pMIXFED: MIXING UP MODEL COEFFICIENTS FOR EFFICIENT PERSONALIZED FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning enables decentralized collaborative learning of machine learning models which presents challenges such as data privacy and client drift for heterogeneous data. Traditional FL methods offer strong generalization but lack personalized solutions for non-IID data. Personalized federated learning (PFL) addresses data heterogeneity by tackling these issues through balancing generalization and personalization level. It, however, still faces challenges such as optimal model partitioning and catastrophic forgetting that reduce quality and accuracy of both local and global models. To address these challenges, we propose "pMixFed", a dynamic, layer-wise PFL approach integrating mixup between shared global and personalized local models. We develop adaptive partitioning between shared and personalized layers of the model, gradual transition of personalization to allow seamless adaptation of local clients, improved generalization across clients, and mitigation of catastrophic forgetting. We provide theoretical analysis of pMixFed. Further, we conduct extensive experiments to demonstrate its superior performance compared with the existing PFL methods. Empirical results hows faster training, increased robustness, and improved handling of heterogeneity when using pMixFed as compared with the state-of-the-art PFL models.

1 INTRODUCTION

The goal in federated learning (FL) Konečnỳ et al. (2016) is to facilitate collaborative learning of several machine learning (ML) models in a decentralized scheme. FL requires addressing data privacy, catastrophic forgetting, and client drift Huang et al. (2022); Singhal et al. (2021); Luo et al. (2023); Qu et al. (2022). Existing FL methods cannot address all these challenges with non-IID data. For instance, although "FedAvg" McMahan et al. (2017) demonstrates strong generalization performance, it fails to provide personalized solutions for a cohort of clients with non-IID datasets. Hence, the global model, or the "average client", may not adequately represent all individual local models in non-IID settings due to client-drift Xiao et al. (2020). Personalized FL (PFL) methods handle data heterogeneity by considering both generalization and personalization during the training stage. Since, there is a trade-off between generalization and personalization in heterogeneous environments, PFL methods leverage heterogeneity and diversity as advantages rather than adversities Pye & Yu (2021); Tan et al. (2022). A group of PFL approaches train personalized local models on each device while collaborating toward a shared global model. Partial PFL, also known as parameter decoupling, involves using a partial model sharing, where only a subset of the model is shared while other parameters remain "frozen" to balance generalization and personalization.

While partial PFL methods are effective in mitigating catastrophic forgetting, strengthening privacy and reducing computation and communication overhead Pillutla et al. (2022); Sun et al. (2023), there are still some unaddressed challenges. First, the answer to *when, where, and how to optimally partition(split) the full model?* is not clear. Recent studies Pillutla et al. (2022); Sun et al. (2023) demonstrated that optimal partitioning architecture also depends on factors such as task type, local model architecture, and device capabilities. An improper partitioning choice could result in underfitting or overfitting, more bias, and catastrophic forgetting. Further, the use of a fixed partitioning strategy across all communication rounds for heterogeneous clients can limit the efficacy of collaborative learning. For instance, if the performance of client suddenly drops due to new incoming data, the partitioning strategy should be changed because the client requires more frozen layers. Another issue is catastrophic forgetting of the previously shared global knowledge after only a few rounds

of local training because the shared global model can be completely overwritten by local updates leading generalization degradation Luo et al. (2023); Huang et al. (2022); Xu et al. (2022). Most importantly, partial models may experience slower convergence compared to full model personalization, as frozen local model updates can diverge in an opposite direction from the globally-shared model. Since the generalized and personalized models are trained on non-IID datasets, there might also be a domain shift, leading to model discrepancy as depicted in Figure 1. These discrepancies arise from variations in local and global objective functions, differences in initialization, and asynchronous updates Yang et al. (2024); Lee et al. (2023). As a result, merging the shared and the personalized layers can disrupt information flow within the network, impede the learning process, and lead to a slower convergence rate. Further, while partial PFL techniques contribute to an overall improved training accuracy, they can reduce the test accuracy on some devices, particularly in devices with limited samples, leading to variations in results in terms of the performance level Pillutla et al. (2022). Hence, there is a need to novel solutions to achieve the following:

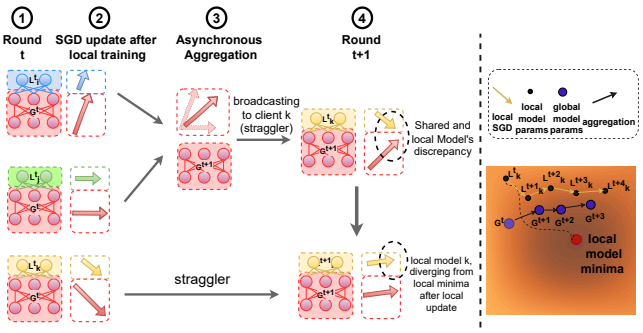

Figure 1: Proposed PFL approach: **(Left)** The global model G^t is formed by aggregating asynchronous local updates L_i^t, L_j^t, and L_k^t. After local training in communication round t, available clients i, j aggregate shared parameters to update the global model G^{t+1}, while the personalized parameters L_k^t remain frozen. Merging these distinct models G^{t+1}, and L_k^t, introduces inconsistencies in model updates. **(Right)** During joint training of generalized and personalized models, the gradients vector from the generalized part may conflict with the gradients from the personalized model, leading to model discrepancy, and slower convergence.

- Dynamic and Adaptive Partitioning: The balance between shared and personalized layers should be dynamically and adaptively adjusted for each client during every communication round, rather than relying on a static suboptiaml partitioning strategy.
- Gradual Personalization Transition: The degree of personalization should transition gradually across layers, as opposed to employing strict partitioning or hard splits which has been depicted in Figure 1. This ability allows nuanced adaptation for individual client needs.
- Improved Generalization Across Clients: The test accuracy should be optimized such that that the global model is unbiased toward specific devices or subsets of clients.
- Mitigation of Catastrophic Forgetting: The strategy should address the issue of catastrophic forgetting by incorporating mechanisms to retain memory of the previous global model.
- Scalability and Adaptability: The approach should be fast, scalable, and easily adaptable to new cold-start clients, while also accounting for model and device heterogeneity.

We propose "pMixFed", a layer-wise, dynamic PFL approach that integrates mixup Zhang et al. (2017) between the shared global and personalized local models' layers during both the broadcasting (global model sharing with local clients) and aggregation (aggregating distributed local models to update the global model) stages within a partial PFL framework. Our main contributions include:

- We develop an online and dynamic interpolation method between local and global models using Mixup Yoon et al. (2021), simultaneously handling data and model heterogeneity.
- Our solutions facilitates a gradual increase in the degree of personalization across layers, rather than relying on a strict cut-off layer, which helps mitigating the client drift problem.
- We introduce a new aggregation technique that is fast and efficient, and it addressed the catastrophic forgetting by keeping the previous global model state.
- "pMixFed" reduces the participant gap (test accuracy for cold-start users) and the out-of-sample gap (test accuracy on unseen data) caused by data heterogeneity through linear interpolation between client updates, thereby mitigating the impact of client drift.

For a discussion on prior works, please refer to the **Appendix A**.

2 PROBLEM FORMULATION

Consider K collaborating clients (or agents), each trying to optimize a local loss function $F_k(\theta)$ on the distributed local dataset $D_k = (x_k, y_k)$, where (x, y) shows the data features and the corresponding labels, respectively. Since the agents collaborate, the parameters θ (parameters of the global model) are shared across the agents. A basic FL objective function aims to optimize:

$$\min_{\theta} F(\theta) = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} F_k(\theta) \qquad , \qquad F_k(\theta) = \frac{1}{|\mathcal{D}_k|} \sum_{(x,y_i)\in\mathcal{D}_k} \ell(f_k(\theta, x_i), y_i), \tag{1}$$

where $|\mathcal{D}| = \sum_{k=1}^{K} |\mathcal{D}_k|$ and $F(\theta)$ is the global loss function of ***FedAvg*** McMahan et al. (2017). FL is performed in at iterative fashion. At each round, each client downloads the current version of the global model and trains it using their local data. Clients then send the updated model parameters to the central server. The central server receives the model updates from the selected clients and aggregates them to update the global model. This iterative process continues until convergence

In Equation 1, the assumption is that the data is collected from an IID distribution, and all clients should train their data according to the exact same model. However, this assumption cannot be applied within many practical FL settings due to the non-IID nature of data Imteaj et al. (2021); Imteaj & Amini (2021). In the the PFL settings with high heterogeneity and non-IID data distribution, the same issue persists. In PFL, the local parameters need to be customized toward each agent. PFL extends FL by solving the following objective Li et al. (2021a); Pillutla et al. (2022):

$$\min_{\theta,\theta_k} \sum_{k=1}^{K} \frac{1}{|\mathcal{D}_k|} (\mathcal{F}_k(\theta_k) + \alpha_k \|\theta_{\mathbf{k}} - \theta\|^2). \tag{2}$$

PFL explicitly handles data heterogeneity through the term $\mathcal{F}_k(\theta_k)$ which accounts for model heterogeneity by considering personalized parameter θ_k for client k. Meanwhile, θ represents the shared global model parameters, and α_k indicates the degree of personalization tuning collaborative learning between personalized local models θ_k and generalized global model θ.

2.0.1 PARTIAL PERSONALIZED MODEL

The limitations of full model personalization methods with global and fully independent local models are discussed in Appendix A. Personalized PFL methods improve PFL by providing more flexibility through allowing clients to choose which parts of their models to be personalized based on their specific needs and constraints, leading to potentially better performance. Let L_k^t be a partial local model k in round t which is partitioned into two parts $\langle L_{l,k}^t; L_{g,k}^t \rangle$, where $l, g \subseteq \{1, \dots M\}$ are the personalized and global layers, respectively, and M is the number of layers. We can integrate both personalized and generalized layers in a local model L_k as:

$$\mathcal{F}_k(\theta_k) = \ell(f_k(\langle L_{g,k}; L_{l,k} \rangle, x_k), y_k) \tag{3}$$

Among different partitioning strategies for partial PFL Pillutla et al. (2022), the most popular technique is to assign local personalized layers $L_{l,k}^t$ to final layers and allow the base layers $L_{g,k}^t$ to share the knowledge similar to ***FedPer*** Arivazhagan et al. (2019). This choice aligns with insights from MAML [1] algorithm, suggesting that initial layers keeps general information while personalized characteristics manifest prominently in the higher layers. Accordingly, we would have:

$$\mathcal{F}_k(\theta_k) = L_l^{(t)}(L_g^{(t)}(x_k)) \xrightarrow{localupdate} L_l'(L_g'(x_k)) \xrightarrow{broadcasting} L_l^{(t+1)}(G^{(t+1)}(x_k)) \tag{4}$$

For simplicity $L_g = L_{g,k}$ and $L_l = L_{g,k}$ where $\{1 \leq g \leq s \leq l \leq M\}$ and s is the split(cut) layer. The objective in solving Equation 4 is to find the optimal s (cut layer) which minimizes the personalization objective: $\sum_{k=1}^{K} \frac{1}{|\mathcal{D}_k|} \mathcal{F}_k(\theta_k) = L_l^{(t)}(L_g^{(t)}(x_k))$.

In partial models, after several rounds of local training, both personalized and global layers of local model are updated. This update could be synchronous like ***FedSim*** or asynchronous as in ***FedAlt***. The Personalized layers will be frozen until the next communication round, $L_l^{(t+1)} = L_l'$ and the global layers will be sent to the server for global model aggregation : $G^{(t+1)} \leftarrow \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} L_g'(x_k)$. In the next broadcasting phase, the shared layers of the local model will be updated as $L_g^{(t+1)} \leftarrow G^{(t+1)}$.

[1] Model-Agnostic Meta-Learning

3 pMIXFED : PARTIAL MIXED UP PERSONALIZED FEDERATED LEARNING

3.1 MIXUP

Mixup is a data augmentation technique for enhancing model generalization Zhang et al. (2017) based on learning to generalize on linear combinations of training examples. Variations of Mixup have consistently excelled in vision tasks, contributing to improved robustness, generalization, and adversarial privacy. Eq.5 demonstrates the Mixup formula for creating augmented samples:

$$\begin{aligned} \bar{x} &= \lambda.x_i + (1-\lambda).x_j \\ \bar{y} &= \lambda.y_i + (1-\lambda).y_j, \end{aligned} \tag{5}$$

where x_i and x_j are two input samples, y_i and y_j are the corresponding labels, λ, $\lambda \sim \text{Beta}(\alpha, \alpha), \lambda \in [0, 1]$, is the degree of interpolation between the two samples where $\alpha = 0$ acts similar to empirical risk minimization (ERM). The generated samples then are used for training.

Mixup relates data points belonging to different classes which has been shown to be successful in mitigating overfitting and improving model generalization Verma et al. (2019); Guo et al. (2019); Zhang et al. (2020). This linear interpolation also serves as a regularization technique that shapes smoother decision boundaries, thereby enhancing the ability of a trained model to generalize to unseen data. Mixup can also increase robustness against adversarial attacks Zhang et al. (2020); Beckham et al. (2019); and improves performance against noise, corrupted labels, and uncertainty as it relaxes the dependency on specific information Guo et al. (2019). Further, early stopping can be effectively employed as Mixum accelerates the training process without compromising the model performance Zou et al. (2023).

3.2 METHODOLOGY

Our goal is to leverage the well-established benefits of Mixup in the context of personalized FL. While Mixup has previously been employed in FL frameworks, such as XORMixup Shin et al. (2020), FEDMIX Yoon et al. (2021), and FedMix Wicaksana et al. (2022), prior studies have primarily focused on using Mixup for data augmentation or data averaging. We propose **pMixFed** by integrating Mixup on the model parameter space, rather using it on the feature space. We apply Mixup between the parameters of the global and the local models in a layer-wise manner for more customized and adaptive PFL. Our approach eliminates the need for static and rigid partitioning strategies. Specifically, during both the broadcasting and aggregation stages of our partial PFL framework, we generate mixed model weights using a an interpolation strategy which is illustrated in Figure 2. Mixup offers the flexibility in combining models by introducing a mix degree for each layer λ_i, which changes gradually according to μ, i.e., the *mix factor*. Parameter μ is also updated adaptively in each communication round and for each client according to the test accuracy of the global and the local model during the evaluation phase of FL. μ is computed as follows:

$$\mu_k^t = 1 - \frac{1}{1 + e^{-t(acc-50)}} \tag{6}$$

A more detailed discussion on parameter μ's rule of update is discussed in section 5.4.2.

As shown in Figure 2, Mixup is applied in two distinct stages of FL. Firstly, when transferring shared knowledge to local models, the local model L_k is mixed up with the current global model G according to the dynamic mixing factor μ, which determines the change ratio of λ_i (layer-wise Mixup degree in Eq. equation 5) across different layers. λ_i gradually is changed from $1 \rightarrow 0$ as we move from the head to the base layer. $\lambda_i = 1$ means sharing the 100% of the global model and $\lambda_i = 0$ means that the corresponding layer in local model is frozen and will not be mixed up with the global model G. Calculation of the Mixup degree of layer i λ_i at both broadcasting and aggregation stages is performed as follows:

$$\begin{aligned} \text{Broadcasting Stage:} \quad & \lambda_i = \begin{cases} 1 & \lambda_i > 1 \\ \mu^c * (n-i) & \lambda_i \leq 1 \end{cases} \\ \text{Aggregation Stage:} \quad & \lambda_i = \begin{cases} 0 & \lambda_i \leq 0 \\ 1 - (i * \mu)^c & \lambda_i > 0, \end{cases} \end{aligned} \tag{7}$$

where n is the number of local model layers and μ is the adaptive mix factor which will be updated in each communication round according to Eq. 6 for each local model individually.

3.2.1 BROADCASTING : GLOBAL TO LOCAL MODEL TRANSFER

This stage involves sharing global knowledge with local clients. In the existing PFL methods, the same weight allocation is typically applied to each heterogeneous local model. In our work, we personalize this process by allowing the local model to select the proportion of layers it requires. For instance, for a cold-start user, more information should be extracted from the shared knowledge model, implying that a few layers should be frozen for personalization. Additionally, we introduce a gradual update procedure where the value of λ gradually decreases from one (indicating fully shared layers) from the base layer to the end, based on the mixing factor μ. The mix layer is adaptively updated in each communication round for each client individually, according to personalization accuracy. With this adaptive and flexible approach, not only can upcoming streaming unseen data be managed, but also the participation gap, considering the addition of new cold start users:

$$
\begin{aligned}
L'^{(t)}_{k,i} &= (\lambda^{(t)}_{k,i}).G^{(t)}_i + (1-\lambda^{(t)}_{k,i}).L^{(t)}_{k,i} \\
L^{(t+1)}_k &= L'^{(t)}_k - \eta \nabla F_k(L'^{(t)}_k).
\end{aligned}
\tag{8}
$$

3.2.2 AGGREGATION :LOCAL TO GLOBAL MODEL TRANSFER

In the existing methods, two types of layers are considered: the personalized and the general layers. The general layers were merely expected to be shared with the global model. This approach poses several challenges such as catastrophic forgetting. The base layers of the global model are the backbone of the shared knowledge Raghu et al. (2019) as they carry the generalization. When we update the general layers of the global model by only averaging between a few local participants, valuable information from the previously shared knowledge will be lost and forgotten. The reason is that in every local update, the general layers are fully updated. This update leads to catastrophic forgetting and could slow down the overall cohort's convergence. To overcome this challenge, we propose a new strategy by applying a Mixup between previous global gradients and the other participants. $\sum \lambda_i = 1$ for each client i, λ_i will also gradually increase from 0 to 1 from the head to the base layer, according to the mix factor μ_i. Then, the base layer will be gradually updated according to the communication round and the generalization accuracy:

$$
\begin{aligned}
G'^{(t)}_{k,i} &= (\lambda^{(t)}_{i,k}).G^{(t)}_i + (1-\lambda^{(t)}_{k,i}).L^{(t+1)}_{k,i}, \\
G^{(t+1)} &= \sum_{k=1}^{K} \frac{|D_k|}{\sum_{k=1}^{K}|D_k|} G'^{(t+1)}_k.
\end{aligned}
\tag{9}
$$

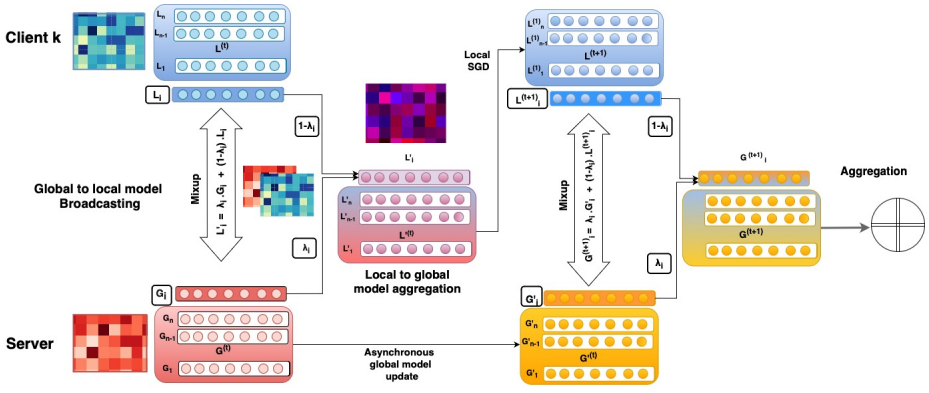

Figure 2: Workflow of pMixFed: Mixup is used in two stages. **1- Broadcasting:** when transferring knowledge to local models, the frozen personalized model $L^{(t)}$ is mixed up with global model $G^{(t)}$ according to the mix factor $\mu^{(t)}$ which determines λ_i, the Mixup degree for each layer. **2-Aggregation:** The updated global model $G^{(t+1)}$ results from a Mixup between the updated local model $L'^{(t)}$ and the current shared model $G'(t)$.

The high-level block-diagram visualization of the proposed method is shown in Figure 1. It is important to note that the sizes of local models $LM^{(0)}_i$ can differ from each other. Consequently, the size of $GM^{(0)}$ should be greater than the maximum size of local models. The parameter λ

in Equation 5 determines the Mixup degree between the shared model GM and the local models $LM_i^{(\cdot)}$, while μ governs the slope of the change in λ across different layers. The degree of Mixup gradually decreases according to the parameter μ from 0 to 1. In this scenario, $\lambda = 0$ for the first base layer, indicating total sharing, while $\lambda = 1$ applies to the final layer, which represents no sharing. The underlying concept is that the base layer contains more general information, whereas the final layers retain client-specific information. The use of the parameter μ, relative to the number of local layers, eliminates the need for a specified cut layer k and allows its application across different model sizes and layers. The parameter μ_i is adaptively updated based on the personalized and global model accuracy for each client. Algorithm 1 shows how Mixup is used as a shared aggregation technique between individual clients and the server. In each training round, only one client is *Mixed up* with the global model, and λ is adaptively learned based on the objective function using online learning. Algorithm 2 shows how Mixup is employed as a shared aggregation technique between the clients and the server. In each training round, only one client is "Mixed up" with the global model and the λ parameter is adaptively learned based on the objective function using online learning.

Algorithm 1 pMixFed: Broadcasting global to local model

Input: Initial states global model: $GM^{(0)}$, local models: $\{LM_i^{(0)}\}_{i=1,\dots,M}$, Number of communication rounds T, number of devices per round m, Number of layers in local models $\{L_i\}$
for $t = 0, 1, \dots, T-1$ **do**
  Server selects K devices $S(t) \subset \{1, \dots, N\}$
  Update μ for each LM_i, $i = \{1, \dots, K\}$
  Server broadcasts $GM^{(t)}$ to each device in $S(t)$
  for each device $m \in S(t)$ **in parallel do**
    $(LM_m^{(t+1)}, GM_m^{(t+1)}) =$ Mixup$[(LM_m'^{(t+1)}, GM_m'^{(t+1)}), \mu]$
    Device sends $GM_m^{(t+1)}$ back to server
    Update $\mu_{i=1,\dots,K}$
  end for
  Server updates $GM^{(t+1)} = \frac{1}{K}\sum_{i \in S(t)} GM_m^{(t+1)}$
end for

Algorithm 2 Proposed Mixup for Aggregation

Input: Initial states global model: $GM^{(0)}$, Number of communication rounds T, Number of local iterations Itr, number of devices per round m, Mixup degree λ
for $t = 0, 1, \dots, T-1$ **do**
  Server broadcasts $GM^{(t)}$ to each device in $S(t)$: $LM_i^{(t)}$
  Update $\lambda_{i=1,\dots,M}$ for each device
  for each device $i \in S(t)$ **do**
    for $epoch = 0, 1, \dots, Itr-1$ **do**
      Train local model:
        $LM_i^{(t+1)} = GM^{(t)}$
        $GM^{(t+1)} =$ Mixup$(LM_i^{(t+1)}, GM^{(t)})$
    end for
    Device sends $GM^{(t+1)}$ back to server
    Adaptively update $\lambda_{i=1,\dots,M}$
  end for
end for

4 Theoretical Analysis

In this section, we provide the convergence analysis of *pMixFed*. Due to space limitations, the complete proofs are included in Appendix B.2. Moreover, We compare the aggregation process and the server global model update of the *FedSGD* algorithm with our proposed mixed aggregation stage in *pMixFed*. In *FedSGD*, the gradients are aggregated and the server will be update the global model according to the aggregated gradients. the *FedSGD* is sometimes preferred over *FedAvg* due to its potentially faster convergence. However, it lacks robustness in heterogeneous environments. *pMixFed* leverages the faster convergence characteristics of *FedSGD* by incorporating early stopping mechanisms, facilitated by the use of mixup. As demonstrated in Section 5.3.1, the mixup factor λ functions analogously to an SGD update at the server, even though *pMixFed* aggregates model weights rather than gradients, similar to *FedAvg*. Section C.4 provides a more detailed explanation of this mechanism.
We begin by introducing the key notations and assumptions used throughout the convergence analysis. **Notations:**

- $t \in \{0, \dots, T-1\}$: communication round index.
- η_l: learning rate for local update, η_g: learning rate for global updates.
- $\lambda_{k,i}$: mixup coefficient for client k at layer i in round t, λ_k: mixup coefficient for client k in round t (assuming uniform across layers).
- $G^{(t)}$: global model parameters at round t, $L_k^{(t)}$: local model parameters of client k at round t.

- $|\mathcal{D}_k|$: size of the dataset at client k, $|\mathcal{D}|$: total size of datasets across all clients.
- $\nabla L_k^{(t+1)}$: gradient of the local model at client k in round t.

Assumption 1 (Smoothness of Local Objectives)**.** *The local objective functions $L_k(\cdot)$ are L-smooth, i.e.,*

$$\|\nabla L_k^{(t+1)} - \nabla L_k^{(t)}\| \leq L\|L_k^{(t+1)} - L_k^{(t)}\|. \tag{10}$$

Assumption 2 (Bounded Gradients)**.** *The gradients at each client are bounded, i.e.,*

$$\|\nabla L_k^{(t+1)}\| \leq G, \quad \forall k, t. \tag{11}$$

These assumptions are standard in convergence analysis and ensure that the optimization process is well-behaved. for more detailed proof refer to B.2.
We hypothesize that the mixup coefficient λ acts similarly to the learning rate η, suggesting that λ plays a role analogous to η in *FedSGD*.
FedSGD Update Rule : In *FedSGD*, the global model $G^{(t)}$ is updated by aggregating the gradients from all clients:

$$G^{(t+1)} = G^{(t)} - \eta_g \sum_{k=1}^{K} \Omega_k \nabla L_k^{(t+1)}, \tag{12}$$

where $\Omega_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$ is the weight associated with client k. Since $L_k^{(t+1)} = L_k^{(t)} - \eta_l \nabla L_k^{(t+1)}$, we can rewrite the update as:

$$G^{(t+1)} = G^{(t)} - \eta_g \sum_{k=1}^{K} \Omega_k \left(\frac{L_k^{(t)} - L_k^{(t+1)}}{\eta_l} \right). \tag{13}$$

pMixFed Update Rule: In *pMixFed*, the global model update incorporates mixup coefficients:

$$G^{(t+1)} = \sum_{k=1}^{K} \Omega_k \left[(1 - \lambda_k) L_k^{(t+1)} + \lambda_k G^{(t)} \right]. \tag{14}$$

Assuming $\lambda_{k,i} = \lambda_k$ for all layers i, and considering that $G^{(t)}$ is sent to all clients at round t, we simplify the update to:

$$G^{(t+1)} = (1 - \lambda_k) \sum_{k=1}^{K} \Omega_k L_k^{(t+1)} + \lambda_k G^{(t)}. \tag{15}$$

4.0.1 Analytical analysis of the Effect of learning rate and mixup degree

To establish the relationship between λ and η_g, we align the *FedSGD* and *pMixFed* update equations. From Equation equation 13, rearranged:

$$G^{(t+1)} = G^{(t)} - \frac{\eta_g}{\eta_l} \sum_{k=1}^{K} \Omega_k \left(L_k^{(t)} - L_k^{(t+1)} \right). \tag{16}$$

Assuming $L_k^{(t)}$ is replaced with $G^{(t)}$ at round t for *FedSGD*, we have $L_k^{(t)} = G^{(t)}$. Substituting this into Equation equation 16:

$$G^{(t+1)} = G^{(t)} - \frac{\eta_g}{\eta_l} \sum_{k=1}^{K} \Omega_k \left(G^{(t)} - L_k^{(t+1)} \right). \tag{17}$$

Simplifying and Comparing with Equation equation 15, we see that if: $\lambda_k = \frac{\eta_g}{\eta_l}$. then the updates are analogous.

$$G^{(t+1)} = G^{(t)} \left(1 - \frac{\eta_g}{\eta_l} \sum_{k=1}^{K} \Omega_k \right) + \frac{\eta_g}{\eta_l} \sum_{k=1}^{K} \Omega_k L_k^{(t+1)}. \tag{18}$$

Theorem 1. *Under Assumptions 1 and 2, the mixup coefficient λ_k in pMixFed acts similarly to the learning rate ratio $\frac{\eta_g}{\eta_l}$ in FedSGD, such that:$\lambda_k = \frac{\eta_g}{\eta_l}$. This implies that the mixup mechanism in pMixFed can be interpreted as a form of learning rate control analogous to FedSGD.For complete proof please refer to D.2.*

Our theoretical analysis indicates that the mixup coefficient λ_k in *pMixFed* plays a role analogous to the learning rate in *FedSGD*. This equivalence provides a deeper understanding of how *pMixFed* leverages the strengths of *FedSGD* while mitigating its weaknesses in heterogeneous settings. By appropriately choosing λ_k, *pMixFed* can achieve faster convergence and improved robustness. Additionally, we conducted experiments to further investigate the impact of the learning rate and mix factor on model performance in 5.3.1.

5 EXPERIMENTAL RESULTS

We conduct comprehensive experiments to demonstrate the superiority of our proposed method. We evaluated our method on three benchmarks with non-IID data distributions.

5.1 EXPERIMENTAL SETUP

Dataset and Data partition: For simulating heterogeneous non-IID distributions, we follow the literature Pillutla et al. (2022) and use a Dirichlet distribution (parameterized by γ) to model the heterogeneous distribution of client data. The objective is to partition the dataset across multiple clients such that each client holds a subset of the classes with varying proportions of data. A smaller γ leads to highly skewed distributions, where clients predominantly receive samples from a limited number of classes, while a larger γ results in a more balanced distribution of classes across clients. In our experiments, γ was set to 0.1. To simulate higher heterogeneity and evaluate generalization, we used different class distributions for the training D^{ts} and test sets D^{tr} compared to the setting in Lin et al. (2020) where the the train-test split has been done after generating the the class distribution for each client. In our experiments, *the training and test data for the same client did not contain the same set of classes and contains different sample sizes.*

Baseline and backbone: We compare two version of our method **(i) pMixFed**: an adaptive and dynamic mixup-based PFL approach, and **(ii) pMixFed-Dynamic**: a dynamic-only mixup variant where the parameter μ is fixed across communication rounds, against several baselines. These baselines include: **FedAvg** McMahan et al. (2017), **FedAlt** Pillutla et al. (2022), **FedSim** Pillutla et al. (2022), **FedBABU** Oh et al. (2021). Additionally, we compare against full model personalization methods, **pFedHN** Shamsian et al. (2021), **Per-FedAvg** Fallah et al. (2020) , and **LG-FedAvg** Liang et al. (2020). For all experiments, the number of local training epochs was set to $r = 2$, and the batch size was fixed at 32. The learning rate, for both global and local updates, was fixed at $lr = 0.001$ across all communication rounds. The client participation rate C varied with the number of clients N, using the configurations: $\{C = 100\%, N = 10\}$, and $\{C = 10\%, N = 100\}$. The number of communication rounds was set to 100 for all experiments. Figure 3 presents the training accuracy versus communication rounds for CIFAR10 and CIFAR100 datasets.

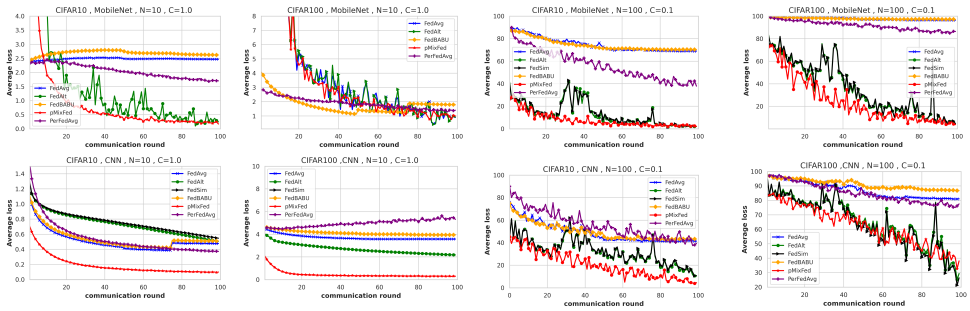

Figure 3: Average loss versus communication rounds for *pMixFed* and PFL baselines for different participation ratios experimented on CIFAR10 and CIFAR100.

Model Architectures: Following the FL literature, we utilized several model architectures. For MNIST, we used a simple CNN consisting of 2 convolutional layers (each with 1 block) and 2 fully connected layers. For CIFAR-10 and CIFAR-100, we employed a CNN with 4 convolutional layers (1 block each) and 1 fully connected layer. Additionally, we used MobileNet, which comprises 14 convolutional layers (2 blocks each) and 1 fully connected layer, for CIFAR-10 and CIFAR-100, for all datasets. For partial model approaches such as **FedAlt** and **FedSim**, the split layer is fixed in the middle of the network: for CNNs, layers 1–2 are shared, while for MobileNet, layers 1–7 are shared. Details about the model architectures can be found in Appendix **??**. The Adam optimizer was used for all experiments, the momentum is set to 0.0.

5.2 COMPARATIVE RESULTS

We have reported the participant-gap Yuan et al. (2021) before and after fine tunning for different approaches in Tables 1 and 2. In the evaluation, the average test accuracy[2] is measured on cold-start clients $|D^{ts}_{k \cap unseen}|, k \subset \{1, .., M\}$, where $D^{ts} \neq D^{tr}$. These clients have not participated in the federation at all. For evaluation, the final global model at the last communication round is saved and used as the initial model for cold-start clients. The global model is then personalized according to each baseline's personalization or fine-tuning algorithm for $r = 2$ local epochs. For **FedAlt**, the local model is reconstructed from the global model and fine-tuned on the test data. For **FedSim**, both the global and local models are fine-tuned partially but simultaneously. In the case of **FedBABU**, the head (fully connected layers) remains frozen during local training, while the body is updated. Since we could not directly apply **pFedHN** in our platform setting, we adapted their method using

[2]Classification accuracy using softmax

the same hyperparameters discussed above and employed hidden layers with 100 units for the hypernetwork and 16 kernels. The local update process for **LG-FedAvg**, **FedAvg**, and **Per-FedAvg** simply involves updating all layers jointly during the fine-tuning process. We observe that pMixFed outperforms the baselines in most cases. However, we can see in 3 that partial models with a hard split are very sensitive to the choice of hyper-parameters and they may fail to train properly. It seems that this problem has been addressed in *pFedMix* more discussed in 5.3.2.

N	C	Methods	CIFAR-10		CIFAR-100		MNIST	
			participant-gap	fine-tune	participant-gap	fine-tune	participant-gap	fine-tune
10	1.0	**pMixFed**	**41.40**	62.34	**34.83**	**48.88**	**79.42**	**99.62**
		pMixFed-Dynamic	**45.67**	**68.07**	**32.30**	**40.51**	78.00	**99.37**
		FedAvg	27.71	59.04	10.56	18.97	76.17	99.13
		FedAlt	39.38	65.08	13.54	20.66	77.93	98.88
		FedSim	39.10	65.03	11.15	18.94	78.49	98.67
		FedBaBU	22.11	58.35	-	14.48	-	99.12
		per-FedAvg	-	59.97	-	19.40	-	99.33
		LG-FedAvg	-	67.45	18.97	28.63	-	99.14
		pFedHN	-	53.12	-	37.67	-	-
100	0.1	**pMixFed**	37.73	**60.27**	**16.25**	**22.71**	**78.42**	95.30
		pMixFed-Dynamic	28.78	**56.89**	**16.14**	**22.42**	74.39	97.53
		FedAvg	**41.97**	58.57	7.90	18.87	77.86	98.05
		FedAlt	28.98	47.93	5.38	10.65	74.80	94.52
		FedSim	26.40	47.14	6.28	11.78	70.57	87.37
		FedBaBU	14.78	56.71	2.25	13.21	69.49	98.02
		per-FedAvg	-	60.58	-	22.42	-	**99.24**
		LG-FedAvg	32.17	53.13	9.02	21.20		98.10

Table 1: Test accuracy of unseen clients trained on the CNN model across three datasets: CIFAR-10, CIFAR-100, and MNIST. Results are reported for two participation rates (C = 0.1 , 1.0) and varying numbers of clients (N = 10,100), both before and after personalization.

N	C	Methods	CIFAR-10		CIFAR-100	
			participant-gap	fine-tune	participant-gap	fine-tune
100	0.1	**pMixFed**	29.22	**73.22**	**15.43**	**59.54**
		pMixFed-Dynamic	29.09	**72.71**	**15.10**	**57.77**
		FedAvg	16.76	31.01	1.03	3.57
		FedAlt	19.34	72.45	4.07	57.03
		FedSim	19.37	72.11	4.08	56.57
		FedBaBU	17.02	29.56	4.89	10.53
		Per-FedAvg	-	57.59	-	13.60
		Lg-FedAvg	**36.31**	47.88	5.01	**36.22**
10	1.0	**pMixFed**	**47.90**	**82.17**	**13.73**	**69.90**
		pMixFed-Dynamic	**48.20**	**81.69**	**12.86**	**70.64**
		FedAvg	30.79	47.08	9.91	37.62
		FedAlt	23.49	72.25	12.43	63.73
		FedSim	24.35	70.19	11.63	64.00
		FedBaBU	11.67	29.15	7.41	35.02
		Per-FedAvg	-	61.24	-	9.40
		Lg-FedAvg	47.16	71.22	28.05	36.22

Table 2: Test accuracy of unseen clients trained on the MobileNet model across two datasets: CIFAR-10, CIFAR-100. Results are reported for two participation rates (C = 0.1 and 10) and varying numbers of clients (N = 10,100), both before and after personalization

5.3 Analytic Experiments

We provide a set of experiments to study and gain more insights about the proposed method.

5.3.1 Effect of mixup degree as learning rate

As discussed in Section D, the effect of the mixup coefficient is similar to that of the learning rate or learning rate decay. To demonstrate this similarity empirically, we designed an experiment where we compared a scenario in which the learning rate follows a scheduler similar to the mixup coefficient μ. The results of this comparison can be seen in Figure **??**. We also conducted a layer-wise experiment, where the learning rate for the shared layers (the first half of the network) was set to a relatively large value, $lr = 0.1$, while the learning rate for the personalized layers (the second half of the network) was set to a much smaller value, $lr = 1 \times 10^{-5}$. This experiment demonstrates a similar effect to freezing the personalized layers, as when $\lambda_i = 0$ for $i \in L_i$ (personalized layers), these layers do not update or participate in the aggregation process. Our findings show that the mixup coefficient μ can achieve the same effect of scheduling the learning rate. Without modifying the learning rate, partitioning the model, or hindering the learning process, we were able to achieve stronger improvements in performance.

5.3.2 Adaptive Robustness to Performance Degradation

During the experiments, we observed the algorithm's ability to adapt and recover from performance degradation, particularly in challenging scenarios such as zero gradients or incoming data variability. For instance, in complex settings, such as with larger models like *MobileNet* or on the CIFAR-100 dataset, partial model approaches encounter sudden accuracy drops due to zero gradients or the addition of new participants. This is mitigated by the adaptive nature of the mixup coefficient, which updates and tunes itself according to the model's performance. Specifically, if the global model $G^{(t)}$ is not powerful enough, the mixup coefficient $\mu^{(t)}$ is decreased, reducing the influence of the history $H\big|_1^T G^t$, and giving more weight to the current local updates $L_k^t, k \in S^t$ during the aggregation process. This adaptive adjustment allows handling performance drops and ensures robust training, even in scenarios with dynamic participants and data shifts. Figure 4 shows this experiment

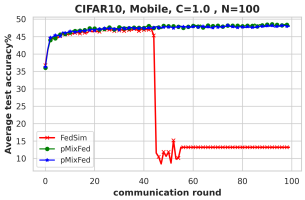

Figure 4: Accuracy drop in FedSim arises due to gradient vanishing in round 42. Applying adaptive mixup to the shared layers of the same model solves the problem.

5.4 Effects of different mix factor

5.4.1 Random vs gradual mix factor from β distribution

The value of λ in Eq. 5, naturally sampled from a $\beta \neq (\alpha, \alpha)$ distribution Zhang et al. (2017) which is on the interval [0,1]. We have also experimented the random λ_i using β distribution with different α. If $\alpha = 1$, the β distribution is uniform meaning that the λ would be sampled uniformly from [0,1]. Moreover $\alpha > 1$, The λ would be more in between, creating a more mixed output between L_k and G. On the contrary, if $\alpha < 1$ the mixed model tend to choose just one of the global and local models where $\lambda = 1 or \lambda = 0$. The effects of different α on mixup degree λ could be seen in figure 7. for more discussion on this, refer to F.1.

5.4.2 Mix Factor(μ): sigmoid vs Simple mode

In this study, we have exploited two different functions to update adaptive mixup factor(μ) in each communication round. This idea is based on the performance of the model which we want to update. In 1st scenario σ function has been adapted as shown in equation 6. Acc refers to the test accuracy of the local model: $\big(L_k^t(x_k, \theta_k^t), y_i\big)$ in the *broadcasting* phase and average test accuracy of the previous global model $\big(G^t(x, \theta^t), y\big)$ on all local test sets $x = \{x_1, x_2, \ldots, x_K\}$. It should be noted that parameter μ is constant in the aggregation stage for all clients as it's dependent to the average performance of the previous global model. [3] the figure 8 illustrates how the Mu changes with different stoop size t. t updates in each communication round ($t = \frac{epoch_{current}}{epochs}$). The reason behind this is that a cold start user usually needs to acquire knowledge more than an experienced user. Hence, the communication round should also impact the mix factor. On the other hand, the 2nd scenario merely uses a simple linear function for updating Mu according to the test accuracy ($\mu_k^t = 1 - \frac{acc}{100}$) The comparison of these two scenarios as well as the effect of different t values on the test accuracy, is depicted in Figure 5.

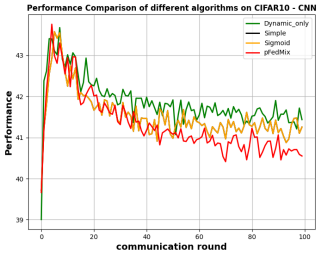

Figure 5: The comparison between test accuracy of different scenarios for updating *Mix factor* in each communication round. **(Dynamic-only)**. In this scenario we used a fixed Mu for all communication rounds. **(Sigmoid)**. an updating strategy based on a sigmoid function 6 has been used with different b values. **(Simple)**. A simple linear function has been adapted for updating Mu. **(pFedMix)** No dynamic or adaptive update of Mu has been used here. Mu has been set to 0.5 for all communication rounds.

5.4.3 Different model sizes

pMixFed is capable of handling variable model sizes across different clients. The global model, M_g, retains the maximum number of layers from all clients, i.e., $M_G = \max(M_1, M_2, \ldots, M_N)$. During the matching process between the global model G_i and the local model L_i, if a layer block from the local model does not match a corresponding global layer, we set $\lambda_i = 0$, meaning that the layer block will neither participate in the broadcasting nor aggregation processes. A more detailed explanation is in section E.1.

6 Conclusions

We introduced pMixFed, a dynamic, layer-wise personalized federated learning approach that uses mixup to integrate the shared global and personalized local models. Our approach features adaptive partitioning between shared and personalized layers, along with a gradual transition for personalization, enabling seamless adaptation for local clients, improved generalization across clients, and reduced risk of catastrophic forgetting. We provided a theoretical analysis of pMixFed to study the properties of its convergence. Our experiments on three datasets demonstrated its superior performance over existing PFL methods. Empirically, pMixFed exhibited faster training times, increased robustness, and better handling of data heterogeneity compared to state-of-the-art PFL models. Future research directions include exploring multi-Modal personalization and adapting pMixFed for working on resource-constrained devices

[3] In the aggregation stage, the focus is on keeping the history of the generalized information to mitigate the catastrophic forgetting problem.

REPRODUCIBILITY STATEMENT

For reproducibility, we have included our source codes and environment setup instructions as a supplementary material.

REFERENCES

Sawsan Abdulrahman, Hanine Tout, Azzam Mourad, and Chamseddine Talhi. Fedmccs: Multicriteria client selection model for optimal iot federated learning. *IEEE Internet of Things Journal*, 8(6):4723–4735, 2021. doi: 10.1109/JIOT.2020.3028742.

Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized federated learning with gaussian processes. *Advances in Neural Information Processing Systems*, 34:8392–8406, 2021.

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

Christopher Beckham, Sina Honari, Vikas Verma, Alex M Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Chris Pal. On adversarial mixup resynthesis. *Advances in neural information processing systems*, 32, 2019.

Duc Bui, Kshitiz Malik, Jack Goetz, Honglei Liu, Seungwhan Moon, Anuj Kumar, and Kang G Shin. Federated user representation learning. *arXiv preprint arXiv:1909.12535*, 2019.

Hsin-Ping Chou, Shih-Chieh Chang, Jia-Yu Pan, Wei Wei, and Da-Cheng Juan. Remix: rebalanced mixup. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pp. 95–110. Springer, 2020.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Fedavg with fine tuning: Local updates lead to representation learning. *Advances in Neural Information Processing Systems*, 35:10572–10586, 2022a.

Liam Collins, Aryan Mokhtari, Sewoong Oh, and Sanjay Shakkottai. Maml and anil provably learn representations. In *International Conference on Machine Learning*, pp. 4238–4310. PMLR, 2022b.

Moming Duan, Duo Liu, Xianzhang Chen, Yujuan Tan, Jinting Ren, Lei Qiao, and Liang Liang. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. In *2019 IEEE 37th international conference on computer design (ICCD)*, pp. 246–254. IEEE, 2019.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3714–3722, 2019.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33:14068–14080, 2020.

Wenke Huang, Mang Ye, and Bo Du. Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10143–10153, 2022.

Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 7865–7873, 2021.

Ahmed Imteaj and M Hadi Amini. Fedparl: Client activity and resource-oriented lightweight federated learning model for resource-constrained heterogeneous iot environment. *Frontiers in Communications and Networks*, 2:657653, 2021.

Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24, 2021.

Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

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

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pp. 5132–5143. PMLR, 2020.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Sunwoo Lee, Anit Kumar Sahu, Chaoyang He, and Salman Avestimehr. Partial model averaging in federated learning: Performance guarantees and benefits. *Neurocomputing*, 556:126647, 2023.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10713–10722, 2021a.

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, 2021b.

Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems*, 33:2351–2363, 2020.

Kangyang Luo, Xiang Li, Yunshi Lan, and Ming Gao. Gradma: A gradient-memory-based accelerated federated learning with alleviated catastrophic forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3708–3717, 2023.

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.

Jaehoon Oh, Sangmook Kim, and Se-Young Yun. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042*, 2021.

Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pp. 17716–17758. PMLR, 2022.

Sone Kyaw Pye and Han Yu. Personalised federated learning: A combinational approach. *arXiv preprint arXiv:2108.09618*, 2021.

Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10061–10071, 2022.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, pp. 9489–9502. PMLR, 2021.

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.

Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

Karan Singhal, Hakim Sidahmed, Zachary Garrett, Shanshan Wu, John Rush, and Sushant Prakash. Federated reconstruction: Partially local federated learning. *Advances in Neural Information Processing Systems*, 34: 11220–11232, 2021.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

Guangyu Sun, Matias Mendieta, Jun Luo, Shandong Wu, and Chen Chen. Fedperfix: Towards partial model personalization of vision transformers in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4988–4998, 2023.

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

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Shashanka Venkataramanan, Ewa Kijak, Laurent Amsaleg, and Yannis Avrithis. Alignmixup: Improving representations by interpolating aligned features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19174–19183, 2022.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pp. 6438–6447. PMLR, 2019.

Han Wang, Luis Muñoz-González, David Eklund, and Shahid Raza. Non-iid data re-balancing at iot edge with peer-to-peer federated learning for anomaly detection. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 153–163, 2021.

Jeffry Wicaksana, Zengqiang Yan, Dong Zhang, Xijie Huang, Huimin Wu, Xin Yang, and Kwang-Ting Cheng. Fedmix: Mixed supervised federated learning for medical image segmentation. *IEEE Transactions on Medical Imaging*, 42(7):1955–1968, 2022.

Qiong Wu, Xu Chen, Zhi Zhou, and Junshan Zhang. Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring. *IEEE Transactions on Mobile Computing*, 21(8):2818–2832, 2020.

Peng Xiao, Samuel Cheng, Vladimir Stankovic, and Dejan Vukobratovic. Averaging is probably not the optimum way of aggregating parameters in federated learning. *Entropy*, 22(3):314, 2020.

Chencheng Xu, Zhiwei Hong, Minlie Huang, and Tao Jiang. Acceleration of federated learning with alleviated forgetting in local training. *arXiv preprint arXiv:2203.02645*, 2022.

Xiyuan Yang, Wenke Huang, and Mang Ye. Fedas: Bridging inconsistency in personalized federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11986–11995, 2024.

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

Honglin Yuan, Warren Morningstar, Lin Ning, and Karan Singhal. What do we mean by generalization in federated learning? *arXiv preprint arXiv:2110.14216*, 2021.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*, 2020.

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

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, pp. 12878–12889. PMLR, 2021.

Difan Zou, Yuan Cao, Yuanzhi Li, and Quanquan Gu. The benefits of mixup for feature learning. In *International Conference on Machine Learning*, pp. 43423–43479. PMLR, 2023.

A RELATED WORK

PFL aims to adapt local models to the individual needs, preferences, and contexts of each participant. Since the inception of FL, various PFL approaches have been explored to address the challenges stemming from heterogeneity in three distinct categories: 1) data-centric strategies Tan et al. (2022) (e.g., Astraea Duan et al. (2019), P2P k-SMOTE Wang et al. (2021), FedMCCS Abdulrahman et al. (2021), and FedHome Wu et al. (2020)), 2) adaptation techniques, and 3) local model personalization.

Data-centric PFL: Data normalization, feature engineering, data augmentation, employing synthetic data, and client selection techniques are the examples of data-centric approaches which tend to address data heterogeneity and class imbalances by manipulating statistical data properties e.g., data size, distribution, and local data selection Tan et al. (2022). Although data-driven methods such as Astraea Duan et al. (2019), P2P k-SMOTE Wang et al. (2021), FedMCCS Abdulrahman et al. (2021), FedAug Jeong et al. (2018) ,Zhao et al. (2018), and FedHome Wu et al. (2020) are reported successful results, these methods all change the natural distribution and statistical properties of federated data as well as injecting biased information and eliminating valuable user information. Moreover, most of the mentioned strategies require proxy data which increases the risk of information leakage and might not comply with privacy regulations of some entities. Also, data-centric methods could be computationally expensive and could not be employed for resource-constrained devices.

Adaptation techniques: In these type of approaches, a single global model resides on the server which can be swiftly adapted to local models in the subsequent phase. These approaches aim for two primary objectives: 1) learning a robust generalized model (representation learning), and 2) achieving fast and efficient local model adaptation (regularization). Some of these techniques employ regularization terms to address the client drift problem and prevent local updates from diverging. FedProx Li et al. (2020), SCAFOLD Karimireddy et al. (2020), FL-MOON Li et al. (2021a), PerFedAVGFallah et al. (2020), pFedMe T Dinh et al. (2020), and FedMD Li & Wang (2019) are examples of the adaptation PFL techniques.These approaches mainly benefit from meta-learning, regularization, and transfer learning, but they also come with several challenges. Meta-learning methods can be computationally expensive, while regularization techniques can slow down the convergence of the training process by adding a regularization term to the objective function. Similarly, using transfer learning algorithms are inefficient in terms of communication overhead, and they may require a public dataset to reinforce the global model on the server. One common and significant complication in adaptation techniques is the requirement for the same model architecture for all local models. This implies that both resource-constrained and computationally powerful devices must use the same model sizes.

Local Model Personalization: Lack of accurate personalization solutions has motivated the development of local model personalization methods that train customized local models. Some of these approaches use multi-task learning (MTL), which is a collaborative technique facilitating information flow between different tasks. MOCHA Smith et al. (2017), FedAMP Huang et al. (2021), FedCurv Shoham et al. (2019), Ditto Li et al. (2021b), pFedHN Shamsian et al. (2021) and pFedGP Achituve et al. (2021) are some of the personalization methods that mainly incorporates various MTL approaches. The disadvantage of using MTL for PFL is that it can be computationally expensive, making it infeasible to deploy this method for large-scale FL across different devices. other type of these approaches are using knowledge distillation(KD) which is usuful when the local training objectives differs . Some examples are FedGen Zhu et al. (2021) and FedGKT He et al. (2020).

Partial PFL: Partial PFL methods which is the main foucus in this paper, have been effective in mitigating catastrophic forgetting, owing to the presence of a shared component while keeping other layers freezed for personalization. FedPer Arivazhagan et al. (2019) initially introduced the concept of employing partial models in FL, where only the first layers containing generalized information are shared, and the final layers are reserved for personalization. Another framework, FURL Bui et al. (2019), utilizes partial PFL for document classification, keeping some feature embeddings private and unshared. LG-FedAVG Liang et al. (2020) adopts compact local representation learning for high-level features, while the base layers are shared to create a global model via adversarial competitive learning. two pionear works that have been used as baseline in this paper, are FedAlt and Fedsim which have been successfully employed partial PFL.FedAlt Singhal et al. (2021), or FedRecon, introduces a stateless FL paradigm where clients don't need to retain previous parameters in memory. Instead, local models are reconstructed from the global model, resembling meta-learning. Accordingly, the new participants could easily shape their local models by reconstructing local parameters using their own local data. In contrast, FedSim Pillutla et al. (2022) updates shared and local models simultaneously during each local iteration. While both algorithms are task-specific, FedAlt outperforms FedSim when local heterogeneity surpasses global heterogeneity. In These algorithms, shared global layers can undergo significant changes after just a few rounds of local training leading to client drift. An experienced user with high personalization accuracy should not freeze the same number of layers as a cold-start user. Initially, a cold-start user needs to learn from the global shared model before gradually increasing personalization by freezing more final layers, As clients' dataset is constantly updated, resulting in a performance drop known as the out-of-sample gap and participation gap Yuan et al. (2021).

Mixup: There are various mixup variations, each designed to address specific challenges or enhance aspects of the original technique. For example, AlignMixup Venkataramanan et al. (2022) improves local spatial alignment, while Manifold Mixup Verma et al. (2019) acts as a regularization technique by training DNNs on linear combinations of hidden layer representations. CutMix Yun et al. (2019) exchanges patches between images,

and Remix Chou et al. (2020) addresses class imbalance by adding extra weight to minority classes. Lastly, AdaMix Guo et al. (2019) optimizes mixing distributions and minimizes overlaps.

B APPENDIX

B.1 HETEROGENITY

One of the primary sources of complications in FL is heterogeneity, which manifests in three domains: model, device, and data Kairouz et al. (2021); Huang et al. (2022); McMahan et al. (2017). Device heterogeneity occurs due to differences in computational capabilities, storage capacity, network bandwidth, and hardware equipment. These differences cause scheduling problems, limitations in model design, high computational and communication time, unreliability, and overall model performance degradation Kairouz et al. (2021). Model heterogeneity, On the other hand, refers to differences between local ML model structures, optimization algorithms, and model size. This type of heterogeneity complicates the aggregation process, generates inconsistency in model Performance, drives scalability issues, and reduces generalization Huang et al. (2022). Data heterogeneity arises from variations in clients' data distribution. These differences could be evident in features, labels, number of samples, class imbalance, and different local data distributions that exist in non-IID federated data McMahan et al. (2017); Kairouz et al. (2021).Data heterogeneity is a major factor contributing to various challenges in the FL environment, such as catastrophic forgetting and client drift problem Huang et al. (2022); Singhal et al. (2021); Luo et al. (2023); Qu et al. (2022). In catastrophic forgetting, a client's model becomes too tailored to its data after just a few rounds of local training, easily forgets the acquired shared knowledge and diminishes generalization ability. This issue arises from the fundamental discrepancies between local data distributions Luo et al. (2023); Qu et al. (2022). On the other hand, the client drift problem, is a common occurrence in FL environments with high data heterogeneity, where the global and local optima gradually diverge with each communication round. This divergence eventually causes the global model to inadequately represent the local models, leading to a lack of sufficient generalization. PFL shares numerous objectives with meta-learning and multi-task learning Li et al. (2021b). According to the previous pioneer works Collins et al. (2022a) where the similarities between meta-learning and FL have been discussed, "FedAvg" algorithm with more than two local updates behaves similar to MAML [4] Finn et al. (2017), Collins et al. (2022b)

B.2 CONVERGENCE PROOF

C CONVERGENCE ANALYSIS OF *pMixFed*

We make the following assumptions to establish the convergence properties of *pMixFed*:

Assumption 3. ***Lipschitz Continuity.*** *The gradient of the local loss function for client k is Lipschitz continuous with constant L_1, i.e.,*

$$\|\nabla \mathcal{L}_k^{(t_1)} - \nabla \mathcal{L}_k^{(t_2)}\| \leq L_1 \|L_k^{(t_1)} - L_k^{(t_2)}\|. \tag{19}$$

Assumption 4. ***Unbiased Gradient Estimation with Bounded Variance.*** *The gradient estimate for local model updates is unbiased and has a bounded variance, i.e.,*

$$\mathbb{E}[\nabla \mathcal{L}_k^{(t)}] = \nabla \mathcal{L}_k^{(t)}, \quad \textit{Var}(\nabla \mathcal{L}_k^{(t)}) \leq \sigma^2. \tag{20}$$

C.1 LOCAL TRAINING CONVERGENCE

We first analyze the local training updates that occur between communication rounds.

Lemma 1. ***Local Model Training Progress.*** *Under Assumptions 3 and 4, after r local updates, the expected loss for client k satisfies:*

$$\mathbb{E}[\mathcal{L}_k^{(t+r)}] \leq \mathcal{L}_k^{(t)} + \left(\frac{L_1 \eta_l^2}{2} - \eta_l\right) \sum_{j=0}^{r-1} \|\nabla \mathcal{L}_k^{(t+j)}\|^2 + \frac{L_1 r \eta_l^2 \sigma^2}{2}. \tag{21}$$

This lemma provides a bound on how the local training process improves the loss function. The bound depends on the learning rate η_l, the smoothness constant L_1, and the variance of the gradient estimates σ^2.

C.2 GLOBAL MODEL AGGREGATION

Next, we consider the effect of aggregating local models at the server after each communication round.

[4] Model-Agnostic Meta-Learning

Lemma 2. ***Global Model Aggregation Dynamics.*** *After aggregating the local models, the global model loss changes as follows:*

$$\mathbb{E}[\mathcal{L}^{(t+1)}] \leq \mathbb{E}[\mathcal{L}^{(t)}] + \eta_g \delta^2, \tag{22}$$

where δ^2 bounds the difference between the global and local model parameters before and after aggregation.

This lemma shows that, during aggregation, the global loss increases by a term proportional to the global learning rate η_g and the parameter variation δ^2.

C.3 CONVERGENCE OF *pMixFed*

With the above lemmas, we can now derive the convergence properties of *pMixFed*.

Theorem 2. ***Convergence Rate of pMixFed.*** *Under the previously stated assumptions, the expected global loss after a complete round of communication and local training satisfies:*

$$\mathbb{E}[\mathcal{L}^{(t+1)}] \leq \mathcal{L}^{(t)} + \left(\frac{L_1 \eta_l^2}{2} - \eta_l\right) \sum_{k=1}^{K} \sum_{j=0}^{r-1} \|\nabla \mathcal{L}_k^{(t+j)}\|^2 + \frac{L_1 r \eta_l^2 \sigma^2}{2} + \eta_g \delta^2. \tag{23}$$

This theorem demonstrates that the global loss decreases with each communication round, with the convergence depending on the local and global learning rates, the gradient norms, and the parameter variation.

Theorem 3. ***Non-Convex Convergence Rate.*** *For non-convex loss functions, pMixFed achieves convergence at the following rate:*

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{k=1}^{K} \|\nabla \mathcal{L}_k^{(t)}\|^2 \leq \mathcal{O}(1/T), \tag{24}$$

where T is the total number of communication rounds.

This final theorem indicates that *pMixFed* achieves non-convex convergence with a rate of $\mathcal{O}(1/T)$, demonstrating that the algorithm improves over time as the number of communication rounds increases.

C.4 THE EFFECT OF AGGREGATING MODEL PARAMETERS AND GRADIENTS ON CATASTROPHIC FORGETTING

the *pMixFed* algorithm combines the advantages of FedSGD and FedAvg by aggregating model weights rather than gradients, while still ensuring convergence even in heterogeneous data settings. The incorporation of the mixup mechanism enhances stability, providing faster convergence rates compared to FedSGD, particularly in non-convex settings.

Considering local SGD steps r and η_l as the local learning rate we can further expand the above formulation:

$$\begin{aligned} G^{t+1} &= G^t - \eta_g \sum_{i=1}^{k} \Omega_k \left[L_k^{(t+r)} - L_k^{(t)}\right] \\ G^{t+1} &= G^t - \eta_g . \eta_l \sum_{i=1}^{k} \Omega_k \sum_{j=0}^{r-1} \nabla L_k^{(t+j)} \end{aligned} \tag{25}$$

Since in FedSGD, the global model $G^{(t)}$ is fully shared with each local model $L_k^{(t)}$ in each communication round *t*, and $\sum_{i=1}^{k} \Omega_k = 1$, we can update the problem as a constrained optimization problem.

$$\begin{aligned} G^{t+1} &= G^t - \eta_g . \eta_l \sum_{j=1}^{r-1} \nabla G^{(t+j)} \\ \sum_{j=2}^{r-1} \nabla G^{(t+j)} &= (1 - \eta_g . \eta_l) G^t - (1 + \eta_g . \eta_l) G^{t+1} \end{aligned}$$

The right term of the equation is the history of global model gradients which could be written as $H\big|_2^{r-1} \nabla G^t = a.G^t - b.G^{t+1}$. The problem of the FedSGD algorithm is that the gradients are too small to keep the state of the previous global model, leading to catastrophic forgetting in partial models that use gradients in their aggregation stage.

D THEORETICAL ANALYSIS

D.1 NOTATIONS AND ASSUMPTIONS

We begin by introducing the key notations and assumptions used throughout the convergence analysis.

Notations:

- $t \in \{0, \ldots, T-1\}$: communication round index.
- η_l: learning rate for local updates.
- η_g: learning rate for global updates.
- $\lambda_{k,i}$: mixup coefficient for client k at layer i in round t.
- λ_k: mixup coefficient for client k in round t (assuming uniform across layers).
- $G^{(t)}$: global model parameters at round t.
- $L_k^{(t)}$: local model parameters of client k at round t.
- $|\mathcal{D}_k|$: size of the dataset at client k.
- $|\mathcal{D}|$: total size of datasets across all clients.
- $\nabla L_k^{(t+1)}$: gradient of the local model at client k in round t.

D.2 PROOF OF THEOROTICAL ANALYSIS

Theorem 4. *Under Assumptions 1 and 2, the mixup coefficient λ_k in pMixFed acts similarly to the learning rate ratio $\frac{\eta_g}{\eta_l}$ in FedSGD, such that:*

$$\lambda_k = \frac{\eta_g}{\eta_l}. \tag{26}$$

This implies that the mixup mechanism in pMixFed can be interpreted as a form of learning rate control analogous to FedSGD.

Proof. Starting from the *pMixFed* update in Equation equation 15:

$$G^{(t+1)} - \lambda_k G^{(t)} = (1 - \lambda_k) \sum_{k=1}^{K} \Omega_k L_k^{(t+1)}. \tag{27}$$

From the rearranged *FedSGD* update in Equation equation 18:

$$G^{(t+1)} = G^{(t)} \left(1 - \frac{\eta_g}{\eta_l}\right) + \frac{\eta_g}{\eta_l} \sum_{k=1}^{K} \Omega_k L_k^{(t+1)}. \tag{28}$$

Setting the two expressions equal:

$$G^{(t)} (1 - \lambda_k) + (1 - \lambda_k) \sum_{k=1}^{K} \Omega_k L_k^{(t+1)} = G^{(t)} \left(1 - \frac{\eta_g}{\eta_l}\right) + \frac{\eta_g}{\eta_l} \sum_{k=1}^{K} \Omega_k L_k^{(t+1)}. \tag{29}$$

This simplifies to:

$$\lambda_k = \frac{\eta_g}{\eta_l}. \tag{30}$$

Given that λ_k must lie in the range $[0, 1]$, this relationship holds when $\eta_g \leq \eta_l$. Since both η_g and η_l are also constrained within the interval $[0, 1]$, our findings are consistent with previous studies Oh et al. (2021); Collins et al. (2022b), which recommend keeping the head frozen by decreasing η_l. Additionally, we conducted experiments to further investigate the impact of the learning rate and mix factor on model performance in 5.3.1. □

Remark1. *Theorem 4 establishes a direct relationship between the mixup coefficient λ_k and the learning rates used in local and global updates. This insight allows us to interpret the mixup mechanism in pMixFed as adjusting the effective learning rate at the server, providing a theoretical foundation for selecting λ_k based on desired convergence properties.*

Remark2. *practice, this relationship suggests that by tuning λ_k, we can control the influence of the global model versus the local models in the aggregation process, similar to adjusting the learning rate in FedSGD. This is particularly beneficial in heterogeneous environments where clients may have varying data distributions.*

Conclusion of Analysis: Our theoretical analysis indicates that the mixup coefficient λ_k in *pMixFed* plays a role analogous to the learning rate in *FedSGD*. This equivalence provides a deeper understanding of how *pMixFed* leverages the strengths of *FedSGD* while mitigating its weaknesses in heterogeneous settings. By appropriately choosing λ_k, *pMixFed* can achieve faster convergence and improved robustness.

D.3 Generalization Efficiency

Due to space limitations, we provide a detailed generalization analysis in Appendix B.2. The analysis demonstrates how the mixup mechanism in *pMixFed* impacts the generalization performance, showing that the algorithm achieves a convergence rate comparable to *FedSGD* under similar assumptions.

E Analytic Experiments:

E.1 Handling Different Model Sizes and the Relationship Between Learning Rate and Mixup Factor

In this section, we explore how *pMixFed* accommodates variable model sizes across clients and investigate the relationship between the learning rate η and the mixup coefficient λ in this context. At the beginning of training, we partition each client's model into shared and private parts according to the *FedSim* or *FedAlt* protocols Pillutla et al. (2022), starting from the end of the global model. This allows for effective aggregation despite heterogeneity in model sizes.

Experiment: Investigating the Relationship Between Learning Rate and Mixup Factor To study the interplay between the learning rate and the mixup factor in the context of varying model sizes, we conducted an experiment with 10 clients, each having different model sizes ranging from 3 to 13 layers. We used MobileNet as the base architecture, with a maximum of 15 layers serving as the global model M_G. **Objective:** The goal was to examine how adjusting the mixup coefficient λ affects the training dynamics, especially when clients have different model capacities. We hypothesized that the mixup factor could compensate for discrepancies in learning rates caused by model size variations.

Experimental Setup:

- **Clients and Models:** 10 clients with local models of sizes $M_k \in \{3, 5, 7, 9, 11, 13\}$ layers.
- **Global Model:** MobileNet with 15 layers ($M_G = 15$).
- **Data Distribution:** Non-IID data distribution across clients to simulate realistic federated learning scenarios.
- **Learning Rates:** A constant local learning rate η_l for all clients. The global learning rate η_g was adjusted in relation to the mixup coefficient.
- **Mixup Coefficients:** For layers where $M_k < M_G$, $\lambda_i = 0$. For shared layers, λ_k was adjusted based on the client's model size.

Results:

Figure 6 illustrates the training accuracy over communication rounds for clients with different model sizes.

Table 3: Summary of performance metrics for different model sizes and mixup coefficients

Model Size (M_k)	Mixup Coefficient (λ_k)	Final client Accuracy (%)
3	0.8	85.2
5	0.7	86.5
7	0.6	87.3
9	0.5	88.0
11	0.4	88.5
13	0.3	88.9

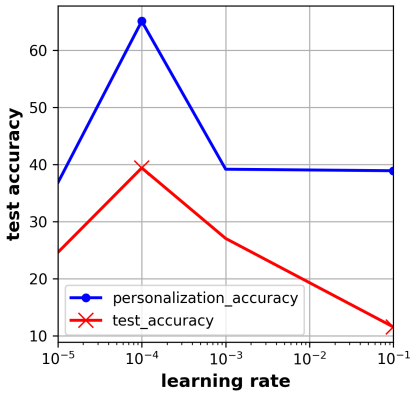

Figure 6: test accuracy curves of partial models with different learning rates lr.

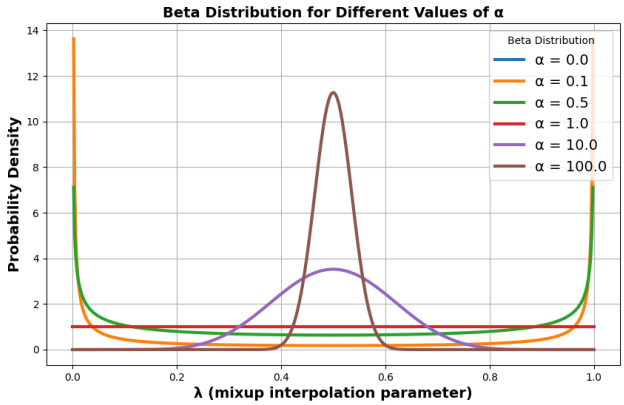

Figure 7: PDF λ (mixup degree) for different values of α in β distribution

F ROBUSTNESS AGAINST ADVERSARIAL ATTACKS

F.1 EFFECTS OF DIFFERENT MIX FACTOR

G EXTENDED ANALYSIS OF MIXUP FACTOR FROM THE BETA DISTRIBUTION

In this section, we provide a more detailed analysis of the mixup factor λ, sampled from the Beta distribution $\beta(\alpha, \alpha)$, as outlined in the main text. This distribution plays a crucial role in determining how the global and local models are mixed, influencing the overall performance of the federated learning system.

G.1 EFFECTS OF THE BETA DISTRIBUTION PARAMETER α

The Beta distribution $\beta(\alpha, \alpha)$ is defined on the interval [0,1], where the parameter α controls the shape of the distribution. The value of λ, used in Eq. 5, is naturally sampled from this distribution. By varying the parameter α, we can adjust how much mixing occurs between the global model G and the local model L_k.

- **Uniform Distribution:** When $\alpha = 1$, the Beta distribution becomes uniform over [0,1]. In this case, λ is sampled uniformly across the entire interval, meaning that each model, G and L_k, has an equal probability of being weighted more or less in the mixup process. This leads to a broad exploration of different combinations of global and local models, allowing for a wide range of mixed models.
- **Concentrated Mixup ($\alpha > 1$):** When $\alpha > 1$, the Beta distribution is concentrated around the center of the interval [0,1]. As a result, the mixup factor λ is more likely to be closer to 0.5, leading to more

balanced combinations of the global and local models. This results in outputs that are more "mixed," with neither model dominating the mixup process. Such a setup can enhance the robustness of the combined model, as it prevents extreme weighting of either model, creating smoother interpolations between them.

- **Extremal Mixup ($\alpha < 1$):** In contrast, when $\alpha < 1$, the Beta distribution becomes U-shaped, with more probability mass near 0 and 1. This means that λ tends to be either very close to 0 or very close to 1, favoring one model over the other in the mixup process. When $\lambda \approx 0$, the local model L_k is chosen almost exclusively, and when $\lambda \approx 1$, the global model G is predominantly selected. This form of mixup creates a more deterministic selection between global and local models, with less mixing occurring.

The behavior of different α values is depicted in Figure 7, where the distribution of the mixup factor λ is visualized. These distributions highlight the varying degrees of mixup, ranging from uniform blending to nearly deterministic model selection.

G.2 Experimental Setup: Investigating the Impact of α

To thoroughly investigate the impact of α on model performance, we designed two distinct experimental setups:

- **Random Sampling:** In this scenario, we set $\alpha = 1$, meaning that the λ values are sampled uniformly from the interval [0,1]. This ensures a wide range of mixup combinations between the global and local models. The random sampling approach helps us assess the general robustness of the model when the mixup degree λ is not biased towards any specific value.
- **Adaptive Sampling:** For this case, we divided the communication rounds into three distinct stages, each consisting of $\frac{epoch_{global}}{3}$ epochs. During these stages, the parameter α is adaptively changed as follows:

$$\alpha = \begin{cases} 0.1, & \text{initial stage (early training)} \\ 100, & \text{middle stage (convergence phase)} \\ 0.1, & \text{final stage (fine-tuning)} \end{cases} \tag{31}$$

  This adaptive strategy mimics the behavior of the original pFedMix algorithm while also allowing for more controlled exploration of different mixup combinations. During the early and late stages, a small α (0.1) encourages more deterministic model selections (i.e., either local or global), while the middle stage with $\alpha = 100$ promotes more balanced mixing. This dynamic adjustment of α enables us to control the degree of mixup at different phases of training.

G.3 Conclusion and Observations

Our extended experiments reveal that the choice of α has a significant impact on the performance of the mixed models. Uniform sampling ($\alpha = 1$) provides a general robustness across different communication rounds, but adaptive sampling offers more fine-grained control over the mixup process. In particular, the adaptive strategy with varying α values allows for better performance tuning across different training phases, as the model benefits from both deterministic selections and balanced mixups at different stages. We've furthur designed an experiment to see the effect of α on the overall performance of our proposed model: (a) random: In this scenario, we chose $\alpha = 1$. (b) adaptive. we divided the communication rounds in three stages : $\frac{epoch_{global}}{3}$, $\alpha = [0.1 \rightarrow 100 \rightarrow 0.1]$. This is the closestt possible to the original adaptive pFedMix algorithm, as the frozen layer.

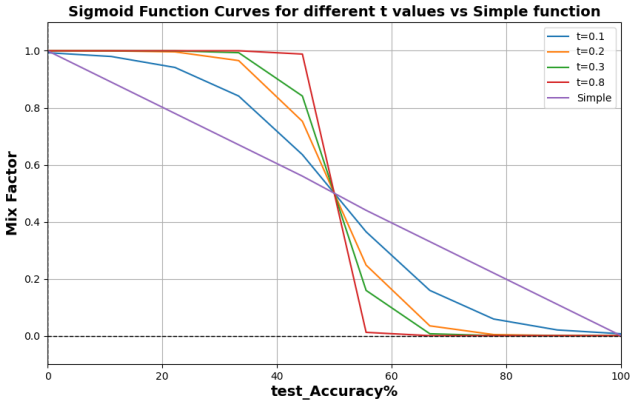

Figure 8: The Sigmoid Mixup factor μ function with different t values vs the **simple** function. The X-axis presents the input which is test accuracy of the local model in broadcasting phase and average test accuracy of the previous global model over all clients in the aggregation.