Federated Learning under Evolving Distribution Shifts

Anonymous authors
Paper under double-blind review

Abstract

Federated learning (FL) is a distributed learning paradigm that facilitates training a global machine learning model without collecting the raw data from distributed clients. Recent advances in FL have addressed several considerations that are likely to transpire in realistic settings such as data distribution heterogeneity among clients. However, most of the existing works still consider clients' data distributions to be static or conforming to a simple dynamic, e.g., in participation rates of clients. In real FL applications, client data distributions change over time, and the dynamics, i.e., the evolving pattern, can be highly non-trivial. Further, evolution may take place from training to testing. In this paper, we address dynamics in client data distributions and aim to train FL systems from time-evolving clients that can generalize to future target data. Specifically, we propose two algorithms, *FedEvolve* and *FedEvp*, which are able to capture the evolving patterns of the clients during training and are test-robust under evolving distribution shifts. FedEvolve explicitly models the temporal evolution by learning two distinct representation mappings that capture the transition between consecutive data domains for each client. And FedEvp learns a single, evolving-domain-invariant representation by aligning current data with prototypes that are continuously updated from all previously seen domains. Through extensive experiments on both synthetic and real data, we show the proposed algorithms can significantly outperform the FL baselines across various network architectures.

1 Introduction

Federated learning (FL) is a widely used distributed learning framework where multiple clients, using their local data, train machine learning models collaboratively, orchestrated by a server (McMahan et al., 2017; Yang et al., 2019; Zhang et al., 2021). A problem that has been extensively studied in FL literature is learning from heterogeneous clients, i.e., ensuring convergence of FL training and avoiding degradation of accuracy when clients' data are **not** identically and independently distributed (non i.i.d.) (Diao et al., 2021; Achituve et al., 2021; Reisizadeh et al., 2020).

Although a variety of approaches such as robust FL (Reisizadeh et al., 2020) and personalized FL (Wang et al., 2019) have been proposed to tackle the issue of data heterogeneity, most of them still assume that the data distribution of each client is *static* and, in particular, remains fixed between training and testing. Some recent works (Jiang & Lin, 2023; Gupta et al., 2022) move one step further by proposing test-robust FL models when there exist distribution shifts between training and testing data. However, they only consider *one-step* shift between training and testing while the training data distribution is still assumed to be static. In practice, FL systems are trained and deployed in dynamic environments that may continually change over time, e.g., satellite data evolves due to spatial environmental changes and seasonal variations, clinical data evolve due to changes in disease prevalence and diverse across regions due to difference in hospital infrastructure, and human language exhibits temporal and regional changes, etc. Existing FL algorithms without considering such evolving distribution shifts may result in inaccurate models and show degradation under evolving shifts, especially when there is a large magnitude of the shift, as shown in Figure 3.

In this paper, we will explore two questions:

- How can data stream with evolving distribution shifts impact FL systems (with or without client heterogeneity)?

- How can we exploit the evolving patterns from training data (source domains) and deploy our model on the unseen future distribution (target domain)?

The goal is to continuously train an FL model from distributed, time-evolving data that can generalize well on future target data. Figure 1 shows one motivating example.

Note that although the problem of learning under evolving distribution shifts has been studied recently in the centralized setting (typically known as evolving domain generalization), e.g., see Wang et al. (2022); Qin et al. (2022); Pham et al. (2024), it remains unclear how evolving distribution shifts can impact FL training and how to design FL algorithms when both evolving distribution shifts and data heterogeneity exist. The most relevant line of research to ours is continual federated learning (CFL) (Yoon et al., 2021; Casado et al., 2022), which aims to train an FL system continuously from a set of distributed time series. However, the primary objective of these works is to stabilize the training process and tackle the issue of catastrophic forgetting (i.e., prevent forgetting the previously learned old knowledge as the model is updated on new data). This differs from our work, where we aim to explicitly learn evolving patterns and leverage them to adapt the model on future unseen data.

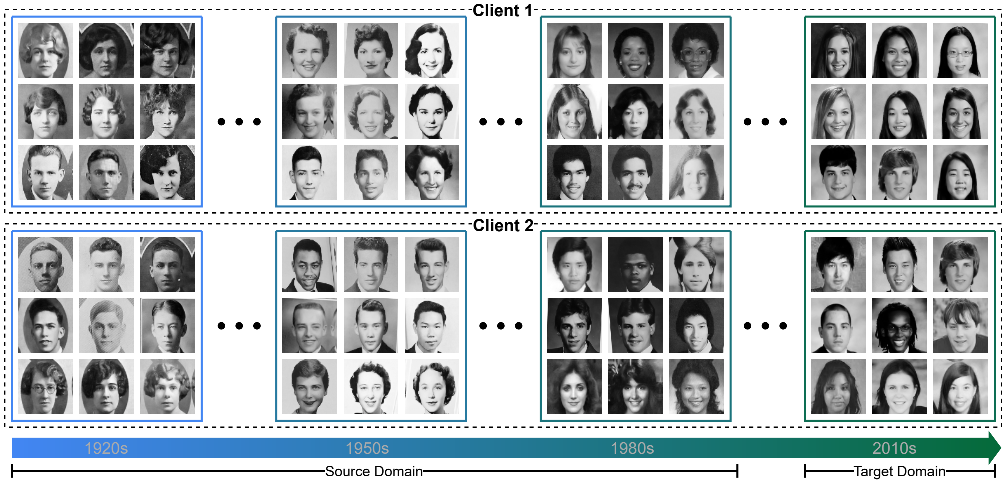


Figure 1: Illustration of evolving distribution shifts and client heterogeneity: Consider an FL system trained from distributed time-evolving photos (Ginosar et al., 2015) for gender classification. In this example, data exhibits obvious evolving patterns (e.g., changes in facial expression and hairstyle, improvement in the quality of images). Besides, clients are non-i.i.d and they have different class distributions. Our goal is to train an FL model that captures the evolving pattern of source domains and generalizes it to the future target domain.

To answer the above two questions, we will examine the performance of existing FL methods on time-evolving data, including a wide range of methods such as traditional FL methods, personalized FL methods, test-time adaptation methods, domain generalization methods, and continual FL methods. We observe that existing methods cannot capture evolving patterns and fail to generalize on future data. We then propose *FedEvolve*, an FL algorithm that learns the evolving patterns of clients during the training process and can generalize to future test data.

Specifically, *FedEvolve* learns the evolving pattern of source domains through *representation learning*. It assumes there exists a mapping function for each client that captures the transition of any two consecutive domains. To learn such transition, each client in *FedEvolve* learns two distinct representation mappings that map the inputs of domains in two consecutive time steps to a representation/latent space. By minimizing the distance between the distributions of these feature representations, *FedEvolve* captures the transition over two consecutive steps.

Although *FedEvolve* shows superior performance in learning from evolving distribution shifts in empirical experiments, the need for two distinct representation mappings brings double overhead during FL training. To reduce the computation cost and communication overhead, we further develop *FedEvp* as a more efficient and versatile version of *FedEvolve* by updating one representation mapping when evolving distribution shifts occur. Moreover, *FedEvp* better tackles heterogeneous data by incorporating the personalization strategy to partially personalize the model on each client's local data.

We illustrate via extensive experiments that our algorithms significantly outperform current benchmarks of FL when the feature domain is evolving, on multiple datasets (Rotated MNIST/EMNIST, Circle, Portraits, Caltran) using different models (MLP, CNN, ResNet). Our main contributions are:

- We identify the evolving distribution shift in FL that the current robust FL, personalized FL, and test-robust FL frameworks have failed to consider.
- We propose *FedEvolve* to actively capture the evolving pattern from evolving source domains and generalize to unseen target domains.
- We propose a more efficient and versatile version of algorithm *FedEvp* that learns domain-invariant representation from evolving prototypes.
- We empirically study how FL systems are affected when both evolving shifts and local heterogeneity exist. Experiments on multiple datasets show the superior performance of our methods compared to previous benchmark models.

2 Related Work

We briefly review related previous works in this section.

Tackle client heterogeneity in FL. Many approaches have been proposed to tackling data heterogeneity issues in FL and they can be roughly categorized into four classes. The first method is to add a regularization term. For example, Li et al. (2020; 2021) proposed to steer the local models towards a global model by adding a regularization term to guarantee convergence when the data distributions among different clients are non-IID. The second method is clustering (Briggs et al., 2020; Ghosh et al., 2020; Sattler et al., 2020). By aggregating clients with similar distribution into the same cluster, the clients within the same cluster have lower statistical heterogeneity. Then, a cluster model that performs well for clients within this cluster can be found to reduce the performance degradation of statistical heterogeneity. The third method is to mix models or data. For example, Zhao et al. (2018) proposed a data-sharing mechanism where clients update models according to both the local train data and a small amount of globally shared data. Wu et al. (2022); Shin et al. (2020) developed mixup data augmentation techniques to let local devices decode the samples collected from other clients. Mansour et al. (2020) find a mixture of the local and global models according to a certain weight. The fourth method is robust FL. For instance, Reisizadeh et al. (2020); Deng et al. (2020b) obtain robust Federated learning models by finding the best model for worst-case performance. Notably, Reisizadeh et al. (2020) only considers the affine transformation of data distributions and Deng et al. (2020b) focuses on varying weight combinations over local clients. In addition, different personalization methods are applied to local clients, such as personalization (Wang et al., 2019; Yu et al., 2020; Arivazhagan et al., 2019; Huang et al., 2023; Bao et al., 2023), representation learning (Arivazhagan et al., 2019; Collins et al., 2021; Chen & Chao, 2022; Jiang & Lin, 2023), and meta-learning (Fallah et al., 2020).

FL with dynamic data distributions. While most previous works on statistical heterogeneity have considered static situations (i.e., the local heterogeneity stays constant during training), another line of literature focuses on FL in a dynamic environment where various distribution drifts occur. Some works aim to tackle drifts caused by time-varying participation rates of clients with local heterogeneity (Rizk et al., 2020; Park et al., 2021; Wang & Ji, 2022; Zhu et al., 2021), while other works assume the global distributions are also evolving (Guo et al., 2021; Casado et al., 2022; Yoon et al., 2021). However, among all previous works, Jiang & Lin (2023); Gupta et al. (2022) are the only ones considering the distribution shift between training and testing, but they assume the training distribution itself is static.

Evolving domain generalization. *Domain Generalization* (DG) has been extensively studied to generalize ML algorithms to unseen domains where different methods including data manipulation (Khirodkar et al., 2019; Robey et al., 2021), representation learning (Blanchard et al., 2017; Deshmukh et al., 2019), and domain adversarial learning (Rahman et al., 2020; Zhao et al., 2020). To go one step further, a few works have considered the evolving patterns of the domains (Hong Liu, 2020; Zhang & Davison, 2021; Kumar et al., 2020; Wang et al., 2022; Qin et al., 2022; Pham et al., 2024), but only Wang et al. (2022); Qin et al. (2022); Pham et al. (2024) consider *Evolving Domain Generalization* (EDG) where the target domain is not accessible. Specifically, Wang et al. (2022) developed an algorithm to learn embeddings of the previous domain and the current domain such that their representations are invariant. Qin et al. (2022) developed a dynamic probabilistic framework to model the underlying latent variables across domains. Pham et al. (2024) went beyond stationary dynamics to consider non-stationary evolving patterns across domains. Unlike these works that do not require access to the target domain during training, Wang et al. (2020) considered the evolving domain adaptation problem, where the unlabeled data from a target domain is available and the goal is to use domain discriminators to learn domain-invariant features and adapt the model to target data. Note that domain adaptation differs from domain generalization, as domain generalization imposes stricter conditions by restricting access to the target domain during training, thereby making it a more challenging setting. However, all these previous works consider the *centralized setting*. Thus, there is a gap for EDG under distributed settings, and in particular for FL.

3 Problem Formulation

Consider a federated learning (FL) system consisting of K clients, whose data distributions vary dynamically over time. For each local client k, Define $\{S_1^k, ..., S_M^k\}$ as the distributions over $\mathcal{X} \times \mathcal{Y}$ for M consecutive local domains with some evolving patterns. Let $\mathcal{D}_m^k$ be the local dataset of client $k \in \{1, \ldots, K\}$ at m-th domain. The clients are heterogeneous and they may have access to different class labels. Given an FL model with parameter h, let $\ell(x, y; h)$ be the corresponding loss evaluated on a labeled data sample (x, y). Our goal is to learn an FL model h that can generalize on subsequent target domains $\{S_{M+1}^k\}_{k=1}^K$. That is, we wish to find h^* that minimizes the total loss at the target domain S_{M+1}^k over K clients:

$$h^* = \arg\min_{h \in \mathbb{R}^d} \sum_{k=1}^{K} \alpha_k L_k(h) \tag{1}$$

where α_k is the weight of client k (e.g., the proportion of sample size), and $L_k(h) := \mathbb{E}_{(x,y)\sim S_{M+1}^k}[\ell(x, y; h)]$ is the local loss of client k evaluated on target domain S_{M+1}^k.

4 Methodology

To learn an FL model from time-evolving data that generalizes well to the future domain, we need to learn the evolving pattern of source domains during federated training. Motivated by (Wang et al., 2022; Snell et al., 2017), we assume there is an evolving pattern that captures the transition between every two consecutive domains S_m^k and S_{m+1}^k for each client. Instead of learning evolving patterns directly in the input space, we consider *representation learning* to learn the evolution in a representation space. Next, we introduce two algorithms *FedEvolve* and *FedEvp*, which align data representation from evolving domains and facilitate local personalization. Specifically, *FedEvolve* is designed to actively identify the evolving pattern between two consecutive domains, while *FedEvp* first learns an evolving invariant representation across all existing domains, then generalizes to the unknown evolving domain.

4.1 FedEvolve

Theoretical motivation. To actively capture the evolving patterns of source domains, *FedEvolve* learns two distinct learnable representation functions f_ϕ, f_ψ[1]. Given two consecutive domains S_m^k and S_{m+1}^k:

- $f_\phi(S_m^k)$ is the **estimated representation** of **subsequent** domain S_{m+1}^k using input S_m^k.
- $f_\psi(S_{m+1}^k)$ is the **representation** of input domain S_{m+1}^k.

---

[1] Theoretically, we can also use one function f to demonstrate the evolving pattern directly in terms of the source domains. However, using two representation mappings f_ϕ, f_ψ brings empirical benefits and makes it easier for the model to learn the evolving patterns accurately in a latent representation space (Snell et al., 2017).

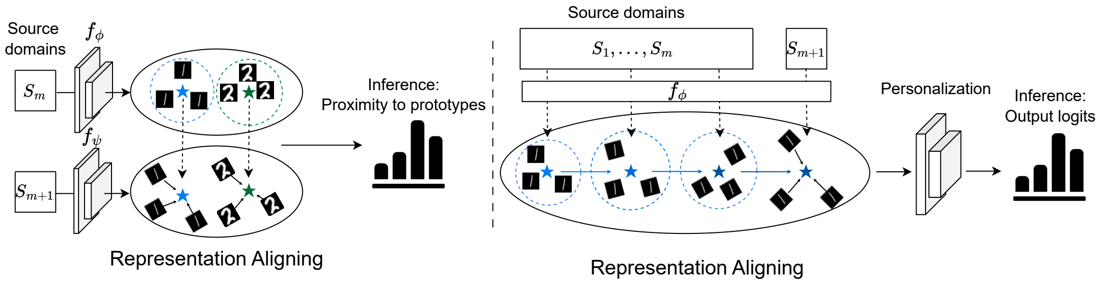


Figure 2: Illustration of *FedEvolve* (left) and *FedEvp* (right): (i). *FedEvolve* consists of two distinct modules ϕ and ψ, where ϕ calculates the prototypes for domain S_m, individually for each class, using mean values as class representations. Then, ψ represents a data batch from the domain S_{m+1}. Both modules are updated based on the distance between S_{m+1} representations and S_m prototypes. During inference, ψ computes the distance to the latest domain's prototypes, then selects the minimal one as the prediction result. (ii). *FedEvp* simplifies *FedEvolve* by removing ψ and integrating a classifier w with ϕ. This decreases the communication cost during federated training. Instead of using localized prototypes from just S_m, *FedEvp* builds global prototypes from domains S_1 to S_m. These prototypes align with the representations of the succeeding domain S_{m+1}, providing an integrated feature representation across diverse domains. By emphasizing consistent feature representation, *FedEvp* ensures its classifier adeptly handles an unseen domain, making predictions resilient and versatile across changing data contexts.

Since we define $\{S_1^k, \cdots, S_M^k\}$ as the data distributions in input space, $f_\phi(S_m^k), f_\psi(S_m^k)$ associated with each domain S_m^k are the corresponding distributions in the representation space.

To measure the distance between two distributions, we adopt Jensen-Shannon divergence d_{JS}. For each client k, define $\{\psi_k^*, \phi_k^*\}$ as the parameter pair that minimizes the average distance between representation distributions generated from consecutive local domains, i.e., $\psi_k^*, \phi_k^* \overset{\mathrm{def}}{=} \arg\min_{\psi,\phi} \frac{1}{M-1} \sum_{m=1}^{M-1} d_{\mathrm{JS}}\left(f_{\psi_k}(S_{m+1}^k) \| f_{\phi_k}(S_m^k)\right)$. The following theorem characterizes an upper bound of the prediction error at the target domains $\{S_{M+1}^k\}_{k=1}^K$.

Theorem 4.1 (Upper bound of error at target domains)**.** *Let* $\widehat{h}$ *be a classifier operated on a representation space, and denote* $L_{f_\psi(S_m^k)}\left(\widehat{h}\right), L_{f_\phi(S_m^k)}\left(\widehat{h}\right)$ *as the expected losses of* $\widehat{h}$ *with respect to distributions* $f_\psi(S_m^k)$*,* $f_\phi(S_m^k)$ *in the corresponding representation space. Suppose the loss function* ℓ *is bounded and define its range as* $G = \max(\ell) - \min(\ell)$*. Then for any* h*,* f_ψ*, and* f_ϕ*, the following holds*

$$\sum_{k=1}^{K} \alpha_k L_{f_\psi(S_{M+1}^k)}\left(\widehat{h}\right) \leq \underbrace{\sum_{k=1}^{K} \alpha_k L_{f_\phi(S_M^k)}\left(\widehat{h}\right)}_{\textbf{\textit{Term 1}}} + \tag{2}$$

$$\sum_{k=1}^{K} \alpha_k \frac{G}{\sqrt{2}(M-1)} \Bigg(\underbrace{\sum_{m=1}^{M-1} \Bigg(\sqrt{d_{JS}\left(f_{\psi_k^*}(S_{m+1}^k) \| f_{\phi_k^*}(S_m^k)\right)}}_{\textbf{\textit{Term 2}}} + \underbrace{\left(\sqrt{d_{JS}\left(f_\psi(S_{m+1}^k) \| f_{\psi_k^*}(S_{m+1}^k)\right)} + \sqrt{d_{JS}\left(f_\phi(S_m^k) \| f_{\phi_k^*}(S_m^k)\right)} \right)}_{\textbf{\textit{Term 3}}}$$

$$+ \underbrace{\left| \sqrt{d_{JS}\left(f_\psi(S_{M+1}^k) \| f_\phi(S_M^k)\right)} - \sqrt{d_{JS}\left(f_\psi(S_{m+1}^k) \| f_\phi(S_m^k)\right)} \right|}_{\textbf{\textit{Term 4}}} \Bigg) \Bigg)$$

The proof of Thm. 4.1 is motivated by Wang et al. (2022) and provided in Appendix B. Thm. 4.1 suggests that the prediction error at unseen target domains can be bounded. Specifically, **Term 1** in the upper bound is the prediction error on *estimated representations* of the target domain. **Term 2** measures the distance between representations generated from consecutive domains and it also indicates *stationarity* of evolving pattern of local source domains—for any given hypothesis classes of f_ϕ, f_ψ, it represents the extent to which we can use one (ψ, ϕ) pair to capture the evolution across domains. **Term 3** measures the client heterogeneity in evolution patterns of the federated system. **Term 4** represents whether the evolution pattern learned from source domains $S_1^k, \cdots, S_M^k$ can be generalized to target domain S_{M+1}^k.

Thm. 4.1 provides insights for algorithm design: to learn an FL model with small prediction error on future target domains $\{S^k_{M+1}\}_{k=1}^K$, we find $\phi, \psi, \widehat{h}$ such that the upper bound in Thm. 4.1 is minimized. Specifically, we aim to find a classification rule $\widehat{h}$ such that predictions on estimated representations $f_\phi(S^k_M)$ are sufficiently accurate (reducing **Term 1**). Meanwhile, the parameter pair (ϕ, ψ) should be close to the optimal parameters (ϕ^*_k, ψ^*_k) of local clients on average (reducing **Term 3**), where (ϕ^*_k, ψ^*_k) should be learned from source domains $\{S^k_1, \cdots, S^k_m\}$ such that representation $f_{\phi^*_k}(S^k_m)$ is sufficiently close to $f_{\psi^*_k}(S^k_{m+1})$ estimated from previous domain S^k_m (reducing JS-distance in **Term 2**). Following this idea, we design *FedEvolve* as detailed below.

***FedEvolve* algorithm.** Because f_ϕ estimates the representation of a domain using the previous domain, we can use it to estimate unknown target domain S^k_{M+1} from source domains $\{S^k_1, ..., S^k_M\}$ for each client k. Let ϕ, ψ be the trainable neural network parameters of f_ϕ, f_ψ, respectively. To learn the evolving pattern, we aim to learn ϕ, ψ such that the estimated future domain representation $f_\phi(S^k_m)$ is sufficiently accurate and close to the actual representation $f_\psi(S^k_{m+1})$, i.e., we need to minimize the distance between $f_\phi(S^k_m)$ and $f_\psi(S^k_{m+1})$. Inspired by (Wang et al., 2022), to align the two representations while capturing the class characteristics across evolving domains, we leverage prototypical learning (Snell et al., 2017) to directly align their representation prototypes. Instead of directly optimizing on representation functions, we maintain prototypes for each evolving domain to handle the representation shifts by learning the prototype differences.

Specifically, for each client k and domain S^k_m, we let the average of representations for each class y learned by $f_{\widetilde{\phi}_k}$ be the *prototype* $C^k_{m,y}$, where $\widetilde{\phi}_k$ is the local parameter learned on client k, i.e.,

$$C^k_{m,y} = \frac{1}{|\mathcal{D}^k_{m,y}|} \sum_{x \in \mathcal{D}^k_{m,y}} f_{\widetilde{\phi}_k}(x) \tag{3}$$

where $\mathcal{D}^k_{m,y} \subseteq \mathcal{D}^k_m$ is a subset of data instances with label y, $|\mathcal{D}^k_{m,y}|$ is the cardinality of this set. For the next domain S^k_{m+1}, Instead of directly minimizing the JS-distance between representation distributions $f_{\widetilde{\psi}_k}(S^k_{m+1})$ and $f_{\widetilde{\phi}_k}(S^k_m)$, *FedEvolve* achieves this by aligning the representations from $f_{\widetilde{\psi}_k}(S^k_{m+1})$ to prototypes $C^k_{m,y}$ computed from S^k_m. Mathematically, we minimize the loss defined below:

$$\ell(x, y) = -\log \frac{\exp\left(-d\left(f_{\widetilde{\psi}_k}(x), C^k_{m,y}\right)\right)}{\sum_{y' \in \mathcal{Y}_{\mathcal{D}^k_{m+1}}} \exp\left(-d\left(f_{\widetilde{\psi}_k}(x), C^k_{m,y'}\right)\right)} \tag{4}$$

where (x, y) is a sample pair from $\mathcal{D}^k_{m+1}$ and $\mathcal{Y}_{\mathcal{D}^k_{m+1}}$ including all class labels in $\mathcal{D}^k_{m+1}$. d is a distance measure (e.g. Euclidean distance, cosine distance) that quantifies the difference between feature representation $f_{\widetilde{\psi}_k}(x)$ and the prototype $C^k_{m,y}$ of class y from the local dataset $\mathcal{D}^k_m$. In this paper, we employ Euclidean distance.

By minimizing equation 4 on all active clients, local models learn the evolving pattern by aligning representations of domain S^k_{m+1} with prototypes from the former domain S^k_m. After local updates, active clients $\mathcal{I}_t$ send local parameters to the server and the server performs an average aggregation to update the global parameters $\phi = \frac{1}{|\mathcal{I}_t|} \sum_{k \in \mathcal{I}_t} \widetilde{\phi}_k, \psi = \frac{1}{|\mathcal{I}_t|} \sum_{k \in \mathcal{I}_t} \widetilde{\psi}_k$. This aggregation rule is chosen to reduce **Term 3** in Thm. 4.1 and the resulting aggregations encapsulate global information with diverse data contributions of all clients. Once consolidated, these models can be directly dispatched to the clients and facilitate continuous model generalizations to the evolving data distributions across the federated network.

After training on source domains, we can use the learned representation functions f_ϕ, f_ψ to predict the target domains $\{S^k_{M+1}\}_{k=1}^K$. Specifically, we first compute the prototypes of $f_\phi(S^k_M)$ on S^k_M. Then, we apply f_ψ to test samples in S^k_{M+1} to generate representations $f_\psi(S^k_{M+1})$ and classify them based on proximity to prototypes. We present the pseudocode of *FedEvolve* in Algorithm 3 in Appendix A and its simplified version in Algorithm 1.

Algorithm 1 FedEvolve (Simplified)

```
Require: Number of clients $K$, client ratio $r$, step size $\eta$, local steps $\tau$, rounds $T$, source domains $M$, global parameters $\phi$, $\psi$, local datasets $\mathcal{D}_m^k$
for each round $t = 1$ to $T$ do
    Server selects $rK$ clients and broadcasts $\phi$, $\psi$
    for each selected client $k$ in parallel do
        Initialize local models $\widetilde{\phi}_k \leftarrow \phi$, $\widetilde{\psi}_k \leftarrow \psi$
        for $\tau$ local updates do
            For domain pairs $(m, m+1)$, sample episodic batches
            Compute class prototypes from domain $m$ using equation 3
            Compute prototypical contrastive loss on domain $m+1$ using equation 4
            Update $\widetilde{\phi}_k$, $\widetilde{\psi}_k$ using gradient descent
        end for
        Send updated parameters to server
    end for
    Server aggregates parameters to update $\phi$ and $\psi$
end for
Output: final global parameters $\phi$, $\psi$
```

4.2 FedEvp

Because the two distinct representation functions f_ϕ and f_ψ in *FedEvolve* are usually large neural networks (e.g., ResNet (He et al., 2016) for complex image datasets), there is a non-negligible additional overhead to transmit extra parameters of the second representation function, rendering deployment challenges in environments with limited computational resources or network bandwidth. To address the potential overhead, we also present *FedEvp*, an efficient and streamlined strategy that achieves similar performance as *FedEvolve*.

Unlike the dual model mechanism of *FedEvolve*, *FedEvp* adopts a single-model strategy to reduce communication costs while simultaneously accelerating training. As shown in the right plot of Figure 2, *FedEvp* aims to learn the evolving-domain-invariant representation using a representation function f_ϕ by continuously aligning data to prototypes from previous domains. If we can develop a representation that is resilient to evolving distributional shifts, a single classifier could effectively serve all domains. To further address local heterogeneity, we also incorporate an efficient personalization step for the classifier.

To ensure a consistent learning process, *FedEvp* maintains evolving prototypes according to the classes of consecutive domains. In essence, the prototypes learned by *FedEvp* consolidate the global information from all previous domains to enable the learning of domain-invariant features. For each class y within client k, an evolving prototype $C_{m,y}^k$ is continually updated as equation 5,

$$C_{m,y}^k = \frac{(m-1)}{m} C_{m-1,y}^k + \frac{1}{m}\left(\frac{1}{|\mathcal{D}_{m,y}^k|}\right) \sum_{x \in \mathcal{D}_{m,y}^k} f_{\widetilde{\phi}_k}(x) \tag{5}$$

where $C_{0,y}^k$ is set to zero, $\mathcal{D}_{m,y}^k$ is the set of all instances in the current domain m that belongs to class y, and $f_{\widetilde{\phi}_k}(x_i)$ denotes the representation of instance x_i under the client k's local model parameters $\widetilde{\phi}_k$. Such an iterative update mechanism ensures that the prototype $C_{m,y}^k$ evolves as new domains are introduced, gradually incorporating information from each one. As a result, $C_{M,y}^k$ becomes a representative prototype of class y across all available training domains for client k.

We then align the data from domain S_{m+1}^k to the prototypes C_m^k to update parameter $\widetilde{\phi}_k$. We adopt the same loss function as *FedEvolve* given in equation 6,

$$\ell_f(x,y) = -\log \frac{\exp\left(-d\left(f_{\widetilde{\phi}_k}(x), C^k_{m,y}\right)\right)}{\sum_{y' \in \mathcal{Y}_{\mathcal{D}^k_{m+1}}} \exp\left(-d\left(f_{\widetilde{\phi}_k}(x), C^k_{m,y'}\right)\right)} \tag{6}$$

where d is the same distance metric as in *FedEvolve*, and $d\left(f_{\widetilde{\phi}_k}(x), C^k_{m,y}\right)$ measures the distance between feature representation $f_{\widetilde{\phi}_k}(x)$ of instance x and the prototype $C^k_{m,y}$ of class y, $\mathcal{Y}_{\mathcal{D}^k_{m+1}}$ is the set of classes in domain S^k_{m+1}.

Indeed, the above idea of continuously aligning data to an evolving prototype also comes with theoretical support. Following Pham et al. (2023), we construct another upper bound of prediction error at the target domain, as detailed in Lemma 4.2 below.

Lemma 4.2 (Upper bound of error at target domain S^k_{M+1})**.** *Let $\widehat{h}$ be a classifier operated on a representation space, and denote $L_{f_{\phi_k}(S^k_m)}\left(\widehat{h}\right)$ as the expected loss of $\widehat{h}$ with respect to distribution $f_{\phi_k}(S^k_m)$. Suppose the loss function ℓ is upper bounded by C. Then the following holds for any $\widehat{h}$ and f_{ϕ_k}:*

$$L_{f_{\phi_k}(S^k_{M+1})}\left(\widehat{h}\right) \leq \underbrace{\sum_{m=1}^{M} L_{f_{\phi_k}(S^k_m)}\left(\widehat{h}\right)}_{\textit{Term 1}} + \sqrt{2}C \underbrace{\min_{m\in[M]} d_{JS}(S^k_{M+1}\|S^k_m)}_{\textit{Term 2}} + \sqrt{2}C \underbrace{\max_{m,n\in[M]} d_{JS}\left(f_{\phi_k}(S^k_m)\|f_{\phi_k}(S^k_n)\right)}_{\textit{Term 3}}$$

Lemma 4.2 provides an upper bound of prediction error at target domain S^k_{M+1}. Note that **Term 2** in the upper bound is fully determined by domains $\{S^k_1, \cdots, S^k_{M+1}\}$ and out of our control. To attain a small error on target domain S^k_{M+1}, Lemma 4.2 suggests that we may learn $\widehat{h}, \phi_k$ such that predictions on source domains $\{S^k_1, \cdots, S^k_{M+1}\}$ are sufficiently accurate (reducing **Term 1**). Meanwhile, we need to learn ϕ_k to minimize the *maximum* possible distance between representations generated from any two source domains (reducing **Term 3**). As domains continuously evolve in a specific direction, a good representation function ϕ_k that minimizes the maximum JS-distance in **Term 3** is to align data from the current domain to the *average* of all previous representations (i.e., evolving prototype in equation 5 is updated by averaging over all previous domains).

Besides minimizing ℓ_f to learn evolving-domain-invariant representation, we introduce a classifier $\widehat{h}_{\widetilde{w}_k}$ operated in a representation space, where $\widetilde{w}_k$ is parameter and is updated by minimizing empirical risk ℓ_e defined as:

$$\ell_e(x,y) = -y \log \frac{\exp\left(\widehat{h}^y_{\widetilde{w}_k}\left(f_{\widetilde{\phi}_k}(x)\right)\right)}{\sum_{y' \in \mathcal{Y}_{\mathcal{D}^k_m}} \exp\left(\widehat{h}^{y'}_{\widetilde{w}_k}\left(f_{\widetilde{\phi}_k}(x)\right)\right)} \tag{7}$$

where $\widehat{h}^y_{\widetilde{w}_k}\left(f_{\widetilde{\phi}_k}(x)\right)$ is the predicted outputs of the class y for instance $(x,y) \in \mathcal{D}^k_{m,y}$, computed by the classifier $\widehat{h}_{\widetilde{w}_k}$. In our experiments, ℓ_e is the classical cross-entropy loss.

After local updates, *FedEvp* aggregates the local parameters of active clients $\mathcal{I}_t$ at the server $\phi = \frac{1}{|\mathcal{I}_t|}\sum_{k\in\mathcal{I}_t} \widetilde{\phi}_k, w = \frac{1}{|\mathcal{I}_t|}\sum_{k\in\mathcal{I}_t} \widetilde{w}_k$. These aggregated global models are then sent back to clients for future updates. As *FedEvp* relies on the classifier using evolving domain invariant features instead of directly using the difference between two consecutive domain representations, the prediction may be influenced by the client's heterogeneity. To handle the issue raised by local heterogeneity, a personalization mechanism, akin to *local fine-tuning*, is further incorporated. Specifically, we personalize each client by updating both the classifier w and the *last layer* of the feature extractor f_ϕ for an additional epoch on the client's local dataset. The pseudocode of *FedEvp* is given in Algorithm 4 in Appendix A and its simplified version is in Algorithm 2..

Algorithm 2 FedEvp (Simplified)

```
Require: Number of clients $K$, client ratio $r$, step size $\eta$, local steps $\tau$, rounds $T$, source domains $M$, global
    parameters $\phi$, $w$, local datasets $\mathcal{D}_m^k$
for each round $t = 1$ to $T$ do
    Server selects $rK$ clients and broadcasts $\phi$, $w$
    for each selected client $k$ in parallel do
        Initialize local models $\widetilde{\phi}_k \leftarrow \phi$, $\widetilde{w}_k \leftarrow w$
        for $\tau$ local updates do
            for each domain $m$ do
                Incrementally update evolving prototypes across domains as equation 5
                Compute evolving alignment loss on domain $m$ and pass evolving prototypes using equation 6
                Update $\widetilde{\phi}_k$, $\widetilde{w}_k$ using gradient descent with cross-entropy loss and alignment loss
            end for
        end for
        Send updated parameters to server
    end for
    Server aggregates parameters to update $\phi$ and $w$
end for
Server Output: global model $\phi$, $w$
for each client $k$ do
    Client Output: personalized model via fine-tuning on $\mathcal{D}^k$
end for
```

5 Experiments

To evaluate our methods, we consider classification tasks using various network architectures and report the average accuracy and standard deviation over three runs. The detailed implementation can be found in Appendix D. The Dirichlet distribution (Yurochkin et al., 2019; Mendieta et al., 2022) is used to control the level of heterogeneity with parameter Dir $\in [0, \infty)$. The smaller Dir implies that the clients are more heterogeneous. Heterogeneous clients may have access to different class labels. We report the average performance across clients and the performance on the server. Both are evaluated on the test domain after the last epoch. The federated training phase follows typical FL steps. In each communication round t, a subset $\mathcal{I}_t$ of K clients join the system and the server distributes aggregated global model parameters to client $k \in \mathcal{I}_t$. Upon receiving these parameters, each client k initializes its local parameters to those and performs τ local updates. We follow the same setting as Jiang & Lin (2023) to use 20 clients in experiments. For datasets with a limited number of samples, we reduce the number of clients to 10. Details can be found in the dataset introduction in Appendix D.

5.1 Datasets and Networks

We evaluate *FedEvolve* and *FedEvp* on both **synthetic data** (Circle) and **real data** (Rotated MNIST, Rotated EMNIST, Portraits, and Caltran). All datasets either come with evolving patterns or are adapted to evolving environments. For all datasets, the last domain is viewed as the target domain. The feature extractor in the neural network is viewed as ϕ and ψ, and the classifier is w mentioned in the previous section.

Circle (Pesaranghader & Viktor, 2016). This synthetic data has 30 evolving domains. 30000 instances within these domains are sampled from 30 two-dimensional Gaussian distributions, with the same variance but different means that are uniformly distributed on a half-circle. We use a 5-layer multilayer perception (MLP) with 3 layers serving as a representation function (f_ϕ and f_ψ in *FedEvolve*, f_ϕ in *FedEvp*) and the remaining 2 layers as a classifier (f_w in *FedEvp*).

Rotated MNIST (Ghifary et al., 2015) and Rotated EMNIST (Cohen et al., 2017). The Rotated MNIST is a variation of the MNIST data, where we rotate the original handwritten digit images to produce different domains. Specifically, we partition the data into 12 domains and rotate the images within each domain by an angle θ, beginning at $\theta = 0°$ and progressing in 15-degree increments up to $\theta = 165°$. We also consider other increments spanning from 0° to 25° to simulate varying degrees of evolving shifts. EMNIST is a more challenging alternative to MNIST with more classes including both hand-written digits and letters. We use the hand-written letters subset and split it into 12 domains by rotating images with a degree of $\theta = \{0°, 8°, ..., 88°\}$. We design a model consisting of a 4-layer convolutional neural network (CNN) for representation layers, followed by two linear layers for classification.

Portraits (Ginosar et al., 2015). It is a real dataset consisting of frontal-facing American high school yearbook photos over a century. This time-evolving dataset reflects the changes in fashion (e.g., high style and smile). We resize images to 32×32 and split the dataset by every 12 years into 9 domains. We use WideResNet (Zagoruyko & Komodakis, 2016) as the representation function to train the gender classifier. Note that the data is only intended to compare various methods.

Caltran (Hoffman et al., 2014). This real surveillance dataset comprises images captured by a fixed traffic camera. We divide the dataset into 12 domains where the samples from every 2-hour block form a domain (evolving shifts arising from changes in light intensity). ResNet18 (He et al., 2016) backbone is used as the representation function and the last linear layer is used as the classifier.

5.2 Baselines

We compare *FedEvolve* and *FedEvp* with various existing FL methods. These baselines cover a broad range of methods including regular FL methods, methods with personalization (PFL) or test-time adaptation (TTA) mechanisms, and methods designed for distribution shifts across domains or sequential tasks e.g. domain generalization methods. Note that this paper focuses on a domain generalization setting rather than domain adaptation. Therefore, domain adaptation methods are not included as baselines.

- *FedAvg* (McMahan et al., 2017): A FL method that learns the global model by averaging the client's local model.
- *GMA* (Tenison et al., 2022): A FL method using gradient masked averaging approach to aggregate local models.
- *FedAvg + FT*: Fine-tunes the global model on local training data, an effective strategy for personalization in FL.
- *MEMO* (Zhang et al., 2022): A TTA method and we adapt it to FL. Following (Jiang & Lin, 2023), we term MEMO applied to the global model as MEMO(G) and to the FedAvg + FT model as MEMO(P).
- *APFL* (Deng et al., 2020a): A PFL method that leverages a weighted ensemble of personalized and global models.
- *FedRep* (Collins et al., 2021) and *FedRoD* (Chen & Chao, 2022): PFL methods that use a decoupled feature extractor and classifier to enhance personalization in FL.
- *Ditto* (Li et al., 2021): A fairness-aware PFL method that has been shown to outperform other fairness FL methods.
- *T3A* (Iwasawa & Matsuo, 2021): A TTA method that is adapted to personalized FL by adding test-time adaptation to *FedAvg + FT*.
- *FedTHE* (Jiang & Lin, 2023): A TTA PFL method that tackles the data heterogeneity issue while learning test-time robust FL under distribution shifts.
- *Flute* (Liu et al., 2024): Flute is a PFL method that facilitates the distillation of the subspace spanned by the global optimal representation from the misaligned local representations.
- *FedSR* (Nguyen et al., 2022): A TTA FL method using the regular domain generalization method.
- *CFL* (Guo et al., 2021): A continual federated learning method that learns from time-series data without forgetting old tasks..
- *CFeD* (Ma et al., 2022): It uses distillation to learn from sequential tasks in continual federated learning.

Table 1: Average accuracy over three runs of experiments on rotated MNIST under i.i.d and non-i.i.d distribution. The client heterogeneity(Dir) is determined by the value of Dirichlet distribution.

	Dir→ ∞		Dir=1.0		Dir=0.1	
Method	Client	Server	Client	Server	Client	Server
FedAvg	65.92$_{\pm 1.01}$	66.34$_{\pm 0.34}$	62.35$_{\pm 0.97}$	63.16$_{\pm 1.78}$	51.68$_{\pm 0.73}$	51.59$_{\pm 2.48}$
GMA	65.94$_{\pm 0.91}$	66.17$_{\pm 0.21}$	61.49$_{\pm 0.30}$	61.68$_{\pm 0.66}$	50.86$_{\pm 1.15}$	51.32$_{\pm 2.47}$
Memo(G)	65.94$_{\pm 1.34}$	66.78$_{\pm 2.30}$	61.39$_{\pm 0.94}$	62.91$_{\pm 2.55}$	49.76$_{\pm 5.58}$	52.06$_{\pm 1.23}$
FedAvgFT	48.70$_{\pm 1.03}$	66.61$_{\pm 0.59}$	57.95$_{\pm 2.91}$	62.61$_{\pm 1.02}$	69.51$_{\pm 1.97}$	51.59$_{\pm 1.70}$
APFL	62.37$_{\pm 1.08}$	65.57$_{\pm 1.54}$	67.58$_{\pm 1.09}$	63.98$_{\pm 2.31}$	70.37$_{\pm 2.19}$	50.66$_{\pm 0.47}$
FedRep	60.04$_{\pm 1.00}$	68.09$_{\pm 3.10}$	63.95$_{\pm 0.75}$	63.49$_{\pm 2.62}$	76.35$_{\pm 1.67}$	52.25$_{\pm 1.75}$
Ditto	65.23$_{\pm 0.87}$	65.35$_{\pm 1.50}$	68.14$_{\pm 0.92}$	64.64$_{\pm 1.45}$	75.55$_{\pm 2.56}$	50.89$_{\pm 2.79}$
FedRod	52.30$_{\pm 1.87}$	67.93$_{\pm 1.05}$	54.00$_{\pm 3.98}$	63.32$_{\pm 2.33}$	64.11$_{\pm 3.68}$	53.02$_{\pm 1.22}$
Memo(P)	51.70$_{\pm 2.48}$	65.35$_{\pm 1.47}$	59.84$_{\pm 0.61}$	64.75$_{\pm 1.59}$	69.46$_{\pm 2.77}$	50.27$_{\pm 2.85}$
T3A	53.94$_{\pm 0.76}$	66.61$_{\pm 0.59}$	61.60$_{\pm 2.49}$	62.61$_{\pm 1.02}$	71.73$_{\pm 1.63}$	51.59$_{\pm 1.70}$
FedTHE	66.84$_{\pm 1.51}$	67.43$_{\pm 0.23}$	67.98$_{\pm 0.43}$	62.55$_{\pm 1.98}$	78.52$_{\pm 3.92}$	53.40$_{\pm 0.74}$
Flute	62.97$_{\pm 1.39}$	63.27$_{\pm 1.13}$	68.86$_{\pm 0.75}$	61.46$_{\pm 0.16}$	78.44$_{\pm 3.54}$	54.71$_{\pm 3.28}$
FedSR	69.91$_{\pm 1.14}$	71.79$_{\pm 1.75}$	67.00$_{\pm 1.23}$	68.01$_{\pm 2.65}$	61.49$_{\pm 2.60}$	59.88$_{\pm 3.54}$
CFL	63.75$_{\pm 0.98}$	64.33$_{\pm 2.17}$	60.29$_{\pm 1.85}$	60.82$_{\pm 1.97}$	50.76$_{\pm 1.41}$	51.04$_{\pm 2.49}$
CFeD	70.22$_{\pm 0.63}$	71.66$_{\pm 0.66}$	68.07$_{\pm 0.72}$	68.64$_{\pm 1.38}$	60.41$_{\pm 2.33}$	61.27$_{\pm 2.93}$
FedEvolve	**84.75** $_{\pm 1.39}$	**84.43**$_{\pm 1.21}$	**79.93**$_{\pm 1.00}$	**77.25**$_{\pm 1.82}$	**83.86**$_{\pm 1.81}$	**71.66**$_{\pm 1.95}$
FedEvp	**75.99**$_{\pm 0.31}$	**77.63**$_{\pm 1.99}$	**77.91**$_{\pm 1.80}$	**73.85**$_{\pm 1.53}$	**83.15**$_{\pm 0.49}$	**61.84**$_{\pm 3.34}$

Table 2: Average accuracy over three runs of experiments on rotated EMNIST-Letter under i.i.d and non-i.i.d distribution.

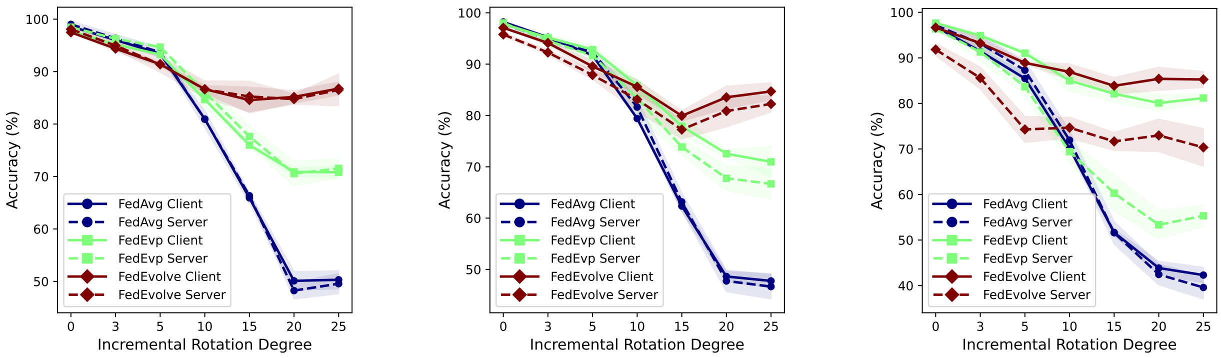
	Dir→ ∞		Dir=1.0		Dir=0.1	
Method	Client	Server	Client	Server	Client	Server
FedAvg	53.83$_{\pm 1.84}$	54.18$_{\pm 1.72}$	52.72$_{\pm 4.45}$	52.77$_{\pm 3.74}$	46.72$_{\pm 2.55}$	45.71$_{\pm 1.77}$
GMA	54.23$_{\pm 1.77}$	55.10$_{\pm 1.71}$	51.23$_{\pm 1.93}$	51.42$_{\pm 0.79}$	48.40$_{\pm 1.75}$	48.61$_{\pm 2.13}$
Memo(G)	53.32$_{\pm 1.38}$	53.85$_{\pm 0.72}$	50.33$_{\pm 2.06}$	50.37$_{\pm 1.10}$	47.53$_{\pm 2.09}$	47.20$_{\pm 1.86}$
FedAvgFT	44.20$_{\pm 2.54}$	54.09$_{\pm 1.30}$	52.16$_{\pm 4.62}$	53.82$_{\pm 2.13}$	66.96$_{\pm 0.68}$	46.87$_{\pm 0.60}$
APFL	44.98$_{\pm 1.57}$	54.33$_{\pm 1.12}$	49.84$_{\pm 1.48}$	50.99$_{\pm 0.62}$	66.80$_{\pm 0.37}$	46.42$_{\pm 2.58}$
FedRep	39.01$_{\pm 2.03}$	46.39$_{\pm 2.49}$	47.26$_{\pm 2.64}$	47.25$_{\pm 0.93}$	67.51$_{\pm 1.35}$	44.12$_{\pm 0.46}$
Ditto	42.38$_{\pm 1.77}$	53.90$_{\pm 1.20}$	53.80$_{\pm 1.89}$	56.22$_{\pm 1.58}$	72.66$_{\pm 0.61}$	55.48$_{\pm 1.94}$
FedRod	44.25$_{\pm 1.60}$	51.79$_{\pm 2.77}$	49.53$_{\pm 0.81}$	50.32$_{\pm 2.61}$	67.31$_{\pm 2.03}$	45.74$_{\pm 3.99}$
Memo(P)	45.42$_{\pm 2.39}$	53.47$_{\pm 1.33}$	51.23$_{\pm 4.94}$	51.10$_{\pm 1.10}$	68.37$_{\pm 1.48}$	47.73$_{\pm 2.26}$
T3A	48.80$_{\pm 2.84}$	54.49$_{\pm 0.46}$	55.93$_{\pm 2.28}$	53.29$_{\pm 1.12}$	71.80$_{\pm 1.95}$	52.08$_{\pm 2.84}$
FedTHE	52.40$_{\pm 3.87}$	53.27$_{\pm 3.60}$	58.08$_{\pm 1.44}$	53.45$_{\pm 1.87}$	69.34$_{\pm 2.10}$	46.15$_{\pm 2.17}$
Flute	48.89$_{\pm 3.04}$	51.52$_{\pm 4.63}$	55.10$_{\pm 3.88}$	46.71 $_{\pm 2.73}$	64.99$_{\pm 3.35}$	40.27$_{\pm 3.01}$
FedSR	55.71$_{\pm 0.09}$	56.92$_{\pm 0.44}$	51.40$_{\pm 4.65}$	55.35$_{\pm 3.93}$	44.38$_{\pm 2.30}$	49.43$_{\pm 2.48}$
CFL	40.65$_{\pm 2.19}$	41.41$_{\pm 1.86}$	45.82$_{\pm 2.34}$	46.13$_{\pm 1.01}$	40.24$_{\pm 3.50}$	39.37$_{\pm 4.29}$
CFeD	56.76$_{\pm 0.65}$	56.17$_{\pm 1.39}$	55.50$_{\pm 4.33}$	55.53$_{\pm 5.73}$	47.20$_{\pm 1.37}$	47.76$_{\pm 2.22}$
FedEvolve	**83.58**$_{\pm 1.45}$	**82.91**$_{\pm 1.36}$	**82.13**$_{\pm 0.48}$	**78.68**$_{\pm 0.25}$	**87.67**$_{\pm 0.55}$	**72.85**$_{\pm 1.03}$
FedEvp	**67.30**$_{\pm 1.35}$	**71.94**$_{\pm 1.50}$	**73.61**$_{\pm 1.70}$	**68.91**$_{\pm 0.30}$	**87.01**$_{\pm 0.22}$	**58.73**$_{\pm 0.96}$

Table 3: Average accuracy across various datasets over three runs. We consider the i.i.d setting that Dir→ ∞.

	Circle		Portraits		Caltran	
	Client	Server	Client	Server	Client	Server
FedAvg	70.40$_{\pm 6.51}$	70.40$_{\pm 6.51}$	94.10$_{\pm 0.13}$	94.10$_{\pm 0.13}$	62.93$_{\pm 2.10}$	64.31$_{\pm 2.13}$
GMA	62.55$_{\pm 6.94}$	62.55$_{\pm 6.94}$	94.18$_{\pm 0.14}$	94.18$_{\pm 0.14}$	63.28$_{\pm 3.48}$	63.85$_{\pm 3.49}$
Memo(G)	-	-	94.38$_{\pm 0.07}$	94.63$_{\pm 0.31}$	63.41$_{\pm 2.81}$	63.82$_{\pm 2.92}$
FedAvgFT	60.85$_{\pm 3.07}$	63.55$_{\pm 5.67}$	90.99$_{\pm 0.74}$	93.21$_{\pm 1.86}$	63.82$_{\pm 0.70}$	63.98$_{\pm 3.22}$
APFL	59.90$_{\pm 2.48}$	63.55$_{\pm 5.67}$	90.54$_{\pm 0.29}$	94.64$_{\pm 0.16}$	62.11$_{\pm 1.85}$	63.17$_{\pm 3.29}$
FedRep	64.37$_{\pm 5.60}$	64.97$_{\pm 6.05}$	90.88$_{\pm 0.63}$	93.50$_{\pm 1.15}$	62.03$_{\pm 3.05}$	64.07$_{\pm 2.41}$
Ditto	62.60$_{\pm 2.64}$	63.10$_{\pm 6.00}$	91.46$_{\pm 0.13}$	94.07$_{\pm 0.30}$	62.44$_{\pm 2.59}$	63.58$_{\pm 3.43}$
FedRod	64.60$_{\pm 2.33}$	65.00$_{\pm 6.55}$	91.57$_{\pm 0.18}$	94.78$_{\pm 0.43}$	64.14$_{\pm 3.94}$	58.29$_{\pm 4.75}$
Memo(P)	-	-	91.30$_{\pm 0.16}$	94.34$_{\pm 0.28}$	63.66$_{\pm 2.93}$	63.58$_{\pm 3.43}$
T3A	62.20$_{\pm 4.11}$	66.50$_{\pm 4.95}$	91.84$_{\pm 0.61}$	94.59$_{\pm 0.34}$	63.90$_{\pm 0.60}$	63.98$_{\pm 3.22}$
FedTHE	64.03$_{\pm 4.79}$	63.27$_{\pm 5.05}$	94.13$_{\pm 0.24}$	93.48$_{\pm 0.98}$	60.48$_{\pm 1.44}$	58.17$_{\pm 3.18}$
Flute	65.69$_{\pm 3.81}$	63.04$_{\pm 3.31}$	94.25$_{\pm 0.10}$	94.53$_{\pm 0.18}$	61.71$_{\pm 2.19}$	61.46$_{\pm 3.70}$
FedSR	72.77$_{\pm 3.38}$	71.62$_{\pm 5.70}$	94.43$_{\pm 0.35}$	94.52$_{\pm 0.35}$	64.57$_{\pm 1.36}$	66.02$_{\pm 1.47}$
CFL	72.12$_{\pm 8.76}$	72.12$_{\pm 8.76}$	92.91$_{\pm 1.07}$	92.91$_{\pm 1.07}$	63.68$_{\pm 3.61}$	63.92$_{\pm 3.15}$
CFeD	71.60$_{\pm 6.77}$	71.60$_{\pm 6.77}$	93.64$_{\pm 0.27}$	93.64$_{\pm 0.27}$	63.48$_{\pm 3.87}$	63.55$_{\pm 3.27}$
FedEvolve	**84.25**$_{\pm 2.45}$	**81.64**$_{\pm 1.95}$	**95.43**$_{\pm 0.17}$	**96.88**$_{\pm 1.35}$	**65.04**$_{\pm 1.66}$	63.54$_{\pm 0.74}$
FedEvp	**73.30**$_{\pm 5.02}$	**74.12**$_{\pm 6.93}$	93.54$_{\pm 0.19}$	**94.92** $_{\pm 0.11}$	**66.59**$_{\pm 1.44}$	**66.34**$_{\pm 0.69}$

Table 4: Accuracy of baselines across various datasets over three runs (Dir=1.0).

	Circle		Portraits		Caltran	
	Client	Server	Client	Server	Client	Server
FedAvg	$66.53_{\pm 4.74}$	$66.53_{\pm 4.74}$	$94.37_{\pm 0.86}$	$94.37_{\pm 0.86}$	$66.34_{\pm 2.41}$	$65.12_{\pm 4.87}$
GMA	$65.93_{\pm 6.01}$	$65.93_{\pm 6.01}$	$93.75_{\pm 0.68}$	$93.75_{\pm 0.68}$	$65.12_{\pm 1.95}$	$63.05_{\pm 4.51}$
Memo(G)	-	-	$93.81_{\pm 0.45}$	$93.81_{\pm 0.45}$	$66.20_{\pm 2.07}$	$65.54_{\pm 0.73}$
FedAvgFT	$65.97_{\pm 1.49}$	$66.93_{\pm 3.30}$	$92.54_{\pm 0.65}$	$94.56_{\pm 0.43}$	$65.12_{\pm 2.84}$	$65.56_{\pm 3.61}$
APFL	$64.23_{\pm 0.80}$	$66.93_{\pm 3.30}$	$92.16_{\pm 0.42}$	$94.47_{\pm 0.38}$	$70.49_{\pm 3.70}$	$65.41_{\pm 3.84}$
FedRep	$66.87_{\pm 4.91}$	$69.07_{\pm 5.42}$	$92.50_{\pm 0.65}$	$94.19_{\pm 0.56}$	$65.27_{\pm 1.86}$	$65.90_{\pm 3.39}$
Ditto	$69.05_{\pm 4.41}$	$64.50_{\pm 5.09}$	$91.86_{\pm 0.87}$	$94.93_{\pm 0.32}$	$65.45_{\pm 3.43}$	$65.61_{\pm 4.52}$
FedRod	$63.70_{\pm 1.96}$	$77.20_{\pm 4.98}$	$92.64_{\pm 0.58}$	$95.26_{\pm 0.31}$	$73.27_{\pm 3.35}$	$64.88_{\pm 4.03}$
Memo(P)	-	-	$92.94_{\pm 0.65}$	$94.48_{\pm 0.32}$	$64.88_{\pm 3.13}$	$62.24_{\pm 3.97}$
T3A	$69.80_{\pm 1.60}$	$69.10_{\pm 1.50}$	$91.93_{\pm 0.50}$	$94.20_{\pm 0.34}$	$67.24_{\pm 2.01}$	$65.61_{\pm 4.52}$
FedTHE	$70.30_{\pm 5.83}$	$74.97_{\pm 3.90}$	$91.77_{\pm 0.85}$	$94.53_{\pm 0.32}$	$71.80_{\pm 3.07}$	$62.02_{\pm 4.22}$
Flute	$70.33_{\pm 4.31}$	$67.04_{\pm 3.66}$	$94.16_{\pm 0.69}$	$94.59_{\pm 0.38}$	$71.61_{\pm 1.61}$	$63.46_{\pm 1.58}$
FedSR	$73.88_{\pm 3.10}$	$72.08_{\pm 4.85}$	$93.99_{\pm 0.79}$	$94.22_{\pm 0.77}$	$62.99_{\pm 2.11}$	$\mathbf{68.35}_{\pm 0.53}$
CFL	$70.82_{\pm 5.43}$	$70.82_{\pm 5.43}$	$93.84_{\pm 0.30}$	$93.84_{\pm 0.30}$	$64.50_{\pm 3.17}$	$65.28_{\pm 3.50}$
CFeD	$68.37_{\pm 8.22}$	$68.38_{\pm 8.22}$	$93.22_{\pm 3.21}$	$94.77_{\pm 0.92}$	$65.30_{\pm 2.92}$	$67.18_{\pm 2.91}$
FedEvolve	$\mathbf{82.52}_{\pm 1.94}$	$\mathbf{83.59}_{\pm 5.91}$	$93.84_{\pm 1.62}$	$\mathbf{96.54}_{\pm 1.39}$	$\mathbf{75.04}_{\pm 4.03}$	$64.06_{\pm 3.83}$
FedEvp	$\mathbf{74.80}_{\pm 1.69}$	$\mathbf{77.93}_{\pm 4.20}$	$\mathbf{94.50}_{\pm 0.28}$	$93.91_{\pm 2.19}$	$\mathbf{73.46}_{\pm 0.90}$	$68.24_{\pm 1.08}$



(a) Dir → ∞ (b) Dir = 1.0 (c) Dir = 0.1

Figure 3: Performance comparison of various methods across different rotation angles on RMNIST for distinct distributions.

5.3 Results

In Figure 3, we examine how the algorithm performance changes as the degree of evolving shifts varies. Tables 14, 2 and 3 show the comparison with baselines, where we report both the averaged performance of clients' local models and the performance of the global model at the server. We also extend the experiments in Table 3 to the setting when clients are heterogeneous (Dir = 1.0) and present the results in Table 4.

Impacts of distribution shifts and local heterogeneity. First, we examine the impact of distribution shifts and client heterogeneity on FL systems. Figure 3 presents the results on RMNIST data under clients with varying degrees of local heterogeneity ($\text{Dir} = \infty, 1.0, 0.1$). Each sub-figure shows how performance changes as the extent of distribution shift changes from no distribution shift (0° incremental angle) to high distribution shift (25° incremental angle):

- In the absence of significant distribution shifts (e.g. rotation incremental angle 0°, 3°, or 5°), Figure 3a shows that, when there is no client heterogeneity, our methods have similar performance as the traditional FL methods. The learning task reduces to the standard FL task, and the classical FL methods maintain competitive performance. As clients become more heterogeneous, Figures 3b and 3c show that all methods experience performance degradation. Importantly, we observe a widening gap between client-side (personalized) and server-side (generalized) performance as heterogeneity increases. This is especially pronounced in Figure 3c where Dir = 0.1, highlighting the value of personalization under extreme heterogeneity. While *FedEvolve*'s server-side accuracy drops slightly below *FedAvg*, its client-side model retains competitive performance, and *FedEvp* shows consistently robust results across both views. This increasing gap indicates that personalized models benefit more from local adaptation when clients become highly non-iid, whereas server aggregation struggles to reconcile the diverging local objectives as we also find in Table 4.

- When the rotation increments increase, *FedAvg* experiences a significant performance drop (e.g., nearly 12% decrease when the incremental angle increases for 5 degrees, see Figure 3a). Such impacts are more significant than the performance drop caused by client heterogeneity, indicating the challenge of evolving shifts. However, our methods are still robust against such shifts and significantly better than baselines. When both strong local heterogeneity and distribution shifts are present (Figure 3c), both the baselines and ours experience a performance drop while ours exhibit a relatively slower decline. Additionally, the client-server performance gap grows under these settings, further validating the importance of personalization. Notably, the superior client-side performance of *FedEvp* under these compounded challenges further validates the effectiveness of the personalization mechanism of *FedEvp*.

Comparison with Baselines. We conduct extensive experiments on five datasets with different levels of client heterogeneity. Table 14 and 2 and the results of Circle data in Table 3 compare different methods in scenarios with strong evolving patterns. We observe that both *FedEvolve* and *FedEvp* outperform the baseline methods. In particular, *FedEvolve* attains the highest accuracy (84.75%, 83.58%, and 84.25% on RMNIST, REMNIST, and Circle respectively), demonstrating its capability to learn from the evolving pattern and effectively address the distribution shifts. This advantage also shows on other datasets (Portraits and Caltran) in Table 3 with less obvious evolving patterns.

For PFL or TTA baselines tuned on local source domains, without client heterogeneity (Dir $\to \infty$), the performance may deteriorate compared to classical FL such as *FedAvg*. Specifically, methods such as *FedAvgFT*, *APFL*, and *FedRep* may experience a drop in client performance compared to the server on certain datasets. These methods originally designed to tackle client heterogeneity without learning evolving patterns suffer performance degradation; this further highlights the importance of considering evolving distribution shifts in FL systems. Nonetheless, when clients are heterogeneous (Dir is 1.0 or 0.1 in Table 14 and 2), their personalization or test-time adaptation can still be beneficial. Methods designed for addressing domain shifts or task shifts like *FedSR*, *CFL*, and *CFeD* tend to achieve better results than other baselines, indicating their capability to mitigate the influence of evolving distribution shifts. However, the gap between their performance and that of ours still emphasizes the need for a specific design to solve the problem.

Among all methods, our proposed *FedEvolve* and *FedEvp* show the best performance and are robust to both client heterogeneity and evolving shifts. *FedEvp* achieves comparable performance with *FedEvolve* but only uses half numbers of parameters as *FedEvolve*. Specifically, when Dir $= 0.1$, *FedEvolve* achieves accuracy of **83.86%** and **87.67%** on RMNIST and REMNIST, while *FedEvp* achieves similar accuracy of **83.15%** and **87.01%**. Thus, a careful design of personalization can prevent the unintended consequence of performance degradation.

Impact of varying the number of domains. Specifically, we conduct experiments under the same settings as shown in Table 5, while controlling for the number of source domains and test prediction performance for the target domain. Our methods are compared to *FedAvg* using reduced numbers of domains: 7 domains (rotation starting at 75° and increasing to 165°), 10 domains (rotation starting at 30° and increasing to 165°), and 12 domains (rotation starting at 0° and increasing to 165°).

As the number of domains increases, *FedAvg* shows significant performance degradation across all heterogeneity settings. This indicates regular methods' vulnerability to evolving distributional shifts. Both *FedEvolve* and *FedEvp* display robustness against increasing domain numbers and maintaining or improving performance. In particular, *FedEvolve* can fully learn the transition of two consecutive domains by incorporating more source domains. However, *FedEvp* remains less sensitive to domain transitions, performing consistently well across different settings. The robustness of our methods contrasts sharply with the performance drop observed in *FedAvg*, highlighting the importance of handling distribution variability in FL. In addition, we study the impact of the unexpected changing pattern on the target domain.

Table 5: Performance under different number of domains.

RMNIST(Dir$\to\infty$)	7	10	12
FedAvg	74.92 ± 1.08	70.07 ± 1.48	65.92 ± 1.01
FedEvolve	78.24 ± 1.18	82.44 ± 1.40	84.57 ± 2.45
FedEvp	80.20 ± 2.09	74.17 ± 1.15	75.99 ± 0.31
RMNIST(Dir=1.0)	7	10	12
FedAvg	71.44 ± 1.21	65.40 ± 0.58	62.35 ± 0.97
FedEvolve	75.07 ± 2.42	80.81 ± 2.33	79.93 ± 1.00
FedEvp	80.19 ± 1.26	78.99 ± 0.82	77.91 ± 1.80
RMNIST(Dir=0.1)	7	10	12
FedAvg	61.18 ± 0.91	54.83 ± 1.24	51.68 ± 0.73
FedEvolve	78.56 ± 1.69	83.44 ± 2.25	83.86 ± 1.81
FedEvp	78.16 ± 4.26	83.92 ± 1.59	82.12 ± 1.84

Impact of Straggler. Stragglers in FL systems introduce heterogeneity at the system level; therefore, we also study how our methods could be resilient to the straggler problem. We report the results in Table 6 when stragglers are present during the training phase. The results show our methods are not significantly affected by stragglers. In this experiment, the straggler ratio represents the probability that a client will train fewer local iterations than the specified number τ. For stragglers, the actual number of local iterations is randomly selected, ranging from 1 to τ.

Table 6: Performance under different straggler ratio.

RMNIST(Dir=1.0)	0	0.1	0.3	0.5	0.7	0.9
FedEvolve	$79.93_{\pm 1.00}$	$78.56_{\pm 4.17}$	$77.50_{\pm 4.87}$	$77.42_{\pm 3.45}$	$75.88_{\pm 2.58}$	$71.87_{\pm 0.98}$
FedEvp	$77.91_{\pm 1.80}$	$77.79_{\pm 1.69}$	$77.11_{\pm 0.83}$	$77.00_{\pm 1.14}$	$76.91_{\pm 0.84}$	$76.60_{\pm 1.54}$

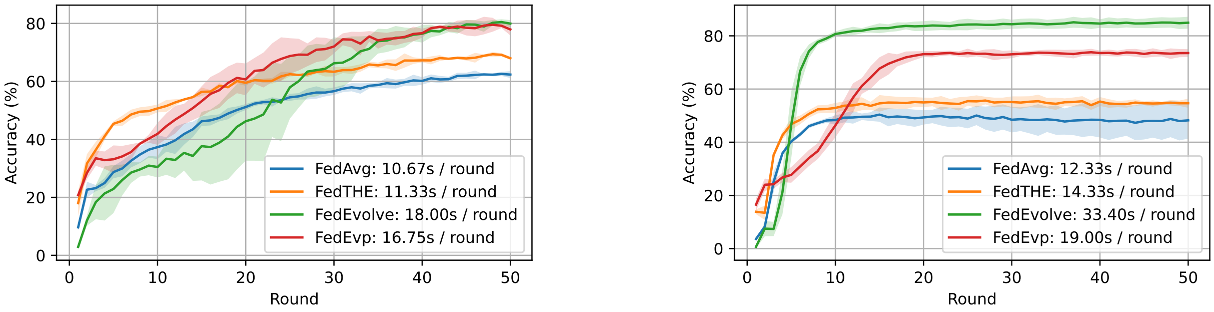
Overhead Comparison. Table 7 compares transmission overhead. We use an CNN as an example to report the number of parameters and server-client transmission time in the MPI environment. Although *FedEvolve* has the higher transmission overhead, its cost-efficient version *FedEvp* has comparable overhead as the baselines. Here we also provide learning-curve plots and wall-clock time per round for FedEvolve and FedEvp in Figure 4.

Table 7: The number of model parameters and transmission time.

	FedRod	FedTHE	FedSR	FedEvolve(Ours)	FedEvp (Ours)	Others
Parameters	382106	382208	391937	741120	379392	379392
Time/ms	21.38 ± 0.45	21.95 ± 1.23	21.62 ± 0.87	46.32 ± 0.78	21.30 ± 0.86	21.26 ± 1.11

Ablation study. We also study the influence of personalization mechanisms of *FedEvp* on the performance in Table 8. The results show personalizing part of the feature extractor and classifier can achieve the best results. We also notice that personalizing the classifier brings the most significant improvement which means the classifier is most sensitive to the client heterogeneity with evolving distribution shifts.

Discussion. The empirical evidence suggests that conventional FL algorithms cannot simultaneously handle the evolving distributional shifts and clients heterogeneity. In addition, evolving distributional shifts could be viewed as a specific form of data heterogeneity affecting client devices. Present personalization strategies, designed for data heterogeneity, fail in adapting models to unseen distributions. Simply tuning clients on known domains without considering shifts between training data and test data, these methods may inadvertently increase the model's bias towards training data resulting in performance that is sometimes inferior to that of non-personalized algorithms. While continual FL frameworks take account of dynamic distributional shifts during training, they primarily concentrate on preventing catastrophic forgetting of prior tasks or domains rather than adapting to new, unseen ones. This focus makes them inadequate for



(a) Dir → ∞

(b) Dir = 1.0

Figure 4: Learning curves on RMNIST (left) and REMNIST (right) datasets under dir = 1.0.

Table 8: Ablation for *FedEvp* (Dir=0.1). We compare the average accuracy on clients for *FedEvp* with three versions: one without any personalization, another that personalizes only the classifier, and a third that personalizes all parameters.

Method	RMNIST Acc	REMNIST Acc
FedEvp	$83.15_{\pm 0.49}$	$87.01_{\pm 0.22}$
FedEvp w/o personalization	$63.59_{\pm 2.38}$	$57.67_{\pm 1.64}$
FedEvp personalize C	$79.21_{\pm 2.29}$	$86.59_{\pm 0.35}$
FedEvp personalize all	$73.06_{\pm 1.07}$	$82.78_{\pm 0.54}$

managing evolving distributional shifts effectively. However, when the distribution of a target domain is predictable based on existing data, our methods explicitly leverage and learn the pattern of distribution transitions, enabling the extrapolation of the model to the target domain. Therefore, our methods mitigate the performance drop and achieve the best results.

6 Conclusions

This paper studies FL under evolving distribution shifts. We explored the impacts of evolving shifts and client heterogeneity on FL systems and proposed two algorithms: *FedEvolve* that precisely captures the evolving patterns of two consecutive domains, and *FedEvp* that learns a domain-invariant representation for all domains with the aid of personalization. Extensive experiments show both algorithms have superior performance compared to SOTA methods.

References

Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized federated learning with gaussian processes. Advances in Neural Information Processing Systems, 34, 2021.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. arXiv preprint arXiv:1912.00818, 2019.

Wenxuan Bao, Tianxin Wei, Haohan Wang, and Jingrui He. Adaptive test-time personalization for federated learning. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL `https://openreview.net/forum?id=rbw9xCU6Ci`.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset. In Proceedings of the international AAAI conference on web and social media, volume 14, pp. 830–839, 2020.

Gilles Blanchard, Aniket Anand Deshmukh, Urun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. arXiv preprint arXiv:1711.07910, 2017.

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), pp. 1–9. IEEE, 2020.

Fernando E Casado, Dylan Lema, Marcos F Criado, Roberto Iglesias, Carlos V Regueiro, and Senén Barro. Concept drift detection and adaptation for federated and continual learning. Multimedia Tools and Applications, pp. 1–23, 2022.

Hong-You Chen and Wei-Lun Chao. On bridging generic and personalized federated learning for image classification. In International Conference on Learning Representations, 2022. URL `https://openreview.net/forum?id=I1hQbx10Kxn`.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In 2017 international joint conference on neural networks (IJCNN), pp. 2921–2926. IEEE, 2017.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In International Conference on Machine Learning, pp. 2089–2099. PMLR, 2021.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. arXiv preprint arXiv:2003.13461, 2020a.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Distributionally robust federated averaging. Advances in Neural Information Processing Systems, 33:15111–15122, 2020b.

Aniket Anand Deshmukh, Yunwen Lei, Srinagesh Sharma, Urun Dogan, James W Cutler, and Clayton Scott. A generalization error bound for multi-class domain generalization. arXiv:1905.10392, 2019.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In International Conference on Learning Representations, 2021. URL `https://openreview.net/forum?id=TNkPBBYFkXg`.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. Advances in Neural Information Processing Systems, 33:3557–3568, 2020.

M. Ghifary, W. Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In 2015 IEEE International Conference on Computer Vision (ICCV), pp. 2551–2559, 2015. doi: 10.1109/ICCV.2015.293.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. Advances in Neural Information Processing Systems, 33:19586–19597, 2020.

Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, and Alexei A Efros. A century of portraits: A visual historical record of american high school yearbooks. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 1–7, 2015.

Yongxin Guo, Tao Lin, and Xiaoying Tang. Towards federated learning on time-evolving heterogeneous data. arXiv preprint arXiv:2112.13246, 2021.

Sharut Gupta, Kartik Ahuja, Mohammad Havaei, Niladri Chatterjee, and Yoshua Bengio. Fl games: A federated learning framework for distribution shifts. In Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022), 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

Judy Hoffman, Trevor Darrell, and Kate Saenko. Continuous manifold based adaptation for evolving visual domains. In 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 867–874, 2014.

Jianmin Wang Yu Wang Hong Liu, Mingsheng Long. Learning to adapt to evolving domains. In NIPS, 2020.

Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. Distributed pruning towards tiny neural networks in federated learning. In 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), pp. 190–201. IEEE, 2023.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. Advances in Neural Information Processing Systems, 34, 2021.

Liangze Jiang and Tao Lin. Test-time robust personalization for federated learning. In The Eleventh International Conference on Learning Representations, 2023. URL `https://openreview.net/forum?id=3aBuJEza5sq`.

Rawal Khirodkar, Donghyun Yoo, and Kris Kitani. Domain randomization for scene-specific car detection and pose estimation. In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1932–1940. IEEE, 2019.

Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. In International Conference on Machine Learning, pp. 5468–5479. PMLR, 2020.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems, 2:429–450, 2020.

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

Renpu Liu, Cong Shen, and Jing Yang. Federated representation learning in the under-parameterized regime. In Forty-first International Conference on Machine Learning, 2024. URL `https://openreview.net/forum?id=LIQYhV45D4`.

Yuhang Ma, Zhongle Xie, Jue Wang, Ke Chen, and Lidan Shou. Continual federated learning based on knowledge distillation. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, volume 3, 2022.

Y. Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. ArXiv, abs/2002.10619, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics, pp. 1273–1282. PMLR, 2017.

Matias Mendieta, Taojiannan Yang, Pu Wang, Minwoo Lee, Zhengming Ding, and Chen Chen. Local learning matters: Rethinking data heterogeneity in federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8397–8406, 2022.

A. Tuan Nguyen, Philip Torr, and Ser-Nam Lim. Fedsr: A simple and effective domain generalization method for federated learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), Advances in Neural Information Processing Systems, 2022. URL `https://openreview.net/forum?id=mrt90D00aQX`.

Tae Jin Park, Kenichi Kumatani, and Dimitrios Dimitriadis. Tackling dynamics in federated incremental learning with variational embedding rehearsal. arXiv preprint arXiv:2110.09695, 2021.

Ali Pesaranghader and Herna L Viktor. Fast hoeffding drift detection method for evolving data streams. In Joint European conference on machine learning and knowledge discovery in databases, pp. 96–111. Springer, 2016.

Thai-Hoang Pham, Xueru Zhang, and Ping Zhang. Fairness and accuracy under domain generalization. ArXiv, 2023.

Thai-Hoang Pham, Xueru Zhang, and Ping Zhang. Non-stationary domain generalization: Theory and algorithm. In The 40th Conference on Uncertainty in Artificial Intelligence, 2024. URL `https://openreview.net/forum?id=AMxdbjUvWg`.

Tiexin Qin, Shiqi Wang, and Haoliang Li. Generalizing to evolving domains with latent structure-aware sequential autoencoder. In Proceedings of the 39th International Conference on Machine Learning, volume 162, pp. 18062–18082. PMLR, 17–23 Jul 2022.

Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. Correlation-aware adversarial domain adaptation and generalization. Pattern Recognition, 100:107124, 2020.

Amirhossein Reisizadeh, Farzan Farnia, Ramtin Pedarsani, and Ali Jadbabaie. Robust federated learning: The case of affine distribution shifts. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/f5e536083a438cec5b64a4954abc17f1-Abstract.html`.

Elsa Rizk, Stefan Vlaski, and Ali H Sayed. Dynamic federated learning. In 2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 1–5. IEEE, 2020.

Alexander Robey, George J Pappas, and Hamed Hassani. Model-based domain generalization. Advances in Neural Information Processing Systems, 34:20210–20229, 2021.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. IEEE Transactions on Neural Networks and Learning Systems, 2020.

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.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. Advances in neural information processing systems, 30, 2017.

Irene Tenison, Sai Aravind Sreeramadas, Vaikkunth Mugunthan, Edouard Oyallon, Eugene Belilovsky, and Irina Rish. Gradient masked averaging for federated learning. arXiv preprint arXiv:2201.11986, 2022.

Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. In Proceedings of the 37th International Conference on Machine Learning, pp. 9898–9907, 2020.

Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. arXiv preprint arXiv:1910.10252, 2019.

Shiqiang Wang and Mingyue Ji. A unified analysis of federated learning with arbitrary client participation. Advances in Neural Information Processing Systems, 35:19124–19137, 2022.

William Wei Wang, Gezheng Xu, Ruizhi Pu, Jiaqi Li, Fan Zhou, Changjian Shui, Charles Ling, Christian Gagné, and Boyu Wang. Evolving domain generalization. arXiv preprint arXiv:2206.00047, 2022.

Bingzhe Wu, Zhipeng Liang, Yuxuan Han, Yatao Bian, Peilin Zhao, and Junzhou Huang. Drflm: Distributionally robust federated learning with inter-client noise via local mixup. arXiv preprint arXiv:2204.07742, 2022.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 10(2):1–19, 2019.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In International Conference on Machine Learning, pp. 12073–12086. PMLR, 2021.

Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. arXiv preprint arXiv:2002.04758, 2020.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In International Conference on Machine Learning, pp. 7252–7261. PMLR, 2019.

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

Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. Knowledge-Based Systems, 216:106775, 2021. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2021.106775. URL `https://www.sciencedirect.com/science/article/pii/S0950705121000381`.

Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. Advances in Neural Information Processing Systems, 35:38629–38642, 2022.

Youshan Zhang and Brian D Davison. Adversarial continuous learning in unsupervised domain adaptation. In Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part II, pp. 672–687. Springer, 2021.

Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. Advances in neural information processing systems, 33:16096–16107, 2020.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. 2018. doi: 10.48550/ARXIV.1806.00582. URL `https://arxiv.org/abs/1806.00582`.

Chen Zhu, Zheng Xu, Mingqing Chen, Jakub Konečný, Andrew Hard, and Tom Goldstein. Diurnal or nocturnal? federated learning of multi-branch networks from periodically shifting distributions. In International Conference on Learning Representations, 2021.

A Algorithm

We present the pseudo-code for *FedEvolve* and *FedEvp* in Alg.3 and Alg.4. We randomly sample a subset of data from the dataset to train the model for each update instead of the whole dataset.

Algorithm 3 FedEvolve

```
Require: Number of clients $K$; client participation ratio $r$; step size $\eta$; the number of local training updates $\tau$; communication rounds $T$; the number of source domains $M$; initial global parameter $\phi$ and global parameter $\psi$ for representation function $f$; distance metric $d$; local datasets $\mathcal{D}_m^k$ and their known classes $\mathcal{Y}_{\mathcal{D}_m^k}$ for $m \in \{1, \dots, M\}, k \in \{1, \dots, K\}$.
for $t \in \{1, \dots, T\}$ do
    server samples $rK$ clients as $\mathcal{I}_t$ from all clients
    server sends $\phi$, $\psi$ to $\mathcal{I}_t$
    for each client $k \in \mathcal{I}_t$ in parallel do
        client $k$ initialize $\widetilde{\phi}_k := \phi$, $\widetilde{\psi}_k := \psi$
        for $\tau$ local training iterations do
            for $m \in \{1, \dots, M-1\}$ do
                $\mathcal{A} \leftarrow RandomSample(\mathcal{D}_m^k)$
                $\mathcal{B} \leftarrow RandomSample(\mathcal{D}_{m+1}^k)$
                for $y \in \mathcal{Y}_{\mathcal{D}_m^k}$ do
                    $\mathcal{A}_y \leftarrow \{(x_i, y_i) \in \mathcal{A} | y_i = y\}$
                    $C_{m,y}^k = \frac{1}{|\mathcal{A}_y|} \sum_{(x_i,y_i)\in\mathcal{A}_y} f_{\widetilde{\phi}_k}(x_i)$
                end for
                $\ell = 0$
                for $(x, y) \in \mathcal{B}$ do
                    $\ell = \ell - \frac{1}{|\mathcal{B}|}[\log \frac{\exp\left(-d\left(f_{\widetilde{\psi}_k}(x), C_{m,y}^k\right)\right)}{\sum_{y'\in\mathcal{Y}_{\mathcal{D}_m^k}} \exp\left(-d\left(f_{\widetilde{\psi}_k}(x), C_{m,y'}^k\right)\right))}]$
                end for
                $\widetilde{\phi}_k, \widetilde{\psi}_k = GradientDescent(\ell; \widetilde{\phi}_k, \widetilde{\psi}_k, \eta)$
            end for
        end for
        client $k$ sends local parameters $\widetilde{\phi}_k, \widetilde{\psi}_k$ to server
    end for
    $\phi = \frac{1}{|\mathcal{I}_t|} \sum_{k\in\mathcal{I}_t} \widetilde{\phi}_k$
    $\psi = \frac{1}{|\mathcal{I}_t|} \sum_{k\in\mathcal{I}_t} \widetilde{\psi}_k$
end for
Output $\phi$ and $\psi$
```

Algorithm 4 FedEvp

```
Require: Number of clients $K$; client participation ratio $r$; step size $\eta$; the number of local training updates $\tau$; communication rounds $T$; the number of source domains $M$; initial global parameter $\phi$ and global parameter $w$ for representation function $f$; distance metric $d$;; local datasets $\mathcal{D}_m^k$ and their known classes $\mathcal{Y}_{\mathcal{D}_m^k}$ for $m \in \{1, \dots, M\}, k \in \{1, \dots, K\}$.
for $t \in \{1, \dots, T\}$ do
    server samples $rK$ clients as $\mathcal{I}_t$ from all clients
    server sends $\phi$, $w$ to $\mathcal{I}_t$
    for each client $k \in \mathcal{I}_t$ in parallel do
        client $k$ initialize $\widetilde{\phi}_k := \phi$, $\widetilde{w}_k := w$
        for $\tau$ local training iterations do
            for $y \in \mathcal{Y}_{\mathcal{D}_m^k}$ do
                $C_{0,y}^k = 0$
            end for
            for $m \in \{1, \dots, M\}$ do
                $\mathcal{A} \leftarrow RandomSample(\mathcal{D}_m^k)$
                $\ell_e \leftarrow -\frac{1}{|\mathcal{A}|} \sum_{(x_i,y_i)\in\mathcal{A}} y_i \log \frac{\exp\left(g_{\widetilde{w}_k}^{y}\left(f_{\widetilde{\phi}_k}(x)\right)\right)}{\sum_{y'\in\mathcal{Y}_{\mathcal{D}_m^k}} \exp\left(g_{\widetilde{w}_k}^{y'}\left(f_{\widetilde{\phi}_k}(x)\right)\right)}$
                for $y \in \mathcal{Y}_{\mathcal{D}_m^k}$ do
                    $\mathcal{A}_y \leftarrow \{(x_i, y_i) \in \mathcal{A} | y_i = y\}$
                    $C_{m,y}^k = \frac{(m-1)}{m} C_{m-1,y}^k + \frac{1}{m}\frac{1}{|\mathcal{A}_y|} \sum_{(x_i,y_i)\in\mathcal{A}_y} f_{\widetilde{\phi}_k}(x_i)$
                end for
                if m≥2 then
                    $\ell_f = 0$
                    for $(x, y) \in \mathcal{A}$ do
                        $\ell_f = \ell_f - \frac{1}{|\mathcal{A}|} \log \frac{\exp\left(-d\left(f_{\widetilde{\phi}_k}(x), C_{m,y}^k\right)\right)}{\sum_{y'\in\mathcal{Y}_{\mathcal{D}_m^k}} \exp\left(-d\left(f_{\widetilde{\phi}_k}(x), C_{m,y'}^k\right)\right))}$
                    end for
                    $\widetilde{\phi}_k, \widetilde{w}_k = GradientDescent(\ell_f + \ell_e; \widetilde{\phi}_k, \widetilde{w}_k, \eta)$
                end if
            end for
        end for
        client $k$ sends local parameters $\widetilde{\phi}_k, \widetilde{w}_k$ to server
    end for
    $\phi = \frac{1}{|\mathcal{I}_t|} \sum_{k\in\mathcal{I}_t} \widetilde{\phi}_k$
    $w = \frac{1}{|\mathcal{I}_t|} \sum_{k\in\mathcal{I}_t} \widetilde{w}_k$
end for
Server Output $\phi$, $w$
for each client $k$ do
    Client Output $\widetilde{\phi}_k$, $\widetilde{w}_k$ = personalize($\phi$, $w$, $\mathcal{D}^k$)
end for
```

B Proof of Thm. 4.1

Proof. Denote $D_{\text{JS}}(P, Q)$ as $\sqrt{d_{\text{JS}}(P \| Q)}$. From Lemma 1 in Wang et al. (2022), we know that:

$$\sum_{k=1}^{K} \alpha_k L_{f_\psi(S_{M+1}^k)}(\widehat{h}) \leq \sum_{k=1}^{K} \alpha_k L_{f_\phi(S_M^k)}(\widehat{h}) + \sum_{k=1}^{K} \alpha_k \frac{G}{\sqrt{2}} D_{\text{JS}}(f_\psi(S_{M+1}^k) \| f_\phi(S_M^k))$$

Next, for each $m \in \{1, 2, ..., M-1\}$, we have:

$$D_{\mathrm{JS}}(f_\psi(S^k_{M+1}) \| f_\phi(S^k_M)) \leq D_{\mathrm{JS}}(f_\psi(S^k_{m+1}) \| f_\phi(S^k_m)) + |D_{\mathrm{JS}}(f_\psi(S^k_{M+1}) \| f_\phi(S^k_M)) - D_{\mathrm{JS}}(f_\psi(S^k_{m+1}) \| f_\phi(S^k_m))|$$

Moreover, since all the distributions have the same support (i.e., the representation space), we can apply the triangle inequality with respect to D_{JS}:

$$D_{\mathrm{JS}}(f_\psi(S^k_{m+1}) \| f_\phi(S^k_m)) \leq D_{\mathrm{JS}}(f_\psi(S^k_{m+1}) \| f_{\psi^*_k}(S^k_{m+1})) + D_{\mathrm{JS}}(f_{\psi^*_k}(S^k_{m+1}) \| f_{\phi^*_k}(S^k_m)) + D_{\mathrm{JS}}(f_{\phi^*_k}(S^k_m) \| f_\phi(S^k_m))$$

Plug these two equations into the first equation:

$$\begin{aligned}\sum_{k=1}^{K} \alpha_k L_{f_\psi(S^k_{M+1})}(\widehat{h}) \leq \sum_{k=1}^{K} \alpha_k L_{f_\phi(S^k_M)}(\widehat{h}) + \sum_{k=1}^{K} \alpha_k \frac{G}{\sqrt{2}} \Big(& D_{\mathrm{JS}}(f_{\psi^*_k}(S^k_{m+1}) \| f_{\phi^*_k}(S^k_m)) + D_{\mathrm{JS}}(f_{\phi^*_k}(S^k_m) \| f_\phi(S^k_m)) \\ & + D_{\mathrm{JS}}(f_\psi(S^k_{m+1}) \| f_{\psi^*_k}(S^k_{m+1})) + \big| D_{\mathrm{JS}}(f_\psi(S^k_{M+1}) \| f_\phi(S^k_M)) - D_{\mathrm{JS}}(f_\psi(S^k_{m+1}) \| f_\phi(S^k_m)) \big| \Big)\end{aligned}$$

We sum from $m = 1$ to $M - 1$ and average to get:

$$\begin{aligned}\sum_{k=1}^{K} \alpha_k L_{f_\psi(S^k_{M+1})}(\widehat{h}) \leq \sum_{k=1}^{K} \alpha_k L_{f_\phi(S^k_M)}(\widehat{h}) + \sum_{k=1}^{K} \alpha_k \frac{G}{\sqrt{2}(M-1)} \Bigg(\sum_{m=1}^{M-1} \Big(& D_{\mathrm{JS}}(f_{\psi^*_k}(S^k_{m+1}) \| f_{\phi^*_k}(S^k_m)) + D_{\mathrm{JS}}(f_\psi(S^k_{m+1}) \| f_{\psi^*_k}(S^k_{m+1})) \\ & + D_{\mathrm{JS}}(f_\phi(S^k_m) \| f_{\phi^*_k}(S^k_m)) + \big| D_{\mathrm{JS}}(f_\psi(S^k_{M+1}) \| f_\phi(S^k_M)) - D_{\mathrm{JS}}(f_\psi(S^k_{m+1}) \| f_\phi(S^k_m)) \big| \Big) \Bigg)\end{aligned}$$

Substitute D_{JS} with $\sqrt{d_{\mathrm{JS}}}$ and we get Thm. 4.1.

□

C Datasets

C.1 Rotated MNIST (Ghifary et al., 2015) and Rotated EMNIST (Ghifary et al., 2015)

For Rotated MNIST (RMNIST), We generate 12 domains by applying the rotations with angles of $\theta = \{0°, 15°, ..., 165°\}$ on each domain respectively. For Rotated EMNIST (REMNIST), we generate 12 domains by applying the rotations with angles of $\theta = \{0°, 8°, ..., 88°\}$ on each domain respectively.

C.2 Circle (Pesaranghader & Viktor, 2016)

We follow (Pesaranghader & Viktor, 2016) to generate this dataset. In this synthetic data set, we have 30 Gaussian distributions centered on a half circle with standard deviation 0.6, and the radius r is set to 10. Each data point has two attributes, and the number of classes is 2. The decision boundary is $(x - x_0)^2 + (y - y_0)^2 \leq r^2$, where (x_0, y_0) are the coordinates of the circle's center (we set it as $(0, 0)$).

C.3 Portraits (Ginosar et al., 2015)

The portraits dataset contains human face images from yearbooks spanning from 1905 to 2013. We partition the data into nine domains by segmenting the dataset into 12-year intervals. All images are resized into 32×32 without any augmentation.

C.4 Caltran(Hoffman et al., 2014)

This real surveillance dataset comprises images captured by a fixed traffic camera deployed in an intersection. The images in this dataset come with time attributes. We categorize the images into 12 distinct domains based on their capture time throughout the day. Specifically, each domain represents a 2-hour interval. As such, a 24-hour day is evenly divided into these 12 domains. We resize images in Caltran to 224×224.

D Implementation

All experiments are conducted on a server equipped with multiple NVIDIA A5000 GPUs, two AMD EPYC 7313 CPUs, and 256GB of memory. The code is implemented with Python 3.8 and PyTorch 1.13.0 on Ubuntu 20.04 based on the implementation in Jiang & Lin (2023). To evaluate our methods, we consider classification tasks using various network architectures and report average accuracy over three different random seeds. All experiments are conducted over 50 communication rounds. The personalization epoch is 1 for PFL methods, including *FedEvp*. For each dataset, we test both SGD and Adam optimizers and use the best one for the dataset. We tune the weight decay rate from {1e-5, 5e-5, 1e-4, 5e-4, 1e-3}.

- **RMNIST**: For the Rotated MNIST dataset, we employ a CNN with four convolutional layers, each equipped with a 3x3 kernel. Group Normalization is applied post-convolution for stabilization using groups of 8 channels. Followed by convolutional layers, there are two linear layers with a hidden dimension of 64. The four convolutional layers and the first linear layer form the representation functions (f_ϕ and f_ψ in *FedEvolve*, f_ϕ in *FedEvp*) with the final linear layer serving as the classifier (f_w in *FedEvp*). We employ an SGD optimizer with a weight decay of 5e-4 and conduct local training for 10 epochs per communication round.
- **REMNIST**: For the Rotated EMNIST dataset, we employ the same CNN as the one in RMNIST. We use Adam optimizer with weight decay of 1e-4 and run local training for 10 epochs per communication round.
- **Circle:** For the Circle dataset, we utilize a five-layer Multi-Layer Perceptron (MLP). This includes three dense layers (2x256, 256x256, 256x256) for feature representation (f_ϕ and f_ψ in FedEvolve, f_ϕ in *FedEvp*), linked by ReLU activations, and two subsequent linear layers (256x64, 64x2) that function as the classifier (f_w in *FedEvp*). Given data constraints, we utilize 10 clients and train for 5 epochs using Adam with a weight decay of 5e-4.
- **Portraits**: For this dataset, images are resized to 32x32 and processed using a WideResNet architecture. The convolution layers along with the average pooling layer act as the representation function. A linear layer is designated as the classifier after representation. The model is trained among 20 clients over 5 epochs using an Adam optimizer with a weight decay of 5e-4.
- **Caltran:** We deploy ResNet18 for the Caltran dataset, with the last linear layer used as the classifier. The representation function comprises four residual convolution blocks and an average pooling layer. With pre-trained weights, we tune the model using an SGD optimizer with a weight decay of 5e-4. Given data limitations, training involves 10 clients over 5 epochs.

Networks for each dataset are presented in Table 9.

For each dataset, we search the learning rate for each algorithm to find the best results. The training detail is given in Table 10. We use the grid search strategy for hyperparameters to tune the models.

- For GMA(Tenison et al., 2022), we set the masking threshold as 0.1, searching from $\{0.1, 0.2, 0.3, ..., 1.0\}$
- For FedRep(Collins et al., 2021), FedRod(Chen & Chao, 2022), and FedTHE(Jiang & Lin, 2023), the last fc layer of the model is used as the head.

Dataset	**Input Dimension**	**Number of Classes**	**Network**
RMNIST	28×28	10	CNN
REMNIST	28×28	26	CNN
Circle	2	2	MLP
Portraits	32×32	2	WideResNet
Caltran	$3 \times 224 \times 224$	2	ResNet18

Table 9: Networks for datasets

Dataset	**Num of Clients**	**Batch Size**	**Learning Rate Range**
RMNIST	20	32	1e-3, 1e-2, 1e-1
REMNIST	20	96	1e-3, 5e-3, 1e-2, 5e-2, 1e-1
Circle	10	32	1e-6, 5e-6, 1e-5, 5e-5, 1e-4
Portraits	20	32	1e-3, 5e-3, 1e-2
Caltran	10	32	1e-5, 5e-5, 1e-4, 5e-4

Table 10: Training Details for datasets

- For Ditto(Li et al., 2021), the regularization factor λ is set to 0.1, searching from $\{0.01, 0.05, 0.1, 0.5\}$.
- For MEMO,(Zhang et al., 2022) we use 32 augmentations and 3 optimization steps.
- For T3A(Iwasawa & Matsuo, 2021), $M = 50$ is used in our experiments, searching from $\{20, 50, 100\}$.
- For FedSR(Nguyen et al., 2022), we follow the same setting in their paper: $\alpha^{L2R} = 0.01$ and $\alpha^{CMI} = 0.001$.

E Supplementary Results

We compared the P-values of our proposed methods, *FedEvolve* and *FedEvp*, with various baseline federated learning algorithms in Table 11. The p-values from our t-test statistical analysis indicated that our methods significantly outperform the baseline methods.

Table 11: P-values comparing FedEvolve and FedEvp with baseline methods on rotated MNIST.

	FedAvg	GMA	Memo(G)	FedAvgFT	APFL	FedRep	Ditto
FedEvolve	5.17×10^{-4}	4.42×10^{-4}	7.69×10^{-4}	1.44×10^{-3}	2.26×10^{-4}	9.41×10^{-4}	7.27×10^{-4}
FedEvp	7.33×10^{-3}	6.76×10^{-3}	9.82×10^{-3}	2.42×10^{-3}	6.20×10^{-6}	4.60×10^{-4}	2.16×10^{-5}
	FedRod	Memo(P)	T3A	FedTHE	FedSR	CFL	CFeD
FedEvolve	6.21×10^{-4}	1.04×10^{-3}	8.09×10^{-4}	8.92×10^{-4}	6.27×10^{-4}	2.46×10^{-4}	1.41×10^{-3}
FedEvp	2.71×10^{-3}	1.20×10^{-3}	7.71×10^{-4}	2.27×10^{-4}	2.77×10^{-2}	4.04×10^{-3}	4.40×10^{-2}

E.1 Impact of Changing Pattern

In previous experiments, we primarily focus on invariant changing patterns in image rotation experiments. Here we examine if our methods are robust against an unexpected pattern. In this experiment, we test the robustness of our methods against an unexpected pattern. Specifically, we simulate an unexpected domain by rotating images from the target domain by an additional 10° and 20°. To prevent confusion between numerals like 6 and 9 when rotated by 180°, we set the incremental rotation degree as 10°. Therefore, the images experience a 120° rotation or a 130° rotation instead of the expected 110°. This experiment aims to evaluate whether our methods can handle deviations from anticipated patterns.

As shown in Table 12, all methods exhibit a significant performance drop when the test data distribution changes substantially, however, our methods still outperform the baseline and the drop is less than the baseline.

Table 12: Average accuracy over three runs of experiments on rotated MNIST with different rotation degrees for the target domain.

	Dir→ ∞		Dir=1.0		Dir=0.1	
Method	Client	Server	Client	Server	Client	Server
FedAvg (110°)	$80.96_{\pm 1.62}$	$80.92_{\pm 0.23}$	$79.41_{\pm 0.30}$	$81.63_{\pm 1.71}$	$70.16_{\pm 0.48}$	$71.93_{\pm 2.06}$
FedAvg (120°)	$66.39_{\pm 0.92}$	$66.61_{\pm 1.17}$	$64.30_{\pm 1.36}$	$65.08_{\pm 1.89}$	$54.74_{\pm 1.40}$	$53.62_{\pm 1.01}$
FedAvg (130°)	$50.53_{\pm 0.88}$	$51.04_{\pm 2.72}$	$48.64_{\pm 1.28}$	$48.41_{\pm 1.16}$	$40.87_{\pm 1.44}$	$40.41_{\pm 0.91}$
FedEvolve (110°)	$86.66_{\pm 0.66}$	$86.62_{\pm 1.60}$	$85.57_{\pm 1.35}$	$83.11_{\pm 2.31}$	$86.92_{\pm 1.72}$	$74.67_{\pm 2.29}$
FedEvolve (120°)	$78.09_{\pm 0.88}$	$77.80_{\pm 0.82}$	$74.85_{\pm 2.86}$	$72.81_{\pm 5.46}$	$78.43_{\pm 4.92}$	$61.57_{\pm 7.44}$
FedEvolve (130°)	$65.13_{\pm 1.79}$	$64.64_{\pm 1.85}$	$62.88_{\pm 2.23}$	$60.47_{\pm 3.91}$	$69.77_{\pm 4.30}$	$50.82_{\pm 5.58}$
FedEvp (110°)	$84.68_{\pm 1.51}$	$86.07_{\pm 1.38}$	$85.84_{\pm 1.33}$	$83.33_{\pm 3.50}$	$84.99_{\pm 2.32}$	$69.41_{\pm 2.47}$
FedEvp (120°)	$72.91_{\pm 0.65}$	$75.66_{\pm 0.82}$	$74.93_{\pm 2.54}$	$71.82_{\pm 2.83}$	$79.48_{\pm 1.89}$	$62.28_{\pm 3.88}$
FedEvp (130°)	$61.67_{\pm 0.31}$	$64.64_{\pm 0.16}$	$65.79_{\pm 2.36}$	$62.99_{\pm 2.11}$	$72.11_{\pm 3.01}$	$53.84_{\pm 4.05}$

Table 13: Average accuracy over three runs of experiments on Reddit text classification.

	Dir→ ∞		Dir=1.0		Dir=0.1	
Method	Client	Server	Client	Server	Client	Server
FedAvg	$62.42_{\pm 0.68}$	$62.42_{\pm 0.68}$	$61.37_{\pm 0.38}$	$61.37_{\pm 0.38}$	$57.34_{\pm 0.86}$	$57.34_{\pm 0.86}$
T3A	$62.13_{\pm 0.48}$	$63.53_{\pm 0.15}$	$62.71_{\pm 0.71}$	$57.50_{\pm 0.04}$	$71.52_{\pm 0.78}$	$53.42_{\pm 1.12}$
FedTHE	$62.22_{\pm 0.44}$	$62.18_{\pm 0.32}$	$63.57_{\pm 0.29}$	$61.15_{\pm 0.39}$	$68.70_{\pm 0.74}$	$57.78_{\pm 1.04}$
CFeD	$62.31_{\pm 0.71}$	$62.31_{\pm 0.71}$	$61.28_{\pm 0.58}$	$61.28_{\pm 0.58}$	$56.36_{\pm 1.46}$	$56.36_{\pm 1.46}$
FedEvolve	$63.00_{\pm 0.58}$	$65.62_{\pm 0.82}$	$67.84_{\pm 1.58}$	$61.75_{\pm 1.39}$	$72.38_{\pm 1.09}$	$52.84_{\pm 1.62}$
FedEvp	$63.46_{\pm 1.82}$	$62.75_{\pm 1.39}$	$73.68_{\pm 0.62}$	$60.99_{\pm 1.50}$	$78.16_{\pm 1.23}$	$56.64_{\pm 2.01}$

Notably, *FedEvp* demonstrates superior performance compared to *FedEvolve* when clients are heterogeneous. This difference arises because *FedEvolve* explicitly learns the distribution transition between consecutive domains, while *FedEvp* learns evolving-domain-invariant features. Consequently, when the distribution transition deviates from the learned pattern, the performance of *FedEvolve* is adversely affected, whereas *FedEvp* remains less influenced by the change.

E.2 Experiment on Text Dataset

To evaluate the generality of our methods beyond image modality, we conduct experiments on a real-world text dataset from Reddit Baumgartner et al. (2020). We partition Reddit post titles from 2014 to 2018 into 10 domains based on 6-month intervals, capturing natural temporal shifts.

We focus on five subreddits that exhibit meaningful evolution over time: `AskReddit`, `news`, `technology`, `movies`, and `worldnews`. These are used as the classification labels. For each domain, we sample 5,000 examples, resulting in a total of 50,000 samples. We distribute this data across 20 clients for federated training. For the model architecture, we use a two-layer bidirectional LSTM (BiLSTM) with a hidden dimension of 256 for text classification. Each method is trained for 10 local epochs per round using the Adam optimizer over 50 rounds. As shown in Table 13, our methods (*FedEvolve* and *FedEvp*) still consistently outperform baselines across all heterogeneity levels. These results demonstrate that our approach generalizes well to the evolving patterns in text data.

E.3 Study of Distance Metrics

To examine the impact of the choice of distance metric in prototype alignment, we evaluate FedEvolve using two common similarity measures: L2 distance and cosine distance. The results across multiple datasets and Dirichlet heterogeneity levels are summarized in Table 14.

Overall, we observe that cosine distance performs better than L2 distance on digit datasets like **RMNIST** and **REMNIST**, while on other datasets such as **Circle** and **Portraits**, L2 distance shows more stable or superior performance. This suggests that the effectiveness of a distance metric may depend on dataset-specific characteristics, including feature distributions and the nature of the domain shifts.

Table 14

FedEvolve	Distance	Dir→ ∞		Dir=1.0		Dir=0.1	
		Client	Server	Client	Server	Client	Server
RMNIST	L2	$84.75_{\pm 1.39}$	$84.43_{\pm 1.21}$	$79.93_{\pm 1.00}$	$77.25_{\pm 1.82}$	$83.86_{\pm 1.81}$	$71.66_{\pm 1.95}$
	Cosine	$88.17_{\pm 0.97}$	$87.34_{\pm 0.99}$	$84.45_{\pm 0.96}$	$84.40_{\pm 1.40}$	$91.73_{\pm 1.35}$	$82.98_{\pm 3.11}$
REMNIST	L2	$83.58_{\pm 1.45}$	$82.91_{\pm 1.36}$	$82.13_{\pm 0.48}$	$78.68_{\pm 0.25}$	$87.67_{\pm 0.55}$	$72.85_{\pm 1.03}$
	Cosine	$82.29_{\pm 0.52}$	$82.29_{\pm 0.29}$	$83.06_{\pm 0.23}$	$78.97_{\pm 0.28}$	$88.70_{\pm 0.99}$	$78.33_{\pm 1.22}$
Circle	L2	$84.25_{\pm 2.45}$	$81.64_{\pm 1.95}$	$82.52_{\pm 1.94}$	$83.59_{\pm 5.91}$	$85.20_{\pm 5.24}$	$84.60_{\pm 7.73}$
	Cosine	$75.95_{\pm 4.78}$	$76.28_{\pm 4.33}$	$74.20_{\pm 3.38}$	$74.78_{\pm 4.40}$	$71.45_{\pm 4.40}$	$72.45_{\pm 6.38}$
Portraits	L2	$95.43_{\pm 0.17}$	$96.88_{\pm 1.35}$	$93.84_{\pm 1.62}$	$96.54_{\pm 1.39}$	$93.03_{\pm 0.43}$	$96.08_{\pm 0.30}$
	Cosine	$94.56_{\pm 1.03}$	$94.81_{\pm 1.53}$	$92.95_{\pm 1.28}$	$93.94_{\pm 2.82}$	$91.03_{\pm 1.64}$	$92.33_{\pm 1.22}$
Caltran	L2	$65.04_{\pm 1.66}$	$63.54_{\pm 0.74}$	$75.04_{\pm 4.03}$	$64.06_{\pm 3.83}$	$84.07_{\pm 5.76}$	$69.88_{\pm 3.14}$
	Cosine	$60.49_{\pm 1.74}$	$59.77_{\pm 4.19}$	$75.30_{\pm 1.51}$	$64.45_{\pm 6.93}$	$81.83_{\pm 6.51}$	$61.94_{\pm 8.21}$
FedEvp	Distance	Dir→ ∞		Dir=1.0		Dir=0.1	
		Client	Server	Client	Server	Client	Server
RMNIST	L2	$75.99_{\pm 0.31}$	$77.63_{\pm 1.99}$	$77.91_{\pm 1.80}$	$73.85_{\pm 1.53}$	$83.15_{\pm 0.49}$	$61.84_{\pm 3.34}$
	Cosine	$80.47_{\pm 0.98}$	$82.40_{\pm 1.32}$	$82.23_{\pm 1.88}$	$82.18_{\pm 1.37}$	$76.07_{\pm 1.25}$	$68.86_{\pm 2.05}$
REMNIST	L2	$67.30_{\pm 1.35}$	$71.94_{\pm 1.50}$	$73.61_{\pm 1.70}$	$68.91_{\pm 0.30}$	$87.01_{\pm 0.22}$	$58.73_{\pm 0.96}$
	Cosine	$71.09_{\pm 1.84}$	$75.72_{\pm 3.14}$	$74.30_{\pm 2.42}$	$70.09_{\pm 2.82}$	$81.58_{\pm 1.07}$	$57.89_{\pm 1.28}$
Circle	L2	$73.30_{\pm 5.02}$	$74.12_{\pm 6.93}$	$74.80_{\pm 1.69}$	$77.93_{\pm 4.20}$	$91.87_{\pm 2.82}$	$89.30_{\pm 4.67}$
	Cosine	$70.62_{\pm 3.00}$	$69.11_{\pm 4.03}$	$68.29_{\pm 2.04}$	$60.14_{\pm 5.07}$	$74.29_{\pm 2.73}$	$71.76_{\pm 4.14}$
Portraits	L2	$93.54_{\pm 0.19}$	$94.92\ _{\pm 0.11}$	$94.50_{\pm 0.28}$	$93.91_{\pm 2.19}$	$94.24_{\pm 0.65}$	$93.13_{\pm 0.57}$
	Cosine	$93.82_{\pm 0.42}$	$95.08_{\pm 0.41}$	$94.02_{\pm 2.88}$	$93.38_{\pm 4.59}$	$93.31_{\pm 1.47}$	$92.38_{\pm 1.50}$
Caltran	L2	$66.59_{\pm 1.44}$	$66.34_{\pm 0.69}$	$73.46_{\pm 0.90}$	$68.24_{\pm 1.08}$	$78.55_{\pm 0.50}$	$67.60_{\pm 3.15}$
	Cosine	$64.21_{\pm 1.95}$	$65.37_{\pm 1.75}$	$72.99_{\pm 2.28}$	$67.56_{\pm 2.09}$	$76.46_{\pm 2.70}$	$66.71_{\pm 2.58}$

Table 15: Comparison of FedAvg and Local-Only Training (Dir → ∞).

Method	RMNIST	REMNIST	Circle	Portraits	Caltran
FedAvg	$65.92_{\pm 1.01}$	$53.83_{\pm 1.84}$	$70.40_{\pm 6.51}$	$94.10_{\pm 0.13}$	$62.93_{\pm 2.10}$
Local	$59.04_{\pm 0.78}$	$44.53_{\pm 1.36}$	$61.84_{\pm 2.34}$	$88.10_{\pm 0.31}$	$59.28_{\pm 2.30}$

E.4 Impact of Collaboration

To study the impact of collaborative learning, we introduce a local-only baseline, where each client trains its model solely on its own data without any communication or aggregation and tests on its own data. We compare this to FedAvg under the i.i.d. setting (Dir → ∞), where data is randomly distributed across clients, ensuring that both local and federated models are exposed to similar data diversity. Table 15 presents performance on five datasets. Across all tasks, FedAvg significantly outperforms the local-only baseline, confirming that collaboration yields better generalization in our problem.

E.5 Impact of Client Numbers

We follow the setting in Jiang & Lin (2023) and initially evaluate performance with 20 clients. To examine the scalability and performance under varying degrees of client participation, we scale up the number of clients to 60. Due to data availability constraints of real-world datasets, we use the MNIST dataset as a representative dataset, given its larger sample size per class. As shown in Table 16, increasing the number of clients does not necessarily lead to significant performance degradation, suggesting that these methods maintain stable accuracy even as the client population grows.

F Limitations and Future Works

- The theoretical analysis and core design of the methods assume that the data evolution follows a predictable pattern that can be learned. This assumption may not hold in all real-world scenarios. Therefore, we also validate our methods on datasets like Portraits, which reflect changes in fashion, or Caltran, where shifts are caused by natural light changes, and the evolution is not subject to a

Table 16: Comparison of average accuracy on rotated MNIST under different client settings (20 and 60 clients).

	Dir→ ∞		Dir=1.0		Dir=0.1	
Method	Client	Server	Client	Server	Client	Server
FedAvg (20)	$65.92_{\pm 1.01}$	$66.34_{\pm 0.34}$	$62.35_{\pm 0.97}$	$63.16_{\pm 1.78}$	$51.68_{\pm 0.73}$	$51.59_{\pm 2.48}$
FedAvg (60)	$64.44_{\pm 1.69}$	$63.50_{\pm 3.34}$	$61.06_{\pm 0.58}$	$60.74_{\pm 1.23}$	$50.82_{\pm 1.17}$	$50.15_{\pm 1.52}$
FedTHE (20)	$66.84_{\pm 1.51}$	$67.43_{\pm 0.23}$	$67.98_{\pm 0.43}$	$62.55_{\pm 1.98}$	$78.52_{\pm 3.92}$	$53.40_{\pm 0.74}$
FedTHE (60)	$65.39_{\pm 0.97}$	$63.82_{\pm 1.23}$	$67.17_{\pm 0.62}$	$61.27_{\pm 2.57}$	$75.10_{\pm 1.56}$	$52.11_{\pm 1.06}$
T3A (20)	$53.94_{\pm 0.76}$	$66.61_{\pm 0.59}$	$61.60_{\pm 2.49}$	$62.61_{\pm 1.02}$	$71.73_{\pm 1.63}$	$51.59_{\pm 1.70}$
T3A (60)	$57.74_{\pm 1.21}$	$63.71_{\pm 1.12}$	$62.15_{\pm 0.87}$	$60.89_{\pm 1.07}$	$75.63_{\pm 0.95}$	$50.96_{\pm 0.92}$
FedEvolve (20)	$84.75_{\pm 1.39}$	$84.43_{\pm 1.21}$	$79.93_{\pm 1.00}$	$77.25_{\pm 1.82}$	$83.86_{\pm 1.81}$	$71.66_{\pm 1.95}$
FedEvolve (60)	$83.06_{\pm 2.27}$	$82.03_{\pm 2.09}$	$79.28_{\pm 1.82}$	$77.15_{\pm 1.46}$	$81.92_{\pm 2.56}$	$68.37_{\pm 3.22}$
FedEvp (20)	$75.99_{\pm 0.31}$	$77.63_{\pm 1.99}$	$77.91_{\pm 1.80}$	$73.85_{\pm 1.53}$	$83.15_{\pm 0.49}$	$61.84_{\pm 3.34}$
FedEvp (60)	$74.09_{\pm 1.84}$	$78.72_{\pm 3.14}$	$76.30_{\pm 3.42}$	$72.09_{\pm 2.82}$	$81.58_{\pm 1.07}$	$59.89_{\pm 1.28}$

Table 17: Accuracy of baselines across various datasets (Dir=0.1).

	Circle		Portraits		Caltran	
	Client	Server	Client	Server	Client	Server
FedAvg	$77.43_{\pm 6.29}$	$77.43_{\pm 6.29}$	$95.08_{\pm 0.68}$	$95.08_{\pm 0.68}$	$64.23_{\pm 3.62}$	$64.15_{\pm 5.55}$
GMA	$79.40_{\pm 6.84}$	$79.40_{\pm 6.84}$	$95.49_{\pm 0.21}$	$95.49_{\pm 0.21}$	$66.61_{\pm 0.78}$	$66.81_{\pm 0.95}$
Memo(G)	-	-	$94.86_{\pm 0.21}$	$94.86_{\pm 0.21}$	$64.80_{\pm 3.76}$	$64.15_{\pm 5.55}$
FedAvgFT	$67.92_{\pm 7.45}$	$74.52_{\pm 8.94}$	$93.78_{\pm 0.52}$	$94.99_{\pm 0.12}$	$63.17_{\pm 4.07}$	$63.74_{\pm 6.02}$
APFL	$67.37_{\pm 6.46}$	$77.55_{\pm 7.35}$	$92.54_{\pm 0.36}$	$95.04_{\pm 0.48}$	$77.46_{\pm 5.75}$	$63.66_{\pm 5.95}$
FedRep	$66.73_{\pm 3.44}$	$70.40_{\pm 7.80}$	$93.69_{\pm 0.28}$	$95.06_{\pm 0.39}$	$65.61_{\pm 3.09}$	$64.07_{\pm 6.05}$
Ditto	$70.30_{\pm 3.75}$	$77.55_{\pm 7.35}$	$93.09_{\pm 0.55}$	$95.33_{\pm 0.02}$	$63.17_{\pm 4.07}$	$63.74_{\pm 6.02}$
FedRod	$69.60_{\pm 4.12}$	$64.37_{\pm 9.55}$	$92.20_{\pm 0.14}$	$95.06_{\pm 0.22}$	$77.73_{\pm 3.70}$	$60.01_{\pm 3.16}$
T3A	$70.57_{\pm 7.07}$	$78.30_{\pm 9.63}$	$92.72_{\pm 0.03}$	$94.83_{\pm 0.11}$	$63.33_{\pm 4.11}$	$63.74_{\pm 6.02}$
Memo(P)	-	-	$95.35_{\pm 0.90}$	$95.21_{\pm 0.28}$	$65.85_{\pm 1.96}$	$63.74_{\pm 6.02}$
FedTHE	$83.38_{\pm 8.65}$	$80.08_{\pm 7.88}$	$94.10_{\pm 0.89}$	$95.45_{\pm 0.26}$	$79.71_{\pm 5.05}$	$58.01_{\pm 5.16}$
Flute	$69.26_{\pm 8.15}$	$66.97_{\pm 7.84}$	$93.11_{\pm 0.28}$	$94.22_{\pm 0.38}$	$64.96_{\pm 1.87}$	$67.64_{\pm 1.04}$
FedSR	$76.77_{\pm 5.55}$	$77.24_{\pm 6.07}$	$94.02_{\pm 0.67}$	$94.38_{\pm 0.89}$	$60.16_{\pm 3.91}$	$65.77_{\pm 1.40}$
CFL	$70.12_{\pm 5.17}$	$71.09_{\pm 6.33}$	$92.40_{\pm 1.41}$	$92.40_{\pm 1.41}$	$65.12_{\pm 3.40}$	$65.20_{\pm 5.34}$
CFeD	$74.97_{\pm 8.08}$	$74.97_{\pm 8.08}$	$94.66_{\pm 0.42}$	$94.66_{\pm 0.42}$	$63.90_{\pm 4.39}$	$63.54_{\pm 6.22}$
FedEvolve	$85.20_{\pm 5.24}$	$84.60_{\pm 7.73}$	$93.03_{\pm 0.43}$	$96.08_{\pm 0.30}$	$84.07_{\pm 5.76}$	$69.88_{\pm 3.14}$
FedEvp	$91.87_{\pm 2.82}$	$89.30_{\pm 4.67}$	$94.24_{\pm 0.65}$	$93.13_{\pm 0.57}$	$78.55_{\pm 0.50}$	$67.60_{\pm 3.15}$

simple, structured pattern. Since the FedEvolve relies on learning patterns between two consecutive distributions, it may underperform when domain shifts are chaotic or abrupt. In addition, in this paper, we only study the domain distribution shift in training data and test data, without considering label or concept drift. Our methods may not be robust under these unevaluated drifts. Future directions include modeling abrupt shifts and bridging our approach with other shifts in machine learning to build a more robust FL system.

- The FedEvolve algorithm incurs significant overhead due to its use of two distinct representation functions. This design results in roughly double the number of parameters to transmit between the server and clients, leading to increased communication costs and training time. Furthermore, in settings where differential privacy is applied, more parameters may require additional noise to meet a given privacy budget, potentially affecting model utility. While the FedEvp was developed specifically to be a more efficient alternative with comparable overhead to standard baselines, the high cost of FedEvolve remains a practical limitation for those edge devices with relatively less computational resources.
- The methods use prototypical learning to align representations across domains. The prototypes for each class are calculated as the mean of the feature representations from that class based on L2 distance. This approach focuses on matching the first moment (the mean) of the distributions and may not effectively capture more complex shifts where the mean is stable but the variance or other higher-order moments change. A potential method is to consider the Mahalanobis distance, which considers the covariance between features. In addition to distance metrics, it is also possible to

consider other variance-based measures (e.g., KL divergence assuming Gaussian distributions around prototypes) to better capture evolving shifts in future studies.