source: http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
[4]Source: https://huggingface.co/datasets/mnaguib/WikiNER
[5]Source: https://www.cs.toronto.edu/~kriz/cifar.html
[6]Source: https://www.robots.ox.ac.uk/~vgg/data/flowers/102/

2) Training Environment: Our experimental platform is built upon Flower [52] and FedNLP [53], utilizing PyTorch as the training backend. All simulations are conducted on four NVIDIA RTX 4090 GPUs with a total of 320GB memory. We perform experiments under various conditions to thoroughly evaluate our models' performance across six different tasks, ensuring a comprehensive evaluation across diverse domains.

3) Hyperparameters and Data Distribution: We initialize all learnable parameters in the Local-LiGO and Global-LiGO operators using Xavier Normal initialization (specifically `nn.init.xavier_normal_` in PyTorch). Regarding hyperparameter configurations, we adopted a consistent strategy across domains. We set the learning rates to a magnitude of 1×10^{-4} for both CV and NLP tasks. To ensure convergence across varying dataset sizes and complexities, we tuned the training duration parameters (E_l, E_g, and R) for each task. Detailed configurations are provided in Table III. For data partitioning, following [53], [54], we use the Dirichlet distribution with $\beta = 0.5$ to generate non-IID data and $\beta = 100$ to generate IID data, maintaining a fixed random seed. We test our models with different numbers of clients, specifically 10 and 20 clients.

TABLE II: Configurations of Different Transformer Models

Name	#Hidden	#Layers	#Heads
small#1	256	2	8
small#2	256	3	8
small#3	256	4	8
intermediate	320	4	8
large	384	6	8

TABLE III: Hyperparameters for Each Task

Task	Batch Size	LR	E_l	E_g	R
20news	32	3e-4	100	10	200
agnews	32	1e-4	10	1	100
wikiner	32	1e-4	50	3	100
CIFAR-10	64	5e-4	50	4	200
CIFAR-100	64	5e-4	50	4	200
Flowers-102	64	5e-4	200	40	100

4) Metrics: We evaluate our method using a comprehensive set of metrics. For classification tasks, we report top-1 *accuracy*, while for sequence tagging, we use *precision*. *Stability* is assessed via the standard deviation of these primary metrics across clients. *Resource Efficiency* is quantified by measuring (1) the number of trainable parameters, (2) the communication cost per round, and (3) client-side computational load, including GPU memory, GFLOPs, and training time per iteration. Finally, we evaluate *Privacy* by reporting the success rate of a standard Membership Inference Attack (MIA) [28].

5) Baselines: We compare our proposed Fed-Grow framework against a set of carefully chosen baselines, each designed to isolate and evaluate a specific aspect of our method's performance.

- *NoAgg (Individual LiGO):* This baseline represents the original, non-federated LiGO algorithm [20] applied by each client individually. Each client grows and trains

TABLE IV: Comparison of Accuracy/Precision and Standard Deviation between Fed-Grow (Agg) and NoAgg [20]

Model	Task	**Accuracy/Precision (%) ↑**											
		10 Clients						20 Clients					
		IID			Non-IID			IID			Non-IID		
		Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ
BERT	20news	**65.295**	49.177	16.118	**53.983**	37.759	16.224	**48.258**	45.470	2.788	**27.787**	18.693	9.094
	agnews	**90.158**	87.474	2.684	**85.671**	73.079	12.592	**86.737**	86.053	0.684	**84.250**	49.303	34.947
	wikiner	**42.581**	28.324	14.257	**43.878**	33.539	10.339	**43.693**	28.803	14.890	**42.558**	27.300	15.258
ViT	CIFAR-10	**76.717**	67.977	8.740	**70.186**	48.771	21.415	**78.223**	70.020	8.203	**71.914**	53.730	18.184
	CIFAR-100	**53.178**	38.127	15.051	**50.109**	26.904	23.205	**54.004**	39.141	14.863	**51.270**	28.457	22.813
	Flowers102	**38.555**	25.000	13.555	**37.928**	20.326	17.602	**31.548**	12.448	19.100	**31.904**	10.349	21.555
Model	Task	**Standard Deviation ↓**											
		10 Clients						20 Clients					
		IID			Non-IID			IID			Non-IID		
		Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ
BERT	20news	**3.546**	24.219	-20.673	**11.960**	23.161	-11.201	**3.518**	14.605	-11.087	**3.518**	6.112	-2.594
	agnews	**1.001**	4.362	-3.361	**4.350**	28.025	-23.675	**1.573**	2.100	-0.527	**14.605**	16.340	-1.735
	wikiner	**3.513**	9.069	-5.556	**1.360**	11.288	-9.928	**2.441**	9.689	-7.248	**2.241**	11.109	-8.868
ViT	CIFAR-10	**1.482**	2.908	-1.426	**1.870**	5.168	-3.298	**1.251**	2.280	-1.029	**1.321**	5.225	-3.904
	CIFAR-100	**1.045**	1.848	-0.803	**1.332**	3.099	-1.767	**1.401**	1.795	-0.394	**1.551**	2.282	-0.731
	Flowers102	**1.419**	1.657	-0.238	**1.399**	2.609	-1.210	**1.467**	2.305	-0.838	**1.038**	1.638	-0.600

a large model from its small pre-trained model in isolation, without any communication or aggregation. We compare Fed-Grow against NoAgg on performance metrics (accuracy/precision) and stability to demonstrate that our collaborative aggregation mechanism is essential for achieving superior and more consistent results.

- *FedAvg (from Scratch):* This is the standard Federated Averaging [32] baseline, where clients collaboratively train the target large model from a random initialization by communicating the full model parameters. We compare against FedAvg on performance under an equivalent communication budget, resource efficiency, and privacy metrics to demonstrate that Fed-Grow is a more efficient and secure alternative to conventional federated training.
- *FedProx (from Scratch):* A well-established baseline that improves upon FedAvg by adding a proximal term to handle data heterogeneity [33]. We compare against FedProx on performance under an equivalent communication budget to test if a standard method for non-IID data can match the performance of our growth-based approach.
- *Federated Knowledge Distillation (FedKD):* This baseline is adapted from [36] where clients' small models act as local "teachers" to collaboratively train the large global "student" model. This paradigm handles model heterogeneity via knowledge transfer. We compare against FedKD on performance and resource efficiency metrics to show that our model growth strategy is a more practical and effective approach than adapting knowledge distillation for this specific task.

B. Performance of Fed-Grow

In this section, we evaluate the performance of our Fed-Grow framework from multiple angles. We first demonstrate its superiority over non-collaborative training and then compare it against key federated baselines to assess its efficiency and effectiveness. Finally, in Appendix, we analyze its robustness to varying degrees of model heterogeneity and client scalability.

1) Comparison with Individual Training (NoAgg): We first establish the benefit of collaboration by comparing Fed-Grow against the **NoAgg** baseline, where each client trains a large model in isolation using the original LiGO algorithm. The results, presented in Table IV and Figure 3, show that our collaborative approach is consistently and significantly superior. Across all datasets and under both IID and non-IID conditions, Fed-Grow achieves higher accuracy and precision. For instance, on CIFAR-100 (non-IID, 10 clients), Fed-Grow improves accuracy by over 23 absolute points compared to "NoAgg". Furthermore, the standard deviation of performance across clients is drastically reduced (e.g., from 28.03 to 4.35 on Agnews non-IID), demonstrating that our framework not only improves average performance but also promotes stability and fairness among participants.

2) Comparison with Federated Baselines under an Equivalent Communication Budget: To provide the most direct and fair assessment of efficiency, we compare the final performance of Fed-Grow against a suite of key federated baselines. For this comparison, we constrain all baseline methods to the same total communication budget as that used by Fed-Grow throughout its entire training process. Specifically, we set the total communication budget to 4GB (each client)

TABLE V: Performance Comparison with Baselines

Dataset	Method	Accuracy/Precision (%)			
		10 Clients		20 Clients	
		IID	Non-IID	IID	Non-IID
20news	**Fed-Grow**	**65.30**	**53.98**	**48.26**	**27.79**
	FedAvg	63.73	50.45	34.52	27.75
	FedProx	30.54	42.90	26.06	14.89
	FedKD	61.62	52.32	46.80	25.27
agnews	**Fed-Grow**	**90.16**	**85.67**	**86.74**	**84.25**
	FedAvg	68.41	58.46	76.84	75.26
	FedProx	41.57	34.07	33.25	36.57
	FedKD	88.81	83.55	83.95	83.16
wikiner	**Fed-Grow**	**42.58**	**43.88**	**43.69**	**42.56**
	FedAvg	39.50	38.92	42.01	40.70
	FedProx	37.10	38.26	37.67	38.28
	FedKD	39.06	38.26	38.90	39.05
CIFAR-10	**Fed-Grow**	**76.72**	**70.19**	**78.22**	**71.91**
	FedAvg	68.48	63.37	68.60	62.57
	FedProx	35.66	31.07	35.46	28.21
	FedKD	68.33	63.13	69.14	63.20
CIFAR-100	**Fed-Grow**	**53.18**	**50.11**	**54.00**	**51.27**
	FedAvg	45.23	42.10	45.05	43.26
	FedProx	36.33	11.47	13.26	12.45
	FedKD	44.67	39.72	42.86	39.95
Flowers-102	**Fed-Grow**	**38.56**	**37.93**	**31.55**	**31.90**
	FedAvg	35.99	34.26	30.50	29.66
	FedProx	29.17	30.05	16.56	15.21
	FedKD	37.74	37.18	16.59	13.98

for the 20news, CIFAR-10, and CIFAR-100 datasets, and to 2GB (each client) for the agnews, wikiner, and Flowers-102 datasets. All baseline methods are run until this budget is exhausted. The comprehensive results are presented in Table V. The results show that Fed-Grow is the superior approach, achieving significantly higher accuracy and precision across nearly all datasets and settings. A deeper analysis reveals two key factors behind this success:

- *The Critical Role of Knowledge Reuse:* The most significant performance gap is observed against FedAvg (from Scratch). While FedProx is specifically designed to mitigate data heterogeneity, its stabilizing mechanism is insufficient to overcome the large performance deficit from a random initialization. This is starkly demonstrated on CIFAR-100 with non-IID data (10 clients), where Fed-Grow achieves 50.11% accuracy, which is over 38 absolute points higher than FedProx's 11.47%. This proves that intelligently reusing knowledge from pre-trained models, as Fed-Grow does, is a more critical factor for success than only addressing the data distribution with a proximal term.
- *A More Effective Knowledge Transfer Mechanism:* The comparison with FedKD highlights the robustness of our growth strategy. While FedKD also leverages the small models, it may struggle when the "teacher" model provides a noisy or underdeveloped signal to a high-capacity "student" model. This is particularly evident on the Flowers-102 dataset with 20 clients, where Fed-Grow's performance is more than double that of FedKD (31.90% vs. 13.98%). Our direct model-growth strategy proves to be a more stable and effective mechanism for integrating distributed knowledge.

In summary, Fed-Grow's unique combination of reusing pre-trained knowledge and employing a direct, collaborative growth mechanism allows it to make far more effective use of the available communication budget, resulting in a stronger final model. The framework's robustness is further demonstrated in Appendix, where we show that it maintains its effectiveness across varying degrees of architectural heterogeneity and scales well with the number of participating clients. These combined strengths highlight its suitability for resource-constrained distributed training.

C. Communication and Computation Resource Analysis

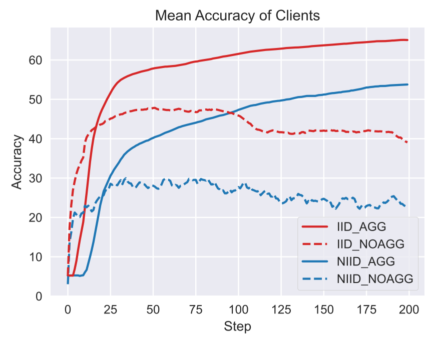
In this part, we provide a detailed analysis of the resource consumption for Fed-Grow and our baseline methods, benchmarked using our computer vision (CIFAR-100) model configurations. We evaluate the parameter and communication costs, as well as the client-side computational burden, to highlight the significant efficiency gains of our approach.

1) Parameter and Communication Efficiency: A primary advantage of Fed-Grow's mapping-based aggregation paradigm is its dramatic reduction in the number of parameters that need to be trained locally and communicated in each federated round. Table VI provides a detailed breakdown of these costs for our three heterogeneous ViT model configurations compared to the FedAvg and FedKD baselines. As shown in Table VI, both FedAvg and FedKD require training and communicating the entire large ViT model in every round, resulting in 7.74M trainable parameters and a staggering communication cost of 15.48M parameters per round. In contrast, Fed-Grow's costs are substantially lower. On average, Fed-Grow reduces the number of trainable parameters by 59.25% and slashes the per-round communication cost by 73.01%. This transformative reduction in communication overhead is a key advantage of our approach.

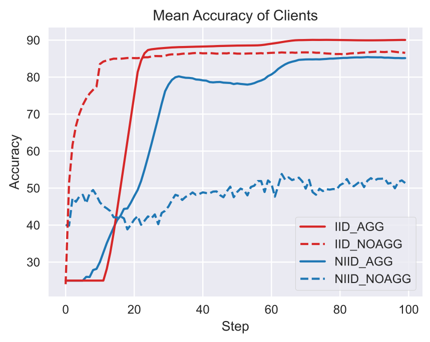
TABLE VI: Comparison of Trainable and Communicated Parameters for Fed-Grow vs. Federated Baselines, using the ViT model configurations for CIFAR-100

Method	Component	#Trainable Params	Comm. Params
Fed-Grow (Case #1)	Small Model	0	0
	Local-LiGO	0.737M	0
	Global-LiGO	2.089M	4.178M
Fed-Grow (Case #2)	Small Model	0	0
	Local-LiGO	1.065M	0
	Global-LiGO	2.089M	4.178M
Fed-Grow (Case #3)	Small Model	0	0
	Local-LiGO	1.393M	0
	Global-LiGO	2.089M	4.178M
FedAvg (from Scratch)	Large Model	7.740M	15.480M
FedKD	Large Model	7.740M	15.480M
Fed-Grow (Averaged Total)		3.154M	4.178M
Improvement vs. Baselines		**59.25%**	**73.01%**

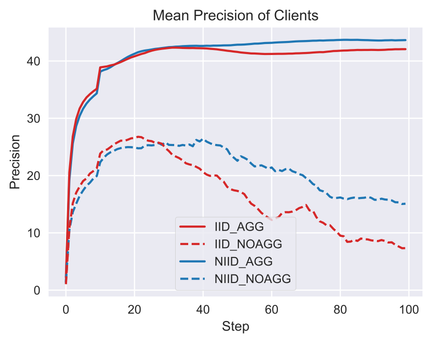
2) Client-Side Computational Burden: Beyond parameter counts, we measured the per-iteration computational load on the client, again using the CIFAR-100 ViT configurations. Table VII presents the results, averaged across the three



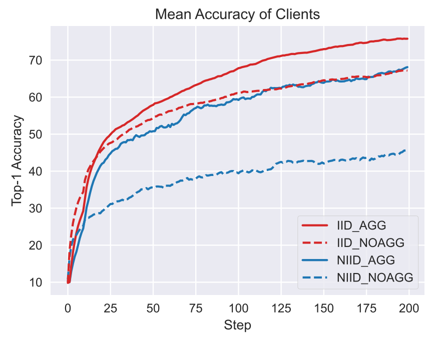
(a) 20news



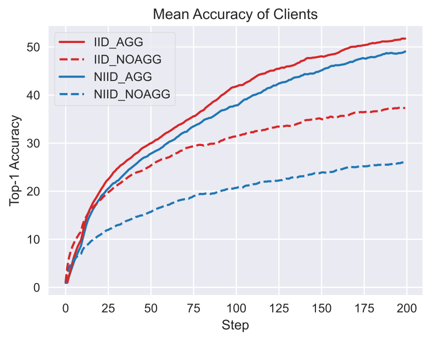
(b) agnews



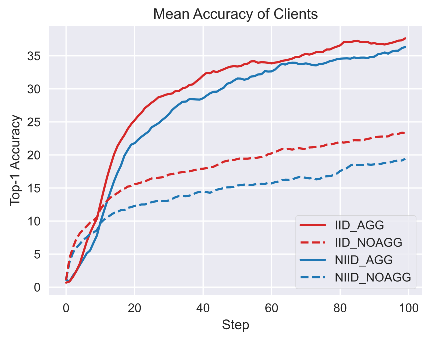
(c) wikiner



(d) CIFAR-10



(e) CIFAR-100



(f) Flowers-102

Fig. 3: Comparison of our proposed Fed-Grow (labeled as Agg) against the individual training baseline (labeled as NoAgg) on six datasets. The subfigures display the mean test accuracy (or precision for wikiner) under both Independent and Identically Distributed (IID) and Non-IID settings with 10 clients.

TABLE VII: Client-Side Resource Consumption per Iteration

Method	Time/iter (ms)	GFLOPs (fwd)	Static Mem (MB)
Fed-Grow (Ours)	28.87	141.76	**56.03**
FedAvg (from Scratch)	27.68	141.76	123.84
FedKD	31.03	176.63	129.68

heterogeneous settings. In terms of wall-clock time, Fed-Grow (28.87 ms) remains highly competitive, showing only a marginal overhead compared to FedAvg (27.68 ms) while being notably faster than FedKD (31.03 ms). Despite this slight increase in per-iteration time, Fed-Grow offers a superior profile for resource-constrained devices by drastically reducing memory usage. As shown in Table VII, it reduces the static memory (for model weights, gradients, and optimizer state) by approximately 54.8% compared to FedAvg. This is a direct result of our mapping-based approach, which only requires gradients and Adam optimizer states for the compact LiGO operators, rather than for the entire large model. This massive memory saving is present throughout the entire training process, making Fed-Grow a far more viable option for devices with limited memory, a common bottleneck in real-world scenarios. FedKD is the most demanding in all computational metrics, requiring more FLOPs, time, and memory due to the need to run both a teacher and student model.

D. Ablation Study on Key Design Components

To validate the critical design choices of our Fed-Grow framework, we conduct a comprehensive ablation study that isolates the contribution of each component in the Dual-LiGO architecture and our aggregation strategy. We investigate three key questions: (1) Is collaborative training necessary? (2) Is the trained Local-LiGO necessary for knowledge transfer? (3) Is our strategy of sharing the Global-LiGO operator superior to the more naive approach of sharing the full model?

The results of these ablations are presented in Table VIII.

- *Necessity of Collaboration (Ablation on Global-LiGO):* The benefit of collaboration is clearly demonstrated by comparing our full method against the NoAgg baseline (which is equivalent to using only a trained Local-LiGO). As shown in our main results (Table IV), the performance without aggregation is drastically lower. This establishes that the collaborative training of the Global-LiGO is essential for achieving high performance.
- *Necessity of Knowledge Transfer (Ablation on Trained Local-LiGO):* To test the importance of the Local-LiGO, we evaluate a Global-LiGO Only variant, where the Local-LiGO is a fixed, randomly initialized projection. As shown in Table VIII, this variant's performance collapses across all datasets. On Agnews (IID, 10 clients), accuracy plummets from 90.16% to just 25.00%. This proves that without a trained Local-LiGO to effectively transfer

knowledge from the pre-trained models, the collaborative process fails.

- *Superiority of the Aggregation Strategy:* We also test the Ideal FedLiGO baseline. In this scenario, each client first uses its private Local-LiGO to generate a large model. These large models are then averaged on the server, and the resulting global model is sent back to the clients. Each client then updates its own Local-LiGO to minimize the distance between its generated model and the averaged global model. This approach, while a form of collaborative refinement, suffers from the same massive communication overhead as standard FedAvg, as the full large model must be transmitted in both directions in every round. While this method benefits from the LiGO-based initialization, it is consistently and significantly outperformed by our full Fed-Grow approach. For instance, on CIFAR-100 with non-IID data (10 clients), our method achieves 50.11% accuracy, whereas the Ideal FedLiGO baseline only reaches 31.17% (Table VIII). This confirms that our strategy of sharing the compact Global-LiGO operator is a more effective and efficient form of collaboration than attempting to align the full model parameters.

In conclusion, this comprehensive set of ablations validates our complete Dual-LiGO design. The results confirm that all components are indispensable and that our strategy of collaborating on the growth process itself is superior to alternative federated approaches.

TABLE VIII: Comprehensive Ablation Studies

Dataset	Method	Accuracy/Precision (%)			
		10 Clients		20 Clients	
		IID	Non-IID	IID	Non-IID
CIFAR-10	**Fed-Grow**	**76.72**	**70.19**	**78.22**	**71.91**
	Global-LiGO Only	72.34	68.11	71.34	66.76
	Ideal FedLiGO	61.77	55.54	62.23	54.66
CIFAR-100	**Fed-Grow**	**53.18**	**50.11**	**54.00**	**51.27**
	Global-LiGO Only	48.70	43.64	46.32	41.27
	Ideal FedLiGO	37.26	31.17	35.90	34.63
Flowers-102	**Fed-Grow**	**38.56**	**37.93**	**31.55**	**31.90**
	Global-LiGO Only	36.36	33.80	30.96	27.66
	Ideal FedLiGO	31.94	30.70	14.50	12.34
20news	**Fed-Grow**	**65.30**	**53.98**	**48.26**	**27.79**
	Global-LiGO Only	15.94	13.28	11.17	5.38
	Ideal FedLiGO	61.62	48.87	41.48	25.26
agnews	**Fed-Grow**	**90.16**	**85.67**	**86.74**	**84.25**
	Global-LiGO Only	25.00	55.52	25.00	25.00
	Ideal FedLiGO	84.21	73.28	76.55	74.64
wikiner	**Fed-Grow**	**42.58**	**43.88**	**43.69**	**42.56**
	Global-LiGO Only	22.98	22.50	23.17	23.16
	Ideal FedLiGO	40.42	36.62	37.07	33.90

E. Privacy Analysis

In this part, we assess the privacy-preserving capabilities of Fed-Grow. Specifically, we discuss how its architecture inherently reduces privacy risks by limiting the information shared and compare its resistance to Membership Inference Attacks (MIA) [28] against the full model sharing baseline.

Our method enhances privacy by sharing only the Global-LiGO component rather than the entire large model. In contrast, methods like FedAvg transmit gradients or parameters corresponding to the full model, which have been shown to be vulnerable to privacy attacks [24], [25], [55], [56]. To quantify this difference, we perform a standard MIA on the CIFAR-10, CIFAR-100, and Flowers-102 datasets.

Table IX shows the results of the attack. Fed-Grow consistently demonstrates a lower MIA success accuracy compared to FedAvg, highlighting its superior privacy protection. This is because the shared Global-LiGO is a more abstract representation of the updates and reveals less about the training data than the full model gradients. Additionally, our approach can be combined with techniques like differential privacy [57] and secure aggregation [58] for enhanced security.

TABLE IX: Membership Inference Attack (MIA) Accuracy for Fed-Grow and FedAvg (from Scratch) Across Various Datasets and Data Distributions

Task	Data Dist.	Method	10 Clients	20 Clients
CIFAR-10	IID	**Fed-Grow**	**58.42 ± 2.68**	**56.94 ± 0.67**
		FedAvg	66.50 ± 1.55	69.56 ± 0.38
	Non-IID	**Fed-Grow**	**55.65 ± 1.99**	**58.03 ± 1.45**
		FedAvg	71.58 ± 1.62	73.01 ± 1.49
CIFAR-100	IID	**Fed-Grow**	**67.49 ± 0.91**	**67.70 ± 0.50**
		FedAvg	71.06 ± 0.62	68.63 ± 0.64
	Non-IID	**Fed-Grow**	**68.68 ± 0.46**	**69.01 ± 0.61**
		FedAvg	74.40 ± 0.62	75.06 ± 0.72
Flowers-102	IID	**Fed-Grow**	**54.27 ± 3.33**	**55.61 ± 2.90**
		FedAvg	58.43 ± 3.85	62.16 ± 2.37
	Non-IID	**Fed-Grow**	**56.34 ± 2.43**	**58.78 ± 2.49**
		FedAvg	62.55 ± 2.27	60.49 ± 2.39

F. Summary

The experimental results demonstrate that Fed-Grow with Dual-LiGO achieves superior performance across key metrics. It consistently outperforms the NoAgg baseline in accuracy and precision (Table IV) while maintaining lower performance variability, indicating greater stability. Crucially, it also surpasses established federated learning baselines, including FedAvg, FedProx, and FedKD, when evaluated under an equivalent communication budget (Table V). The framework is robust to model heterogeneity (Appendix) and scales effectively with increasing numbers of clients under both IID and non-IID settings (Appendix). Our ablation studies further validate our design, confirming that both components of the Dual-LiGO architecture are indispensable for achieving high performance (Table VIII). Furthermore, Fed-Grow significantly reduces trainable parameters by 59.25% and communication costs by 73.01% compared to standard federated approaches (Table VI). Its privacy-preserving design, validated by lower Membership Inference Attack success rates (Table IX), ensures enhanced security by sharing only the Global-LiGO component. These results highlight Fed-Grow as a highly effective, efficient, and secure solution for federated learning in diverse and resource-constrained environments.

VI. Conclusion

To enable resource-constrained users to afford the high cost of training large-scale transformers, this paper studies a federated framework named Fed-Grow, in which multiple participants cooperatively expand a large-scale transformer from their pre-trained small models. To efficiently exchange the useful knowledge and protect the privacy of users in model scaling, the Dual-LiGO architecture is designed based on Fed-Grow, with its Local-LiGO part used to tackle the heterogeneity problem caused by the pre-trained models, and its Global-LiGO part shared by participants to exchange the implicit knowledge from the pre-trained models and the experience learned in model scaling. Compared with the classical federated learning, our approach no longer shares the models, let alone the raw data, which strengthens the privacy of our approach. The numerical results from the simulation show that our approach outperforms several baselines in terms of accuracy, precision, stability, and resource consumption on computations and communications.

A. Discussions on Limitations

We discuss two primary limitations regarding the current scope of Fed-Grow. First, its scalability to billion-parameter foundation models has not been validated; at such a scale, the LiGO operator itself may become too large for resource-constrained devices, necessitating future research into more efficient operator representations like low-rank factorization. Furthermore, our framework assumes a simplified, synchronous communication model and does not account for real-world network heterogeneity, where varying client bandwidths and latencies can introduce bottlenecks like the "bucket effect". Addressing these practical challenges through techniques like operator decomposition and adaptive aggregation is an important direction for future work but is outside the scope of this paper.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 62572280, U24A20244, and U23A20302.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[6] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[7] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[8] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," 2023.

[9] Z. Zhang, Y. Xia, H. Wang, D. Yang, C. Hu, X. Zhou, and D. Cheng, "Mpmoe: Memory efficient moe for pre-trained models with adaptive pipeline parallelism," *IEEE Trans. Parallel Distributed Syst.*, vol. 35, no. 6, pp. 843–856, 2024.

[10] B. Zhuang, J. Liu, Z. Pan, H. He, Y. Weng, and C. Shen, "A survey on efficient training of transformers," *arXiv preprint arXiv:2302.01107*, 2023.

[11] Z. Guo, Y. Tang, J. Zhai, T. Yuan, J. Jin, L. Wang, Y. Zhao, and R. Li, "A survey on performance modeling and prediction for distributed dnn training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 12, pp. 2463–2478, 2024.

[12] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[13] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large batch optimization for deep learning: Training bert in 76 minutes," *arXiv preprint arXiv:1904.00962*, 2019.

[14] M. Zhang and Y. He, "Accelerating training of transformer-based language models with progressive layer dropping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 011–14 023, 2020.

[15] L. Hou, R. Y. Pang, T. Zhou, Y. Wu, X. Song, X. Song, and D. Zhou, "Token dropping for efficient bert pretraining," *arXiv preprint arXiv:2203.13240*, 2022.

[16] L. Gong, D. He, Z. Li, T. Qin, L. Wang, and T. Liu, "Efficient training of bert by progressively stacking," in *International conference on machine learning*. PMLR, 2019, pp. 2337–2346.

[17] C. Yang, S. Wang, C. Yang, Y. Li, R. He, and J. Zhang, "Progressively stacking 2.0: A multi-stage layerwise training method for bert training speedup," *arXiv preprint arXiv:2011.13635*, 2020.

[18] C. Chen, Y. Yin, L. Shang, X. Jiang, Y. Qin, F. Wang, Z. Wang, X. Chen, Z. Liu, and Q. Liu, "bert2bert: Towards reusable pretrained language models," *arXiv preprint arXiv:2110.07143*, 2021.

[19] Y. Qin, Y. Lin, J. Yi, J. Zhang, X. Han, Z. Zhang, Y. Su, Z. Liu, P. Li, M. Sun *et al.*, "Knowledge inheritance for pre-trained language models," *arXiv preprint arXiv:2105.13880*, 2021.

[20] P. Wang, R. Panda, L. T. Hennigen, P. Greengard, L. Karlinsky, R. Feris, D. D. Cox, Z. Wang, and Y. Kim, "Learning to grow pretrained models for efficient transformer training," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=cDYRS5iZ16f

[21] Y. Wang, J. Su, H. Lu, C. Xie, T. Liu, J. Yuan, H. Lin, R. Sun, and H. Yang, "Lemon: Lossless model expansion," *arXiv preprint arXiv:2310.07999*, 2023.

[22] C. Fu, H. Huang, Z. Jiang, Y. Ni, L. Nai, G. Wu, L. Cheng, Y. Zhou, S. Li, A. Li, and J. Zhao, "Triple: Revisiting pretrained model reuse and progressive learning for efficient vision transformer scaling and searching," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 17 107–17 117.

[23] Y. Pan, Y. Yuan, Y. Yin, Z. Xu, L. Shang, X. Jiang, and Q. Liu, "Reusing pretrained models by multi-linear operators for efficient training," *Advances in Neural Information Processing Systems*, vol. 36, pp. 3248–3262, 2023.

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

[25] J. C. Zhao, S. Bagchi, S. Avestimehr, K. S. Chan, S. Chaterji, D. Dimitriadis, J. Li, N. Li, A. Nourian, and H. R. Roth, "Federated learning privacy: Attacks, defenses, applications, and policy landscape-a survey," *arXiv preprint arXiv:2405.03636*, 2024.

[26] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[27] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[28] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[29] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, "Chasing sparsity in vision transformers: An end-to-end exploration," 2021.

[30] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641*, 2015.

[31] X. Li, Y. Yao, X. Jiang, X. Fang, X. Meng, S. Fan, P. Han, J. Li, L. Du, B. Qin *et al.*, "Flm-101b: An open llm and how to train it with $100 k budget," *arXiv preprint arXiv:2309.03852*, 2023.

[32] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[33] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[34] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 8, 2022, pp. 8485–8493.

[35] N. Su, C. Hu, B. Li, and B. Li, "Titanic: Towards production federated learning with large language models," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 611–620.

[36] C. Wu, F. Wu, L. Lyu, Y. Huang, and X. Xie, "Communication-efficient federated learning via knowledge distillation," *Nature communications*, vol. 13, no. 1, p. 2032, 2022.

[37] T. Ashraf, F. B. A. Mir, and I. A. Gillani, "Transfed: A way to epitomize focal modulation using transformer-based federated learning," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 543–552.

[38] Q. Zhao, C. Qu, C. Chen, M. Fan, and Y. Wang, "Fedmcp: Parameter-efficient federated learning with model-contrastive personalization," in *2024 IEEE 30th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2024, pp. 246–253.

[39] C. Xie, D.-A. Huang, W. Chu, D. Xu, C. Xiao, B. Li, and A. Anandkumar, "Perada: Parameter-efficient federated learning personalization with generalization guarantees," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 838–23 848.

[40] W. Lu, X. Hu, J. Wang, and X. Xie, "Fedclip: Fast generalization and personalization for clip in federated learning," *arXiv preprint arXiv:2302.13485*, 2023.

[41] G. Long, T. Shen, J. Jiang, M. Blumenstein *et al.*, "Dual-personalizing adapter for federated foundation models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 39 409–39 433, 2024.

[42] K. Yin and J. Mao, "Personalized federated learning with adaptive feature aggregation and knowledge transfer," *arXiv preprint arXiv:2410.15073*, 2024.

[43] C. Mclaughlin and L. Su, "Personalized federated learning via feature distribution adaptation," *Advances in Neural Information Processing Systems*, vol. 37, pp. 77 038–77 059, 2024.

[44] Y. Zhou, G. Duan, T. Qiu, L. Zhang, L. Tian, X. Zheng, and Y. Zhu, "Personalized federated learning incorporating adaptive model pruning at the edge," *Electronics*, vol. 13, no. 9, p. 1738, 2024.

[45] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *2020 fourth world conference on smart trends in systems, security and sustainability (WorldS4)*. IEEE, 2020, pp. 794–797.

[46] S. Cao, H. Zhang, T. Wen, H. Zhao, Q. Zheng, W. Zhang, and D. Zheng, "Fedqmix: Communication-efficient federated learning via multi-agent reinforcement learning," *High-Confidence Computing*, vol. 4, no. 2, p. 100179, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667295223000776

[47] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 331–339.

[48] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.

[49] S. Tedeschi, V. Maiorca, N. Campolungo, F. Cecconi, and R. Navigli, "Wikineural: Combined neural and knowledge-based silver data creation for multilingual ner," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 2521–2533.

[50] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009, technical Report.

[51] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian conference on computer vision, graphics & image processing*. IEEE, 2008, pp. 722–729.

[52] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, H. L. Kwing, T. Parcollet, P. P. d. Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[53] B. Y. Lin, C. He, Z. Zeng, H. Wang, Y. Huang, C. Dupuy, R. Gupta, M. Soltanolkotabi, X. Ren, and S. Avestimehr, "Fednlp: Benchmarking federated learning methods for natural language processing tasks," *arXiv preprint arXiv:2104.08815*, 2021.

[54] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 965–978.

[55] K. Pillutla, K. Malik, A.-R. Mohamed, M. Rabbat, M. Sanjabi, and L. Xiao, "Federated learning with partial model personalization," PMLR, pp. 17 716–17 758, 2022.

[56] G. Lu, Z. Xiong, R. Li, N. Mohammad, Y. Li, and W. Li, "Defeat: A decentralized federated learning against gradient attacks," *High-Confidence Computing*, vol. 3, no. 3, p. 100128, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667295223000260

[57] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[58] L. Zhao, J. Jiang, B. Feng, Q. Wang, C. Shen, and Q. Li, "Sear: Secure and efficient aggregation for byzantine-robust federated learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3329–3342, 2021.

Shikun Shen received the B.E. degree from the Computer School, Wuhan University, in 2022. He is currently pursuing his PhD degree with the School of Computer Science and technology, Shandong University. His research interests include wireless networks, distributed computing, and edge intelligence.

Yifei Zou received the B.E. degree in 2016 from Computer School, Wuhan University, and the PhD degree in 2020 from the Department of Computer Science, The University of Hong Kong. He is currently an Assistant Professor with the school of computer science and technology, Shandong University. His research interests include wireless networks, ad hoc networks and distributed computing.

Yuan Yuan received the BSc degrees from the School of Mathematical Sciences, Shanxi University in 2016, and the Ph.D. degree from the School of Computer Science and Technology, Shandong University, Qingdao, China, in 2021. She is currently a postdoctoral fellow at the Shandong University-Nanyang Technological University International Joint Research Institute on Artificial Intelligence, Shandong University. Her research interests include distributed computing and distributed machine learning.

Hanlin Gu received his B.S. degree in Mathematics from University of Science and Technology of China in 2017. He received his Ph.D. degree in Mathematics from Hong Kong University of Science and Technology in 2022. He now works as a senior Researcher at WeBank AI Group, WeBank. His research interests include federated learning, privacy-preserving methodology. He has published a series paper in TPAMI, TDSC, ECCV, ECML, IJCAI, PAKDD and so on.

Peng Li received the PhD degree in computer science from The University of Aizu, Japan. He is currently a professor at Xi'an Jiaotong University, China. His research interests mainly focus on cloud/edge computing, wired/wireless networking, distributed AI systems, and blockchain. He has authored or co-authored over 100 papers in major conferences and journals. He won the 2020 Best Paper Award of *IEEE Transactions on Computers*. He serves as the chair of SIG on Green Computing and Data Processing in IEEE ComSoc Green Communications and Computing Technical Committee. He is a guest editor of *IEEE Journal of Selected Areas on Communications*, the editor of *IEEE Open Journal of the Computer Society* and *IEICE Transactions on Communications*. He is a senior member of IEEE.

Xiuzhen Cheng received her M.S. and Ph.D. degrees in computer science from the University of Minnesota – Twin Cities in 2000 and 2002, respectively. She is a professor in the School of Computer Science and Technology, Shandong University. Her current research interests include cyber physical systems, wireless and mobile computing, sensor networking, wireless and mobile security, and algorithm design and analysis. She has served on the editorial boards of several technical journals and the technical program committees of various professional conferences/workshops. She also has chaired several international conferences. She worked as a program director for the US National Science Foundation (NSF) from April to October in 2006 (full time), and from April 2008 to May 2010 (part time). She received the NSF CAREER Award in 2004. She is Fellow of IEEE and a member of ACM.

Falko Dressler received his M.Sc. and Ph.D. degrees from the Dept. of Computer Science, University of Erlangen in 1998 and 2003, respectively. He is a full professor and Chair for Data Communications and Networking at the School of Electrical Engineering and Computer Science, TU Berlin. Dr. Dressler has been associate editor-in-chief for IEEE Trans. on Mobile Computing and Elsevier Computer Communications as well as an editor for journals such as IEEE/ACM Trans. on Networking, IEEE Trans. on Network Science and Engineering, Elsevier Ad Hoc Networks, and Elsevier Nano Communication Networks. He has been chairing conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. Dr. Dressler is an IEEE Fellow as well as an ACM Distinguished Member. He is a member of the German National Academy of Science and Engineering (acatech). He has been serving on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. His research objectives include adaptive wireless networking (radio, visible light, molecular communications) and embedded system design (from microcontroller to Linux kernel) with applications in ad hoc and sensor networks, the Internet of Things, and cooperative autonomous driving systems.

Dongxiao Yu received the BSc degree in 2006 from the School of Mathematics, Shandong University and the PhD degree in 2014 from the Department of Computer Science, The University of Hong Kong. He became an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. He is currently a professor in the School of Computer Science and Technology, Shandong University. His research interests include wireless networks, distributed computing and graph algorithms.

Appendix

A. *Effects of Small Model Heterogeneity*

To assess the robustness of our method to varying degrees of small model heterogeneity, we designed two additional heterogeneous cases beyond the original three small models. Experiments were conducted on the CIFAR-100 and agnews dataset under IID and Non-IID conditions with 10-20 clients. The model heterogeneity cases are shown in Table X, Table XI and the final experimental results are presented in Table XII. From the table, we can see that regardless of the heterogeneity case, our method has a significant improvement over the NoAgg baseline, which indicates that Fed-Grow effectively handles model heterogeneity, maintaining superior performance even as the degree of heterogeneity varies.

TABLE X: Configurations of Heterogeneity Case 1

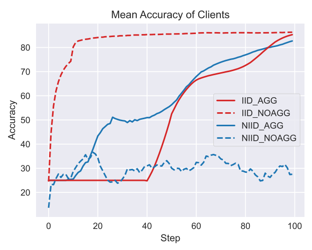
Name	#Hidden	#Layers	#Heads
small#1	256	2	8
small#2	256	3	8
small#3	256	4	8
small#4	192	2	8
small#5	192	3	8
small#6	192	4	8
intermediate	320	4	8
large	384	6	8

TABLE XI: Configurations of Heterogeneity Case 2

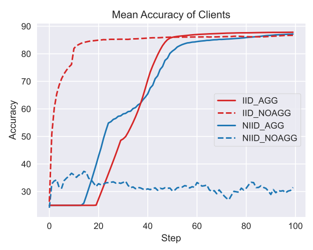
Name	#Hidden	#Layers	#Heads
small#1	256	2	8
small#2	256	3	8
small#3	256	4	8
small#4	256	5	8
small#5	256	6	8
intermediate	320	7	8
large	384	8	8

B. *Different Number of Clients*

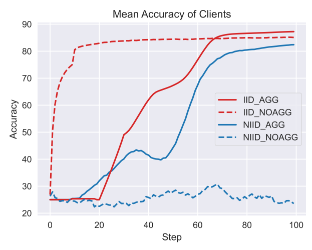
We conduct a robustness analysis to test the performance of Fed-Grow with Dual-LiGO under different client settings. We use two datasets, CIFAR-100 and agnews, and test Fed-Grow under both IID and non-IID data distributions. We vary the number of clients from 10 to 50. The results are shown in Table XIII and the accuracy curves are provided in Fig. 4 and 5, which show that Fed-Grow with Dual-LiGO is robust to different client settings, as it consistently outperforms baseline in both IID and Non-IID settings. In most cases, Fed-Grow has a clear advantage over baseline, especially in the Non-IID setting, where the gap is very large. This demonstrates that a client with Fed-Grow can effectively leverage the information from other clients to improve the performance of its transformer, while the baseline suffers from limited and heterogeneous data. The numerical results shows that Fed-Grow is robust to data heterogeneity and sparsity and can adapt to different client settings.



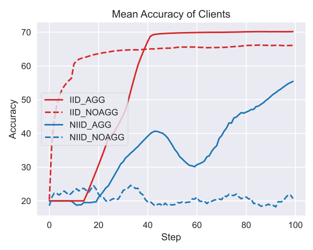
(a) 20 clients



(b) 30 clients

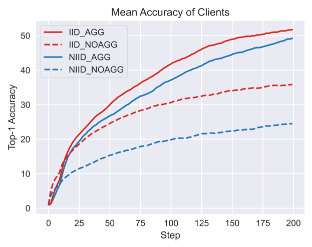


(c) 40 clients

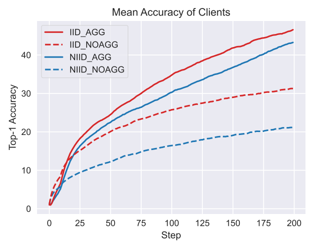


(d) 50 clients

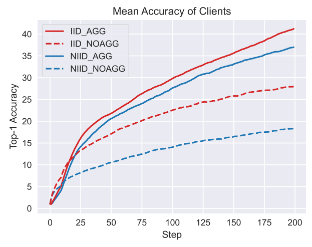
Fig. 4: Impact of client scalability on the agnews dataset. The curves compare the accuracy of Fed-Grow (Agg) and the isolated baseline (NoAgg) under IID and Non-IID settings as the number of clients increases from 20 to 50.



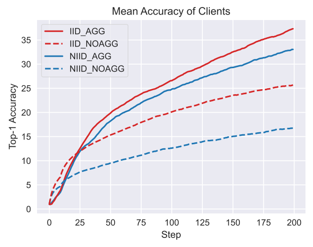
(a) 20 clients



(b) 30 clients



(c) 40 clients



(d) 50 clients

Fig. 5: Impact of client scalability on the CIFAR-100 dataset. The curves compare the accuracy of Fed-Grow (Agg) and the isolated baseline (NoAgg) under IID and Non-IID settings as the number of clients increases from 20 to 50.

C. *Experiment on Significantly Larger Global Model*

To further validate the effectiveness of Fed-Grow in scenarios with a significant scale gap between local and global models, we conducted an additional set of experiments. In this "Large-Scale Expansion" setting, we simulated a realistic environment where client devices hold small, edge-compatible

TABLE XII: Mean Accuracy of Fed-Grow (Agg) vs. NoAgg under Additional Heterogeneity Settings

Task	Heterogeneity Case	Accuracy (%) ↑											
		10 Clients						20 Clients					
		IID			Non-IID			IID			Non-IID		
		Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ
CIFAR-100	Case 1	**57.812**	38.306	19.506	**50.587**	28.622	21.965	**53.341**	36.585	16.756	**50.512**	25.670	24.842
	Case 2	**53.949**	47.568	6.381	**56.276**	35.449	20.827	**55.923**	43.394	12.529	**53.600**	32.784	20.816
agnews	Case 1	**90.224**	87.697	2.526	**81.961**	62.487	19.474	**87.145**	86.000	1.145	**69.066**	45.224	23.842
	Case 2	**89.342**	86.895	2.447	**79.789**	62.671	17.118	**86.132**	78.882	7.250	**77.408**	43.105	34.303

TABLE XIII: Accuracy (%) of Fed-Grow vs. NoAgg on CIFAR-100 and Agnews Datasets with a Varying Number of Clients

#Clients	CIFAR-100						Agnews					
	IID			Non-IID			IID			Non-IID		
	Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ
10	**53.18**	38.13	+15.05	**50.11**	26.90	+23.21	**90.16**	87.47	+2.69	**85.67**	73.08	+12.59
20	**54.00**	39.14	+14.86	**51.27**	28.46	+22.81	**86.74**	86.05	+0.69	**84.25**	49.30	+34.95
30	**47.57**	31.34	+16.23	**43.73**	21.36	+22.37	**88.00**	87.20	+0.80	**87.63**	50.26	+37.37
40	**42.06**	28.50	+13.56	**37.21**	18.48	+18.73	**87.33**	85.74	+1.59	**82.96**	38.58	+44.38
50	**37.48**	32.85	+4.63	**25.92**	17.12	+8.80	**70.23**	66.48	+3.75	**68.13**	34.14	+33.99

TABLE XIV: Comparison of Accuracy/Precision between Fed-Grow (Agg) and NoAgg under Large-Scale Expansion Settings (Global Model: 512 Hidden/10 Layers)

Model	Task	Accuracy/Precision (%) ↑											
		10 Clients						20 Clients					
		IID			Non-IID			IID			Non-IID		
		Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ
BERT	20news	**58.83**	32.01	26.82	**49.80**	23.37	26.43	**48.94**	25.27	23.67	**29.52**	8.78	20.74
	agnews	**88.82**	86.71	2.11	**80.16**	44.61	35.55	**87.89**	86.84	1.05	**70.03**	24.92	45.11
	wikiner	**42.67**	20.53	22.14	**41.03**	4.37	36.66	**43.97**	4.25	39.72	**23.09**	2.15	20.94
ViT	CIFAR-10	**78.42**	69.28	9.14	**71.10**	43.17	27.93	**77.35**	70.31	7.04	**65.57**	42.06	23.51
	CIFAR-100	**55.02**	40.92	14.10	**52.61**	28.07	24.54	**54.13**	37.78	16.35	**51.77**	27.41	24.36
	Flowers102	**41.69**	24.40	17.29	**41.74**	19.74	22.00	**34.22**	14.42	19.80	**35.94**	9.27	26.67
Model	Task	Standard Deviation ↓											
		10 Clients						20 Clients					
		IID			Non-IID			IID			Non-IID		
		Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ	Agg	NoAgg	Δ
BERT	20news	**2.81**	22.86	-20.05	**12.07**	20.70	-8.63	**2.19**	19.98	-17.79	**11.61**	15.21	-3.60
	agnews	**1.07**	2.96	-1.89	**2.68**	33.46	-30.78	**1.69**	2.21	-0.52	**9.67**	30.56	-20.89
	wikiner	**2.12**	8.05	-5.93	**2.42**	8.92	-6.50	**2.33**	6.64	-4.31	**2.52**	8.92	-6.40
ViT	CIFAR-10	**1.42**	6.80	-5.38	**3.13**	9.76	-6.63	**1.64**	4.46	-2.82	**2.51**	13.65	-11.14
	CIFAR-100	**1.61**	3.23	-1.62	**3.16**	3.36	-0.20	**1.34**	3.90	-2.56	**1.81**	4.46	-2.65
	Flowers102	**1.47**	4.20	-2.73	**1.73**	4.56	-2.83	**3.43**	4.45	-1.02	**4.28**	5.31	-1.03

models, while the collaborative goal is to grow a global model that is significantly larger.

Specifically, we utilized the same heterogeneous small models for clients as defined in Table II (2-4 layers with 256 hidden dimensions), but increased the target global model size to 10 layers with a hidden dimension of 512. This configuration results in a global model that is approximately 10× larger in terms of parameters compared to the local small models.

The results are presented in Table XIV. Consistent with our main results, Fed-Grow demonstrates robust performance even under this high scale disparity. Regarding accuracy, Fed-Grow significantly outperforms isolated training; for instance, on the CIFAR-100 dataset with non-IID data distribution (10 clients), Fed-Grow achieves 52.61% accuracy compared to the NoAgg baseline of 28.07%, an improvement of over 24 absolute points. Furthermore, regarding stability, Fed-Grow drastically reduces the performance variance among clients (lower standard deviation) compared to the volatile results of isolated training. These results confirm that our proposed framework effectively bridges the gap between resource-constrained edge devices and large-scale global models, ensuring efficient knowledge transfer even when the scale discrepancy is substantial.