DynaFed: Tackling Client Data Heterogeneity with Global Dynamics

Renjie Pi[1]* Weizhong Zhang[1]* Yueqi Xie[1]* Jiahui Gao[2] Xiaoyu Wang[1]
Sunghun Kim[1] Qifeng Chen[1]
[1]The Hong Kong University of Science and Technology
[2]The University of Hong Kong

Abstract

The Federated Learning (FL) paradigm is known to face challenges under heterogeneous client data. Local training on non-iid distributed data results in deflected local optimum, which causes the client models drift further away from each other and degrades the aggregated global model's performance. A natural solution is to gather all client data onto the server, such that the server has a global view of the entire data distribution. Unfortunately, this reduces to regular training, which compromises clients' privacy and conflicts with the purpose of FL. In this paper, we put forth an idea to collect and leverage global knowledge on the server without hindering data privacy. We unearth such knowledge from the dynamics of the global model's trajectory. Specifically, we first reserve a short trajectory of global model snapshots on the server. Then, we synthesize a small pseudo dataset such that the model trained on it mimics the dynamics of the reserved global model trajectory. Afterward, the synthesized data is used to help aggregate the deflected clients into the global model. We name our method DynaFed, *which enjoys the following advantages: 1) we do not rely on any external on-server dataset, which requires no additional cost for data collection; 2) the pseudo data can be synthesized in early communication rounds, which enables* DynaFed *to take effect early for boosting the convergence and stabilizing training; 3) the pseudo data only needs to be synthesized once and can be directly utilized on the server to help aggregation in subsequent rounds. Experiments across extensive benchmarks are conducted to showcase the effectiveness of* DynaFed. *We also provide insights and understanding of the underlying mechanism of our method.*

1. Introduction

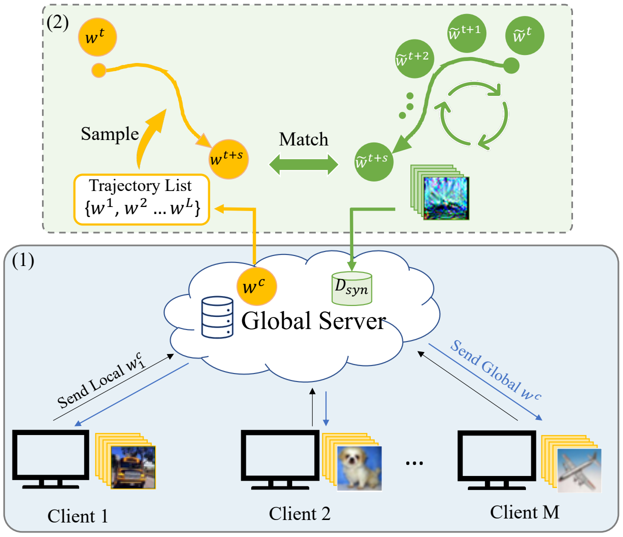


Figure 1. Illustration of DynaFed. Firstly, we run the standard FedAvg for L communication rounds and save the checkpoints to form a trajectory of the global model at the server. Then, we synthesize a pseudo dataset $\mathcal{D}_{\text{syn}}$, with which the network can be trained to mimic the dynamics of the global trajectory. In this way, the knowledge that captures the essential information about the global data distribution is transferred from the global dynamics to $\mathcal{D}_{\text{syn}}$. Afterward, $\mathcal{D}_{\text{syn}}$ is adopted to help aggregate the deflected clients into the global model at the server.

Federated learning (FL) has become a popular distributed training paradigm to alleviate the server's computational burden and preserve clients' data privacy [3, 24, 34, 48]. In the FL paradigm, the clients only have access to their private datasets, while the server is responsible for aggregating the clients' updates into a global model. The most prevalent approaches in FL are based on local-SGD [37] (also referred to as FedAvg), where the client model is updated locally for multiple steps before being sent and merged on the server. Such approaches save communication costs and perform well given the client data are *iid*-distributed. However, in real-world applications such as healthcare [23, 24, 39, 40] and bio-metrics [2], the client data usually demonstrates heterogeneity (highly *non-iid*), which deflects the local optimum from the global optimum [25, 35, 55] and makes the locally trained clients biased. Therefore, naively averaging the client

*Joint first authors

models results in slow convergence and performance drop.

To alleviate the difficulty of training under heterogeneity, a few lines of work have been frequently discussed. The first line attempts to modify the local training process, including imposing regularization on the client models [1, 25, 32, 34] and data sharing or augmentation [38, 41, 51, 55]. However, these solutions have a high requirement for the server's control of local clients. An orthogonal line focuses on refining the global model in the server aggregation process [6, 36, 43, 47, 49]. The majority of these methods typically require a large external dataset on the server, then use it to align the outputs of the global model with that of the client ensemble [6, 15, 36, 47]. Unfortunately, such a large-scale task-related dataset is often hard to acquire in reality. To circumvent this limitation, a few data-free knowledge distillation (KD) approaches are recently proposed [53,56]. These methods attempt to transfer the knowledge contained in the global model to a generator, which is subsequently leveraged to produce pseudo data to either help local training at the clients [56] or finetune the global model at the server [53]. It is clear that these methods require a global model with reasonable performance to ensure the generation of helpful pseudo data. However, such requirement is hard to achieve in practice since the global model often performs poorly under heterogeneity, especially in the early rounds of training. Other solutions include personalized FL [16, 29, 33, 42], simlarity clustering [4, 10, 13], meta learning [22, 31], etc.

Even though the above-mentioned works propose techniques to alleviate the challenges posed by heterogeneity to some extent, they do not tackle the issue from its root cause: the data is unevenly scattered at different clients and is kept locally due to privacy concerns, which is also the main obstacle for training an accurate global model. Ideally, imagine if we can collect all the client data to the server, then training can be directly conducted at the server, and the heterogeneity issue no longer exists. However, this reduces to regular training and conflicts with the original purpose of FL to protect client privacy. We then raise a natural question: *is it possible to derive the essential information about the global data distribution on the server to help training without compromising client privacy?*

Despite the global model typically performing poorly due to heterogeneity, the changes in its parameters are steered jointly by the data scattered at different clients. Therefore, the update dynamics of the global model contain knowledge about global data distribution. Driven by this intuition, we propose DYNAFED to explicitly unearth such knowledge hidden in the global dynamics and transfer it to a pseudo dataset $\mathcal{D}_{\text{syn}}$ at the server. $\mathcal{D}_{\text{syn}}$ can then approximate the global data distribution on the server to aid aggregation. More specifically, inspired by recent works in dataset condensation [5,44,54], we formulate the data synthesis process into a learning problem, which minimizes the distance between the trajectory trained with $\mathcal{D}_{\text{syn}}$ and the global model trajectory derived with $\mathcal{D}$. Fine-tuning the aggregated global model with $\mathcal{D}_{\text{syn}}$ effectively alleviates the performance degradation caused by deflected clients. An appealing feature of our DYNAFED is that the data can be synthesized using just the global model's trajectory of the first few rounds, which enables $\mathcal{D}_{\text{syn}}$ to take effect and help aggregation from early rounds. In addition, the synthesizing process only needs to be conducted once in practice, after which the derived $\mathcal{D}_{\text{syn}}$ can be directly applied in subsequent rounds to help aggregate the deviated client models.

Notably, our framework can be readily applied to the majority of FL approaches, since we rely on only the history of the global model's parameters for synthesizing $\mathcal{D}_{\text{syn}}$, which is available in the conventional setting of FedAvg-based methods. Furthermore, because we extract global knowledge using the global dynamics, rather than any client-specific information as in [12,17,19,21,50], the derived $\mathcal{D}_{\text{syn}}$ comprises of information mixed with the entire global data distribution, thus prevents leakage of client privacy.

Our DYNAFED possesses the following advantages compared with previous approaches: 1) It leverages the knowledge of the global data distribution to alleviate the aggregation bias of the global model without depending on any external datasets; 2) DYNAFED is able to generate informative data in the early rounds of federated learning, which significantly helps convergence and stabilizes training in subsequent rounds; 3) Compared with [53, 56], which need to keep updating the generator throughout all communication rounds, the data synthesis process in our method only needs to be done once, which reduces the computational overhead. In summary, we make the following contributions:

- We propose a practical approach named DYNAFED for tackling the heterogeneity problem, which extracts and exploits the hidden information from the global model's trajectory. In this way, the server can access the essential knowledge of the global data distribution to reinforce aggregation;
- We experimentally show the synthesized dataset helps stabilize training, boost convergence and achieve significant performance improvement under heterogeneity;
- We provide insights and detailed analysis into the working mechanisms of the proposed DYNAFED both experimentally and theoretically.

2. Related Work

Regularization-based Methods FedAvg [37] is the most widely used technique in FL, which periodically aggregates the local models to the global model in each communication round. FedProx [34] proposes to impose a proximal term during local training, such that the local model does not drift too far from its global initialization; Scaffold [25] introduces a control variate and variance reduction to alleviate the drift of local training. Moon [32] proposes to leverage the similarity between model representations to regularize the client local training. FedDyn [1] introduces the linear and quadratic penalty terms to correct the clients' objective during local training. These methods are orthogonal to our approach and can be jointly used.

Data-Dependent Knowledge Distillation Methods This line of work attempts to distill client ensemble knowledge into the global model. [36] proposes to use an unlabeled external dataset on the server to match the global model's outputs with that of the client ensemble. On top of this, [6] further proposes to sample and combine higher-quality client models via a Bayesian model ensemble. Subsequently, some advanced techniques, such as pre-training [15] and weighted consensus distillation scheme [8] are proposed. These methods typically require a large amount of data following similar distribution as the task data, which is usually hard to acquire in practice. Besides, these methods need to conduct KD with a large dataset in every communication round, which introduces prohibitive computational overhead.

Data-free Knowledge Distillation Methods Recently, a few works have proposed to perform KD in a data-free manner by synthesizing data with generative models [53,56]. [56] proposes to train a lightweight generator on the server, which produces a feature embedding conditioned on the class index. The generator is then sent to the clients to regularize local training. [53] trains a class conditional GAN [14] on the server, where the global model acts as the discriminator. The pseudo data is then used to finetune the global model. These methods all depend on the global model for training the generator. Unfortunately, the model performance is often poor under high data heterogeneity, which makes the quality of the pseudo data questionable. On the other hand, due to the use of update dynamics of the global model rather than an individual model, our method can synthesize high-quality data containing rich global information even if the global model performs poorly.

3. Preliminary

Federated Learning. Suppose we have M clients in a federated learning system. For each client $m \in [M]$, a private dataset $\mathcal{D}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{D}_m|}$ is kept locally. The overall optimization goal of the FL system is to jointly train a global model which performs well on the combination of local data, denoted as $\mathcal{D} = \cup_{m=1}^{M} \mathcal{D}_m$, where $\mathcal{D}$ is from a global distribution. Let $\alpha_m = \frac{|\mathcal{D}_m|}{|\mathcal{D}|}$ denote the portion of data samples on client m. Let $\boldsymbol{w} \in \mathbb{R}^d$ denote the model parameter to optimize, and $\mathcal{L}_m(\boldsymbol{w}, \mathcal{D}_m) = \frac{1}{|\mathcal{D}_m|} \sum_{\xi \in \mathcal{D}_m} \ell(\boldsymbol{w}, \xi)$ denote the empirical risk with the loss function $\ell(\cdot,\cdot)$. The optimization problem of a generic FL system can be formulated as follows:

$$\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \mathcal{D}) = \sum_{m=1}^{M} \alpha_m \mathcal{L}_m(\boldsymbol{w}, \mathcal{D}_m). \tag{1}$$

FedAvg. The main-stream solutions of Eqn. 1 rely on local-SGD to reduce the communication cost of transferring gradients. FedAvg [37] is the most prevalent approach, which uses a weighted average to aggregate the locally trained models into a global model in each communication round. Typically, not all the clients participate in every communication round. Suppose $\mathcal{M}^c \subset [M]$ is the set of the participated clients in the c-th round, and $P^c = \sum_{m \in \mathcal{M}^c} \alpha_m$. The aggregation process for the global model $\boldsymbol{w}^{c+1}$ at the end of c-th communication round can be formulated as:

$$\boldsymbol{w}^{c+1} = \frac{1}{P^c} \sum_{m \in \mathcal{M}^c} \alpha_m \boldsymbol{w}_m^c, \tag{2}$$

where $\boldsymbol{w}_m^c$ denotes the client m's locally trained model. After the aggregation, the updated global model $\boldsymbol{w}^{c+1}$ is then distributed to and client and utilized to initiate the $c+1$-th round of the training.

4. Proposed Method

In this section, we introduce our proposed method DYNAFED. The overall framework is illustrated in Figure 1. Firstly, we collect a trajectory of the global model's updates in the early phase of federated training, with which we construct a synthetic dataset $\mathcal{D}_{\text{syn}} = \{\boldsymbol{X}, \boldsymbol{y}\}$ on the server side. Then, we utilize $\mathcal{D}_{\text{syn}}$ to aid the server-side aggregation, which effectively helps recover the performance drop caused by deflected local models.

4.1. Acquiring Global Knowledge by Data Synthesis

Our goal is to construct a pseudo dataset $\mathcal{D}_{\text{syn}}$, which achieves a similar effect during training as the global dataset $\mathcal{D}$. In other words, the trajectory of a network trained with $\mathcal{D}_{\text{syn}}$ should have similar dynamics with the trajectory trained with $\mathcal{D}$. To be precise, we denote the trajectory trained on $\mathcal{D}$ as a sequence $\{\boldsymbol{w}^{\text{t}}\}_{t=0}^{L}$, where L is the length of the trajectory. In order to reduce the number of unrolling steps during optimization, we align the trajectory in a segment-by-segment manner. Without loss of generality, we consider a segment from $\boldsymbol{w}^t$ to $\boldsymbol{w}^{t+s}$, then the problem becomes the following: starting from $\boldsymbol{w}^t$, the network should arrive at a place close to $\boldsymbol{w}^{t+s}$ after being trained for s steps on $\mathcal{D}_{\text{syn}}$. We further formulate the data synthesis task into a learning problem as follows:

$$\min_{\boldsymbol{X}, \boldsymbol{y}} \mathbb{E}_{t \sim U(1, L-s)} [d(\tilde{\boldsymbol{w}}, \boldsymbol{w}^{\text{t+s}})] \tag{3}$$

$$s.t. \; \tilde{\boldsymbol{w}} = \mathcal{A}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}^{\text{t}}, s) \tag{4}$$

In the inner loop expressed by Eqn.4, we run the trainer $\mathcal{A}(\cdot)$ that trains a neural network initialized from $\boldsymbol{w}^{\text{t}}$ on the synthetic dataset $\mathcal{D}_{\text{syn}} = \{\boldsymbol{X}, \boldsymbol{y}\}$ for s steps, which arrives at $\tilde{\boldsymbol{w}}$. In the outer loop, we minimize the distance between $\boldsymbol{w}^{\text{t+s}}$ and $\tilde{\boldsymbol{w}}$, denoted as $d(\tilde{\boldsymbol{w}}, \boldsymbol{w}^{\text{t+s}})$, by optimizing over $(\boldsymbol{X}, \boldsymbol{y})$. The expectation of the uniform distribution U is adopted to take into account all the segments along the trajectory. $d(\cdot, \cdot)$ is a general distance measure, which can take the form of euclidean distance or cosine distance, etc. During the optimization, we treat both $(\boldsymbol{X}, \boldsymbol{y})$ to be learnable variables, where $\boldsymbol{X}$ is initialized with random noise, and $\boldsymbol{y}$ is initialized with equal probabilities over all labels. The detailed algorithm is shown in Algorithm 1.

Algorithm 1 DataSyn

```
Require: Global trajectory $\{w^c\}_1^L$, learning rate $\eta$, training rounds $N$, inner steps $s'$.
  Randomly initialize $\boldsymbol{X}_0$ and pair them with $\boldsymbol{y}$ to form the synthetic dataset.
  for training iteration $n = 1, 2 \ldots N$ do
      Sample $t \sim U(1, L - s)$, then take $\boldsymbol{w}^{\text{t}}$ and $\boldsymbol{w}^{\text{t+s}}$ from the trajectory.
      Get the trained paramters $\tilde{\boldsymbol{w}} = \mathcal{A}(\boldsymbol{X}_n, \boldsymbol{y}_n, \boldsymbol{w}^{\text{t}}, s')$
      Calculate the distance $d(\tilde{\boldsymbol{w}}, \boldsymbol{w}^{\text{t+s}})$ and obtain gradient w.r.t $\boldsymbol{X}_n$ and $\boldsymbol{y}_n$ as $\nabla_{\boldsymbol{X}_n} d(\tilde{\boldsymbol{w}}, \boldsymbol{w}^{\text{t+s}})$ and $\nabla_{\boldsymbol{y}_n} d(\tilde{\boldsymbol{w}}, \boldsymbol{w}^{\text{t+s}})$.
      Update $\boldsymbol{X}_n$ and $\boldsymbol{y}_n$ using gradient descent $\boldsymbol{X}_{n+1} = \boldsymbol{X}_n - \eta \nabla_{\boldsymbol{X}_n} d(\tilde{\boldsymbol{w}}, \boldsymbol{w}^{\text{t+s}})$, $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n - \eta \nabla_{\boldsymbol{y}_n} d(\tilde{\boldsymbol{w}}, \boldsymbol{w}^{\text{t+s}})$
  end for
  Output: Optimized synthetic data $\mathcal{D}_{\text{syn}}$.
```

通过梯度优化的方式生成数据，目标是模型参数与更新历史相近

Note that since the size of $\mathcal{D}_{\text{syn}}$ is much smaller than $\mathcal{D}$, the effective step size should have a different scale. Therefore, in Equation 4, we may increase the number of steps from s to s' to account for this scale mismatch.

In this way, the knowledge hidden in the dynamics of the global model can be transferred to $\mathcal{D}_{\text{syn}}$, which then acts as an approximation of the global data distribution to aid the server-side aggregation.

4.2. Overall Algorithm of DynaFed

In this section, we present our DYNAFED that integrates the data synthesis process into the federated learning framework.

Firstly, to construct the global trajectory on the server, we collect the global model's checkpoints from the first few communication rounds of FedAvg mainly considering the following two factors: 1) collecting a long trajectory induces prohibitive cost due to the expensive global communication, 2) the change in global model's parameters becomes insignificant during late rounds, which makes it difficult to extract knowledge from the dynamics.

Note that although $\mathcal{D}_{\text{syn}}$ contains rich global knowledge, it is not sufficient to replace the global data distribution $\mathcal{D}$ due to the following reasons: 1) the objective in Eqn.3 can not be ideally solved due to the two-level optimization procedure, 2) to make the scale of the optimization problem acceptable, the size of $\mathcal{D}_{\text{syn}}$ can not be as large as $\mathcal{D}$, 3) there exists an inconsistency between the trainer $\mathcal{A}(\cdot)$ and the one that produces the global trajectory, i.e., FedAvg. Therefore, instead of simply conduct regular training with $\mathcal{D}_{\text{syn}}$, we leverage such $\mathcal{D}_{\text{syn}}$ to help aggregation by finetuning the global model on the server side.

Algorithm 2 DYNAFED

```
Require: Client data $\mathcal{D}_m = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{D}_m|}$, global parameters $\boldsymbol{w}$, client parameters $\{\boldsymbol{w}_m\}_{m=1}^{|\mathcal{M}|}$. Total communication rounds $C$, local steps $T$, data learning rate $\eta$, data training rounds $N$, trajectory length $L$, inner steps for data synthesis $s'$.
  for communication round $c = 1, 2 \ldots C$ do
      Sample active clients $\mathcal{M}^c$ randomly. Distribute $\boldsymbol{w}^c$ to active clients and initialize their parameters.
      for all users $m \in \mathcal{M}^c$ do
          $\{\boldsymbol{w}_m^c\}_{m=1}^{|\mathcal{M}^c|} \leftarrow$ LocalTrain$(\boldsymbol{w}^c, T)$
      end for
      Clients send updated $\{\boldsymbol{w}_m^c\}_{m=1}^{|\mathcal{M}^c|}$ to server.
      $\boldsymbol{w}^{c+1} \leftarrow \frac{1}{|\mathcal{M}^c|} \sum_{m \in \mathcal{M}^c} \boldsymbol{w}_m^c$        Model Averaging
      if $c = L$ then
          $\mathcal{D}_{\text{syn}} \leftarrow$ DataSyn$(\{\boldsymbol{w}^c\}_{c=1}^L, \eta, s', N)$

                                                          在第L轮生成数据，基于历史global model更新路径
      else if $c > L$ then
          $\boldsymbol{w}^{c+1} \leftarrow$ Finetune$(\mathcal{D}_{\text{syn}}, \boldsymbol{w}^{c+1})$
      else
          Add $\boldsymbol{w}^{c+1}$ into trajectory list.
      end if                  之后都基于生成数据finetune global model
  end for
```

The rundown of the entire algorithm is presented in Algorithm 2. Firstly, the global model's trajectory in the earliest L rounds is collected and stored on the server. Then, the server executes the data synthesis procedure to generate the pseudo data $\mathcal{D}_{\text{syn}}$. In all subsequent rounds, $\mathcal{D}_{\text{syn}}$ is leveraged on the server to help reduce the negative impact of deflected client models by finetuning the global model.

Our method enjoys the following appealing properties:

- In contrast to data-dependent KD methods [6, 36], DYNAFED extracts the knowledge of the global data distribution from the global model trajectory, which does not depend on any external datasets;
- Compared with data-free KD methods [53, 56], DYNAFED is able to synthesize informative pseudo data in the early rounds of federated learning without requiring the global model to be well trained, which significantly helps convergence and stabilizes training;
- The data synthesis only needs to be conducted once. Besides, since only a few samples are synthesized, refining the global model on the server requires negligible time.

Remark 1. *We emphasize that our* DYNAFED *does not raise privacy concerns given following reasons: 1) we rely on only the trajectory of the global model's parameters, which is available in the conventional setting of all FedAvg-based methods; 2) we keep $\mathcal{D}_{syn}$ at the server rather than sending it to the clients; 3) we extract global knowledge using the global dynamics, rather than any client-specific information as in [12, 17, 19, 21, 50], the derived $\mathcal{D}_{\text{syn}}$ comprises of information mixed with the entire global data distribution, thus prevents leakage of client privacy; 4) our* DYNAFED *shares a similar flavor as dataset condensation (DC) methods, which aims to generate informative pseudo data containing global knowledge, rather than real-looking data. The privacy-preserving ability of DC was also discussed in previous work [11].*

5. Theoretical Analysis

In this section, we present some insights to understand our data synthesis process from the neural tangent kernel (NTK) theory and also give the convergence results for DYNAFED. Detailed proofs are given in the appendix.

In our data synthesis process, we align the segments (e.g., from t to $t+s$) of the trajectories trained on $\mathcal{D}$ and $\mathcal{D}_{\text{syn}}$ for any t. Essentially, those segments are the cumulative gradients calculated using $\mathcal{D}$ and $\mathcal{D}_{\text{syn}}$, which approximate to $\nabla\mathcal{L}(\boldsymbol{w}_t, \mathcal{D}_{\text{syn}})\Delta t$ and $\nabla\mathcal{L}(\boldsymbol{w}^t, \mathcal{D})\Delta t$. Therefore, we can expect $\nabla\mathcal{L}(\boldsymbol{w}, \mathcal{D}_{\text{syn}})$ would be close to $\nabla\mathcal{L}(\boldsymbol{w}, \mathcal{D})$, which is verified by our experimental result (left of Figure.6). Given a sample $\boldsymbol{x}$, if continuous-time gradient descent is adopted as the training solver, the dynamics of neural network function trained on $\mathcal{D}_{\text{syn}}$ and $\mathcal{D}$ take the forms of

$$\begin{cases} \frac{df(\boldsymbol{x},\boldsymbol{w}^t)}{dt} = \nabla_{\boldsymbol{w}^t} f(\boldsymbol{x},\boldsymbol{w}^t)^\top \nabla\mathcal{L}(\boldsymbol{w}^t, \mathcal{D}_{\text{syn}}), \\ \frac{df(\boldsymbol{x},\boldsymbol{w}^t)}{dt} = \nabla_{\boldsymbol{w}^t} f(\boldsymbol{x},\boldsymbol{w}^t)^\top \nabla\mathcal{L}(\boldsymbol{w}^t, \mathcal{D}). \end{cases} \tag{5}$$

which is close to an ordinary differential equation according to the NTK theory [20]. Note that the right-hand sides of the two equations in (5) are close if $\nabla\mathcal{L}(\boldsymbol{w}, \mathcal{D}_{\text{syn}}) \approx \nabla\mathcal{L}(\boldsymbol{w}, \mathcal{D})$. In this case, the following lemma about the continuous dependence of differentiable equation shows that the neural functions $f(\boldsymbol{x}, \boldsymbol{w}^t)$ learned with pseudo data $\mathcal{D}_{\text{syn}}$ are similar to that learned with real global data $\mathcal{D}$ during the whole training process. This further indicates that $\mathcal{D}_{\text{syn}}$ achieves similar effect as $\mathcal{D}$ during training.

Lemma 1 (Continuous Dependence [45])**.** *Suppose $\tilde{F}(t, f)$ and $F(t, f)$ are two continuous functions in a region G satisfying*

$$|\tilde{F}(t, f) - F(t, f)| \le \epsilon, \forall (t, f) \in G.$$

Further, we assume $F(t, f)$ satisfy the L_F-Lipschitz condition w.r.t., f. Let $\tilde{f}(t)$ and $f(t)$ be the solutions of initial problems,

$$\frac{d\tilde{f}}{dt} = \tilde{F}(t, \tilde{f}) \text{ and } \frac{df}{dt} = F(t, f),$$

with $\tilde{f}(t_0) = f_0$ and $f(t_0) = f_0$. Then, in a common region $|t - t_0| \le \alpha$, we have the following estimation:

$$|\tilde{f}(t) - f(t)| \le \frac{\epsilon}{L_F}\left(e^{\alpha L_F} - 1\right).$$

To analyze the convergence of DYNAFED, we need to define some additional notations and rewrite our method as follows. Suppose with the $\mathcal{D}_{\text{syn}}$ generated from the data synthesis process, in DYNAFED we run SGD for τ_1 and τ_2 iterations in each local training round and finetuning process, respectively. Let the sets $\mathcal{I}$ and $\mathcal{J}$ be

$$\begin{aligned} \mathcal{I} &= \{t | t = k(\tau_1 + \tau_2) + c, k = 0, 1, 2, \ldots, c \in [\tau_1]\}, \\ \mathcal{J} &= \{t | t = k(\tau_1 + \tau_2) + \tau_1, k = 0, 1, 2, \ldots\}, \end{aligned} \tag{6}$$

where $[\tau_1] = \{0, 1, \ldots, \tau_1 - 1\}$. Therefore, when $t \in \mathcal{I}$, we perform local training, while when $t \notin \mathcal{I}$, we conduct finetuning. $\mathcal{J}$ denotes the time index for aggregation. The detailed steps of DYNAFED can be rewritten as

$$\boldsymbol{v}_{t+1}^m = \begin{cases} \boldsymbol{w}_m^t - \eta_t \nabla\ell(\boldsymbol{w}_m^t, \xi_m^t), & \text{if } t \in \mathcal{I} \\ \boldsymbol{w}_m^t - \eta_t \nabla\mathcal{L}(\boldsymbol{w}_m^t, \mathcal{D}_{\text{syn}}), & \text{if } t \notin \mathcal{I} \end{cases},$$

$$\boldsymbol{w}_m^{t+1} = \begin{cases} \boldsymbol{v}_m^{t+1}, & \text{if } t+1 \notin \mathcal{J} \\ \sum_{m=1}^M \alpha_m \boldsymbol{v}_m^{t+1}, & \text{if } t+1 \in \mathcal{J} \end{cases},$$

where $\xi_m^t \sim \mathcal{D}_m$. Based on the notations, we define a sequence:

$$\bar{\boldsymbol{w}}^t = \sum_{m=1}^M \alpha_m \boldsymbol{w}_m^t.$$

Hence, our algorithm is an integration of FedAvg and a biased GD. For $\bar{\boldsymbol{w}}^t$, note that $\bar{\boldsymbol{w}}^t = \boldsymbol{w}_1^t = \cdots = \boldsymbol{w}_M^t$ in the finetuning process and we have the following convergence results:

Theorem 1 (Convergence)**.** *For $\tilde{L}$-smooth, μ-strongly convex loss functions $\ell(\cdot, \cdot)$. We assume $\|\nabla\mathcal{L}(\boldsymbol{w}, \mathcal{D}_{syn}) - \nabla\mathcal{L}(\boldsymbol{w}, \mathcal{D})\| \le \delta\|\nabla\mathcal{L}(\boldsymbol{w}, \mathcal{D})\| + \epsilon$ holds with two small non-negative scalars δ and ϵ. Let $\eta_t = \frac{c}{t}$ for a proper constant c. Then,* DYNAFED *satisfies*

$$\mathbb{E}\mathcal{L}(\bar{\boldsymbol{w}}^T, \mathcal{D}) - \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}) \le \frac{C}{T}, \tag{7}$$

where $\mathbf{w}^*$ is the minimum of $\mathcal{L}(\boldsymbol{w}, \mathcal{D})$ and C is a constant, whose detailed formula is given in the appendix.

More detailed discussions about the convergence result are given in the Appendix.

6. Experiments

Benchmark Datasets and Experimental Settings. We conduct experiments over four commonly used datasets: FashionMNIST [46], CIFAR10 [27], CINIC10 [9] and CIFAR100 [27]. Among them, FashionMNIST is a dataset containing grey-scale images of fashion products. CIFAR10 is an image classification dataset containing daily objects. CINIC10 is a dataset combining CIFAR10 and samples from similar classes that are downsampled from ImageNet [28]. These three datasets contain 10 classes. CIFAR100 contains the same data as CIFAR10, but categorizes the data into 100 classes. For each dataset, we mainly conduct experiments with heterogeneous client data distribution. We follow prior work [6, 36] to use Dirichlet distribution for simulating the non-IID data distribution, where the degree of heterogeneity is defined by α, smaller α value corresponds to more severe heterogeneity.

Baseline Methods We consider various state-of-the-art solutions against non-IID data distribution in the context of federated learning. Specifically, we compare with the following approaches 1) the vanilla aggregation strategy FedAVG [37]; 2) regularization-based strategies FedProx [34], Scaffold [25]; 3) data-dependent knowledge distillation strategies that need external dataset FedDF [36] and FedBE [6], ABAVG [47]; (4) data sharing [52] or data-free knowledge distillation [56] methods. Note that we do not compare with [53] since the code is not published. Please refer to Appendix for detailed settings of the baseline methods.

Configurations Unless specified otherwise, we follow [7, 15, 48] and adopt the following default configurations throughout the experiments: we run 200 global communication rounds with local epoch set to 1. There are 80 clients in total, and the participation ratio in each round is set to 40%. Experiments using other participation ratios are in the Appendix. We report the global model's average performance in the last five rounds evaluated using the test split of the datasets. For the construction of global trajectory,

Method	$\alpha = 0.01$ FMNIST	 CIFAR10	 CINIC10	$\alpha = 0.04$ FMNIST	 CIFAR10	 CINIC10	$\alpha = 0.16$ FMNIST	 CIFAR10	 CINIC10
FedAVG	74.50±1.32	39.30±3.42	31.60±5.50	81.74±1.98	51.19±2.85	45.35±3.00	89.54±1.51	69.74±1.29	55.40±2.05
FedProx	76.88±1.83	42.13±3.64	32.56±4.59	83.06±2.53	58.93±2.14	46.30±2.87	89.53±1.13	70.20±0.74	57.78±2.08
Scaffold	77.92±0.87	42.04±2.26	34.90±3.34	82.25±1.35	54.23±1.90	46.22±2.18	88.54±0.32	68.57±0.91	54.30±0.83
FedDF*	72.36±2.08	39.73±3.98	31.97±4.31	81.65±0.97	54.20±2.93	45.79±2.95	89.70±0.97	70.71±0.94	55.78±1.02
FedBE*	72.33±1.79	38.36±3.74	32.04±3.73	81.31±1.25	53.49±2.36	45.50±2.88	89.62±0.75	70.23±0.76	55.42±1.37
ABAVG*	75.98±1.99	39.95±1.37	32.75±4.18	84.88±1.84	57.25±3.42	47.39±3.36	89.53±1.12	70.55±2.41	56.02±1.49
FedGen†	75.59±1.12	40.19±2.14	32.59±3.25	81.46±1.08	56.60±1.29	45.57±2.70	89.95±0.89	70.89±0.54	55.34±1.13
FedMix†	81.34±0.68	50.48±1.23	37.15±1.81	84.23±0.50	62.77±1.07	50.22±1.41	89.05±0.24	70.33±0.55	56.74±0.45
DynaFed†	**87.52±0.15**	**65.53±0.34**	**48.04±0.70**	**89.45±0.11**	**70.07±0.12**	**55.43±0.24**	**91.35±0.07**	**74.69±0.14**	**59.80±0.10**

Table 1. Comparison of test performances achieved by different FL methods with different degrees of data heterogeneity α across multiple datasets. We report the mean test accuracy of last five communication rounds. *Methods assume the availability of proxy data. † Methods are based on data sharing or generation. We observe that our approach outperforms other methods by a large margin, and its advantage is more prominent on more challenging datasets with higher heterogeneity. Specifically, DYNAFED demonstrates relative improvement over the FedAvg baseline by 17.5%, 64.5%, 52.0%, and 82.2% on FMNIST, CIFAR10, CINIC10, and CIFAR100, respectively.

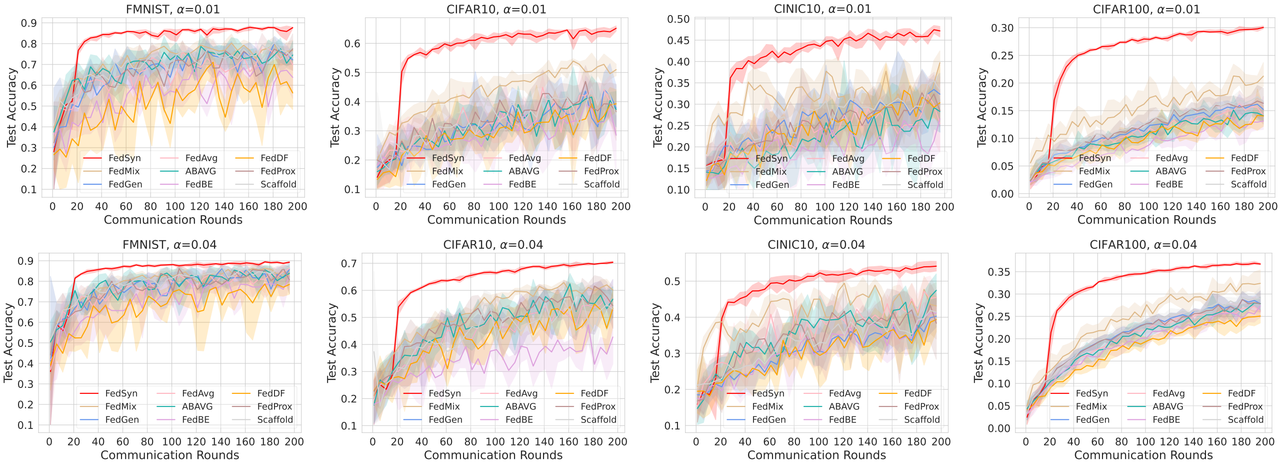


Figure 2. Visualization of global model's test performance on various datasets throughout the global communication rounds. We can see that the global model rapidly converges to a satisfactory test accuracy once $\mathcal{D}_{\text{syn}}$ participates in refining the global model. Furthermore, $\mathcal{D}_{\text{syn}}$ also helps reduce the fluctuation of model performances between communication rounds, which significantly boosts the training stability. DYNAFED requires less than 10% communication rounds to achieve comparable performance with the baseline methods.

Method	$\alpha = 0.01$	$\alpha = 0.04$	$\alpha = 0.08$	$\alpha = 0.16$
FedAVG	16.54±2.18	26.56±1.53	34.54±1.02	39.65±0.94
FedProx	18.46±1.05	28.58±1.46	34.82±0.54	40.98±0.49
Scaffold	17.33±1.21	28.46±1.18	35.04±0.35	40.57±0.33
FedDF*	16.02±1.94	26.94±1.25	34.77±0.88	39.76±0.44
FedBE*	15.78±2.34	28.03±0.34	33.91±0.79	39.45±0.79
ABAVG	16.52±1.98	29.14±0.57	34.66±0.98	41.00±0.23
FedGen†	16.51 ±1.32	27.03±1.14	34.56±0.78	39.96±0.58
FedMix†	23.54±0.96	32.18±0.59	36.30±0.42	41.09±0.14
DynaFed†	**30.14±0.19**	**36.79±0.12**	**40.02±0.09**	**42.47±0.06**

Table 2. Comparison of test performances on CIFAR100 with different degrees of data heterogeneity α.

we first run FedAvg [37] and use the checkpoints from the first 20 communication rounds ($L = 20$). We set the time difference s between the start and end checkpoint to 5, and the target checkpoint is averaged with 2 checkpoints sampled between $\boldsymbol{w}^{\mathrm{t}}$ and $\boldsymbol{w}^{\mathrm{t+s}}$. More details can be found in the Appendix.

6.1. Main Experiments with Data Heterogeneity

We demonstrate the superior performance of our DYNAFED by conducting experiments on heterogeneous client data across comprehensive datasets and various heterogeneity values α. Specifically, we use three datasets with 10 classes (shown in Table 1): FashionMNIST [46], CIFAR10 [27] and CINIC10 [9], heterogeneity degree α set to 0.01, 0.04 and 0.16; and CIFAR100 containing 100 classes, with α values 0.01, 0.04 ,0.08 and 0.16 (shown in Table 2). DYNAFED significantly boosts the convergence, stabilizes training, and brings considerable performance improvement compared with previous approaches. Specifically, with heterogeneity value $\alpha = 0.01$, DYNAFED demonstrates relative improvement over the FedAvg baseline by 17.5%, 64.5%, 52.0%, and 82.2% on FMNIST, CIFAR10, CINIC10, and CIFAR100, respectively.

As demonstrated in Figure 2 , the performance of DYNAFED is rapidly boosted as soon as the synthesized data starts refining the global model on the server. This verifies that DYNAFED does not depend on the global model's performance in data synthesis,

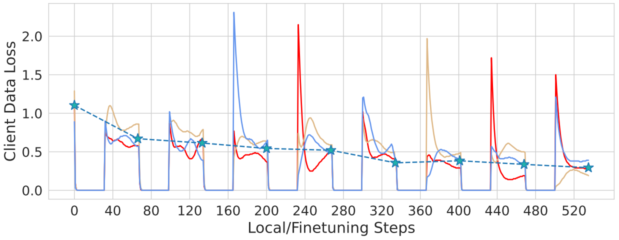
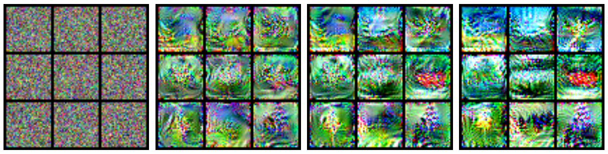


Figure 3. Loss curves over each client's data throughout local training and finetuning. Each of the 3 colors represents the loss over one client's data. The stars are the global model's average losses over all client data after finetuning with $\mathcal{D}_{\text{syn}}$. During local training, the client losses quickly converge to near zero. However, due to deflection caused by heterogeneity, the losses over some clients' data dramatically increase after aggregation. Finetuning with $\mathcal{D}_{\text{syn}}$ decreases those losses and reduces the aggregation bias.



Iteration 0 Iteration 125 Iteration 250 Iteration 500

Figure 4. Visualization of learned synthetic data on 3 classes from CIFAR10 throughout the optimization process. In the beginning, the pixels are randomly initialized and contain little information. As the optimization goes on, some patterns emerge in the synthetic images but remain unrecognizable.

which is consistent with our analysis in Section 4. This characteristic enables faster convergence to achieve good performance with fewer communication rounds. As shown in Figure 2, DYNAFED requires less than 20% communication rounds to achieve comparable performance with the baseline methods.

6.2. Detailed Analysis

We conduct a detailed analysis of DYNAFED and aim to provide answers to the following questions: (1) Does $\mathcal{D}_{\text{syn}}$ contain information about global data distribution while protecting client privacy? (2) Can we leverage just the dynamics of the early rounds to synthesize $\mathcal{D}_{\text{syn}}$? (3) How many pseudo samples do we need to synthesize to ensure effectiveness? (4) Does $\mathcal{D}_{\text{syn}}$ still help convergence under more severe heterogeneity and longer local training?

$\mathcal{D}_{\text{syn}}$ Contains Global Information and Preserves Privacy. We conduct experiment with CIFAR10 and set $\alpha = 0.01$, where client datasets are extremely *non-iid*. We track the losses calculated over each client's data throughout local training as well as the global model's finetuning. During local training, we calculate the client models' losses over their own datasets, i.e., $\mathcal{L}_m(\boldsymbol{w}_m, \mathcal{D}_m)$. During finetuning, we calculate the global model's losses over each client's dataset, i.e., $\mathcal{L}_m(\boldsymbol{w}, \mathcal{D}_m)$. To prevent cluttering, we randomly select 3 client datasets for illustration. The result is shown in Figure 3, each color represents the loss over one clients' dataset. We observe that the client models easily overfit during local training due to the extreme class imbalance. The deflected client models make the aggregated global model demonstrate high loss values over some clients' data. Remarkably, we observe that finetuning with $\mathcal{D}_{\text{syn}}$ is able to recover the global model to a reasonable state, which achieves small losses over all client datasets. This verifies that $\mathcal{D}_{\text{syn}}$ contains information of the global data distribution. Furthermore, Figure 4 presents the synthesized data of CIFAR10, client-specific information can not be observed.

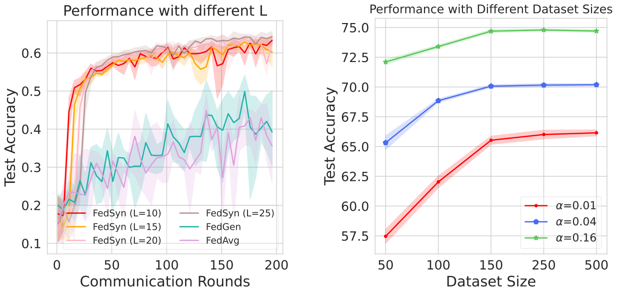


Figure 5. Left: The test accuracy curves for different choices of trajectory length L on the CIFAR10 dataset with $\alpha = 0.01$. By leveraging the dynamics of the global model's trajectory in the first few rounds, e.g., L $\in \{10, 15, 20, 15\}$, the derived $\mathcal{D}_{\text{syn}}$ already helps achieve faster convergence and stable training. In contrast, the FedGen approach only brings slight performance gain during the later phase of training due to the dependence on the global model's performance. Right: We show the performance of DYNAFED with different sizes of $\mathcal{D}_{\text{syn}}$ for various α. The performance gain is significant with just 150 synthesized samples.

$\mathcal{D}_{\text{syn}}$ Can be Learned with Early Trajectory. We conduct experiments with different choices of trajectory length L in left of Figure 5. We observe that even if with $L = 10$, the result is comparable with performance obtained with longer trajectory $L = 25$. Compared with the baseline methods, DYNAFED achieves significant convergence speedup and performance boost. These results support our claim in Section 4 that our method can take effect early during training. By contrast, the data-free KD method FedGen [56] that trains a generator to produce pseudo data starts to show a slight improvement only in the late stage of training since it depends explicitly on the global model's performance when training the generator.

How the Size of $\mathcal{D}_{\text{syn}}$ Impacts the Performance. As shown in the right of Figure 5, we conduct experiments with different sizes of $\mathcal{D}_{\text{syn}}$ and various heterogeneity degrees α. We find that a small $\mathcal{D}_{\text{syn}}$ suffices for good performance, while larger $\mathcal{D}_{\text{syn}}$ brings only marginal performance boost. This property not only saves the cost for synthesizing $\mathcal{D}_{\text{syn}}$, but also makes the finetuning of the global model more efficient.

$\mathcal{D}_{\text{syn}}$ is Able to Mimic the Global Dynamics. In the left of Figure 6, we calculate the cosine distance between the target checkpoint $\boldsymbol{w}^{\text{t+s}}$ and the parameters $\tilde{\boldsymbol{w}}$ trained from $\boldsymbol{w}^{\text{t}}$ for s' steps with $\mathcal{D}_{\text{syn}}$, a randomly sampled real dataset of the same size as $\mathcal{D}_{\text{syn}}$, and a dataset consisted of noisy pixels, respectively. We can see that in terms of mimicking the global trajectory, $\mathcal{D}_{\text{syn}}$ not only significantly outperforms the noise dataset, but also achieves only half of the distance obtained with real dataset, which is not

Method	$\alpha = 0.01$		$\alpha = 0.04$	
	5 epochs	10 epochs	5 epochs	10 epochs
FedAVG	33.23±3.54	29.93±4.62	50.28±2.17	46.09±2.95
FedProx	42.60±2.30	42.86±2.84	58.40±1.35	54.30±1.98
Scaffold	39.43±1.86	36.52±2.04	55.46±1.25	50.05±1.57
FedDF*	31.68±3.16	39.85±3.79	52.31±2.38	50.90±2.53
FedBE*	35.49±2.88	34.19±3.34	49.78±1.79	51.34±1.90
ABAVG*	37.87±2.57	35.08±3.03	56.81±1.94	52.17±2.32
FedGen†	35.64±2.52	35.03±3.58	57.60±1.55	54.48±2.03
FedMix†	47.36±1.24	41.53±1.37	60.74±0.95	56.35±1.33
DynaFed†	**61.45±0.46**	**59.04±0.64**	**68.35±0.20**	**66.30±0.34**

Table 3. Test performances on CIFAR10 achieved by different FL algorithms under various degrees of data heterogeneity and local training epochs. Total communication rounds of 100 and 50 are set with local training epochs of 5 and 10, respectively. As can be seen, DYNAFED significantly surpasses other methods.

Method	CIFAR10		CINIC10	
	$\alpha = 0.01$	$\alpha = 0.04$	$\alpha = 0.01$	$\alpha = 0.04$
	$Acc = 0.45$	$Acc = 0.55$	$Acc = 0.33$	$Acc = 0.45$
FedAVG	132.0±15.0	117.0±8.0	189.3±10.5	138.7±5.6
FedProx	113.3±16.4	102.0±4.7	156.7±7.0	118.3±4.0
Scaffold	105.0±10.4	100.3±3.5	158.0±5.4	110.0±3.4
FedDF*	145.7±13.1	117.3±5.8	180.7±5.0	170.0±7.0
FedBE*	165.0±12.7	122.7±4.5	185.3±14.6	174.3±6.8
ABAVG*	109.7±5.4	110.7±5.0	150.0±8.4	127.0±5.8
FedGen†	115.7±10.4	110.3±5.7	167.0±12.1	128.3±5.5
FedMix†	77.3±3.7	89.3±3.5	79.0±7.8	82.3±5.5
DynaFed†	**22.3±0.6**	**22.0±1.0**	**21.3 ±1.5**	**22.7±1.4**

Table 4. Comparison of **the number of communication rounds** to reach target accuracy. With the knowledge of global data distribution stored in $\mathcal{D}_{\text{syn}}$ at the server, the convergence speed of our DYNAFED is significantly accelerated.

Method	$\alpha = 0.01$		$\alpha = 0.04$	
	MLP	ConvNet	MLP	ConvNet
FedAVG	65.64±1.69	74.51±1.32	73.26±1.49	81.74±1.98
FedProx	68.09±1.47	76.88±1.83	79.83±1.70	83.06±2.53
Scaffold	67.60±1.53	77.92±0.87	78.09±1.35	82.25±1.35
FedDF*	64.59±1.70	72.36±2.08	77.20±1.58	81.65±0.97
FedBE*	65.97±1.64	72.33±1.79	75.42±1.35	81.31±1.25
ABAVG*	69.19±1.50	75.98±1.99	81.64±1.20	84.44±1.84
FedGen†	68.67±1.45	75.59±1.12	77.94±1.38	56.60±1.08
FedMix †	70.30±0.92	81.34±0.68	81.95±0.64	84.23±0.50
DynaFed†	**73.89±0.24**	**87.52±0.15**	**83.54±0.42**	**89.45±0.11**

Table 5. Performance comparison across different network architectures. We conduct the experiment on FMNIST dataset using MLP and ConvNet to demonstrate the generalization of DYNAFED for different network architectures.

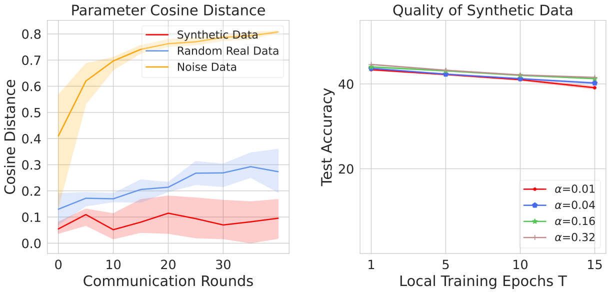


Figure 6. Left: the cosine distance between the target checkpoint $\boldsymbol{w}^{\text{t+s}}$ and the parameters $\tilde{\boldsymbol{w}}$ obtained by training with different datasets from $\boldsymbol{w}^{\text{t}}$ for s' steps. We can see that the distance derived using $\mathcal{D}_{\text{syn}}$ is constantly smaller compared with other data. Right: Test performances of a model trained from scratch using only $\mathcal{D}_{\text{syn}}$ generated under various local epochs and client heterogeneity. The two plots together verify the quality of the generated $\mathcal{D}_{\text{syn}}$.

accessible in FL setting. This verifies the ability of $\mathcal{D}_{\text{syn}}$ to mimic global trajectory.

DYNAFED is Robust to Longer Local Training. Longer local training is generally required in FL to reduce the total number of global communication rounds. Under different heterogeneity degrees, we conduct experiments to evaluate the impact of longer local training epochs on DYNAFED. Specifically, we conduct experiments with total communication rounds of 100 and 50 with ocal training epochs of 5 and 10, respectively. The results are presented in Table 3, from which we observe the following: 1) DYNAFED consistently outperforms other methods by a large margin even with longer local training epochs; 2) the performance of DYNAFED is less sensitive to the length of local training, which benefits from the $\mathcal{D}_{\text{syn}}$ containing information about the global data distribution. Therefore, DYNAFED is able to achieve similar performance with less global communication rounds, which is the major bottleneck in the efficiency of FL.

We further conduct experiments on varying local epochs to measure the quality of $\mathcal{D}_{\text{syn}}$. Specifically, we use it to train a network from scratch and evaluate its test performance. Shown in right of Figure 6, the quality of $\mathcal{D}_{\text{syn}}$ stays similar with longer local training and more severe heterogeneity. This further explains the superior performance of DYNAFED with longer local training.

6.3. Architecture Generalization and Efficiency

To showcase the generalization ability of our approach over different network architecture choices, we conduct experiments on FMNIST using different network architectures. Specifically, we choose ConvNet and MLP following [37, 56]. As shown in Table 5, under various client data heterogeneity, our method demonstrates superior performances with both network architectures.

Thanks to the rich knowledge of global data distribution contained in $\mathcal{D}_{\text{syn}}$, using it to refine the global model greatly boosts the convergence speed of training. As shown in Figure 2, as soon as $\mathcal{D}_{\text{syn}}$ is used to refine the global model, its performance rapidly increases to a reasonable accuracy, which reduces many rounds of communication. In Table 4, we also quantitively compare the convergence speed of different FL algorithms by showing the number of communication rounds needed to reach the highest test accuracy achievable by the baselines. As can be observed, our DYNAFED requires only less than 20% communication rounds to reach a target accuracy comparable to other methods.

7. Conclusion

In this paper, we propose a novel approach DYNAFED to tackle the data heterogeneity issue, which synthesizes a pseudo dataset to extract the essential knowledge of the global data distribution from the dynamics of the global model's trajectory. Extensive experiments show that DYNAFED demonstrates relative improvement over the the FedAvg baseline up to 82.2% on CIFAR100. Further, we believe our work is able to provide insights for extracting global information on the server side, which goes beyond tackling the data heterogeneity issue.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021. 2, 3

[2] Divyansh Aggarwal, Jiayu Zhou, and Anil K Jain. Fedface: Collaborative learning of face recognition model. In *2021 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–8. IEEE, 2021. 1

[3] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019. 1

[4] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020. 2

[5] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4750–4759, 2022. 2

[6] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. In *ICLR*, 2021. 2, 3, 4, 5, 14

[7] Hong-You Chen and Wei-Lun Chao. On bridging generic and personalized federated learning for image classification. In *International Conference on Learning Representations*, 2021. 5, 13

[8] Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. In *IJCAI*, 2022. 3

[9] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018. 5, 6

[10] Don Kurian Dennis, Tian Li, and Virginia Smith. Heterogeneity for the win: One-shot federated clustering. In *International Conference on Machine Learning*, pages 2611–2620. PMLR, 2021. 2

[11] Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? *arXiv preprint arXiv:2206.00240*, 2022. 4

[12] Liam Fowl, Jonas Geiping, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021. 2, 4

[13] Avishek Ghosh, Justin Hong, Dong Yin, and Kannan Ramchandran. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*, 2019. 2

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 3

[15] Hang Gu, Bin Guo, Jiangtao Wang, Wen Sun, Jiaqi Liu, Sicong Liu, and Zhiwen Yu. Fedaux: An efficient framework for hybrid federated learning. In *IEEE International Conference on Communications, ICC 2022, Seoul, Korea, May 16-20, 2022*, pages 195–200. IEEE, 2022. 2, 3, 5, 13

[16] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. *Advances in Neural Information Processing Systems*, 33:2304–2315, 2020. 2

[17] Ali Hatamizadeh, Hongxu Yin, Holger R Roth, Wenqi Li, Jan Kautz, Daguang Xu, and Pavlo Molchanov. Gradvit: Gradient inversion of vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10021–10030, 2022. 2, 4

[18] Bin Hu, Peter Seiler, and Laurent Lessard. Analysis of biased stochastic gradient descent using sequential semidefinite programs. *Mathematical Programming*, 187(1):383–408, 2021. 12

[19] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in Neural Information Processing Systems*, 34:7232–7241, 2021. 2, 4

[20] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018. 5

[21] Jinwoo Jeon, Kangwook Lee, Sewoong Oh, Jungseul Ok, et al. Gradient inversion with generative image prior. *Advances in Neural Information Processing Systems*, 34:29898–29908, 2021. 2, 4

[22] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019. 2

[23] Arthur Jochems, Timo M Deist, Issam El Naqa, Marc Kessler, Chuck Mayo, Jackson Reeves, Shruti Jolly, Martha Matuszak, Randall Ten Haken, Johan van Soest, et al. Developing and validating a survival prediction model for nsclc patients through distributed learning across 3 countries. *International Journal of Radiation Oncology* Biology* Physics*, 99(2):344–352, 2017. 1

[24] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021. 1

[25] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. In *ICML*, 2020. 1, 2, 5, 14

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 13

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5, 6

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 5

[29] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020. 2

[30] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 13

[31] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022. 2

[32] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021. 2, 3

[33] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021. 2

[34] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys*, 2020. 1, 2, 5, 14

[35] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019. 1, 12, 13

[36] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020. 2, 3, 4, 5, 14

[37] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017. 1, 2, 3, 5, 6, 8, 13, 14

[38] Seungeun Oh, Jihong Park, Eunjeong Jeong, Hyesung Kim, Mehdi Bennis, and Seong-Lyun Kim. Mix2fld: Downlink federated learning after uplink federated distillation with two-way mixup. *IEEE Communications Letters*, 24(10):2211–2215, 2020. 2

[39] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020. 1

[40] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10(1):1–12, 2020. 1

[41] MyungJae Shin, Chihoon Hwang, Joongheon Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Xor mixup: Privacy-preserving data augmentation for one-shot federated learning. *arXiv preprint arXiv:2006.05148*, 2020. 2

[42] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020. 2

[43] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020. 2

[44] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12196–12205, 2022. 2

[45] Walter Wolfgang, Wolfgang Walter, and Wolfgang Ludwig Walter. *Ordinary differential equations*, volume 182. Springer Science & Business Media, 1998. 5

[46] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 5, 6

[47] Jianhang Xiao, Chunhui Du, Zijing Duan, and Wei Guo. A novel server-side aggregation strategy for federated learning in non-iid situations. In *2021 20th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 17–24. IEEE, 2021. 2, 5, 14

[48] Yueqi Xie, Weizhong Zhang, Renjie Pi, Fangzhao Wu, Qifeng Chen, Xing Xie, and Sunghun Kim. Optimizing server-side aggregation for robust federated learning via subspace training. *arXiv preprint arXiv:2211.05554*, 2022. 1, 5, 13

[49] Yousef Yeganeh, Azade Farshad, Nassir Navab, and Shadi Albarqouni. Inverse distance aggregation for federated learning with non-iid data. In *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, pages 150–159. Springer, 2020. 2

[50] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018. 2, 4

[51] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. In *ICLR*, 2021. 2

[52] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. *arXiv preprint arXiv:2107.00233*, 2021. 5, 14

[53] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10174–10183, 2022. 2, 3, 4, 5

[54] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *ICLR*, 1(2):3, 2021. 2

[55] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018. 1, 2

[56] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning.

In *International Conference on Machine Learning*, pages 12878–12889. PMLR, 2021. 2, 3, 4, 5, 7, 8, 14

A. Proofs of Theorem 1

From lines 502 to 518, we rewrite the detailed steps of DYNAFED for the convenience of analysis. The key idea is to treat our aggregation as a local training process where each client works with exactly the same parameters $\boldsymbol{w}_m^t$ and gradients in each time step. And thus $\bar{\boldsymbol{w}}^t = \boldsymbol{w}_1^t = \cdots = \boldsymbol{w}_M^t$ in the finetuning/aggregation process. Therefore, the convergence of $\bar{\boldsymbol{w}}^t$ is actually that of our learned model.

Firstly, we need the following lemmas.

Assumption 1. *The variance of the stochastic gradients in each client is bounded, i.e.,*

$$\mathbb{E}\|\nabla \mathcal{L}_m(\boldsymbol{w}_m^t, \xi_m^t) - \nabla \mathcal{L}_m(\boldsymbol{w}_m^t, \mathcal{D}_m)\|^2 \leq \sigma_m^2,$$

where ξ_m^t is sampled from $\mathcal{D}_m$ uniformly at random, $m = 1, 2, \dots, M$.

Assumption 2. *The expectation of $\|\nabla \mathcal{L}_m(\boldsymbol{w}_m^t, \xi_m^t)\|^2$ is bounded, i.e.,*

$$\mathbb{E}\|\nabla \mathcal{L}_m(\boldsymbol{w}_m^t, \xi_m^t)\|^2 \leq G^2, \tag{8}$$

where ξ_m^t is sampled from $\mathcal{D}_m$ uniformly at random, $m = 1, 2, \dots, M$.

Assumption 3. *For our $\mathcal{D}_{syn}$, we assume*

$$\|\nabla \mathcal{L}(\boldsymbol{w}, \mathcal{D}_{syn}) - \nabla \mathcal{L}(\boldsymbol{w}, \mathcal{D})\| \leq \delta \|\nabla \mathcal{L}(\boldsymbol{w}, \mathcal{D})\| + \epsilon, \tag{9}$$

where $\delta \geq 0$ and $\epsilon \geq 0$ are two small scalars.

Assumptions 1 and 2 are widely used in stochastic optimization as well as FL [35]. Assumption 3 is a standard assumption for biased gradient [18]. We restate Theorem 1 as follows:

Theorem 2 (Convergence). *Under Assumptions 1, 2 and 3, for $\tilde{L}$-smooth, μ-strongly convex loss functions $\ell(\cdot, \cdot)$, we assume $\delta \tilde{L} < \mu$. Let $\eta_t = \frac{c}{t+\gamma}$ for a proper constant c and γ. Then,* DYNAFED *satisfies*

$$\mathbb{E}\mathcal{L}(\bar{\boldsymbol{w}}^T, \mathcal{D}) - \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}) \leq \frac{C}{T}, \tag{10}$$

where $\boldsymbol{w}^*$ is the minimum of $\mathcal{L}(\boldsymbol{w}, \mathcal{D})$ and C is a constant, whose detailed formula is given in Eqn.(17).

Before giving the detailed proof, we need the following two lemmas.

Lemma 2 (One Step of Local Training). *Under the same assumptions with Theorem 2, for $t \in \mathcal{I}$, it follows that*

$$\mathbb{E}\|\bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}^*\|^2 \leq (1 - \eta_t \mu)\mathbb{E}\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}^*\|^2 + \eta_t^2 B_{loc}, \tag{11}$$

where

$$B_{loc} = \sum_{m=1}^{M} \alpha_m^2 \sigma_m^2 + 6\tilde{L}\Gamma + 8(\tau_1 - 1)^2 G^2 \text{ with } \Gamma = \mathcal{L}(\boldsymbol{w}^*, \mathcal{D}) - \sum_{m=1}^{M} \alpha_m \min_{\boldsymbol{w}} \mathcal{L}_m(\boldsymbol{w}, \mathcal{D}_m).$$

Proof. of Lemma 2. Noe that in DYNAFED, the local training is equivalent to that of FedAvg, therefore, the intermediate results in [35] hold for DYNAFED.

The result in Eqn.(11) can be directly obtained from the proof of Theorem 1 in [35]. □

In our aggregation, each step is essentially a biased gradient descent method, for which we have the following lemma from [18].

Lemma 3 (One Step of Aggregation [18]). *Under the same assumptions with Theorem 2, for $t \notin \mathcal{I}$, it follows that*

$$\mathbb{E}\|\bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}^*\|^2 \leq \rho_t^2 \mathbb{E}\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}^*\|^2 + \eta_t^2 B_{agg}, \tag{12}$$

where $\rho_t^2 = 1 - (\mu - \delta\tilde{L})\eta_t + \mathcal{O}(\eta_t^2)$ and B_{agg} is a positive constant.

The lemma above is actually one of the implications of Theorem 1 in [18]. Please refer to Sections 1 and 3.2 of [18] for more details.

Remark 2. *Note that when $t \notin \mathcal{I}$, we perform model aggregation. Therefore, Lemma 3 demonstrates that each step in our aggregation process can reduce the expectation of the squared distance between $\bar{\boldsymbol{w}}_t$ and the optimal solution $\boldsymbol{w}^*$, i.e., our aggregation can improve equality of the aggregated model during fintuning. This is benefited from our synthesized data $\mathcal{D}_{syn}$. It is consistent with our exmperimental results in Figure 3 in the main paper.*

Now we turn to prove Theorem 2.

Proof. of Theorem 2: From Eqn.(12), we know when $\eta_t = \frac{c}{t+\gamma}$ is sufficiently small, which can be satisfied by choosing proper c or sufficiently large γ, the following holds for $t \notin \mathcal{I}$:

$$\mathbb{E}\|\bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}^*\|^2 \leq (1 - \tilde{\mu}\eta_t)\mathbb{E}\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}^*\|^2 + \eta_t^2 B_{\text{agg}}, \tag{13}$$

where $\tilde{\mu}$ can be any positive constant with $\tilde{\mu} < \mu - \delta\tilde{L}$.

By defining $\tilde{B} = \max\{B_{\text{loc}}, B_{\text{agg}}\}$, we can unify the Eqn.(11) and Eqn.(12) into

$$\mathbb{E}\|\bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}^*\|^2 \leq (1 - \tilde{\mu}\eta_t)\mathbb{E}\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}^*\|^2 + \eta_t^2 \tilde{B}, \text{ for all } t. \tag{14}$$

We assume $c\tilde{\mu} > 1$, which can be satisfied by choosing proper γ to make Eqn.(13) hold. Let

$$v = \max\{\frac{c^2\tilde{B}}{c\tilde{\mu} - 1}, (\gamma + 1)\mathbb{E}\|\bar{\boldsymbol{w}}_1 - \boldsymbol{w}^*\|^2\}. \tag{15}$$

We claim that

$$\mathbb{E}\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}^*\|^2 < \frac{v}{t + \gamma}. \tag{16}$$

We prove this claim by induction. Firstly Eqn.(16) hold for $t = 1$ due to the definition of v. Assume it also holds for some $t \geq 1$, we have

$$\begin{aligned}
\mathbb{E}\|\bar{\boldsymbol{w}}_{t+1} - \boldsymbol{w}^*\|^2 &\leq (1 - \tilde{\mu}\eta_t)\mathbb{E}\|\bar{\boldsymbol{w}}_t - \boldsymbol{w}^*\|^2 + \eta_t^2\tilde{B} \\
&\leq (1 - \tilde{\mu}\eta_t)\frac{v}{t+\gamma} + \eta_t^2\tilde{B} \\
&\leq (1 - \frac{\tilde{\mu}c}{t+\gamma})\frac{v}{t+\gamma} + \frac{c^2}{(t+\gamma)^2}\tilde{B} \\
&\leq \frac{t+\gamma-1}{(t+\gamma)^2}v + \frac{1}{(t+\gamma)^2}\left(c^2\tilde{B} - (\tilde{\mu}c - 1)v\right) \\
&\leq \frac{t+\gamma-1}{(t+\gamma)^2}v \\
&\leq \frac{v}{t+1+\gamma}.
\end{aligned}$$

Therefore, our claim holds for all $t > 0$.

Hence, for $\tilde{L}$-smoothness of $\mathcal{L}$, we have

$$\mathbb{E}\left[\mathcal{L}(\bar{\boldsymbol{w}}^T, \mathcal{D}) - \mathcal{L}(\bar{\boldsymbol{w}}^*, \mathcal{D})\right] \leq \frac{\tilde{L}}{2}\mathbb{E}\|\bar{\boldsymbol{w}}_T - \boldsymbol{w}^*\|^2 \leq \frac{\tilde{L}v}{2(T+\gamma)} \leq \frac{C}{T},$$

where

$$C = \frac{\tilde{L}v}{2}. \tag{17}$$

□

Remark 3. *We give the result of full device participation setting above. We would like to point out that for the setting of partial device participation, we can have a similar result with different C. The reason is the only difference with the full device participation setting is the local training process and the aggregation process is kept the same. The Eqn.(21) in [35] shows that Lemma 2 holds with a different constant B in this setting. To avoid redundancy, we omit this result in this appendix and please refer to [35] for more details.*

B. Detailed Experiment Settings

Detailed Configurations Unless specified otherwise, we follow [7, 15, 48] and adopt the following default configurations throughout the experiments: we run 200 global communication rounds with local epoch set to 1. In the experiments in the main paper, a total of 80 clients are adopted, and the participation ratio in each round is set to 40%. Experiments using 20% participation ratio are in Table 6 and Figure 7. For the network choices, we use ConvNet [30] with 3 layers, and the hidden dimension is set to 128. The local learning rate is set to 10^{-3} with Adam optimizer [26]. We report the global model's average performance in the last five rounds evaluated using the test split of the datasets. For the construction of global trajectory, we first run FedAvg [37] and use the checkpoints from the first 20 communication rounds ($L = 20$). For the data synthesis process on the server, we use Adam optimizer with learning rate 5×10^{-2} for optimizing the data and the label in the outer loop, SGD with learning rate 10^{-5} is used to train the network on the synthetic data. We set the time difference s between the start and end checkpoint to 5. The total number of iteration for synthesis is $N = 1000$, which takes around 1 GPU hour on RTX 3080 Ti. The subsequent finetuning takes negligible time since the amount of synthesized data is set to 150.

Method	$\alpha = 0.01$	$\alpha = 0.04$	$\alpha = 0.16$
FedAVG	33.62±4.36	50.36±3.10	68.05±1.39
FedProx	39.87±2.34	52.78±2.69	69.99±1.08
Scaffold	38.65±2.21	53.13±1.35	70.01±0.87
FedDF*	35.25±2.90	50.02±3.34	68.82±1.07
FedBE*	29.98±3.02	48.97±3.86	68.84±0.96
ABAVG	37.26±2.89	57.88±0.78	72.05±0.88
FedGen†	36.28±3.54	52.11±2.36	70.17 ±1.20
FedMix†	46.77±1.93	59.80±1.34	70.59±0.31
DynaFed†	**62.53±0.57**	**67.54±0.44**	**73.59±0.12**

Table 6. Comparison of test performances on CIFAR10 with different degrees of data heterogeneity α. The client participation ratio per round is set to 20%. Our DYNAFED outperforms other approaches by a large margin, and the superiority is more evident under more severe heterogeneity. Specifically, DYNAFED has a relative performance gain of 86% over the FedAvg baseline when $\alpha = 0.01$.

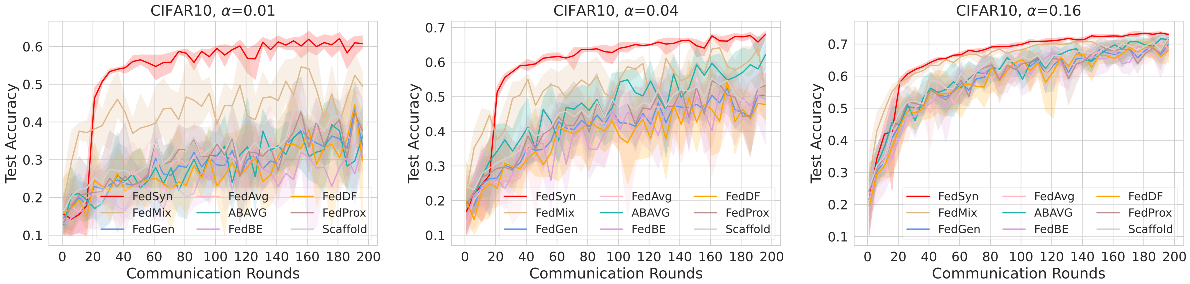


Figure 7. Visualization of global model's test performance on CIFAR10 with 20% participation ratio throughout the global communication rounds. We can see that the global model rapidly converges to a satisfactory test accuracy once $\mathcal{D}_{\text{syn}}$ participates in refining the global model. Furthermore, $\mathcal{D}_{\text{syn}}$ also helps reduce the fluctuation of model performances between communication rounds, which significantly boosts the training stability. DYNAFED requires less than 10% communication rounds to achieve comparable performance with the baseline methods.

B.1. Detailed Descriptions of Baselines

(1) FedAVG [37]: The most prevalent aggregation strategy for federated learning, which simply averages the weights sent by the clients to form the global model. (2)FedPROX [34]: a method that alleviates the client heterogeneity by regularizing the drift of local models with the global model via a penalty term during local training. (3)Scaffold [25]: a method for client heterogeneous that introduces control variates to current local gradients and performs variance reduction to stabilize training. (4)FedDF [36]: a method using knowledge distillation with unlabelled server data, which distills the ensemble knowledge from the client ensemble into the global model.(5)FedBE [6]: a method based on FedDF, which uses bayesian ensemble-based knowledge distillation with unlabelled server data.(6) ABAVG [47]: An method using validation accuracy to reweight the clients, which needs labelled server data. (7) FedGen [56]: A data-free knowledge distillation method that trains a generator that generates an embedding based on the class label. This generator is trained using the global model as the discriminator. The generator is sent to the clients to help local training. (8) FedMix [52]: a data-sharing strategy that uses linear combination between data points to preserve client privacy, which still has the potential to leak privacy, as stated in the original paper.

C. Experiment with Other Participation Ratio

To verify our DYNAFED is able to work well under different client participation ratios, we conduct additional experiments on CIFAR10 with the participation ratio set to 20%. As shown in Table 6 and Figure 7, our method demonstrates superior performances against other baseline methods.