

Scaling Synthetic-Image Pre-Training for Federated Fine-Tuning of Large Vision Models

Qianpiao Ma
Nanjing University of Science and
Technology
Nanjing, China
maqianpiao@njust.edu.cn

Xiaozhu Song
Nanjing University of Science and
Technology
Nanjing, China
songxiaozhu@njust.edu.cn

Junlong Zhou^{*}
Nanjing University of Science and
Technology
Nanjing, China
jlzhou@njust.edu.cn

Yue Zeng
Nanjing University of Science and
Technology
Nanjing, China
zengyue@njust.edu.cn

Jianchun Liu
University of Science and Technology
of China
Hefei, China
jcliu17@ustc.edu.cn

Huaqing Tu
Zhejiang Gongshang University
Hangzhou, China
thq@mail.zjgsu.edu.cn

Abstract

Federated fine-tuning (FedFT) enables adapting pre-trained large vision models (LVMs) on distributed, privacy-sensitive devices, while its practical deployment is hindered by three critical challenges: resource constraints, system heterogeneity, and non-IID data. While prior studies partially address these issues, *e.g.*, by pre-training initial models on synthetic images to mitigate the adverse effects of non-IID data, or leveraging parameter-efficient fine-tuning (PEFT) methods like low-rank adaptation (LoRA) to reduce resource consumption, they remain inadequate and fragmented. Specifically, existing synthetic image generation methods fail to capture device-specific feature distributions, while current PEFT-based FedFT methods often undervalue weaker devices that may provide critical information. More importantly, the separate optimization of pre-training and FedFT neglects their inherent connection, lacking a holistic perspective to maximize training efficiency. To overcome these limitations, we propose FeDiSyn, a unified framework that holistically considers the interplay between pre-training and FedFT to minimize the overall LVM training time. Specifically, FeDiSyn introduces: (i) a scaling law for FedFT pre-training to determine the optimal number of synthetic images, balancing pre-training benefit against generation/pre-training cost, (ii) diffusion-based synthetic image generation that captures device-specific feature distributions for pre-training to tackle non-IID data, and (iii) a contribution-aware LoRA configuration and bandwidth allocation algorithm for FedFT to ensure that informative devices are effectively utilized while addressing system heterogeneity. Experimental results on the real-world testbed demonstrate that FeDiSyn reduces completion time by over 52.5% and communication cost by over 97.2%, while achieving comparable accuracy to state-of-the-art solutions.

^{*}Corresponding author.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

ICPP '26, Singapore, Singapore

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2657-6/26/09

<https://doi.org/10.1145/3832810.3832835>

CCS Concepts

• Computing methodologies → Distributed artificial intelligence.

Keywords

Federated fine-tuning, Synthetic image, Pre-training, Scaling law, Large vision models

ACM Reference Format:

Qianpiao Ma, Xiaozhu Song, Junlong Zhou, Yue Zeng, Jianchun Liu, and Huaqing Tu. 2026. Scaling Synthetic-Image Pre-Training for Federated Fine-Tuning of Large Vision Models. In *Proceedings of the 55th International Conference on Parallel Processing (ICPP '26)*, September 28–October 01, 2026, Singapore, Singapore. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3832810.3832835>

1 Introduction

Since the landmark success of AlexNet in 2012 [16], deep learning powered by large-scale labeled datasets has revolutionized the field of image classification. Initially, most vision models are trained via supervised learning on relatively small datasets. More recently, however, a two-stage paradigm has emerged for training large vision models (LVM): first pre-training on massive datasets, then fine-tuning on downstream task-specific data. In particular, pure transformer-based architectures such as the vision transformer (ViT) [5] have demonstrated more outstanding performance than the convolutional networks when pre-trained at scale.

In parallel, the Internet of Things (IoT) [51][49] has grown explosively, generating massive amounts of data from edge devices that offers a rich source for training LVMs. However, centralized training on large-scale datasets is increasingly limited by privacy regulations (*e.g.*, GDPR) and by bandwidth bottlenecks when accessing local data. This motivates the adoption of federated fine-tuning (FedFT) [48], a technique that integrates model fine-tuning into the federated learning (FL) framework [26]. In a typical FedFT setting, a parameter server first broadcasts a pre-trained global model to a set of distributed devices. Each device fine-tunes the model on its local data and sends the updated parameters back to the server for aggregation. This cycle of local fine-tuning and global aggregation repeats until the model converges or reaches a target accuracy.

Table 1: Comparison of FedFT methods for LVMs.

Method	Synthetic Image for Pre-training	PEFT Method	Handling Resource Constraints	Handling System Heterogeneity	Handling Label Distribution Skew	Handling Feature Distribution Skew
PerAda [41]	No	Adapter	✓	×	×	×
pFedSeq [29]	No	Adapter	✓	×	×	×
CaFPT [8]	No	Prompt	✓	×	×	×
FedPrefix [36]	No	Prompt	✓	×	×	×
SFLF [40]	No	LoRA	✓	×	×	×
FedRA [35]	No	LoRA	✓	✓	×	×
CAFF[30]	No	LoRA	✓	✓	×	×
FPS [3]	Yes	-	×	×	✓	×
GPT-FL [47]	Yes	-	×	×	✓	×
FGL [45]	Yes	-	×	×	✓	×
FeDiSyn (Ours)	Yes	LoRA	✓	✓	✓	✓

Challenges of FedFT for LVMs. Although FedFT offers a promising solution for fine-tuning LVMs leveraging distributed data, it still faces several practical challenges: (1) *Resource Constraints*. Modern LVMs like ViT and SwinT [21], usually contain billions of parameters, requiring significant computation/communication resources for fine-tuning/transmission. For example, ViT-L/16 requires roughly 3.1GB of traffic per communication and over 2,900 TFLOPs of computation. In contrast, device resources are typically limited [39]. For example, devices like the NVIDIA Jetson TX2 offer less than 2TFLOPs of computation capacity, while network bandwidth often falls below 100Mbps, rendering full-parameter FedFT infeasible on devices. (2) *System Heterogeneity*. Devices differ widely in hardware capabilities and network conditions. For example, the computational power of an iPhone 16 amounts to only about 12% of that of an RTX 3090 desktop GPU [18], while network condition varies significantly, ranging from 4G (~100Mbps), 5G (~500Mbps) to fiber connections (~1Gbps). As a result, the time required for fine-tuning and transmitting parameters can differ substantially across devices. Faster devices may idle while waiting for slower ones, leading to a *straggler problem* [23] that reduces overall FedFT efficiency. (3) *Non-IID Data*. Local data on devices are often non-independent and non-identically distributed [22], which involves two aspects. First, the *label distribution* varies across devices, e.g., some classes may be overrepresented or missing entirely on certain devices. Second, even for the same label, the *feature distributions* may differ due to variations in style or domain, e.g., cartoon vs. natural images [42].

Status Quo and Limitations. To address resource constraints in FedFT, some studies [8, 29, 30, 35, 36, 40, 41] have deployed the parameter-efficient fine-tuning (PEFT) methods (e.g., Adapter tuning, LoRA and Prompt tuning). PEFT inserts lightweight trainable modules into the LVM while keeping the pre-trained model frozen. Since these modules typically account for less than 5% of the pre-trained model’s parameters, PEFT significantly reduces the computation and communication overhead. However, most of these works [8, 29, 36, 40, 41] simply transplant centralized PEFT methods into distributed settings without accounting for system heterogeneity across devices. A few studies [30, 35] attempt to mitigate heterogeneity by assigning varying LoRA depths based on devices’ capabilities. This strategy, however, overlooks the potential importance of weaker devices in FedFT [2]: assigning them smaller LoRA depths can significantly degrade fine-tuning performance.

Moreover, these FedFT approaches are typically initialized with off-the-shelf pre-trained models without synthetic data augmentation, indicating that convergence performance could be further improved [15].

Recent theoretical work [12] has shown that a carefully initialized pre-trained global model can mitigate the adverse effects of non-IID data and accelerate convergence in FedFT. However, models pre-trained on public datasets (e.g., ImageNet [4]) may not reflect the device-specific data distributions [3], making them suboptimal as initialization points for FedFT. Several studies [45, 47] have introduced diffusion models (DM) [34] to generate synthetic images that approximate the label distribution on each device. These synthetic images are then used to pre-train a global model, serving as the initialization for FedFT. However, these approaches rely solely on text-conditioned synthesis (e.g., labels or text embeddings) and may fail to capture the unique feature distribution of each device [50], leading to a style gap between synthetic and real data that diminishes the effectiveness of augmentation.

Overall, none of these approaches adequately addresses the challenges outlined above, summarized in Table 1. More importantly, these approaches treat synthetic pre-training and FedFT as *isolated* stages, and ignore their synergy: insufficient images weaken pre-training effectiveness for FedFT, whereas excessive images introduce substantial computation overhead, thereby prolonging generation and pre-training time. Therefore, *the key challenge lies in determining the optimal number of generated images, so as to balance pre-training benefit for FedFT against generation/pre-training cost*.

Overview of the Proposed Approach. To this end, we propose FeDiSyn, a unified framework that holistically considers pre-training and FedFT to minimize the overall completion time for LVM training, which is built upon “one law, two optimizations”. *First*, we introduce a scaling law for synthetic pre-training under FedFT setting, which captures the relationship among accuracy, total training time, and the synthetic data scaling factor, providing quantitative guidance for determining the optimal number of generated images. *Second*, for pre-training optimization, each device extracts features by encoding local images into latents, which are then uploaded to the server. By combining these latents with semantically enhanced labels via a DM, the server generates synthetic images to pre-train a global model. Since the synthetic images capture cross-device label and feature distributions, the pre-trained model yields a superior initialization for FedFT, effectively addressing both label and feature

distribution skews. *Third*, for FedFT optimization, FeDiSyn introduces a contribution-aware LoRA configuration and bandwidth allocation algorithm. By configuring potentially important devices sufficient LoRA depth and balancing completion times via bandwidth allocation, the FedFT performance can be further improved while addressing system heterogeneity.

Our main contributions are summarized as follows.

- We propose FeDiSyn, a unified framework that holistically considers the close interplay between pre-training and FedFT to reduce the overall LVM training time. To our knowledge, this is the first work to explore the scaling law for synthetic pre-training under FedFT setting, and systematically optimize these two stages in a unified manner.
- We develop a synthetic pre-training method that tackles non-IID data from both label and feature distribution perspectives. In addition, we design a contribution-aware LoRA configuration and bandwidth allocation algorithm to improve FedFT performance while addressing system heterogeneity.
- We implement the proposed FeDiSyn framework and related algorithms on a real-world testbed. Experimental results on various models and datasets show that the proposed solution outperforms the state-of-the-art solutions.

2 Background and Motivation

2.1 Pre-Training Model Initialization for FedFT

The Importance of Pre-Training and Initialization. As a central challenge in FedFT, non-IID data is typically addressed through various aspects, such as device sampling [2, 24] and aggregation mechanism [23, 25], while the impact of model initialization is often overlooked in most studies. Several recent works have highlighted the critical importance of pre-training for the initial model in FedFT. Nguyen *et al.* [27] observe that models initialized with pre-trained weights typically exhibit significantly lower training loss values at initialization compared to those initialized randomly. Karimireddy *et al.* [15] further demonstrate that this lower initial loss resulting from pre-training facilitates faster convergence during subsequent FedFT. Chen *et al.* [3] note that pre-training shapes a better conditioned loss surface and steers the model into a location closer to the optimal solution, enhancing the stability of model aggregation. These phenomena are theoretically supported by [12], which proves that pre-training effectively reduces sensitivity to certain hyperparameters within FedFT by diminishing misaligned filters at initialization.

Synthetic Image for Pre-Training LVMs. Although pre-training initialization can enhance FedFT performance, models pre-trained on public datasets (e.g., ImageNet) show limited gains due to domain mismatch between the public data and the device-specific data. For example, in chest X-ray analysis, an ImageNet pre-trained model achieves only 0.2% accuracy improvement over random initialization, whereas a domain-relevant synthetic image pre-trained model yields a 5.9% gain [3]. To address this, GPT-FL [47] and FGL [45] upload labels or text descriptions of device datasets to the server for synthetic image generation. However, these approaches rely solely on text-conditioned generation, which fails to capture device-specific feature distributions. As a result, the

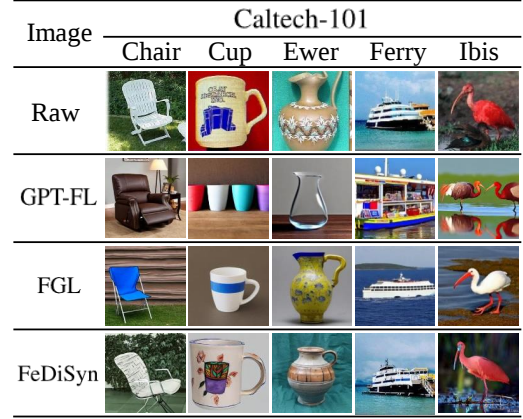


Fig. 1: Comparison of FeDiSyn and other synthetic image generation methods on Caltech-101.

synthetic images may not align well with real local data, limiting their improvement for FedFT.

In contrast, FeDiSyn uploads latents encoding local data features to generate synthetic images, as elaborated in Section 3. For example, generated images for Caltech-101 [7] in Fig. 1 show that FeDiSyn can obtain synthetic images that better align with the feature distributions of raw data compared to GPT-FL and FGL.

2.2 Scaling Law of Synthetic-Image Pre-Training for FedFT

Following the groundbreaking success of large-scale models, scaling laws have emerged as a critical framework for quantifying the power-law relationships between model/data scale and performance [14]. Prior studies have demonstrated that pre-training data volume has a significant impact on model performance [28, 44], which has motivated recent extensions of these principles to synthetic data, exploring how generative data scale influences convergence [6, 13]. However, these investigations focus exclusively on centralized training paradigms and are therefore not directly applicable to the FedFT setting, where the capabilities of servers for pre-training and edge devices for fine-tuning jointly determine the overall training time. As illustrated in Fig. 2 on Caltech-101, the synthetic scaling factor λ introduces a trade-off in FedFT: increasing λ accelerates convergence but incurs substantial overhead in image generation and pre-training. Therefore, the total completion time follows a U-shaped curve, which, in this example, attains its minimum near $\lambda = 0.5$.

To provide guidance for synthetic pre-training in the FedFT setting, we propose a novel scaling law that quantifies the relationship between synthetic volume, target accuracy, and total training time, enabling the analytical determination of the optimal scaling factor λ , as elaborated in Section 5.

2.3 Federated Fine-Tuning with LoRA for LVMs

LoRA [11] fine-tuning is a specific PEFT approach via low-rank matrix decomposition. In LoRA-based FedFT, each device only fine-tunes and uploads the lightweight inserted LoRA layers while keeping the pre-trained model frozen. The server aggregates the uploaded LoRA parameters and then distributes them back to all

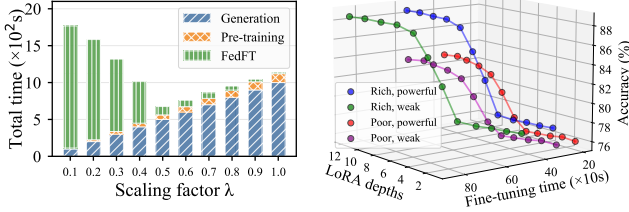


Fig. 2: The effect of different λ on total time to reach 70% accuracy on Caltech-101. **Fig. 3: The tradeoff between heterogeneity and performance.**

devices. Most existing LoRA-based FedFT approaches for LVMs [35, 40] insert LoRA layers into all transformer layers, causing significant computation overhead. To reduce this burden, a few studies [19, 30] insert LoRA only to a subset of layers, enabling more efficient fine-tuning on resource-constrained devices. Furthermore, these works typically address system heterogeneity by assigning LoRA depths based solely on device capabilities: powerful devices receive deeper LoRA layers, while resource-constrained devices get smaller ones. However, weak devices may play a critical role in FedFT, and assigning them shallow LoRA depths can significantly degrade performance. We conduct a case study with ViT-B/16 fine-tuned on CIFAR-10 [17] in Fig. 3, which compares accuracy and fine-tuning time across four device types: (i) rich-data/powerful-capability, (ii) rich-data/weak-capability, (iii) poor-data/powerful-capability, and (iv) poor-data/weak-capability. As shown, powerful but data-scarce devices (red line) gain little benefit from deeper LoRA depths, whereas data-rich devices with weak-capability (green line) achieve accuracy comparable to powerful ones (blue line) when given a deeper LoRA, albeit with longer fine-tuning time. These observations highlight a trade-off: It is worth setting a larger LoRA depth for resource-constrained yet critical devices, but these devices may become stragglers that delay the overall FedFT process.

To this end, we propose a contribution-aware LoRA configuration and bandwidth allocation strategy in FeDiSyn. LoRA depth for each device is determined by its expected performance gain, while bandwidth is allocated to balance device completion times, effectively addressing system heterogeneity. Details are elaborated in Section 5.

3 System Design of FeDiSyn

3.1 Preliminaries

We consider a K -class classification problem performed through FedFT with label space $C = \{c_1, c_2, \dots, c_K\}$. The system contains a centralized server with relatively abundant computational resources and a set of devices $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$. Each device v_i owns a local dataset $\mathcal{D}_i$, with size $D_i = |\mathcal{D}_i|$. The global dataset across all devices is represented as $\mathcal{D}$, with size $D = |\mathcal{D}| = \sum_{v_i \in \mathcal{V}} D_i$. The data labeled as c_k on v_i is denoted as $\mathcal{D}_i^k$ with size $D_i^k = |\mathcal{D}_i^k|$, and it holds that $\sum_{c_k \in C} D_i^k = D_i$. The loss function of v_i is defined as

$$f_i(\mathbf{w}) \triangleq \frac{1}{D_i} \sum_{\xi \in \mathcal{D}_i} f_i(\mathbf{w}; \xi), \quad (1)$$

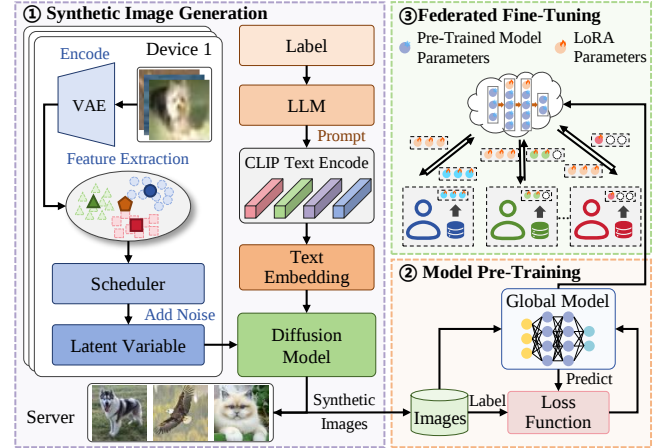


Fig. 4: Overview of FeDiSyn.

where $\mathbf{w}$ is the parameter vector, and $f_i(\mathbf{w}; \xi)$ is the loss over sample ξ in $\mathcal{D}_i$. The global loss function on all the distributed datasets is

$$F(\mathbf{w}) \triangleq \sum_{v_i \in \mathcal{V}} \frac{D_i}{D} f_i(\mathbf{w}). \quad (2)$$

The learning problem is to find the optimal parameter vector $\mathbf{w}^*$ so as to minimize $F(\mathbf{w})$, i.e., $\mathbf{w}^* = \arg \min_{\mathbf{w}} F(\mathbf{w})$.

3.2 System Overview

As shown in Fig. 4, the process of FeDiSyn consists of three steps.

① Synthetic image generation: Devices encode private images into low-dimensional latents using a lightweight variational autoencoder (VAE). These latents are clustered to obtain class-specific centroids and a global average representation, which are then sent to the server with their labels. The server uses a large language model (LLM) to expand labels into diverse text prompts, encodes them via CLIP's [31] text encoder, and feeds the resulting embeddings along with the uploaded latents into a DM (e.g., Stable Diffusion) to generate synthetic images, where latents govern stylistic features (e.g., lighting, texture) and text embeddings condition semantics (e.g., class-specific details). **② Model pre-training:** The server pre-trains a global model over synthetic images, yielding an initial model that captures cross-device features for subsequent FedFT. **③ Federated fine-tuning:** The initialized model is sent to devices for LoRA-based fine-tuning under the FedFT framework. Specifically, each device keeps the pre-trained model frozen and updates only LoRA parameters, which are aggregated by the server and redistributed until convergence. The following sections detail each step.

3.3 Synthetic Image Generation

Data Feature Extraction. To align synthetic images with device-specific styles while preserving privacy, each device v_i extracts the features of its raw images locally in the latent space. Specifically, each sample $\mathcal{D}_{i,j}^k \in \mathcal{D}_i^k$ of class c_k on device v_i is encoded by a VAE $\mathcal{E}$ to obtain latents $e_{i,j}^k = \mathcal{E}(\mathcal{D}_{i,j}^k)$. Subsequently, for the same class c_k , device v_i clusters the latents $e_{i,j}^k, \forall j$ into Z cluster centroids $e_{i,z}^k, z \in [1, Z]$, and calculate their average representation as $\bar{e}_i^k$. $\bar{e}_i^k$ and $e_{i,z}^k$ capture the style characteristics of the device distribution

[43]. Then the device uses a scheduler to add noise to the encoded latents at a randomly chosen diffusion timestep $s \in \{0, \dots, S\}$ [32] as

$$\begin{aligned}\tilde{\mathbf{e}}_{i,z}^{k,s} &= \sqrt{\alpha_s} \mathbf{e}_{i,z}^k + \sqrt{1 - \alpha_s} \epsilon, \epsilon \sim \mathcal{N}(0, I), z \in [1, Z], \\ \tilde{\mathbf{e}}_i^{k,s} &= \sqrt{\alpha_s} \mathbf{e}_i^k + \sqrt{1 - \alpha_s} \epsilon, \epsilon \sim \mathcal{N}(0, I),\end{aligned}\quad (3)$$

where α_s controls noise intensity at timestep s . The device retains the raw data locally and only uploads the cluster centroids $\tilde{\mathbf{e}}_{i,z}^{k,S}$, the average representation $\tilde{\mathbf{e}}_i^{k,S}$, and its class label c_k to the server.

Label Collection and Prompt Engineering. The server collects the data volume D_i^k labeled as c_k of each device v_i and calculates the total amount of data labeled as c_k across all devices by $D^k = \sum_{v_i \in \mathcal{V}} D_i^k$. To enrich semantic variety, a deployed LLM expands each label into multiple descriptive prompts. It has been verified that language enhancement is an effective way for enlarging synthetic image diversity for pre-training [9]. For example, for the label “dog”, the prompt can be semantically expanded as follows:

Q: Write a sentence using the word “dog” to describe the characteristics and actions of the object.

A: A golden retriever playing on the grass.

These prompts are encoded into text embeddings via CLIP’s text encoder, ensuring semantic alignment between text and image latent spaces. To preserve the real data distribution, the synthetic image size for class c_k is set as $\hat{D}^k = \lambda D^k$. λ is the scaling factor, which is determined according to our proposed scaling law in Section 5.

Synthetic Image Generated by Diffusion Model. On the server, for each class c_k , the received cluster centroids $\tilde{\mathbf{e}}_{i,z}^{k,S}$ and average representation $\tilde{\mathbf{e}}_i^{k,S}$ are weighted as $\tilde{E}_{i,z}^{k,S} = \rho \tilde{\mathbf{e}}_{i,z}^{k,S} + (1 - \rho) \tilde{\mathbf{e}}_i^{k,S}$, where ρ is a factor with a value between 0 and 1. Next, each latent $\tilde{E}_{i,z}^{k,S}$ is paired with a text embedding and fed into a latent DM, where latents guide stylistic consistency and text embeddings enforce data diversity. At each reverse step, a PNDM scheduler [20] is employed for sampling, iteratively denoising $\tilde{E}_{i,z}^{k,S}$ over S steps via

$$\phi(\tilde{E}_{i,z}^{k,S}, \epsilon_s, s, s-\delta) = \frac{\sqrt{\alpha_{s-\delta}}}{\sqrt{\alpha_s}} \tilde{E}_{i,z}^{k,S} - \frac{(\alpha_{s-\delta} - \alpha_s)}{\sqrt{\alpha_s} (\sqrt{(1-\alpha_{s-\delta})\alpha_s} + \sqrt{(1-\alpha_s)\alpha_{s-\delta}})} \epsilon_s, \quad (4)$$

where δ is the step interval, and the gradient term ϵ_s is computed using numerical methods such as the Runge-Kutta method or the linear multi-step method. Ultimately, all $\tilde{E}_{i,z}^{k,S}$, $\forall i, k$ undergo a certain number of denoising process to produce synthetic images $\hat{\mathcal{D}}$ with size $\hat{D} = \sum_{c_k \in \mathcal{C}} \hat{D}^k = \lambda \sum_{c_k \in \mathcal{C}} D^k$ at server side.

Remark 1 (Privacy Analysis). *Transmitting latent representations is a widely used strategy for data augmentation in FL. Considering the limited amount of latent information uploaded by each device, our approach introduces minimal privacy leakage. Furthermore, similar to [43], we generate images using DM from noise-added features. As a result, even when the original images contain sensitive content (e.g., text or faces), the generated images exhibit entirely different semantic details. Therefore, despite potential visual similarities, they do not reveal or compromise the underlying private information.*

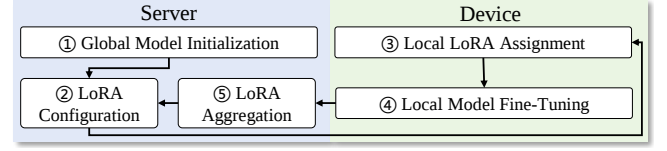


Fig. 5: Federated fine-tuning with LoRA.

3.4 Model Pre-Training over Synthetic Image

With the generated synthetic images $\hat{\mathcal{D}}$ on the server, a global model is pre-trained in a standard supervised fashion. Similar to Eq. (2), we denote $\hat{F}(\mathbf{w})$ as the loss function over the synthetic images $\hat{\mathcal{D}}$ for model parameters $\mathbf{w}$. Starting from an initial parameter vector $\hat{\mathbf{w}}^0$, the server performs $\hat{T}$ iterations of model pre-training by

$$\hat{\mathbf{w}}^t = \hat{\mathbf{w}}^{t-1} - \eta \nabla \hat{F}(\hat{\mathbf{w}}^{t-1}), \quad (5)$$

where η is the learning rate, and ∇ is the gradient operator. The global model after $\hat{T}$ iterations of pre-training is denoted by $\bar{\mathbf{w}} = \hat{\mathbf{w}}^{\hat{T}}$, which acts as the initial model for the subsequent FedFT process.

3.5 Federated Fine-Tuning with LoRA

After the pre-training model $\bar{\mathbf{w}}$ is obtained, FeDiSyn performs FedFT with LoRA for the LVM through the five phases shown in Fig. 5.

① Global Model Initialization. The server first injects the LoRA layers into the pre-trained model $\bar{\mathbf{w}}$, i.e., embedding initial LoRA matrices $\tilde{\mathbf{w}}^0 = \{\tilde{\mathbf{w}}_l^0, \forall l \in [0, L-1]\}$ within L transformer layers of the neural network. Then the embedded pre-trained model $\mathbf{w}^0 = \{\bar{\mathbf{w}}, \tilde{\mathbf{w}}^0\}$ is distributed to all devices in $\mathcal{V}$.

② LoRA Configuration. Prior studies [19, 46] show that inserting LoRA into deeper layers yields better fine-tuning performance than shallow layers. Therefore, we define LoRA depth as the number of layers with LoRA modules, counted from the deepest output layer upward. At each round t , the server assigns each device v_i a depth x_i^t based on its heterogeneous resources, thus only the last x_i^t layers $L_i^t = \{l | L - x_i^t \leq l < L\}$ are fine-tuned. The LoRA configuration strategy for all devices at round t is denoted as $\mathcal{X}^t = \{x_i^t | v_i \in \mathcal{V}\}$, which will be described in Section 5.

③ Local LoRA Assignment. At each round t , the server distributes the configuration information $\mathcal{X}^t$ and the global LoRA parameters $\tilde{\mathbf{w}}^t = \{\tilde{\mathbf{w}}_l^t | \forall l \in [0, L-1]\}$ to all devices in $\mathcal{V}$. The l -th LoRA layer of device v_i ’s local model is assigned as

$$\tilde{\mathbf{w}}_{i,l}^t = \tilde{\mathbf{w}}_l^t. \quad (6)$$

④ Local Model Fine-Tuning. After the local LoRA configuration, v_i fine-tunes the local model $\mathbf{w}_i^t = \{\bar{\mathbf{w}}, \tilde{\mathbf{w}}_i^t\}$ on its local dataset $\mathcal{D}_i$. The update of each LoRA layer $\tilde{\mathbf{w}}_{i,l}^t$ at round t is expressed as

$$\tilde{\mathbf{w}}_{i,l}^{t+1} = \begin{cases} \tilde{\mathbf{w}}_{i,l}^t - \eta \nabla f_l(\tilde{\mathbf{w}}_{i,l}^t), & L - x_i^t \leq l < L \\ \tilde{\mathbf{w}}_{i,l}^t, & \text{otherwise} \end{cases}. \quad (7)$$

where η is the learning rate and ∇ is the gradient operator.

⑤ LoRA Aggregation. After local fine-tuning, each device v_i uploads the fine-tuned LoRA parameters $\tilde{\mathbf{w}}_{i,l}^{t+1}$, $L - x_i^t \leq l < L$ to the server. Upon receiving the updated LoRA parameters from all devices, for each layer l , the server aggregates them as

$$\tilde{\mathbf{w}}_l^{t+1} = \frac{\sum_{v_i \in \mathcal{V}} [l \geq L - x_i^t] D_i \tilde{\mathbf{w}}_{i,l}^{t+1}}{\sum_{v_i \in \mathcal{V}} [l \geq L - x_i^t] D_i}, \quad (8)$$

where $[\cdot]$ is the Iverson notation. As a result, the global LoRA parameters are updated as $\tilde{\mathbf{w}}^{t+1} = \{\tilde{\mathbf{w}}_l^{t+1} | l \in [0, L-1]\}$, and the server configures LoRA for the next round $t+1$. The process repeats until the model converges or reaches satisfactory accuracy.

4 Convergence Analysis

We make the following assumptions on the loss functions $F(\mathbf{w})$ for convergence analysis, which are widely used in prior works [39].

Assumption 1 (Lipschitz). *The loss function F has a $\mathcal{L}$ -Lipschitzian gradient for $\forall \mathbf{w}_1, \mathbf{w}_2$, i.e., $\|\nabla F(\mathbf{w}_1) - \nabla F(\mathbf{w}_2)\| \leq \mathcal{L}\|\mathbf{w}_1 - \mathbf{w}_2\|$.*

Assumption 2 (Gradient Bounded). *The expected squared norm of gradients is uniformly bounded for $\forall \mathbf{w}_i^t$, i.e., $\mathbb{E}\|\nabla F(\mathbf{w}_i^t)\|^2 \leq G^2$.*

For LoRA fine-tuning, each LoRA matrix is further decomposed as $\tilde{\mathbf{w}}_i^t = \mathcal{B}_i^t \mathcal{A}_i^t, \forall i, t$, where $\mathcal{B}_i^t$ and $\mathcal{A}_i^t$ are low-rank matrices.

Assumption 3 (LoRA Bounded). *There exist constants $C_{\mathcal{A}}, C_{\mathcal{B}} > 0$, such that $\|\mathcal{A}_i^t\|_2 \leq C_{\mathcal{A}}, \|\mathcal{B}_i^t\|_2 \leq C_{\mathcal{B}}$.*

Based on the above assumptions, we derive the convergence bound of FeDiSyn by the following theorem.

Theorem 1 (Convergence of FeDiSyn). *After T rounds of fine-tuning in FeDiSyn, the expectation of the gradient of the LoRA matrices satisfies*

$$\begin{aligned} & \frac{1}{TN} \sum_{t=1}^T \sum_{v_i \in \mathcal{V}} \left(\mathbb{E}\|\nabla F(\mathcal{A}_i^t)\|^2 + \mathbb{E}\|\nabla F(\mathcal{B}_i^t)\|^2 \right) \\ & \leq \frac{1}{N} \sum_{v_i \in \mathcal{V}} \frac{\mathbb{E}[F(\mathbf{w}^0)] - \mathbb{E}[F(\mathbf{w}_i^*)]}{\eta T} + \kappa, \end{aligned} \quad (9)$$

where $\mathbf{w}_i^*$ denotes the optimal model on v_i , κ is a constant satisfying $[G^3 C_{\mathcal{A}} C_{\mathcal{B}} + \frac{3\mathcal{L}}{2}(G^2 C_{\mathcal{A}}^2 + G^2 C_{\mathcal{B}}^2 + \eta^2 G^4 C_{\mathcal{A}}^2 C_{\mathcal{B}}^2)]\eta + \frac{G^2}{2} \leq \kappa$.

Due to limited space, we omit the proof of Theorem 1 here.

Remark 2 (LoRA-based FedFT Convergence). *It is noticed that, unlike prior convergence analyses for FedFT [19, 40], which typically study the convergence behavior only at the level of the overall model parameter $\mathbf{w}$, our analysis is developed specifically for LoRA-based FedFT and explicitly characterizes the convergence of the two low-rank matrices $\mathcal{A}$ and $\mathcal{B}$. As a result, our convergence bound provides a fine-grained theoretical understanding of the training in LoRA-based FedFT, and better reveals how the low-rank decomposition influences convergence.*

5 Problem Formulation and Algorithm Description

5.1 Problem Formulation

The total completion time of FeDiSyn includes the image generation, pre-training, and FedFT time. Let $\hat{h}$ denote the time to generate one image on server, then the total image generation time is expressed as $H^{\text{gene}} = \hat{D}\hat{h} = \lambda D\hat{h}$. Next, the server pre-trains LVM on synthetic data for $\hat{T}$ rounds, and the pre-training time can be expressed as $H^{\text{pre}} = \lambda D\hat{h}\hat{T}$, where $\hat{h}$ is the single round training time for one image on server.

For FedFT, although deeper LoRA layers improve its performance, practical deployments are constrained by resource limitations such as computation, communication, and memory. Let $\bar{h}_i$ denote the

time for data loading and performing a full forward pass through the LVM on v_i per sample, and h_i denote the additional time for backpropagating through per LoRA layer. Since the computation capacity of v_i is fixed, both $\bar{h}_i$ and h_i can be measured in advance. Therefore, the fine-tuning time on device v_i at round t is given by $H_i^{t,\text{comp}} = D_i(\bar{h}_i + x_i^t h_i)$. The transmission time primarily arises from uploading LoRA parameters via FDMA, which allows simultaneous transmissions from multiple devices [40]. Let B_i^t denote the bandwidth allocated to v_i at round t , then the uplink transmission rate $r_i^t = B_i^t \log(1 + \frac{p_i h_i^t}{\sigma^2})$ is modeled by the Shannon capacity, where p_i is the transmission power, h_i^t is the channel gain, and σ^2 is the noise power. Therefore, the transmission time for v_i at round t is $H_i^{t,\text{comm}} = \frac{\tau x_i^t}{r_i^t}$, where τ is the parameter size per LoRA layer. The duration of round t is determined by the slowest device, i.e.,

$$H^t = \max_{v_i \in \mathcal{V}} \{H_i^{t,\text{comp}} + H_i^{t,\text{comm}}\}. \quad (10)$$

For memory, let $\bar{m}$ denote the fixed memory to load the LVM and store forward activations, and m represent the extra memory incurred by gradient storage per LoRA layer during backpropagation. Then, the total memory overhead for fine-tuning on v_i at round t is

$$m_i^t = \bar{m} + x_i^t m. \quad (11)$$

Let the total completion time $H = H^{\text{gene}} + H^{\text{pre}} + \sum_{t=1}^T H^t$, we formulate the problem as follows:

$$(\mathbf{P1}) : \min_{\lambda, \mathcal{X}^t, \mathcal{B}^t} H \quad (12a)$$

$$\text{s.t. } F(\mathbf{w}^T) - F(\mathbf{w}^*) \leq \epsilon \quad (12b)$$

$$\sum_{v_i \in \mathcal{V}} B_i^t \leq B \quad \forall t \in [0, T] \quad (12c)$$

$$m_i^t \leq M_i \quad \forall i \in [1, N] \quad (12d)$$

Constraint (12b) represents the global model converges after T rounds, where ϵ is the convergence threshold. Constraint (12c) ensures the total allocated bandwidth does not exceed the budget B in each round. Constraint (12d) ensures each device v_i 's memory usage does not exceed its available memory M_i . Our objective is to decide the data scaling factor λ , the LoRA depth configuration $\mathcal{X}^t = \{x_i^t | v_i \in \mathcal{V}\}$ and bandwidth allocation $\mathcal{B}^t = \{B_i^t | v_i \in \mathcal{V}\}$ for each round, minimizing the total completion time, i.e., $\min_{\lambda, \mathcal{X}^t, \mathcal{B}^t} H$.

5.2 Algorithm Description

In this section, we introduce the algorithm description for our FeDiSyn to solve problem **P1**, as described in Alg. 1.

① **Determining Scaling Factor.** The key challenge of solving problem **P1** is determining the scaling factor λ to balance the time required for synthetic image generation/pre-training and FedFT. From Section 5.1, the total time for generation and pre-training is $H^{\text{gene}} + H^{\text{pre}} = \lambda D\hat{h} + \lambda D\hat{h}\hat{T} = \lambda(\hat{h} + \hat{h}\hat{T})D = \lambda h_0$, where $h_0 \triangleq (\hat{h} + \hat{h}\hat{T})D$ is defined as the unit time for generation and pre-training. Conversely, empirical analysis across multiple datasets suggests that the subsequent FedFT time is negatively correlated with λ and grows exponentially with the target accuracy Acc . To quantify this relationship, we propose the following *scaling law* to describe the total completion time as

$$H(\lambda, Acc) = h_0 \lambda + \frac{1}{\kappa_1 \lambda^{\kappa_2}} e^{\kappa_3 Acc} + \kappa_4, \quad (13)$$

Algorithm 1 Diffusion Model-Driven Synthetic Image Pre-Training for Federated Fine-Tuning (FeDiSyn)**Input:** Data volume D_i , available memory M_i , bandwidth budget B **Output:** The fine-tuned model w^T

```

1: /* Step 1. Determining Scaling Factor.*/
2: Conduct a set of small-scale experiments to measure  $H(\lambda, Acc)$ 

3: Fit coefficients  $\kappa_1$ - $\kappa_4$  using nonlinear least squares method
4: Obtain optimal scaling factor  $\lambda^* = \sqrt{\frac{\kappa_2 e^{\kappa_3 Acc}}{h_0 \kappa_1}}$ 
5: /* Step 2. Generating Synthetic Images and Pre-training.*/
6: Obtain the centroids  $e_{i,z}^k$  and average representation  $e_i^k$  on  $v_i$ 
7: Devices upload noisy latents  $\tilde{e}_{i,z}^{k,s}$ ,  $\tilde{e}_i^{k,s}$  and labels of each  $c_k$ 
8: Server expands the labels  $C$  into prompts
9: Server performs image generation with  $\lambda^*$  yields  $\hat{D}$ 
10: Pre-train LVM over  $\hat{D}$  and distribute LoRA-inserted models
11: /* Step 3. Federated Fine-tuning.*/
12: for each  $t \in T$  do
13:   /* Contribution-aware LoRA configuration and bandwidth allocation algorithm (CLCBA).*/
14:   for each  $v_i \in \mathcal{V}$  do
15:      $x_i^t \leftarrow 1$ ,  $x_i^{\max} \leftarrow \min(\lfloor (M_i - \bar{m})/m \rfloor, L)$ 
16:      $R_{\max}^t \leftarrow 0$ 
17:     while true do
18:       for each  $v_i \in \mathcal{V}$  do
19:         if  $x_i^t < x_i^{\max}$  then
20:            $x_i^t \leftarrow x_i^t + 1$ 
21:           Obtain  $\mathcal{B}^t$  under  $\mathcal{X}^t$  by water-filling based bandwidth allocation sub-algorithm (WFBA, Alg. 2)
22:           Calculate completion time  $H^t$  by Eq. (10)
23:           Calculate performance gain  $R^t$  by Eq. (14)
24:            $\Delta R_i^t \leftarrow R^t - R_{\max}^t$ ,  $x_i^t \leftarrow x_i^t - 1$ 
25:         if  $\exists \Delta R_i^t > 0$  then
26:            $i \leftarrow \text{argmax}_{v_i \in \mathcal{V}} \Delta R_i^t$ ,  $R_{\max}^t \leftarrow R_{\max}^t + \Delta R_i^t$ ,  $x_i^t \leftarrow x_i^t + 1$ 
27:         else
28:           break
29:       Obtain  $\mathcal{B}^t$  under  $\mathcal{X}^t$  by Alg. 2
30:       After LoRA assignment, devices perform local fine-tuning
31:       Server aggregates the LoRA matrices uploaded by devices
32: return The fine-tuned model  $w^T$ 

```

where $h_0 \lambda$ represents the linear growth of generation and pre-training time, and the remaining terms capture the diminishing returns of synthetic data on FedFT acceleration. The architecture-specific coefficients κ_1 - κ_4 , which depend on the dataset, model and device capacity, can be estimated in a short calibration phase using a set of small-scale pilot runs, detailed in Section 6.2. To find the optimal scaling factor λ^* that minimizes the total training time, we take the derivative of Eq. (13) with respect to λ as $H'_\lambda = h_0 - \frac{\kappa_2}{\kappa_1 \lambda^{\kappa_2+1}} e^{\kappa_3 Acc}$. Taking $H'_\lambda = 0$ yields the optimal scaling factor $\lambda^* = \sqrt{\frac{\kappa_2 e^{\kappa_3 Acc}}{h_0 \kappa_1}}$. We derive the following key insights regarding λ^* .

Remark 3 (Impact of Server Capability). *As the server's computational capacity increases ($h_0 \rightarrow 0$), the optimal scaling factor λ^* grows. When the server possesses plenty of computational capacity, it can rapidly generate a vast synthetic images. As a result, a larger λ accelerates FedFT without incurring a significant generation time.*

Remark 4 (Impact of Target Accuracy). *The presence of Acc in the exponent, $\lambda^* \propto e^{\frac{\kappa_3}{\kappa_2+1} Acc}$, reveals that the demand for synthetic data grows exponentially with the target accuracy. For low-accuracy requirements, the overhead of large-scale synthesis outweighs its benefits. However, for high-accuracy requirements, the FedFT time increases so sharply that a massive synthetic pre-training for model initialization becomes essential to reduce the total training time.*

② Generating Synthetic Images and Pre-Training.

After the optimal λ^* is determined, the server generates a corresponding number of synthetic images (Alg. 1, Lines 6-9) and pre-trains the LVM on the generated images dataset, then distributes LoRA-inserted model to devices (Alg. 1, Lines 10), as described in Section 3.3 and 3.4.

③ Federated Fine-Tuning. The main idea of FedFT is to determine the LoRA configuration $\mathcal{X}^t$ to maximize the performance gain, while balancing completion times across devices through a bandwidth allocation strategy $\mathcal{B}^t$. To handle device heterogeneity in FedFT, we design a contribution-aware LoRA configuration and bandwidth allocation (CLCBA) algorithm (Alg. 1, Lines 12-31). Since the local gradient $g_i(\tilde{w}_{i,l}^t) = \nabla f_i(\tilde{w}_{i,l}^t)$ captures the contribution of l -th layer to the loss function [18, 37], we use the L2 norm $G_i^t = \sum_{l \in L_i} \|g_i(\tilde{w}_{i,l}^{t-1})\|_2^2$ of the last round $t-1$ to quantify the contribution of v_i at round t . The performance gain is defined as the sum of all devices' contributions divided by current round's duration

$$R^t \triangleq \frac{\sum_{v_i \in \mathcal{V}} G_i^t}{H^t} = \frac{\sum_{v_i \in \mathcal{V}} \sum_{l \in L_i} \|g_i(\tilde{w}_{i,l}^{t-1})\|_2^2}{H^t}. \quad (14)$$

We first initialize the depths of all devices as 1, i.e., set $x_i^t = 1$ for all $v_i \in \mathcal{V}$ (Line 15). In each iteration, we choose a device that obtains the maximum performance gain among all devices to add one LoRA layer. This process continues until the maximum performance gain is less than its value in the last iteration (Alg. 1, Line 17-28). Meanwhile, in each iteration of the process of Alg. 1, we call the water-filling based bandwidth allocation sub-algorithm (WFBA) to balance completion times across devices, which minimizes the completion time H^t in Eq. (14).

The WFBA sub-algorithm is shown in Alg. 2. First, WFBA calculates the maximum computation time $H_{\max}^{t, \text{comp}}$ of all devices (Alg. 2, Lines 1-4). Since the waterline must exceed the maximum computation time $H_{\max}^{t, \text{comp}}$, WFBA initializes the waterline as $\mathcal{H}_0^t = 2H_{\max}^{t, \text{comp}}$ and iteratively solve its optimal value using Newton's method (Alg. 2, Lines 5-10). Finally, WFBA calculates the optimal bandwidth allocation B_i^t for each device v_i based on the converged waterline and returns the bandwidth configuration $\mathcal{B}^t$ (Alg. 2, Lines 11-13). The intuitive diagram of CLCBA to obtain $\mathcal{X}^t$ and $\mathcal{B}^t$ are shown in Fig. 6.

Algorithm 2 Water-Filling based Bandwidth Allocation (WFBA)

Input: Bandwidth budget B , depth configuration $\mathcal{X}^t$, transmission power p_i , and channel gain h_i^t

Output: Optimal bandwidth allocation $\mathcal{B}^t$

```

1: for  $i \leftarrow 1$  to  $N$  do
2:   Calculate  $H_i^{t, \text{comp}} \leftarrow D_i(\bar{h}_i + x_i^t h_i)$ 
3:    $\gamma_i^t \leftarrow \tau x_i^t / \log(1 + \frac{p_i h_i^t}{\sigma^2})$ 
4:    $H_{\max}^{t, \text{comp}} \leftarrow \max_i H_i^{t, \text{comp}}$ 
5:    $q \leftarrow 0, \mathcal{H}_q^t \leftarrow 2H_{\max}^{t, \text{comp}}$ 
6:   while  $|\Delta H^t| > \zeta$  do
7:      $\Phi(\mathcal{H}_q^t) = \sum_{i=1}^N \frac{\gamma_i^t}{\mathcal{H}_q^t - H_i^{t, \text{comp}}} - B$ 
8:      $\mathcal{H}_{q+1}^t = \mathcal{H}_q^t - \frac{\Phi(\mathcal{H}_q^t)}{\Phi'(\mathcal{H}_q^t)}$ 
9:      $\Delta H^t \leftarrow \mathcal{H}_{q+1}^t - \mathcal{H}_q^t$ 
10:     $q \leftarrow q + 1$ 
11:  for  $i \leftarrow 1$  to  $N$  do
12:     $B_i^t \leftarrow \gamma_i^t / (\mathcal{H}_q^t - H_i^{t, \text{comp}})$ 
13:  return  $\mathcal{B}^t$ 

```

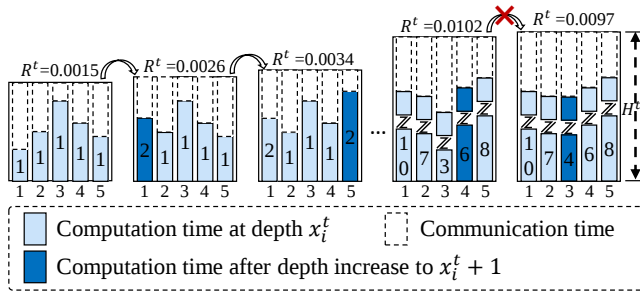


Fig. 6: An example of CLCBA. Initially, devices $v_1 - v_5$ have $x_i^t = 1$. In the first iteration, increasing the depth of v_1 yields $R^t = 0.0026$, and further increasing v_5 's depth raises R^t to 0.0034. This continues until the maximum $R^t = 0.0102$. Further increasing the depth of other devices will decrease R^t to 0.0097, so CLCBA terminates and returns the LoRA configuration $\mathcal{X}^t$. In this process, WFBA is invoked to obtain $\mathcal{B}^t$, balancing the completion time of all devices.

6 Performance Evaluation

6.1 System Setup

Environment Settings. We implement a FedFT experimental testbed comprising one server and 10 heterogeneous devices, as shown in Fig. 7. The server is equipped with an NVIDIA GeForce RTX 4090 GPU with 48GB GDDR6X. We use 1 Jetson AGX Xavier kit, 5 Jetson Orin NX kits, 3 Jetson AGX Orin kits, and 1 Jetson AGX Thor kit as heterogeneous devices. Particularly, we simulate more practical wireless communication in an edge environment instead of high-bandwidth laboratory networks, where communication time is typically negligible compared to computation time. Similar to [40], the server is at (0, 0), and each device is uniformly distributed within a circular area of 50 meters radius centered at coordinates (300, 0). The large-scale fading is represented as $(d/d_0)^{-3.5}$, where $d_0 = 10$ meters is the reference distance. The noise power is set as $\sigma^2 = 10^{-11}$ W, with a total system bandwidth of $B = 10$ MHz.



Fig. 7: Hardware devices of our testbed.

Models and Datasets. We train ViT-B/16 [5] on CIFAR-10 [17] and Caltech-101 [7] datasets, and train ViT-L/16 [5] on Food-101 [1] dataset. ViT-B/16 and ViT-L/16 consist of 86 and 307 million parameters, respectively.

- **CIFAR-10** [17] is composed of 60,000 color images, which are divided into 10 classes, each containing 6,000 images. The dataset is further split into 50,000 training images and 10,000 test images.
- **Caltech-101** [7] comprises 101 classes objects, totaling 9,144 samples. The number of samples per category ranges from 40 to 800 images, with each category averaging approximately 50 samples. We divide it into a 4:1 training-test split ratio.
- **Food-101** [1] contains a series of food datasets divided into 101 classes with high inter-class similarity. It comprises 101,000 samples, with each class including 750 training and 250 test samples.

We analyze FedFT performance under both IID and non-IID data among devices. Similar to [33], for the non-IID data case, we adopt the Dirichlet distribution [10] with $\alpha_{\text{dir}} = 0.3$ to partition the dataset. In addition, we deploy the LLaMa-7B [38] to enrich prompts and Stable-Diffusion-v1-5 [32] to generate synthetic images as described in Section 3.3.

Benchmarks. We compare FeDiSyn against two solutions with synthetic image pre-training (GPT-FL [47], FGL [45]) and two solutions deploying LoRA-based FedFT (SFLF [40], CAFF [30]). Specifically, GPT-FL and FGL generate images based on uploaded labels or text embeddings from devices, while SFLF and CAFF determine each device's LoRA depth based on its capacity. For a fair comparison, FeDiSyn, SFLF, and CAFF adopt the same LoRA rank.

- **GPT-FL**[47] is a synthetic image pre-training based approach that expands device-uploaded labels into prompts and generates images for pre-training LVM, while using FedAvg [26] for FedFT.
- **FGL**[45] is a synthetic image pre-training based approach where devices use BLIP to generate prompts for local images, and the server employs these prompts to generate images for pre-training LVM, while utilizing FedAvg for FedFT.
- **SFLF**[40] is a LoRA-based approach for FedFT, where the LVM is initialized with ImageNet parameters, and all devices fine-tune all transformer layers of the LVM.
- **CAFF**[30] is an advanced LoRA-based approach that initializes LVM with ImageNet parameters, and assigns appropriate LoRA depths to each device to address system heterogeneity.

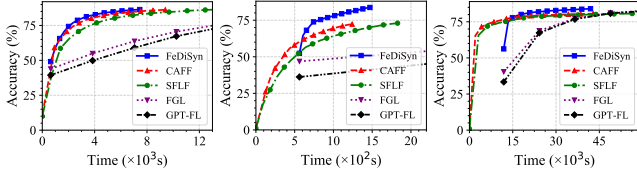


Fig. 8: The test accuracy over different datasets (IID). *Left: CIFAR-10; Middle: Caltech-101; Right: Food-101.*

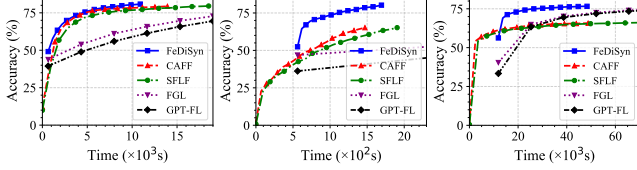


Fig. 9: The test accuracy over different datasets (non-IID). *Left: CIFAR-10; Middle: Caltech-101; Right: Food-101.*

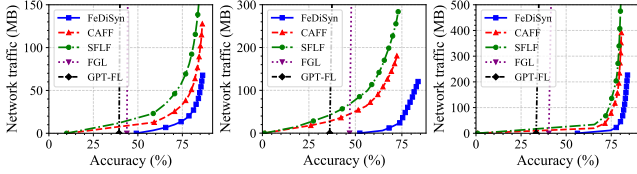


Fig. 10: The network traffic over different datasets (IID). *Left: CIFAR-10; Middle: Caltech-101; Right: Food-101.*

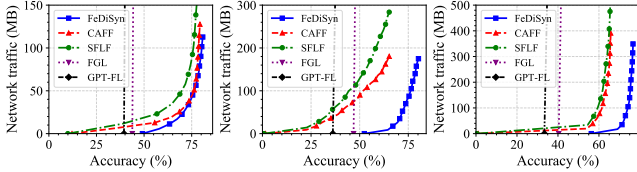


Fig. 11: The network traffic over different datasets (non-IID). *Left: CIFAR-10; Middle: Caltech-101; Right: Food-101.*

6.2 Evaluation Results

Training Performance. We evaluate the FedFT process of our FeDiSyn method and the benchmarks on the CIFAR-10, Caltech-101 and Food-101 datasets under both IID and non-IID data distributions, as illustrated in Figs. 8 and 9. Overall, FeDiSyn consistently achieves the fastest convergence and highest accuracy on all datasets under both IID and non-IID data. For example, as shown in the third plot of Fig. 8, under IID data on Food-101, FeDiSyn reaches 80% accuracy in 17,584s, while CAFF, SFLF, FGL, and GPT-FL require 31,905s, 41,885s, 44,783s, and 45,383s, respectively, meaning FeDiSyn reduces 44.9%, 58.0%, 60.7%, and 61.3% completion time compared to these benchmarks. Moreover, under non-IID data, FeDiSyn’s advantage becomes even more pronounced compared to other solutions. As shown in the third plot of Fig. 9, under non-IID data, FeDiSyn reduces 62.0%, 70.5%, 52.5%, and 56.2% time for reaching 65% accuracy compared to CAFF, SFLF, FGL, and GPT-FL, respectively. This is because FeDiSyn pre-training over synthetic images that enhances the generalization ability of initial model, alleviating the impact of non-IID data [12].

Network Traffic Consumption. In Fig. 10 and 11, we compare the network traffic consumption by different solutions to achieve certain accuracies on CIFAR-10 Caltech-101 and Food-101, respectively. By the results, FeDiSyn significantly reduces the network traffic compared with the benchmarks. For example, under non-IID

Table 2: The fine-tuning time (10^2 s) and network traffic (MB) to reach target accuracy.

Dataset	Metric	GPT-FL	FGL	SFLF	CAFF	FeDiSyn
CIFAR-10 (IID, 85%)	Time	232.5	220.3	103.5	73.3	55.1
	Traffic	23243.5	21798.8	172.7	99.8	50.3
CIFAR-10 (Non-IID, 80%)	Time	348.6	341.3	203.7	161.1	92.3
	Traffic	31051.5	29840.5	254.0	146.0	93.5
Caltech-101 (IID, 70%)	Time	113.7	96.4	14.6	10.3	6.8
	Traffic	14547.6	12198.6	226.2	150.1	15.7
Caltech-101 (Non-IID, 65%)	Time	147.5	116.7	18.9	14.5	6.3
	Traffic	18564.2	14755.1	279.5	177.8	10.1
Food-101 (IID, 80%)	Time	453.8	447.8	418.9	319.1	175.8
	Traffic	30763.4	30375.9	457.3	290.1	42.5
Food-101 (Non-IID, 65%)	Time	284.9	262.2	423.1	328.4	124.7
	Traffic	13676.3	12002.8	399.5	262.1	7.4

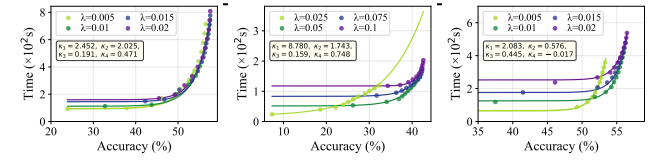


Fig. 12: Fitting Eq. (13) with small λ . *Left: CIFAR-10; Middle: Caltech-101; Right: Food-101.*

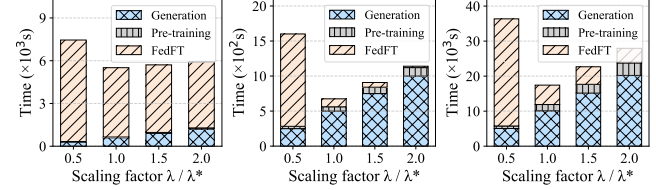


Fig. 13: The time to reach target accuracy under different scaling factor λ . *Left: CIFAR-10 (85%); Middle: Caltech-101 (70%); Right: Food-101 (75%).*

data on Food-101, the consumed network traffic of FeDiSyn, CAFF, SFLF, FGL, and GPT-FL is 7.4MB, 262.1MB, 399.5MB, 12,002.8MB, and 13,676.3MB, respectively. FeDiSyn can reduce network traffic by 97.2%, 98.1%, 99.9%, and 99.9% compared with these methods. Compared with the LoRA-based methods CAFF and SFLF, FeDiSyn adopts a more efficient LoRA configuration strategy, thereby reducing network traffic when achieving the target accuracy. In addition, FGL and GPT-FL rely on full parameter FedFT, which incurs extremely high communication overhead.

The complete experimental results of FedFT performance and network traffic consumption are provided in Table 2.

Impact of Synthetic Image Volume. To estimate the values of coefficients $\kappa_1 - \kappa_4$ in Eq. (13), we fit this equation in a short calibration phase using a set of small values of λ via nonlinear least-squares (e.g., Levenberg-Marquardt method). Fig. 12 shows that Eq. (13) well describes the relationship among total training time, accuracy and the volume of generated data. Then we obtain

the optimal $\lambda^* = \frac{\kappa_2 + 1}{\sqrt{\kappa_2 e^{\kappa_3 Acc} / h_0 \kappa_1}}$ by the estimated coefficients.

Fig. 13 shows the time to reach target accuracy under different scaling factor $\lambda = 0.5\lambda^*, \lambda^*, 1.5\lambda^*$ and $2\lambda^*$ on CIFAR-10, Caltech-101 and Food-101. Acc is set as 85%, 70% and 80% for CIFAR-10, Caltech-101 and Food-101, respectively. We observe that the generation and pre-training time increase linearly with λ . For example, for Food-101, the generation times are 5,050s, 10,100s, 15,150s, and

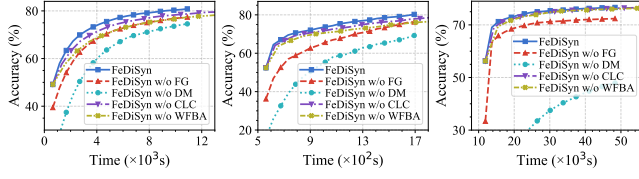


Fig. 14: Ablation study on accuracy. Left: CIFAR-10; Middle: Caltech-101; Right: Food-101.

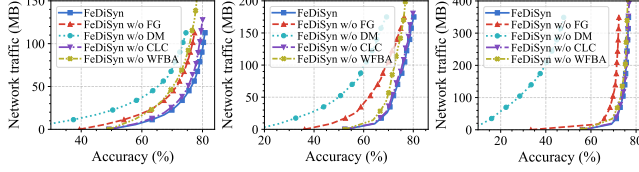


Fig. 15: Ablation study on Bandwidth. Left: CIFAR-10; Middle: Caltech-101; Right: Food-101.

20,200s for $\lambda = 0.5\lambda^*, \lambda^*, 1.5\lambda^*$, and $2\lambda^*$, respectively. However, when the value of λ is too large or too small, the total completion time to achieve the target accuracy will be relatively long. For example, for Food-101, when λ increases from $0.5\lambda^*$ to $2\lambda^*$, the time required to achieve 80% accuracy is 36,351.0s, 17,583.9s, 22,671.1s, and 27,943.9s, respectively. This is because insufficiently generated images weaken pre-training effectiveness for FedFT, whereas excessive images introduce substantial computational overhead, thereby prolonging image generation. The experimental results in Figs. 12–13 show that the values of λ^* obtained by the empirical model have the shortest completion time compared to other values.

Ablation Study. FeDiSyn comprises two primary modules: 1) a feature-guided (FG) synthetic generation module with a diffusion model (DM), 2) a LoRA-based FedFT module deploying contribution-aware LoRA configuration (CLC) and water-filling bandwidth allocation (WFBA). We conduct fine-grained ablation experiments on CIFAR-10, Caltech-101, and Food-101 in Figs. 14 and 15 to evaluate the individual contribution of each sub-module. FeDiSyn consistently outperforms all variants, including those without FG, CLC, WFBA, or those utilizing GANs instead of DMs. For example, on CIFAR-10, FeDiSyn achieves 74% accuracy with 42.4%, 61.4%, 22.8%, and 41.7% time reduction and reduces network traffic usage by 47.0%, 65.6%, 12.5%, and 51.7%, compared to FeDiSyn w/o FG, FeDiSyn w/o DM, FeDiSyn w/o CLC, and FeDiSyn w/o WFBA, respectively. We also observe that the relative importance of synthetic pre-training grows with model scale. For example, for ViT-B/16 training on CIFAR-10 and Caltech-101, the performance gaps among different variants are relatively small, but for ViT-L/16 training on Food-101, FeDiSyn w/o CLC and w/o WFBA significantly outperform FeDiSyn w/o FG and w/o DM, even approaching the performance of FeDiSyn. This is because larger models possess a higher degree of representational complexity, making a high-quality pre-training for initialization far more critical than fine-tuning stage.

7 Conclusion

In this paper, we propose FeDiSyn, a unified framework that holistically considers both pre-training and federated fine-tuning (FedFT) of large vision models to address resource constraints, system heterogeneity, and non-IID data. FeDiSyn introduces a scaling law for synthetic pre-training under FedFT setting to determine the optimal

number of synthetic images. A diffusion model is deployed on the server to generate synthetic images that align with the device’s local image label and feature distributions, thereby efficiently pre-training a superior initial global model for FedFT. During FedFT, we design a contribution-aware LoRA configuration and bandwidth allocation algorithm to enhance FedFT performance while addressing system heterogeneity. Experimental results indicate the superiority of our proposed framework compared with state-of-the-art solutions.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62402537, 62572244, 62402226 and U25A20442, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20241453, and in part by the Fundamental Research Funds for the Central Universities under Grant 30925010408. The corresponding author of this paper is Junlong Zhou.

References

- [1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101 – Mining Discriminative Components with Random Forests. In *European conference on computer vision*. 446–461.
- [2] Huancheng Chen and Haris Vikalo. 2024. Heterogeneity-guided client sampling: Towards fast and efficient non-iid federated learning. *Advances in Neural Information Processing Systems* 37 (2024), 65525–65561.
- [3] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han Wei Shen, and Wei-Lun Chao. 2023. On the Importance and Applicability of Pre-Training for Federated Learning. *International Conference on Learning Representations*.
- [4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. 248–255.
- [5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xi-aohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations*.
- [6] Lijie Fan, Kaifeng Chen, Dilip Krishnan, Dina Katabi, Phillip Isola, and Yonglong Tian. 2024. Scaling laws of synthetic images for model training... for now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7382–7392.
- [7] Li Fei-Fei, Rob Fergus, and Pietro Perona. 2004. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *Computer Vision and Pattern Recognition Workshop* (2004).
- [8] Tao Guo, Song Guo, and Junxiao Wang. 2024. Explore and Cure: Unveiling Sample Effectiveness With Context-Aware Federated Prompt Tuning. *IEEE Transactions on Mobile Computing* 23, 12 (2024), 14044–14054.
- [9] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. 2023. Is synthetic data from generative models ready for image recognition? *International Conference on Learning Representations* (2023).
- [10] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. *arXiv preprint arXiv:1909.06335* (2019).
- [11] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations* (2022).
- [12] Divyansh Jhunjhunwala, Pranay Sharma, Zheng Xu, and Gauri Joshi. 2025. Initialization Matters: Unraveling the Impact of Pre-Training on Federated Learning. *Transactions on machine learning research* (2025).
- [13] Feiyang Kang, Newsha Ardalani, Michael Kuchnik, Youssef Emad, Mostafa El-houshi, Shubhabrata Sengupta, Shang-Wen Li, Ramya Raghavendra, Ruoxi Jia, and Carole-Jean Wu. 2025. Demystifying Synthetic Data in LLM Pre-training: A Systematic Study of Scaling Laws, Benefits, and Pitfalls. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 10739–10758.
- [14] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv:2001.08361*

- [15] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian U Stich, and Ananda Theertha Suresh. 2021. Breaking the centralization barrier for cross-device federated learning. *Advances in Neural Information Processing Systems* 34 (2021), 28663–28676.
- [16] Alex Krizhevsky et al. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).
- [17] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. *Learning multiple layers of features from tiny images*. Citeseer.
- [18] Jianchun Liu, Rukuo Li, Hongli Xu, Qianpiao Ma, Jiaming Yan, and Liusheng Huang. 2026. FedQuad: Adaptive Layer-Wise LoRA Deployment and Activation Quantization for Federated Fine-Tuning. *IEEE Transactions on Mobile Computing* 25, 5 (2026), 6320–6334.
- [19] Jun Liu, Yunming Liao, Hongli Xu, Yang Xu, Jianchun Liu, and Chen Qian. 2025. Adaptive parameter-efficient federated fine-tuning on heterogeneous devices. *IEEE Transactions on Mobile Computing* (2025).
- [20] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. 2022. Pseudo Numerical Methods for Diffusion Models on Manifolds. In *International Conference on Learning Representations*.
- [21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
- [22] Qianpiao Ma, Xiaozhu Song, Junlong Zhou, Haibo Wang, Yunming Liao, Jianchun Liu, and Hongli Xu. 2026. Asynchronous Federated Learning Over Non-IID Data via Over-the-Air Computation. *IEEE Transactions on Networking* 34 (2026), 2165–2180.
- [23] Qianpiao Ma, Yang Xu, Hongli Xu, Zhida Jiang, Liusheng Huang, and He Huang. 2021. FedSA: A Semi-Asynchronous Federated Learning Mechanism in Heterogeneous Edge Computing. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3654–3672.
- [24] Qianpiao Ma, Yang Xu, Hongli Xu, Jianchun Liu, and Liusheng Huang. 2024. FedUC: A Unified Clustering Approach for Hierarchical Federated Learning. *IEEE Transactions on Mobile Computing* 23, 10 (2024), 9737–9756.
- [25] Qianpiao Ma, Junlong Zhou, Xiangpeng Hou, Jianchun Liu, Hongli Xu, Jianeng Miao, and Qingmin Jia. 2025. Air-FedGA: A Grouping Asynchronous Federated Learning Mechanism Exploiting Over-The-Air Computation. In *2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 1–12.
- [26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of Machine Learning Research*, Vol. 54. 1273–1282.
- [27] John Nguyen, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. 2023. Where to begin? exploring the impact of pre-training and initialization in federated learning. *International Conference on Learning Representations*.
- [28] Tim Pearce, Tabish Rashid, David Bignell, Raluca Georgescu, Sam Devlin, and Katja Hofmann. 2025. Scaling Laws for Pre-training Agents and World Models. In *Forty-second International Conference on Machine Learning*.
- [29] Danni Peng, Yuan Wang, Huazhu Fu, Jinpeng Jiang, Yong Liu, Rick Siow Mong Goh, and Qingsong Wei. 2025. Look Back for More: Harnessing Historical Sequential Updates for Personalized Federated Adapter Tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 19857–19865.
- [30] Kilian Pfeiffer, Mohamed Aboelenen Ahmed, Ramin Khalili, and Jörg Henkel. 2024. Efficient Federated Finetuning of Tiny Transformers with Resource-Constrained Devices. *arXiv preprint arXiv:2411.07826* (2024).
- [31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139. 8748–8763.
- [32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10684–10695.
- [33] Yizhou Shi, Qianpiao Ma, Yan Xu, Junlong Zhou, Ming Hu, Yunming Liao, and Hongli Xu. 2026. DySTop: Dynamic Staleness Control and Topology Construction for Asynchronous Decentralized Federated Learning. *IEEE Transactions on Mobile Computing* 25, 8 (2026), 11662–11678.
- [34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*. 2256–2265.
- [35] Shangchao Su, Bin Li, and Xiangyang Xue. 2024. Fedra: A random allocation strategy for federated tuning to unleash the power of heterogeneous clients. In *European Conference on Computer Vision*. Springer, 342–358.
- [36] Guangyu Sun, Matias Mendieta, Jun Luo, Shandong Wu, and Chen Chen. 2023. FedPerfix: Towards Partial Model Personalization of Vision Transformers in Federated Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4988–4998.
- [37] Yuchang Sun, Yuexiang Xie, Bolin Ding, Yaliang Li, and Jun Zhang. 2025. Exploring Selective Layer Fine-Tuning in Federated Learning. In *IEEE International Symposium on Information Theory (ISIT)*.
- [38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
- [39] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221.
- [40] Zixin Wang, Yong Zhou, Yuanming Shi, and Khaled B. Letaief. 2025. Federated Fine-Tuning for Pre-Trained Foundation Models Over Wireless Networks. *IEEE Transactions on Wireless Communications* 24, 4 (2025), 3450–3464.
- [41] Chulin Xie, De-An Huang, Wenda Chu, Daguang Xu, Chaowei Xiao, Bo Li, and Anima Anandkumar. 2024. Perada: Parameter-efficient federated learning personalization with generalization guarantees. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 23838–23848.
- [42] Yunlu Yan, Huazhu Fu, Yuexiang Li, Jinheng Xie, Jun Ma, Guang Yang, and Lei Zhu. 2025. A simple data augmentation for feature distribution skewed federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 25749–25758.
- [43] Mingzhao Yang, Shangchao Su, Bin Li, and Xiangyang Xue. 2024. Exploring one-shot semi-supervised federated learning with pre-trained diffusion models. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 38. 16325–16333.
- [44] Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When Scaling Meets LLM Finetuning: The Effect of Data, Model and Finetuning Method. In *The Twelfth International Conference on Learning Representations*.
- [45] Jie Zhang, Xiaohua Qi, and Bo Zhao. 2023. Federated generative learning with foundation models. *arXiv preprint arXiv:2306.16064* (2023).
- [46] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *The Eleventh International Conference on Learning Representations*.
- [47] Tuo Zhang, Tiantian Feng, Samiul Alam, Dimitrios Dimitriadis, Sunwoo Lee, Mi Zhang, Shrikant S Narayanan, and Salman Avestimehr. 2025. Gpt-fl: Generative pre-trained model-assisted federated learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 1761–1770.
- [48] Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. 2023. FedPETuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Association for Computational Linguistics*. 9963–9977.
- [49] Junlong Zhou, Yufan Shen, Liying Li, Cheng Zhuo, and Mingsong Chen. 2022. Swarm intelligence-based task scheduling for enhancing security for IoT devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 6 (2022), 1756–1769.
- [50] Tianfei Zhou, Ye Yuan, Binglu Wang, and Ender Konukoglu. 2024. Federated Feature Augmentation and Alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 12 (2024), 11119–11135.
- [51] Guangxu Zhu, Dongzhu Liu, Yuqing Du, Changsheng You, Jun Zhang, and Kaibin Huang. 2020. Toward an intelligent edge: Wireless communication meets machine learning. *IEEE communications magazine* 58, 1 (2020), 19–25.