GPT-FL: Generative Pre-Trained Model-Assisted Federated Learning

Anonymous authors
Paper under double-blind review

Abstract

In this work, we propose GPT-FL, a generative pre-trained model-assisted federated learning (FL) framework. At its core, GPT-FL leverages generative pre-trained models to generate diversified synthetic data. These generated data are used to train a downstream model on the server, which is then fine-tuned with private client data under the standard FL framework. We show that GPT-FL consistently outperforms state-of-the-art FL methods in terms of model test accuracy, communication efficiency, and client sampling efficiency. Through comprehensive ablation analysis, we discover that the downstream model generated by synthetic data plays a crucial role in controlling the direction of gradient diversity during FL training, which enhances convergence speed and contributes to the notable accuracy boost observed with GPT-FL. Also, regardless of whether the target data falls within or outside the domain of the pre-trained generative model, GPT-FL consistently achieves significant performance gains, surpassing the results obtained by models trained solely with FL or synthetic data.

1 Introduction

Federated learning (FL) is a privacy-preserving machine learning paradigm that allows a collection of clients to collaboratively train a machine learning model without sharing their private data Zhang et al. (2021). Most existing FL studies such as McMahan et al. (2016); Bonawitz et al. (2019) follow the standard FL architecture, where each participating client trains a local model using its own private data and a central server aggregates these locally trained models to update a global model and send it back to the clients for the next round of training. However, although many efforts have been made Sahu et al. (2018); Karimireddy et al. (2019); Reddi et al. (2020); Mishchenko et al. (2022), the performance of standard FL is still constrained by client drift caused by the heterogeneity in private data distribution across the clients.

To enhance the performance of FL, recent studies propose to incorporate data collected from public spaces such as the internet into the FL process Lin et al. (2020); Li et al. (2021); Itahara et al. (2020); Cho et al. (2022). However, the performance of such public data-based approaches is heavily dependent on the quality of the collected public data. Unfortunately, obtaining the desired public data can be extremely challenging in practice and there is a lack of principled guidance on how to obtain them. To address the issues of public data-based approaches, FL methods based on synthetic data emerge Zhang et al. (2022); Zhu et al. (2021); Pi et al. (2022); Wijesinghe et al. (2023). In Zhang et al. (2022); Zhu et al. (2021), a generative model is trained through knowledge distillation (KD) and the synthetic data are generated from the generative model in an *interleaved* manner *throughout* the federated training iterations. Unfortunately, these approaches are confronted with two limitations: (1) since the training of the generative model and the federated training process interleave, the quality of the synthetic data generated by the generative model before it converges can be extremely unstable. Such low-quality synthetic data would in turn jeopardize the federated training process; (2) given that KD requires clients to report model weights as teachers to transfer knowledge, they are incompatible with secure aggregation protocols Bonawitz et al. (2017); So et al. (2021), which limits their privacy guarantee compared to standard FL.

Table 1: Comparison of `GPT-FL` with existing FL methods.

	External Data	Limited to Smaller Client Model	Generate Data during FL	Data Generator Location	Client Access to Public/Generated Data	Support Data Modality	Compatibility with Secure Aggregation
FedAvg McMahan et al. (2016)	No	No	N/A	N/A	N/A	Image/Audio/Text	Yes
FedOpt Reddi et al. (2020)							
FedProx Sahu et al. (2018)							
SCAFFOLD Karimireddy et al. (2019)							
FedDF Lin et al. (2020)	Public Data	No	N/A	N/A	Not Required	Image/Audio/Text	No
DS-FL Itahara et al. (2020)					Required		
Fed-ET Cho et al. (2022)					Not Required		
MOON Li et al. (2021)	No	No	N/A	N/A	Not Required	Only Image	Yes
FedGen Zhu et al. (2021)	Generated Data	Yes	Yes	Client	Required	Only Image	No
FedFTG Zhang et al. (2022)				Server	Not Required	Only Image	No
DynaFed Pi et al. (2022)				Server	Not Required	Only Image	Yes
GPT-FL (Ours)		**No**	**No**	**Server**	**Not Required**	**Image/Audio/Text**	**Yes**

In this work, we propose `GPT-FL`, a generative pre-trained model-assisted FL framework that effectively addresses the issues of existing methods. The key idea behind `GPT-FL` is to leverage the knowledge from the generative pre-trained models and to *decouple* synthetic data generation from the federated training process. Specifically, `GPT-FL` prompts the generative pre-trained models to generate diversified synthetic data. These generated data are used to train a downstream model on the server in the centralized manner, which is then fine-tuned with the private client data under the standard FL framework. By doing this, the proposed `GPT-FL` is able to combine the advantages of previous methods while addressing their limitations.

The proposed `GPT-FL` exhibits multifold merits compared to prior arts (Table 1): (1) In contrast to public data-based FL methods, `GPT-FL` gets rid of the dependency on the availability of the desired public data, offering much more flexibility in its applications. (2) Compared to other generative data-based approaches, the leverage of generative pre-trained models and the decoupling between synthetic data generation from the federated training process make the generated synthetic data in `GPT-FL` not impacted by private data distribution on the clients and the structure of the model to be trained. (3) By leveraging the computational resources on the server, `GPT-FL` provides a much more efficient way to utilize external data by incorporating them into the pre-training of the downstream model, which significantly reduces the communication and computation costs of FL. (4) The generation of downstream models using synthetic data takes place on the server. As such, it thereby eliminates the need for clients to bear any additional computational burden. (5) Lastly, as `GPT-FL` does not alter the standard FL framework, it is fully compatible with secure aggregation protocols as in standard FL methods. More importantly, `GPT-FL` does not introduce any additional hyper-parameters beyond the standard FL framework. This significantly simplifies the hyper-parameter optimization process, making `GPT-FL` much more practically useful.

We evaluate the performance of `GPT-FL` by comparing it against state-of-the-art FL methods under three categories: standard FL methods, public data-based methods, and generated data-based methods on five datasets that cover both image and audio data modalities. We highlight five of our findings: (1) `GPT-FL` consistently outperforms state-of-the-art FL methods under both low and high data heterogeneity scenarios with significant advantages in communication and client sampling efficiency. (2) Under a zero-shot setting, *i.e.* no real-world data is available, the downstream model after centralized training with synthetic images as part of `GPT-FL` achieves higher performance compared to the global model based on standard FL training with private data. On the contrary, the centralized training with synthetic audio performs worse than FL setups due to the impact of data modality and the quality of the generative pre-trained models. (3) GPT-FL does not fully rely on generated data. Regardless of whether the target data falls within or outside the domain of the pre-trained generative model, `GPT-FL` can largely improve model performance beyond relying solely on private data in a standard FL framework. (4) The downstream model generated by synthetic data controls gradient diversity during FL training, improving convergence speed and leading to significant accuracy gains with `GPT-FL`. (5) `GPT-FL` effectively leverages existing pre-trained downstream models to improve performance in the FL setting, similar to methods under the standard FL framework.

2 Related Work

Standard Federated Learning. In standard federated learning (FL), clients perform local model training on their private data whereas the central server aggregates these locally trained models to update a global model, which is then sent back for the next round of training. To enhance privacy, Secure Aggregation (SA) protocols Bonawitz et al. (2017); So et al. (2021) have been proposed to encrypt each model update and reveal only the sum of the updates to the server. However, the performance of FL is jeopardized by client drift which is caused by the heterogeneity of private data distribution. To tackle this issue, FedProx Sahu et al. (2018) introduces a proximal term to the local subproblem to constrain the local update closer to the global model; SCAFFOLD Karimireddy et al. (2019) leverages a variance reduction technique to mitigate the effect of drifted local updates; and FedOpt Reddi et al. (2020) proposes to update the global model by applying a gradient-based server optimizer to the average of the clients' model updates. A recent proposed algorithm ProxSkip Mishchenko et al. (2022); Malinovsky et al. (2022) offers an effective acceleration on convergence and communication by skipping the expensive prox operator for local training in most rounds.

FL with Public Data. To further mitigate client drift, recent studies propose to utilize public data (e.g., collected from the internet) in the process of federated training. For example, FedDF Lin et al. (2020) leverages public data at the server to aggregate client models through knowledge distillation (KD). DS-FL Itahara et al. (2020) proposes a similar approach based on semi-supervised FL. MOON Li et al. (2021) proposes to use contrastive loss to further improve the performance. Fed-ET Cho et al. (2022) introduces a weight consensus distillation scheme using public data to train a large server model with smaller client models. Mixed FL Augenstein et al. (2022) proposes offloading some intensive computations from clients to the server with public data to reduce communication and client computation load. Xu et al. (2023) reveals that applying pre-training on public data would improve privacy and system utility to federated learning with differential privacy. Nguyen et al. (2022) and Chen et al. (2022) also highlight the importance and applicability of utilizing pre-training in FL. FreD Hou et al. (2023) presents a privacy-preserving method for pre-finetuning foundation models in FL by computing the Fréchet distance between embeddings from a Large Language Model (LLM) on public and private federated datasets.. A recent study Wang et al. (2023) offers to leveraging public datasets and LLMs for differentially private on-device FL model training, enhancing the privacy-utility tradeoff through distillation in NLP tasks.

However, utilizing public data for FL has several limitations: the performance of FL heavily relies on the selected public data. However, it is unclear to which extent should the publish data be related to the training data to guarantee effective knowledge distillation, making it challenging to find appropriate public data for every use case Stanton et al. (2021); Alam et al. (2022); Zhang et al. (2022). Moreover, the involvement of KD requires clients to send model weights to the server. This requirement makes it incompatible with secure aggregation protocols, making them vulnerable to backdoor attacks Wang et al. (2020). Furthermore, some proposed methods Li et al. (2021); Lin et al. (2020) require clients to process the public data. Such requirement adds an extra computational burden to clients.

FL with Synthetic Data. To address the issues of public data-based approaches, FL methods based on synthetic data have been proposed Zhang et al. (2022); Zhu et al. (2021); Pi et al. (2022); Wijesinghe et al. (2023). In particular, FedGen Zhu et al. (2021) proposes to train a lightweight generator on the server using an ensemble of local models in a data-free manner. The generator is then sent to the clients to regularize local training. FedFTG Zhang et al. (2022) trains a GAN-based generator where the global model acts as the discriminator. The generated data are then used to fine-tune the global model on the server after model aggregation. However, training of the generator relies heavily on the global model, which can lead to poor performance under high data heterogeneity. Additionally, the quality of training the generator is impacted by the structure of the global model Kim et al. (2022), making the quality of the synthetic data unstable during training. Furthermore, these approaches are limited to image-related tasks, restricting their applicability to other data modalities. Specifically, both FedGen and FedFTG rely on training MLP-based or GAN-based lightweight generator networks to ensemble user information in a data-free manner, where the lightweight generator may have limitations in generating high-fidelity data. In addition, the MLP-based model is impractical to model temporal structures to signals such as audio and speech. Finally, some approaches Zhang et al. (2022); Zhu et al. (2021) could not support secure aggregation protocols due to the KD-based training, which could compromise the privacy of client data. As an alternative, DynaFed Pi et al. (2022) proposes to generate synthetic data via gradient inversion by applying multi-step parameter matching on global model trajectories and using the synthesized data to help aggregate the deflected clients into the global model. However, using gradient inversion for generating synthetic data could encounter limitations when dealing with high-resolution images Huang et al. (2021). In addition, this approach could not be directly used for other data modalities such as audio Dang et al. (2021). In this work, we propose `GPT-FL` as a solution to address these limitations.

3 GPT-FL: Generative Pre-Trained Model-Assisted Federated Learning

3.1 Overview

Consider the standard FL setting McMahan et al. (2016), in which the FL system is composed of a server with $\mathcal{K}$ clients, whose data is only locally kept without sharing. The clients cooperate in training a global model $\mathcal{W}_g$ with parameters θ with the aim to solve the optimization problem formulated as follows

$$\min_{\theta} f(\theta) := \sum_{i=1}^{\mathcal{K}} p_i F_i(\theta) \tag{1}$$

where $F_i(\cdot)$ represents the local objective of client i and p_i denotes the aggregation weight of client i satisfying $p_i \geq 0$ and $\sum_{i=1}^{\mathcal{C}} p_i = 1$.

Algorithm 1: GPT-FL.

```
Input: $\mathcal{G}$: generative pre-trained model, $T$: FL communication rounds
// Generate downstream model weight $\theta_{init}$ from $\mathcal{G}$ on server $S$
for each client $i \in \{1, \cdots, K\}$ in parallel do
    submit label names set $y_i$ to server $S$
end
global label name set $y_s \leftarrow$ aggregate $\langle y_i \rangle$
synthetic dataset $D' \leftarrow \mathcal{G}(y_s)$
$\theta_{init} \leftarrow$ centralized training $D'$ on server $S$
// Finetune trained downstream model on private client data with FL
Server $S$ sets $\theta_s^0 = \theta_{init}$
for $r \in \{0, \cdots, T-1\}$ communication rounds do
    Sample $n$ clients uniformly at random to define $\mathcal{C}$, and send $\theta_s^r$ to clients in $\mathcal{C}$
    Clients $c \in \mathcal{C}$ in parallel do:
        $\theta_c^r \leftarrow$ local model update
    Server S do:
        $\theta_s^{r+1} \leftarrow$ aggregate $\langle \theta_c^r \rangle$
end
Output: $\theta_s^T$
```

Under such formulation, the overall architecture of `GPT-FL` is illustrated in Figure 1. As shown, `GPT-FL` consists of three steps. First, prompts are created based on the label names at the server. These prompts are then utilized to guide the generative pre-trained models to generate synthetic data. The server uses these generated synthetic data to train a downstream model and distributes the trained model to the clients. Lastly, the clients use the trained model as the starting point, and finetune the model with their private data under the standard FL framework until it converges. The pseudocode of `GPT-FL` is presented in Algorithm 1. In the following, we describe the details in each step.

3.2 Create Prompts based on Label Names

As the first step of `GPT-FL`, a prompt that describes the desired content of the data is required to guide the synthetic data generation process. To do so, `GPT-FL` requires the clients to provide the set of label names[1]

[1]To protect user data privacy in FL setting, `GPT-FL` only requests the set of distinct label names instead of detailed label name distributions, and generates a uniform number of prompts for each label name.

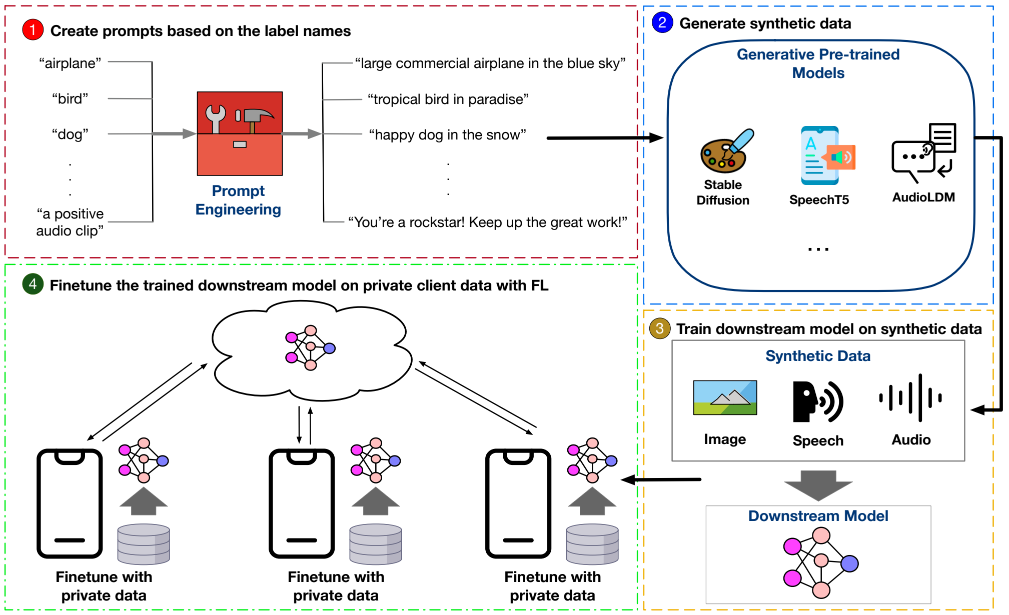
1 Create prompts based on the label names
"airplane"
"bird"
"dog"
"a positive audio clip"
Prompt Engineering
"large commercial airplane in the blue sky"
"tropical bird in paradise"
"happy dog in the snow"
"You're a rockstar! Keep up the great work!"
2 Generate synthetic data
Generative Pre-trained Models
Stable Diffusion
SpeechT5
AudioLDM
3 Train downstream model on synthetic data
Synthetic Data
Image
Speech
Audio
Downstream Model
4 Finetune the trained downstream model on private client data with FL
Finetune with private data
Finetune with private data
Finetune with private data

Figure 1: Overview of the proposed GPT-FL Framework.

of their private local data to generate prompts. However, prior research Shipard et al. (2023); He et al. (2022) shows that using only label names to generate prompts could restrict the quality and diversity of the generated synthetic data. Moreover, in FL, the server does not have access to detailed descriptions of the private data. To address these issues, `GPT-FL` incorporates large language models (LLMs) such as GPT-3 to expand each input class's details and use them as prompts for synthetic data generation. As an example, for the label name "airplane", `GPT-FL` uses the following query for the LLM to generate the prompt as follows:

```
Q: " _ _ _ _ airplane _ _ _ _"
 Please fill in the blank and make it as a prompt to generate the image.
A: Large commercial airplane in the blue sky.
```

Moreover, inspired by Shipard et al. (2023), we randomly set the unconditional guidance scale of the Stable Diffusion model between 1 and 5 to further enrich the data diversity. In addition to the aforementioned techniques, it is worth noting that `GPT-FL` is flexible and compatible with other prompt engineering techniques that can be used to generate diversified synthetic data.

3.3 Integration of Invertible Bloom Lookup Tables (IBLT) to Enhance Label Privacy

It should be noted that `GPT-FL` can employ Invertible Bloom Lookup Tables (IBLT) to encode label names before sending them to the server so that the label information of each client is not leaked to the server Gascón et al. (2023). Specifically, each client locally encodes its unique label names into IBLT, a probabilistic data structure that can encode items in an open domain efficiently. The server linearly aggregates these IBLTs via the secure aggregation Bonawitz et al. (2016) and decodes the aggregated table for the union of unique label names without revealing individual label information.

Within the `GPT-FL` framework, the set of distinct label names is sourced from an open domain. The server lacks detailed length information on the set, making it challenging to directly encode the label names properly for secure aggregation. To address this, we propose to locally encode the unique label names into IBLT Goodrich & Mitzenmacher (2011) data structure, a randomized data structure efficient in storing key-value pairs within an open domain. IBLT is a bloom filter-type linear data structure that supports the efficient listing of inserted elements and their precise counts, with table size scaling linearly with unique keys. IBLT sketches are amenable to linear summation, thus compatible with secure aggregation protocols.

In the `GPT-FL` framework's IBLT integration, each client locally encodes its distinct label names into IBLT and transmits it to the server. The server performs linear aggregation of these IBLTs through a secure multi-party computation protocol, subsequently decoding the aggregated table to obtain total label name counts without revealing individual label information. By leveraging the collective label name histogram, the server determines the union of distinct label names for data generation, maintaining the privacy of client-specific details. This approach finds validation in prior research Gascón et al. (2023), where IBLT demonstrated its efficacy in addressing private heavy hitters within federated analytics.

To better demonstrate the integration of IBLT in `GPT-FL`, we provide an illustrated experiment as an example. The experiment is conducted with the TensorFlow Federated IBLT API TensorFlow (2023). We partition the CIFAR-10 dataset heterogeneously amongst 100 clients using the Dirichlet distribution $Dir_K(\alpha)$ with α equal to 0.1. As the server does not know the length of the dataset initially, we set the capacity of the IBLT sketch to 50, which is much larger than the total number of unique labels inside CIFAR-10 (i.e., 10). Each client encodes its unique set of label names into IBLT and sends it to the server. The server would aggregate them via the secure aggregation protocol, which means the server can not access the individual IBLT but only knows the summation of IBLTs. After decoding the aggregated IBLT, the server only gets the following information:

```
Number of clients participated: 100
Discovered label names and counts:
{'dog': 49, 'automobile': 59, 'bird': 50, 'horse': 32,
'cat': 46, 'frog': 27, 'deer': 44, 'truck': 37, 'airplane': 50, 'ship': 35}
```

The decode information only contains the number of participated clients and the histogram of the label name, which the server could infer the union of distinct label names for data generation. For example, the notation "'dog':49" denotes there are 49 clients who include the label 'dog' within their local datasets, but the server lacks knowledge regarding the specific client identities associated with this 'dog' label in the localized data. It is crucial to emphasize that the server remains unable to access specific client details, such as the labels held by individual clients. As suggested in the previous work TensorFlow (2023); Gascón et al. (2023), this algorithm could be further enhanced by adding a differential privacy mechanism. In conclusion, this IBLT-based algorithm will allow parties to jointly compute the union of unique label names without revealing individual label information, addressing concerns about privacy and confidentiality.

3.4 Generate Synthetic Data from Generative Pre-trained Model

Next, the generated prompts are used as the inputs to the generative pre-trained models to generate synthetic data. In this work, we utilize the state-of-the-art Latent Diffusion Model Rombach et al. (2021) loaded with Stable Diffusion V2.1 weights to generate synthetic images for image-based FL applications; and we utilize the state-of-the-art SpeechT5 model Ao et al. (2021) for text-to-speech and AudioLDM model Liu et al. (2023) for text-to-audio to generate synthetic speech and audio data, respectively.

It should be noted that the proposed `GPT-FL` is a general framework that supports other generative pre-trained models and data modalities beyond images and audio. Notably, `GPT-FL` treats the generative pre-trained models as a service provider, utilizing only their APIs for synthetic data generation without modifying any internal parameters or structures. This design concept enables AI practitioners and engineers to efficiently develop customized AI models without deploying the foundation model on the server side, making it compatible with the current trend of constructing downstream applications with API access to generative pre-trained models. For example, popular large language models such as ChatGPT do not provide access to their model parameters or architecture for deployment purposes, while they allow third-party to develop AI products using the provided API services. This design feature enables `GPT-FL` to be widely applicable to a range of pre-trained models, facilitating flexible and efficient synthetic data generation for FL.

3.5 Train Downstream Model on Generated Synthetic Data

With the generated synthetic data, `GPT-FL` trains a downstream model on the server in a centralized manner, and distributes the trained model to the clients participated in FL. This trained model acts as the initialized model for the following federated training process. One note should be emphasized from our empirical experiences is that training with synthetic data is prone to overfitting, as synthetic data tend to be highly patternized compared to real data. To mitigate the effects of overfitting, we adopt relatively large weight decay hyperparameters and small learning rates compared to training with real data. The detailed hyper-parameter selections are listed in Appendix A.

3.6 Finetune Trained Downstream Model on Private Client Data with FL

Lastly, the clients use the trained model distributed from the server as the starting point, and finetune the model with their private data under the standard FL framework until the finetuning converges. As such, `GPT-FL` does not alter the standard FL framework, making it fully compatible with secure aggregation protocols as in standard FL methods. More importantly, unlike existing generated data-based approaches Zhang et al. (2022); Zhu et al. (2021); Pi et al. (2022), `GPT-FL` does not introduce any additional hyper-parameters beyond the standard FL framework. This significantly simplifies the hyper-parameter optimization process, making `GPT-FL` much more practically useful.

4 Experiments

Datasets, Models, and Tasks. We evaluate the performance of `GPT-FL` on five datasets from three FL applications: image classification, speech keyword spotting, and environmental sound classification. For image classification, we conduct experiments on CIFAR-10, CIFAR-100 Krizhevsky (2009), and Oxford 102 Flower Nilsback & Zisserman (2008) using ConvNet Pi et al. (2022), ResNet18, ResNet50 He et al. (2015), and VGG19 Simonyan & Zisserman (2014). Among them, CIFAR-10 and CIFAR-100 contain images from diverse objects whereas Oxford 102 Flower only contains images of flowers but with higher resolutions for fine-grained classification.

For audio-related tasks, we choose the Google Command speech dataset Warden (2018) for keyword spotting and ESC-50 dataset Piczak for environmental sound classification. We followed the previous work Zhang et al. (2023) to use the same model for these two datasets, where the model consists of two convolution layers followed by one Gated Recurrent Units (GRU) layer and an average pooling layer is connected to the GRU output, which is then fed through two dense layers to generate the predictions. More detailed information about the data-preprocessing method and model setups is described in the Appendix.

Data Heterogeneity. For CIFAR-10 and CIFAR-100, the training dataset is partitioned heterogeneously amongst 100 clients using the Dirichlet distribution $Dir_K(\alpha)$ with α equal to 0.1 and 0.5 following the previous work Cho et al. (2022). With the same method, we partition Flowers102 into 50 subsets respectively due to its relatively small size.

For audio datasets, Google Speech Command is partitioned over speaker IDs, making the dataset naturally non-IID distributed. It contains a total of 105,829 audio recordings collected from 2,618 speakers. The training set includes the recordings from 2,112 speakers and the test set includes the rest. To create non-IID data distributions on ESC-50, we followed the previous work Zhang et al. (2023) to partition ESC-50 into 100 subsets using $Dir_K(\alpha)$ with α equal to 0.1.

Baselines and Evaluation Metrics. We compare `GPT-FL` against three categories of baselines: 1) standard FL methods without the use of public or generated synthetic data – FedAvg, FedProx, and Scaffold; 2) FL methods that involve the use of public data – MOON, FedDF, DS-FL, and Fed-ET; and 3) FL methods that utilize generated synthetic data – FedGen and DynaFed[2]. We use the test accuracy of the trained model as our evaluation metric. We run experiments with three different random seeds and report the average and standard deviation. The details of the hyper-parameter selection of each dataset and experiment are described in Appendix.

[2]We did not compare with FedFTG because its code is not open-source, and we could not reproduce their results following the paper.

[3]Zhu et al. (2021); Pi et al. (2022) only reported results on ConvNet. We tested these two methods on VGG19 but they are not converged.

Table 2: Model accuracy comparison between `GPT-FL` and existing FL methods. For public data-based methods MOON, FedDF, DS-FL and Fed-ET, the results on CIFAR-10 and CIFAR-100 are obtained from Cho et al. (2022), and the results on Flowers102 are marked as N/A given the practical challenge on finding a set of suitable public data that can boost its performance.

Method	Training Model	High Data Heterogeneity ($\alpha = 0.1$)			Low Data Heterogeneity ($\alpha = 0.5$)		
		CIFAR-10	CIFAR-100	Flowers102	CIFAR-10	CIFAR-100	Flowers102
FedAvg	VGG19	71.19 ($\pm$ 0.27)	30.21 ($\pm$ 0.32)	30.30 ($\pm$ 0.16)	74.82 ($\pm$ 0.23)	33.12 ($\pm$ 0.13)	34.75 ($\pm$ 0.90)
FedProx	VGG19	72.45 ($\pm$ 0.13)	31.51 ($\pm$ 0.11)	33.23 ($\pm$ 0.24)	75.24 ($\pm$ 0.19)	33.64 ($\pm$ 0.08)	40.56 ($\pm$ 0.19)
SCAFFOLD	VGG19	75.12 ($\pm$ 0.20)	30.61 ($\pm$ 0.57)	26.75 ($\pm$ 0.50)	78.69 ($\pm$ 0.15)	34.91 ($\pm$ 0.61)	33.21 ($\pm$ 0.41)
MOON	VGG19	75.68 ($\pm$ 0.51)	33.72 ($\pm$ 0.89)	N/A	81.17 ($\pm$ 0.41)	42.15 ($\pm$ 0.72)	N/A
FedDF	VGG19	73.81 ($\pm$ 0.42)	31.87 ($\pm$ 0.46)	N/A	76.55 ($\pm$ 0.32)	37.87 ($\pm$ 0.31)	N/A
DS-FL	VGG19	65.27 ($\pm$ 0.53)	29.12 ($\pm$ 0.51)	N/A	68.44 ($\pm$ 0.47)	33.56 ($\pm$ 0.55)	N/A
Fed-ET	VGG19	78.66 ($\pm$ 0.31)	35.78 ($\pm$ 0.45)	N/A	81.13 ($\pm$ 0.28)	41.58 ($\pm$ 0.36)	N/A
FedGen	ConvNet [3]	42.05 ($\pm$ 0.93)	26.64 ($\pm$ 0.66)	Not Converged	54.86 ($\pm$ 0.13)	34.03 ($\pm$ 0.42)	Not Converged
DynaFed	ConvNet [3]	71.59 ($\pm$ 0.10)	36.08 ($\pm$ 0.15)	Not Converged	75.66 ($\pm$ 0.21)	43.82 ($\pm$ 0.30)	Not Converged
GPT-FL	VGG19	**82.16 ($\pm$ 0.13)**	**47.80 ($\pm$ 0.32)**	**70.56 ($\pm$ 0.34)**	**82.17 ($\pm$ 0.20)**	**48.39 ($\pm$ 0.17)**	**74.84 ($\pm$ 0.43)**
GPT-FL	ConvNet	**72.62 ($\pm$ 0.24)**	**42.66 ($\pm$ 0.19)**	**37.91 ($\pm$ 0.43)**	**77.18 ($\pm$ 0.21)**	**47.89 ($\pm$ 0.28)**	**48.61 ($\pm$ 0.51)**

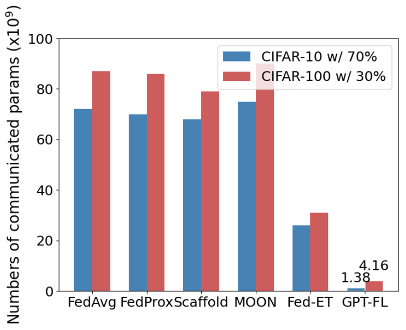

Figure 2: Communication costs of standard FL methods, public data-based methods and `GPT-FL` to achieve the target test accuracy.

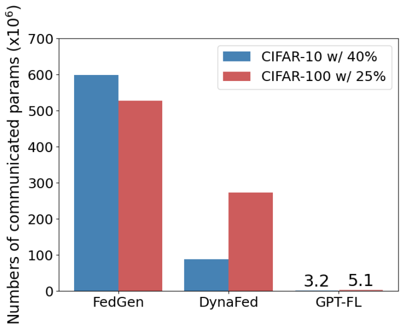

Figure 3: Communication costs of generated data-based methods and `GPT-FL` to achieve the target test accuracy.

4.1 Performance Comparison with State-of-the-Art FL Methods

First, we compare the performance of `GPT-FL` with state-of-the-art FL methods. To enforce fair comparisons, in this experiment, we choose to evaluate on the three image datasets (CIFAR-10, CIFAR-100 and Flowers102) since baseline methods MOON, FedGen and DynaFed only support image data. Moreover, we used the same models (VGG19 and ConvNet) and experiment settings as previous work Cho et al. (2022); Pi et al. (2022). In each communication round, we randomly sample 10 clients from 100 clients for CIFAR and use all 50 clients for Flowers102. We choose FedAvg as the FL optimizer. All the training starts from random initialization and total number of communication rounds is set to 500.

Overall Performance. Table 2 summarizes our results. We make three key observations: (1) `GPT-FL` consistently outperforms all the baselines we selected in Table 2 under both low and high data heterogeneity scenarios across all three datasets. (2) In direct comparison with state-of-the-art generated data-based FL methods, although FedGen and DynaFed perform reasonably well on CIFAR-10 and CIFAR-100, they do not converge on Flowers102 whose images have higher resolutions than CIFAR. Moreover, both FedGen and DynaFed fail to converge when training a larger VGG19 model on Flowers102 and even lower-resolution CIFAR-10/100. In contrast, `GPT-FL` not only converges but also achieves state-of-the-art accuracy on Flowers102. More importantly, `GPT-FL` is able to support larger model, and its accuracy is significantly higher than the smaller ConvNet. (3) For Flowers102, as both public data-based and generated data-based FL methods are confronted with challenges, the only viable options are standard FL methods and `GPT-FL`. As shown, with the same model, `GPT-FL` outperforms standard FL methods by a significant margin. We also conduct ablation studies in the Appendix A.3, demonstrating that `GPT-FL` complements and benefits other FL strategies, such as DynaFed.

Communication Efficiency. Besides model accuracy, we also compare the communication costs of `GPT-FL` with existing FL methods on CIFAR-10/100 under high data heterogeneity, where communication cost is measured as the total number of model parameters communicated between the server and clients during federated training until reaching a target model test accuracy. Specifically, Figure 2 shows the communication cost comparison between standard FL methods, public data-based methods[4], and `GPT-FL` under VGG19; and Figure 3 shows the communication cost comparison between generative data-based methods and `GPT-FL` under ConvNet. The target test accuracies in Figure 3 are set to be lower given the low accuracies achieved by FedGen. As shown, `GPT-FL` has the least communication cost among all the methods, achieving up to 94% communication reduction compared to the best-performed public data-based baseline Fed-ET and 98% communication reduction compared to the best-performed generated data-based baseline DynaFed. These results highlight the significant advantage of `GPT-FL` in communication reduction over state-of-the-art FL methods.

Client Sampling Efficiency. One critical hyper-parameter of FL is the client sampling rate in each communication round during the federated training process. In Figure 4, we plot the test model accuracies obtained by `GPT-FL` under low, medium, and high client sampling rates on CIFAR-10/100 with VGG19 under high data heterogeneity. As shown, even with a single participating client per round, `GPT-FL` is able to achieve 80.44% and 43.07% test accuracy on CIFAR-10 and CIFAR-100 respectively. This performance already surpasses all the other FL methods listed in Table 2, which employs 9 times more clients for training per round. These results highlight the significant advantage of `GPT-FL` in client sampling efficiency over state-of-the-art FL methods, making `GPT-FL` a very attractive solution in challenging scenarios where not many clients can participate at the same time.

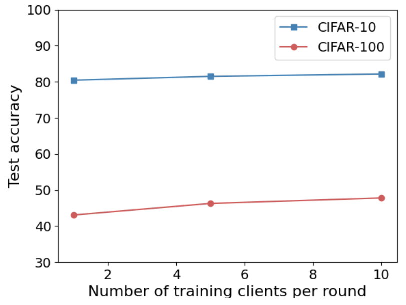

Figure 4: Test accuracy of `GPT-FL` for CIFAR-10/100 under different client sampling rates.

Quality of the Generated Synthetic Data As shown in Table 2, `GPT-FL` outperforms both generated data-based approaches FedGen and DynaFed significantly across all experimental conditions. One plausible reason for this could be associated with the quality of the generated synthetic data. Specifically, both FedGen and DynaFed rely on training MLP-based generator networks to ensemble user information in a data-free manner, where the lightweight generator may have limitations in generating high-fidelity data. The results of Flowers102 provide empirical evidence that such a lightweight generator has constrained capabilities in synthesizing image output on input images with larger sizes, making it challenging for the global model to converge. To illustrate this, Figure 5 and Figure 6 illustrate the synthetic images generated by `GPT-FL` and DynaFed, respectively. As shown, the learned generator of DynaFed fails to generate high-fidelity data as in `GPT-FL`.

4.2 Understanding GPT-FL

(1) Can we only rely on centralized training with synthetic data to achieve competitive results compared to Federated Learning with private data?

To answer this question, we compare the model performance between generated downstream model by centralized training with synthetic data and the global model by standard FL training with private data on both image and audio benchmark datasets. Different from the previous section, we select ResNet18 and ResNet50 models for CIFAR-10 and CIFAR-100 dataset, respectively. We choose the models proposed in the FedAudio Benchmark Zhang et al. (2023) for audio tasks. We report the best F1 score for the audio datasets. The results are summarized in Table 3.

Impact of Out-of-Domain Data Generation. We choose the ESC-50 and Google Speech Commands datasets to examine the impact of out-of-domain data generation for the generative pre-trained model. We did not conduct a similar analysis for the image datasets as the LAION-5B Schuhmann et al. (2022) open-source

[4]We do not compare with FedDF and DS-FL as they do not achieve competitive model accuracy.

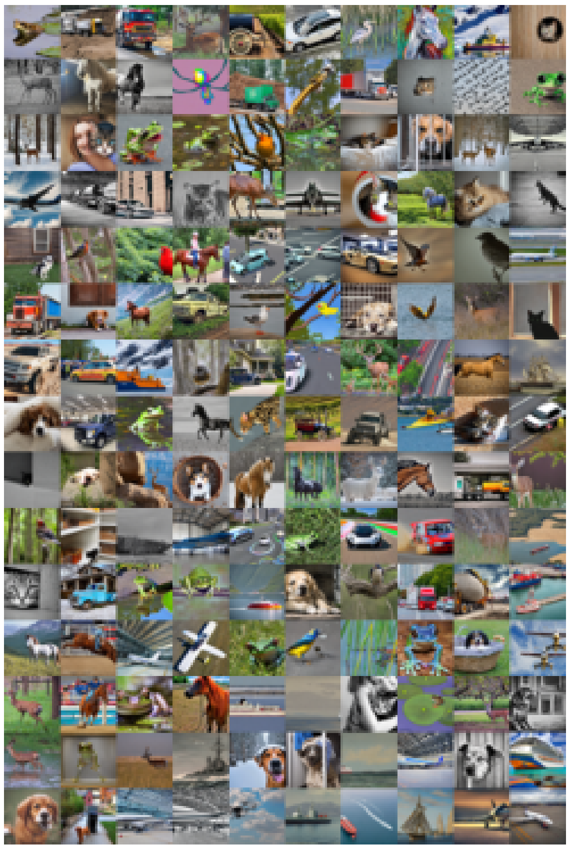
Figure 5: Synthetic CIFAR-10 data by `GPT-FL`.

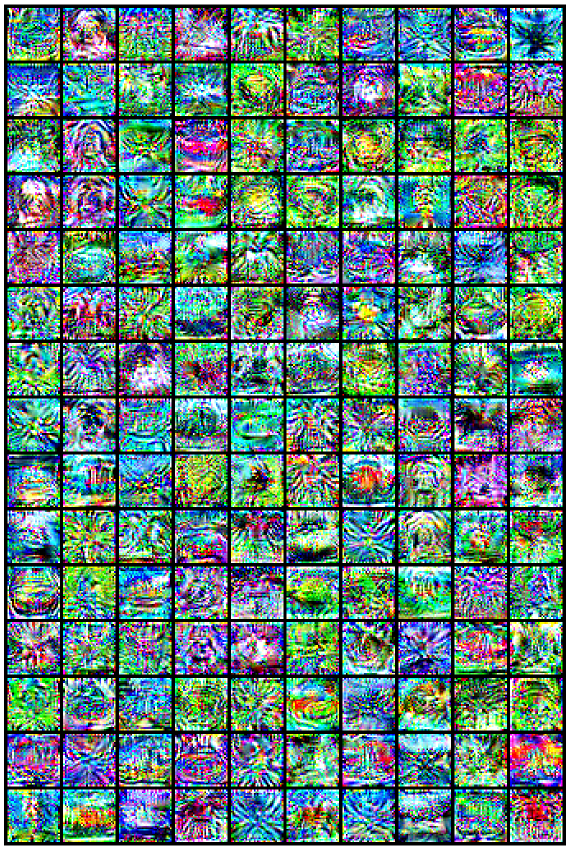
Figure 6: Synthetic CIFAR-10 data by DynaFed.

Table 3: Accuracy performance of the generated downstream model and standard FL on benchmark datasets. "1x Synthetic" represents the size of synthetic data is one time as the real data.

	Dataset	1x Synthetic	2x Synthetic	3x Synthetic	FedAvg	FedOpt
Image Data	CIFAR-10	61.48 (± 0.08)	67.41 (± 0.40)	**75.65 (± 0.09)**	64.48 (± 0.13)	72.68 (± 0.22)
	CIFAR-100	24.70 (± 0.00)	33.41 (± 0.01)	**41.76 (± 0.03)**	25.89 (± 0.67)	20.85 (± 0.14)
	Flowers102	24.94 (± 0.57)	28.26 (± 0.14)	**31.29 (± 0.18)**	30.30 (± 0.16)	26.43 (± 0.09)
Audio Data	Google Command	24.78 (± 0.04)	25.65 (± 0.07)	26.24 (± 0.01)	73.68 (± 0.49)	**83.01 (± 0.23)**
	ESC-50	6.89 (± 0.29)	8.68 (± 0.35)	12.72 (± 0.31)	22.76 (± 1.01)	**32.49 (± 0.57)**

dataset for training the Stable Diffusion model we used is a vast collection of publicly available datasets, including nearly all relevant ones for our experiments.

Our experiments show that synthetic image outperforms synthetic audio regarding model performance when using centralized training. We observed that centralized training with synthetic images achieves higher accuracy than FL setups for all three image benchmark datasets. In contrast, centralized training with synthetic audio performs worse than FL setups for ESC-50 and Google Speech Command datasets. The finding from the Google Speech Command experiments aligns with the previous study Li et al. (2018) that utilizes pure synthetic speech data to train the automatic speech recognition system leading to substantial performance degradation. One plausible explanation is related to the relatively small training data sizes (approximately 400M sentences) and constrained domain knowledge (book corpus) compared to training other generative pre-trained models like Stable Diffusion. For example, using human inspection, we discovered that the TTS model fails to synthesize simple spoken words like "house". This deficiency may originate from the lack of short-spoken utterance samples in training data. In addition, the synthesized speech often lacks diversity due to the limited range of speakers represented in the training dataset. On the other hand, there is insufficient knowledge of the audio generation models, making the model performance of using the synthesized audio data as training data remains unknown. However, our manual inspection revealed that the model frequently encounters difficulties in generating audio samples, such as generated audio related to water sounds often sounds like music. This issue could be largely associated with the relatively small data size in pre-training compared to other foundation models.

Impact of Numbers of Synthetic Data. With both image and audio data, one commonality is centralized training with synthetic data can benefit from increasing the number of synthetic data. To validate this finding, we test the impact of numbers of synthetic data on the performance of the generated model on the Flowers102 dataset, where we increase the size of the synthetic data up to ten times that of the real data. As shown in Figure 7, our experimental results demonstrate that as we enlarge the amount of synthetic data, the performance of the model improves. One justification for this finding is that enlarging the number of synthetic data enriches the diversity and increases overlap between the synthetic and real data, allowing the model to learn more robust and generalizable features. Even if the data is generated randomly by the label name without any other diversity-enriching guidance from the real data, with more synthetic data, there is an increasing chance that some of these additional synthetic data overlap with the real data, allowing the model to perform better on the real test data.

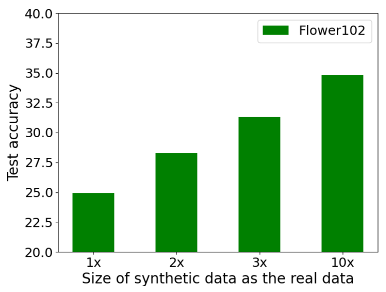

Figure 7: Impact of synthetic data sample number to the generated downstream model.

(2) Why `GPT-FL` needs to fine-tune the downstream models federatively?

Evaluating Client-isolated Fine-tuning against `GPT-FL` Performance. To underscore the effectiveness of federation in fine-tuning, we present an ablation study comparing the performance of local fine-tuning in isolation against FL fine-tuning. This study utilizes the Google Speech Command and CIFAR-100 datasets. We partition the CIFAR-10 dataset using the Dirichlet distribution $Dir_K(\alpha)$ with α equal to 0.1 into 100 clients, and partition the Google Speech Command dataset over speaker IDs into 2,618 clients. We select the VGG19 model for CIFAR-100 dataset to align with Table 2. In the isolated fine-tuning scenario, we select 10 clients at random, allowing each to fine-tune the synthetic-data-based downstream model independently with its local data for 500 epochs. The average accuracy from these clients is then computed. For the `GPT-FL` setup, we maintain the experimental parameters as per the setups described in Table 5. The results of these experiments are summarized in Table 4. We also have conducted an ablation study to explore the impact of distribution drift in synthetic data on `GPT-FL`'s performance in Appendix A.4, which highlights the efficacy of the `GPT-FL` approach under diverse data distribution conditions.

Table 4: Accuracy performance comparison between locally fine-tune in isolation and `GPT-FL` with FedAvg.

Dataset	Locally Fine-tune in Isolation	`GPT-FL` w/ FedAvg
CIFAR-100	35.53% (± 0.57)	**47.80% (± 0.32)**
Google Speech Command	23.00% (± 0.13)	**81.90% (± 0.20)**

The results show that fine-tuning in isolation at the client level yields significantly lower accuracy compared to the `GPT-FL` approach using FedAvg, which fine-tunes synthetic-data-based downstream model models federatively using private data. The primary reason for this disparity is the limited amount and skewed label distribution of the local data available to each client, which is insufficient for individually tuning the model to achieve optimal performance. These findings clearly demonstrate the value of federated learning in fine-tuning, especially given the limitations of local data in terms of volume and diversity.

(3) What benefits does `GPT-FL` bring?

We explore the benefits that `GPT-FL` provides for custom models that are built on top of downstream models generated from synthetic data. Specifically, we want to examine how fine-tuning these downstream models with private data under the FL framework can lead to performance improvements. To demonstrate how

GPT-FL can be integrated with existing FL server optimizers, we evaluate its performance with both FedAvg and FedOpt as the server aggregator. Our experimental results are presented in Table 5.

Table 5: Accuracy comparison between generated downstream model, standard FL and GPT-FL. Differ from the experiments shown in Table 1, the CIFAR-10 and Flowers102 datasets are trained with ResNet18 model and the CIFAR-100 dataset is trained with ResNet50 model. "ΔMetric" represents the accuracy increment by GPT-FL on top of the generated downstream model.

Dataset	3x Synthetic	FedAvg	FedOpt	GPT-FL w/ FedAvg	GPT-FL w/ FedOpt	ΔMetric
CIFAR-10	75.65 (± 0.09)	64.48 (± 0.13)	72.68 (± 0.22)	**81.38 (± 0.05)**	79.08 (± 0.17)	↑ **5.73**
CIFAR-100	41.76 (± 0.03)	25.89 (± 0.67)	20.85 (± 0.14)	**62.83 (± 0.31)**	48.80 (± 0.12)	↑ **21.07**
Flowers102	31.29 (± 0.18)	30.30 (± 0.16)	26.43 (± 0.09)	70.56 (± 0.34)	**77.57 (± 0.03)**	↑ **46.28**
Google Command	26.24 (± 0.01)	73.68 (± 0.49)	83.01 (± 0.23)	81.90 (± 0.20)	**83.46 (± 0.11)**	↑ **57.22**
ESC-50	12.72 (± 0.31)	22.76 (± 1.01)	32.49 (± 0.57)	41.80 (± 0.32)	**43.46 (± 0.30)**	↑ **30.74**

Effectiveness of Private Data. Our experiments demonstrate the effectiveness of incorporating private data with FL into the finetuning process of the downstream model generated from synthetic data. As shown in Table 5, regardless of the modality and quality of the synthetic data used to generate the downstream model, FL fine-tuning leads to significant performance gains, outperforming the ones trained solely with FL or CL combined with synthetic training by a large margin. Furthermore, we observe that fine-tuning with private data can especially benefit the cases for out-of-domain synthetic data, such as in the audio data. For example, GPT-FL with FedOpt could achieve 43.46 test accuracy in the ECS-50 dataset, which nearly provides two times increment than standard FL and three times increment than centralized training by synthetic data. These results suggest that leveraging private data with FL in the fine-tuning process can greatly enhance the performance of synthetic data-generated models, making them more suitable for real-world applications.

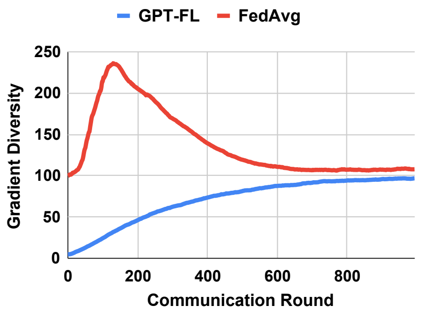

Figure 8: Smoothed Gradient diversity of client updates during training on Google speech commands dataset.

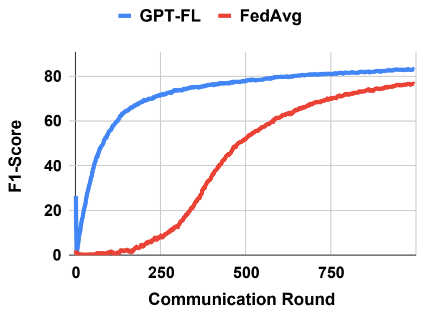

Figure 9: Learning curve of the global model during training on Google speech commands dataset.

Generated Downstream Model Helps FL Optimization. To gain a comprehensive understanding of why the custom models built using GPT-FL provide benefits to performance improvements, we decided to compare the gradient diversity between model weights initialized by GPT-FL and random initialization. Specifically, we apply the definition of the gradient diversity introduced from Yin et al. (2018) by adapting the gradients g_i to client update Δ_i:

$$\Delta_S = \frac{\sum_{i \in S} ||\Delta_i||^2}{||\sum_{i \in S} \Delta_i||^2} \tag{2}$$

where S is the set of sampled clients in each communication round and i represents the client index. As shown in Figure 8, the gradient diversity plot for FedAvg reveals that GPT-FL displays lower initial gradient diversity compared to random initialization. Over training time, both GPT-FL and random initialization converge to similar gradient diversity levels, consistent with the performance curve in Figure 9, where a larger Δ_S corresponds to slower convergence rate. This aligns with prior findings Nguyen et al. (2022); Chen et al. (2022), indicating that starting from a pre-trained model leads to less variation in local client updates, potentially addressing the client drift issue.

Table 6: Accuracy performance comparison between generated downstream model, standard federated learning and `GPT-FL`. All the training is initialized by ImageNet-based pre-train model.

Dataset	**3x Synthetic**	**FedAvg**	**FedOpt**	**`GPT-FL` w/ FedAvg**	**`GPT-FL` w/ FedOpt**
CIFAR-10	72.65 (± 0.05)	66.10 (± 0.03)	79.08 (± 0.39)	75.87 (± 0.73)	**82.20 (± 0.61)**
CIFAR-100	42.30 (± 0.01)	62.83 (± 0.03)	45.27 (± 0.10)	**66.84 (± 0.05)**	66.03 (± 0.02)
Flowers102	41.05 (± 0.26)	80.73 (± 0.01)	87.33 (± 0.29)	86.18 (± 0.04)	**88.66 (± 0.40)**

Harmonization With Existing Pre-train Model. As the standard FL framework, `GPT-FL` could also benefit from other existing pre-train models. Specifically, besides training from scratch, `GPT-FL` could utilize the existing pre-train model to start training the synthetic data to generate downstream model and finetune it again with private data in FL. Table 6 presents the performance evaluation of `GPT-FL` on top of the pre-trained models for image datasets. We follow the approach from prior work Nguyen et al. (2022); Chen et al. (2022) and use the ImageNet pre-trained model available in the PyTorch Torchvision library. Our experiments show that `GPT-FL` achieves better results compared to training solely with FL or synthetic data, as reported in Table 5. Notably, the improvement in performance is consistent across three image benchmark datasets, with a gain ranging from 1% to 11% compared to the results in Table 5. These results demonstrate that `GPT-FL` can effectively leverage pre-trained models to improve performance in the FL setting.

5 Conclusion

We present `GPT-FL`, a generative pre-trained model-assisted federated learning framework. `GPT-FL` leverages the generative pre-trained model to generate diversified synthetic data for a wide range of data modalities before FL training. This synthetic data is then utilized to construct a downstream model, which undergoes fine-tuning with private data within a standard FL framework. Our experimental results showcase the remarkable performance of `GPT-FL` when compared to state-of-the-art FL methods. Moreover, through detailed ablation studies, we demonstrate that `GPT-FL` is a flexible and applicable framework solution to the challenges associated with cross-device FL scenarios.

Limitations and Future works. Due to the limitations in our computational resources, we cannot further scale up the synthetic data volume in our study, as it may take several weeks for the generation. Besides, we do not investigate the larger model sizes in our current study, which we will pursue it as our future work. In addition, we want to explore the expansions of the GPT-FL framework. GPT-FL seamlessly integrates with the vanilla FL framework, allowing for harmonization with most of the existing FL methods. We are interested in exploring the combination of the public-data-based FL aggregation scheme and the GPT-FL framework by replacing the public data with synthetic data. While we recognize the importance of theoretical analysis, the inherent complexity and opacity of current LLMs pose significant challenges to formulating comprehensive theoretical frameworks. Our proposed questions are pivotal in the context of leveraging LLMs and foundational models, which often present as "black-box" systems with limited theoretical support. Our detailed experimental analysis, therefore, serves as a foundational step toward understanding the interplay between synthetic data generation, model fine-tuning, and federated learning in the context of LLMs and foundation models. The proof of theoretical guarantees is designated for future work. We believe these insights are invaluable for guiding future theoretical and empirical studies in this rapidly evolving field.

References

Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *ArXiv*, abs/2212.01548, 2022.

Junyi Ao, Rui Wang, Long Zhou, Shujie Liu, Shuo Ren, Yu Wu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Sean Augenstein, Andrew Hard, Lin Ning, K. Singhal, Satyen Kale, Kurt Partridge, and Rajiv Mathews. Mixed federated learning: Joint decentralized and centralized learning. *ArXiv*, abs/2205.13655, 2022. URL `https://api.semanticscholar.org/CorpusID:249151949`.

K. A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. URL `https://arxiv.org/abs/1611.04482`.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. B. McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. B. McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *ArXiv*, abs/1902.01046, 2019.

Hong-You Chen, Cheng-Hao Tu, Zi hua Li, Hang Shen, and Wei-Lun Chao. On the importance and applicability of pre-training for federated learning. In *International Conference on Learning Representations*, 2022. URL `https://api.semanticscholar.org/CorpusID:253158053`.

Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. In *International Joint Conference on Artificial Intelligence*, 2022.

Trung Dang, Om Thakkar, Swaroop Indra Ramaswamy, Rajiv Mathews, Peter Chin, and Franccoise Beaufays. A method to reveal speaker identity in distributed asr training, and how to counter it. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4338–4342, 2021.

Adrià Gascón, Peter Kairouz, Ziteng Sun, and Ananda Theertha Suresh. Federated heavy hitter recovery under linear sketching. *ArXiv*, abs/2307.13347, 2023. URL `https://api.semanticscholar.org/CorpusID:260154975`.

Michael T. Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 792–799, 2011. URL `https://api.semanticscholar.org/CorpusID:11589877`.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip H. S. Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? *ArXiv*, abs/2210.07574, 2022.

Charlie Hou, Hongyuan Zhan, Akshat Shrivastava, Sida I. Wang, Sasha Livshits, Giulia C. Fanti, and Daniel Lazar. Privately customizing prefinetuning to better match user data in federated learning. *ArXiv*, abs/2302.09042, 2023. URL `https://api.semanticscholar.org/CorpusID:257019611`.

Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. In *Neural Information Processing Systems*, 2021.

Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22:191–205, 2020.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, 2019.

Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization. In *International Conference on Machine Learning*, 2022.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL `https://api.semanticscholar.org/CorpusID:18268744`.

Jason Li, Ravi Gadde, Boris Ginsburg, and Vitaly Lavrukhin. Training neural speech recognition systems with synthetic speech augmentation. *arXiv preprint arXiv:1811.00707*, 2018.

Qinbin Li, Bingsheng He, and Dawn Xiaodong Song. Model-contrastive federated learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10708–10717, 2021.

Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *ArXiv*, abs/2006.07242, 2020.

Haohe Liu, Zehua Chen, Yiitan Yuan, Xinhao Mei, Xubo Liu, Danilo P. Mandic, Wenwu Wang, and MarkD . Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *ArXiv*, abs/2301.12503, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

Grigory Malinovsky, Kai Yi, and Peter Richt'arik. Variance reduced proxskip: Algorithm, theory and application to federated learning. *ArXiv*, abs/2207.04338, 2022. URL `https://api.semanticscholar.org/CorpusID:250426060`.

H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016.

Konstantin Mishchenko, Grigory Malinovsky, Sebastian U. Stich, and Peter Richt'arik. Proxskip: Yes! local gradient steps provably lead to communication acceleration! finally! In *International Conference on Machine Learning*, 2022. URL `https://api.semanticscholar.org/CorpusID:246996874`.

John Nguyen, Jianyu Wang, Kshitiz Malik, Maziar Sanjabi, and Michael G. Rabbat. Where to begin? on the impact of pre-training and initialization in federated learning. *ArXiv*, abs/2210.08090, 2022.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, 2008.

Renjie Pi, Weizhong Zhang, Yueqi Xie, Jiahui Gao, Xiaoyu Wang, Sunghun Kim, and Qifeng Chen. Dynafed: Tackling client data heterogeneity with global dynamics. *arXiv preprint arXiv:2211.10878*, 2022.

Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018. ACM Press. ISBN 978-1-4503-3459-4. doi: 10.1145/2733373.2806390. URL `http://dl.acm.org/citation.cfm?doid=2733373.2806390`.

Sashank J. Reddi, Zachary B. Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecný, Sanjiv Kumar, and H. B. McMahan. Adaptive federated optimization. *ArXiv*, abs/2003.00295, 2020.

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2021.

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

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models. *ArXiv*, abs/2210.08402, 2022.

Jordan Shipard, Arnold Wiliem, Kien Nguyen Thanh, Wei Xiang, and Clinton Fookes. Diversity is definitely needed: Improving model-agnostic zero-shot classification via stable diffusion. 2023.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Jinhyun So, Chaoyang He, Chien-Sheng Yang, Songze Li, Qian Yu, Ramy E. Ali, Basak Guler, and Salman Avestimehr. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. In *Conference on Machine Learning and Systems*, 2021.

Samuel Stanton, Pavel Izmailov, P. Kirichenko, Alexander A. Alemi, and Andrew Gordon Wilson. Does knowledge distillation really work? *ArXiv*, abs/2106.05945, 2021.

TensorFlow. Private heavy hitters, 2023. URL `https://www.tensorflow.org/federated/tutorials/private_heavy_hitters`.

Boxin Wang, Yibo Zhang, Yuan Cao, Bo Li, H. B. McMahan, Sewoong Oh, Zheng Xu, and Manzil Zaheer. Can public large language models help private cross-device federated learning? *ArXiv*, abs/2305.12132, 2023. URL `https://api.semanticscholar.org/CorpusID:258833462`.

Hongyi Wang, Kartik K. Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *ArXiv*, abs/2007.05084, 2020.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *ArXiv*, abs/1804.03209, 2018.

Achintha Wijesinghe, Songyang Zhang, and Zhi Ding. Ps-fedgan: An efficient federated learning framework based on partially shared generative adversarial networks for data privacy. *ArXiv*, abs/2305.11437, 2023. URL `https://api.semanticscholar.org/CorpusID:258823139`.

Zheng Xu, Yanxiang Zhang, Galen Andrew, Christopher Choquette, Peter Kairouz, Brendan Mcmahan, Jesse Rosenstock, and Yuanbo Zhang. Federated learning of gboard language models with differential privacy. In Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 629–639, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-industry.60. URL `https://aclanthology.org/2023.acl-industry.60`.

Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1998–2007. PMLR, 2018.

Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10164–10173, 2022.

Tuo Zhang, Lei Gao, Chaoyang He, Mi Zhang, Bhaskar Krishnamachari, and Salman Avestimehr. Federated learning for the internet of things: Applications, challenges, and opportunities. *IEEE Internet of Things Magazine*, 5:24–29, 2021.

Tuo Zhang, Tiantian Feng, Samiul Alam, Sunwoo Lee, Mi Zhang, Shrikanth S. Narayanan, and Salman Avestimehr. Fedaudio: A federated learning benchmark for audio tasks. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096500.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. *Proceedings of machine learning research*, 139:12878–12889, 2021.

A Appendix

A.1 Experiment Settings

A.1.1 Computing Infrastructure

All experiments are conducted via CPU/GPU simulation. The simulation experiments are performed on two computing servers with ten GPUs. The server is equipped with AMD EPYC 7502 32-Core Processor and 1024G memory. The GPU is NVIDIA RTX A100.

A.1.2 Datasets and Models

CIFAR-10. The CIFAR-10 dataset Krizhevsky (2009) consists of 60,000 32x32 color images in 10 classes. It has 50,000 training images and 10,000 test images. We normalize the images using the mean and standard deviation of the dataset. For evaluation, we use ConvNet Pi et al. (2022), ResNet18 He et al. (2015), and VGG19 Simonyan & Zisserman (2014) models. Following the previous work Pi et al. (2022), the ConvNet has 3 layers with a hidden dimension of 128. The dataset is partitioned using a Dirichlet distribution to emulate a realistic non-iid distribution, following prior work Cho et al. (2022).

CIFAR-100. The CIFAR-100 dataset Krizhevsky (2009) is similar to CIFAR-10 but contains 100 classes, with 600 images per class. We apply the same partitioning method as CIFAR-10. For evaluation, we use ConvNet Pi et al. (2022), ResNet50 He et al. (2015), and VGG19 Simonyan & Zisserman (2014) models. The ConvNet architecture is the same as used for CIFAR-10.

Oxford Flowers 102. The Oxford Flowers 102 Nilsback & Zisserman (2008) (Flowers102) dataset consists of 102 types of flowers, with each type containing between 40 and 258 images. The images exhibit significant variations in scale, angle, and lighting. Some flower categories also have substantial variations within the category and contain several closely related categories. It is divided into training, validation, and test sets. The training and validation sets consist of 10 images per class, totaling 1020 images each. The test set contains the remaining 6149 images, with a minimum of 20 images per class. We resize all images to 224x224 pixels for consistency. For evaluation, we use ConvNet Pi et al. (2022), ResNet18 He et al. (2015), and VGG19 Simonyan & Zisserman (2014) models. We apply the same partitioning method as CIFAR-10. The ConvNet architecture is the same as used for CIFAR-10.

Google Command. The Google Command dataset Warden (2018) comprises 105,829 audio recordings collected from 2,618 speakers. The training set includes recordings from 2,112 speakers, the validation set includes 256 speakers, and the test set includes 250 speakers. It consists of 35 common words from everyday vocabulary, such as "Yes," "No," "Up," and "Down." For evaluation, we use a lightweight model based on related work Zhang et al. (2023) for a 35-class keyword spotting task, where the model consists of two convolution layers followed by one Gated Recurrent Units (GRU) layer and an average pooling layer is connected to the GRU output, which is then fed through two dense layers to generate the predictions. In this work, to pre-process the raw audio data, a sequence of overlapping Hamming windows is applied to the raw speech signal with a time shift of 10 ms. We calculate the discrete Fourier transform (DFT) with a frame length of 1,024 and compute the Mel-spectrogram with a dimension of 128. The Mel-spectrogram is used for training the keyword spotting model. We follow Zhang et al. (2023) for this setup.

ESC-50. The ESC-50 dataset Piczak consists of 2000 environmental audio recordings suitable for environmental sound classification. The dataset contains 5-second-long recordings categorized into 50 semantical classes, with 40 examples per class. These classes are loosely arranged into five major categories: animals, natural soundscapes & water sounds, human & non-speech sounds, interior/domestic sounds, and exterior/urban noises. We employ the same data pre-processing method and model architecture as used in the Google Command dataset.

A.1.3 Hyperparameter Settings

To determine the optimal hyperparameters, we conducted a search within specified ranges. The client learning rate was searched in the range of 1.00E-09 to 1.00E-00, the server learning rate in the range of 1.00E-09 to

1.00E-00, weight decay in the range of 0.1 to 0.9, input batch size in the range of 8 to 256, and epoch number for centralized training in the range of 100 to 500. The hyperparameter settings for the public data-based methods and standard FL methods in Table 2 followed the settings from the previous work Cho et al. (2022). The specific hyperparameter selections for the other experiments are provided below.

Hyperparameter Selection in Table 2. The detailed experiment setups for Table 2 are listed in Table 7, Table 8, Table 9 and Table 10. For the experiments related to FedGen[5] and DynaFed[6], we evaluate them with their official implementation code on GitHub.

Table 7: Experimental setup details of `GPT-FL` with VGG19 in Table 2

		CIFAR-10	**CIFAR-100**	**Flowers102**
Local Epoch		1	1	1
Communication Rounds		500	500	500
Cohort Size		10	10	50
Batch Size		32	32	32
Client Learning Rate	High Data Heterogeneity	1.00E-07	1.00E-06	5.00E-03
	Low Data Heterogeneity	1.00E-07	1.00E-06	5.00E-03
Optimizer		SGD	SGD	SGD
Momentum		0.9	0.9	0.9
Weight Decay		5.00E-04	5.00E-04	5.00E-04

Table 8: Experimental setup details of `GPT-FL` with ConvNet in Table 2

		CIFAR-10	**CIFAR-100**	**Flowers102**
Local Epoch		1	1	1
Communication Rounds		500	500	500
Cohort Size		10	10	50
Batch Size		32	32	32
Client Learning Rate	High Data Heterogeneity	2.00E-07	1.00E-04	1.00E-04
	Low Data Heterogeneity	5.00E-06	1.00E-04	5.00E-03
Optimizer		AdamW	AdamW	SGD
Betas		(0.9, 0.999)	(0.9, 0.999)	N/A
Eps		1.00E-08	1.00E-08	N/A
Weight Decay		5.00E-04	5.00E-04	5.00E-04

Table 9: Experimental setup details of FedGen with ConvNet in Table 2

		CIFAR-10	**CIFAR-100**	**Flowers102**
Local Epoch		1	5	5
Communication Rounds		500	500	500
Cohort Size		10	10	50
Batch Size		32	32	32
Generator Batch Size		32	32	32
Client Learning Rate	High Data Heterogeneity	1.00E-02	1.00E-02	1.00E-02
	Low Data Heterogeneity	1.00E-02	1.00E-02	1.00E-02
Ensemble Learning Rate		1.00E-04	1.00E-04	1.00E-04
Personal Learning Rate		1.00E-02	1.00E-02	1.00E-02
Optimizer		Adam	Adam	Adam
Betas		(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Eps		1.00E-08	1.00E-08	1.00E-08
Weight Decay		1.00E-02	1.00E-02	1.00E-02

Hyperparameter Selection in Table 3 and Table 5. For the centralized training in Table 3 and Table 5, we used the following hyperparameter settings. For image data, the batch size was set to 32, and the optimizer

[5]FedGen: https://github.com/zhuangdizhu/FedGen

[6]DynaFed: https://github.com/pipilurj/DynaFed/tree/main

Table 10: Experimental setup details of DynaFed with ConvNet in Table 2

		CIFAR-10	**CIFAR-100**	**Flowers102**
Local Epoch		1	1	1
Communication Rounds		500	500	500
Cohort Size		10	10	50
Batch Size		32	32	32
Synthetic Images Learning Rate		5.00E-02	5.00E-02	5.00E-02
Distill Interval		1	1	1
Distill Iteration		20	8	20
Distill Step		3000	200	500
Distill Learning Rate		1.00E-04	1.00E-04	1.00E-04
Client Learning Rate	High Data Heterogeneity	1.00E-02	1.00E-02	1.00E-02
	Low Data Heterogeneity	1.00E-02	1.00E-02	1.00E-02
Ensemble Learning Rate		1.00E-04	1.00E-04	1.00E-04
Personal Learning Rate		1.00E-02	1.00E-02	1.00E-02
Optimizer		Adam	Adam	Adam
Betas		(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Eps		1.00E-08	1.00E-08	1.00E-08
Weight Decay		1.00E-02	1.00E-02	1.00E-02

was AdamW with weight decay set to 0.9 and cosine annealing learning rate decay. The initial learning rate was 1.00E-04 for CIFAR-10/CIFAR-100 and 3.00E-04 for Flowers102. For audio data, the batch size was set to 64, and the optimizer was Adam with weight decay set to 1.00E-04. The initial learning rate was 5.00E-05 for both datasets.

For the standard FL training in Table 3 and Table 5, the hyperparameter settings are as follows. For image data, the batch size is set to 32, and SGD is used as the local optimizer with weight decay set to 5.00E-04. When using FedOpt as the server aggregator, Adam is chosen as the server optimizer. Specifically, for the CIFAR-10 dataset, the local learning rate is set to 1.00E-01 with FedAvg as the server aggregator, and for FedOpt as the server aggregator, the local learning rate is set to 1.00E-02 and the server learning rate is set to 1.00E-03. For the CIFAR-100 dataset, the local learning rate is set to 1.00E-01 with FedAvg as the server aggregator, and for FedOpt as the server aggregator, both the local and server learning rates are set to 1.00E-01. For the Flowers102 dataset, the local learning rate is set to 1.00E-01 with FedAvg as the server aggregator, and for FedOpt as the server aggregator, the local learning rate is set to 1.00E-02 and the server learning rate is set to 1.00E-02. For all audio data, the experimental settings strictly follow the FedAudio benchmark Zhang et al. (2023).

For the `GPT-FL` training in Table 3 and Table 5, the hyperparameter settings are as follows. For image data, the batch size is set to 32, and SGD is used as the local optimizer with weight decay set to 5.00E-04. When using FedOpt as the server aggregator, Adam is chosen as the server optimizer. Specifically, for the CIFAR-10 dataset, the local learning rate is set to 5.00E-04 with FedAvg as the server aggregator, and for FedOpt as the server aggregator, the local learning rate is set to 3.00E-04 and the server learning rate is set to 7.00E-04. For the CIFAR-100 dataset, the local learning rate is set to 1.00E-04 with FedAvg as the server aggregator, and for FedOpt as the server aggregator, the local learning rate is set to 5.00E-04 and the server learning rate is set to 1.00E-03. For the Flowers102 dataset, the local learning rate is set to 5.00E-03 with FedAvg as the server aggregator, and for FedOpt as the server aggregator, the local learning rate is set to 1.00E-04 and the server learning rate is set to 1.00E-04. For audio data, the batch size is set to 16, and SGD is used as the local optimizer with weight decay set to 5.00E-04. When using FedOpt as the server aggregator, Adam is chosen as the server optimizer. We set the local learning rate to 5.00E-02 with FedAvg as the server aggregator, and for FedOpt as the server aggregator, the local learning rate is set to 1.00E-03 and the server learning rate is set to 5.00E-04 for both two datasets.

Hyperparameter Selection in Table 6. For the centralized training in Table 6, the hyperparameter selection is follows. For all image data, we set the batch size to 32, and choose AdamW Loshchilov & Hutter (2017) as the optimizer with weight decay equal to 0.9 and cosine annealing learning rate decay. For the CIFAR-10 dataset, we used an initial learning rate of 8.00E-06; for the CIFAR-100 dataset, we used an initial learning rate of 5.00E-06; for the Flowers102 dataset, we used an initial learning rate of 2.00E-05.

For the standard FL training in Table 6, we use the hyperparameter setting as follows. For all image data, we set the batch size to 32, and choose SGD as the local optimizer with weight decay equal to 5.00E-04. With FedOpt as the server aggregator, we choose Adam as the server optimizer. For the CIFAR-10 dataset, we choose the local learning rate as 1.00E-01 with FedAvg as the server aggregator and choose the local learning rate as 1.00E-03 and the server learning rate as 1.00E-03 with FedOpt as the server aggregator. For the CIFAR-100 dataset, we choose the local learning rate as 1.00E-02 with FedAvg as the server aggregator and choose the local learning rate as 5.00E-03 and the server learning rate as 7.00E-03 with FedOpt as the server aggregator. For the Flowers102 dataset, we choose the local learning rate as 1.00E-02 with FedAvg as the server aggregator and choose the local learning rate as 1.00E-04 and the server learning rate as 5.00E-04 with FedOpt as the server aggregator.

For `GPT-FL` training in Table 6, we use the hyperparameter setting as follows. For all image data, we set the batch size to 32, and choose SGD as the local optimizer with weight decay equal to 5.00E-04. With FedOpt as the server aggregator, we choose Adam as the server optimizer. For CIFAR-10 dataset, we choose the local learning rate as 1.00E-07 with FedAvg as the server aggregator and choose the local learning rate as 1.00E-07 and the server learning rate as 1.00E-05 with FedOpt as the server aggregator. For CIFAR-100 dataset, we choose the local learning rate as 1.00E-04 with FedAvg as the server aggregator and choose the local learning rate as 1.00E-04 and the server learning rate as 1.00E-05 with FedOpt as server aggregator. For Flowers102 dataset, we choose the local learning rate as 1.00E-02 with FedAvg as the server aggregator and choose the local learning rate as 1.00E-04 and the server learning rate as 1.00E-04 with FedOpt as the server aggregator.

A.2 Additional Ablation Analysis

A.3 Harmonization with Other FL Strategy

Because `GPT-FL` does not alter the structure of FL framework, it is complementary to other FL training methods as we shown in Table 1 and Table 2. Table 11 shows the performance evaluation of `GPT-FL` + DynaFed with CIFAR-10 and CIFAR-100. We use the same experiment setup as we used in Table 2. The results confirm that `GPT-FL` can improve upon existing FL methods while maintaining their original structures.

Table 11: Accuracy performance of existing FL baseline on top of `GPT-FL`. "ΔMetric" represents the accuracy increment by `GPT-FL` on top of DynaFed.

Dataset	**DynaFed**	**DynaFed + `GPT-FL`**	**ΔMetric**
CIFAR-10	71.59%	**74.20%**	**2.61%**
CIFAR-100	36.08%	**39.23%**	**3.15%**

A.4 Impact of Label Distribution in Synthetic Data

In this section, we investigate the impact of distribution drift within synthetic data on GPT-FL's performance. Initially, our experiments generated an equal number of data samples for each label, resulting in an IID distribution. In the following experiment, we partitioned the synthetic CIFAR-10 data into 10 distinct silos using a Dirichlet distribution, where varying values of the parameter alpha were employed to simulate different degrees of data distribution non-IIDness—from more IID (α=100) to highly non-IID (α=1) scenarios. For each configuration, we selected a single silo of synthetic data for downstream model training and documented the data distribution characteristics utilized in the analysis. The distribution of the synthetic CIFAR-10 labels with various degrees we used for experiments is shown in Figure 10. We adopt the same experiment setup as we used in Table 5. The experiment results are shown in Table 12.

These results demonstrate a clear relationship between the non-IIDness of label distributions and model performance, with the GPT-FL framework showing robustness even in the face of significant distribution drift. Notably, the federated learning fine-tuning step consistently enhances performance, particularly when downstream models are trained with highly skewed data. This finding substantiates the discussions in the

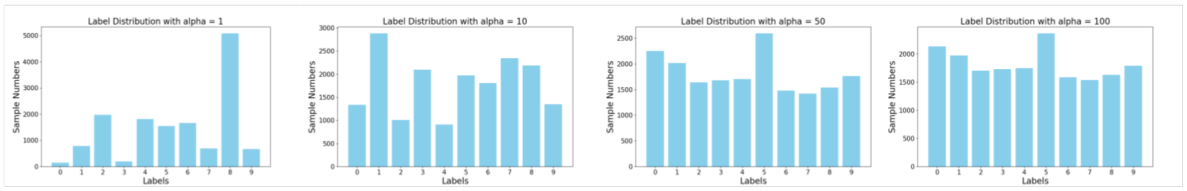

Figure 10: Distribution of synthetic CIFAR-10 labels across various degrees of Non-IIDness in experiments documented in Table 12.

Table 12: Accuracy performance among various label distribution of synthetic data.

	$\alpha = 1$	$\alpha = 10$	$\alpha = 50$	$\alpha = 100$
Synthetic Data Only	56.28%	65.04%	69.04%	69.68%
GPT-FL w/ FedAvg	70.02%	75.31%	77.80%	80.17%
GPT-FL w/ FedOpt	68.55%	76.30%	78.26%	77.11%

second part of Section 4.2 of our manuscript and highlights the efficacy of the GPT-FL approach under diverse data distribution conditions.

It should be noted that the highest performance in Table 12 is slightly lower than the performance in Table 5. This discrepancy primarily stems from utilizing approximately ten times less synthetic data in the experiments of Table 12 compared to those in Table 5. This observation corroborates our discussion on the influence of synthetic data volume on performance outcomes, as illustrated in Figure 7.

Furthermore, it's important to emphasize that in the practical deployment of the GPT-FL algorithm, the server is responsible for managing the nuances of data generation—including the quantity and distribution of data—based on coarse label names collected from clients. This mechanism significantly mitigates the likelihood of training downstream models on highly non-IID distributed datasets, ensuring a more controlled and effective learning environment.