Conditional Moment Alignment for Improved Generalization in Federated Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this work, we study model heterogeneous Federated Learning (FL) for classification where different clients have different model architectures. Unlike existing works on model heterogeneity, we neither require access to a public dataset nor do we impose constraints on the model architecture of clients and ensure that the clients' models and data are private. We prove a generalization result, that provides fundamental insights into the role of the representations in FL and propose a theoretically grounded algorithm **Fed**erated **C**onditional **M**oment **A**lignment (Fed-CMA) that aligns class conditional distributions of each client in the feature space. We prove the convergence and empirically, we show that Fed-CMA outperforms other baselines on CIFAR-10, MNIST, EMNIST, FEMNIST in the considered setting.

1 Introduction

In Federated Learning, it is often possible that the client models have different architectures due to different computational hardware (such as GPU memory). In such cases, naive parameter aggregating as in FedAvg (McMahan et al., 2017) or FedProx (Li et al., 2020) are no longer possible. While some knowledge distillation approaches use a public dataset to address this, a public dataset may not always be available and the role of the public dataset on the generalization performance is not yet understood. Also, the type of architecture used in deep learning has a huge impact on the performance. The users may consider the architecture as a trade secret (often due to the amount of resources spent in designing it) and may not be willing to share the model architecture. We term this as *model privacy*. This also implies that the users must be free to choose the type of model architecture. In this work, we consider the data and model heterogeneous FL problem with these additional constraints: (i) **model privacy**, (ii) **unrestricted model space** and (iii) **no public dataset**. Note that, we restrict our study to the use of neural networks. See Table 1 for a summary of relevant works. Below, we provide an example where these constraints are practical.

Consider the following application where a product manufacturer works with multiple OEMs (Original Equipment Manufacturers) from different countries (due to regulatory restrictions) for developing a new product line. The product manufacturer intends to improve the diagnostics and prognostics which often involves a classification task. Since it is a new product line, data is scarce at every OEM, thus requiring the OEMs to collaborate. The OEMs may differ in the type of sensors leading to *data heterogeneity* and can not share the data due to *data privacy* concerns. They may also have *model heterogeneity* due to different computational hardware (such as GPU memory) that restricts the type of model architectures they use. Moreover, in such scenarios, the OEMs may not be willing to share the model architecture as they may consider it as a trade secret and due to the amount of resources spent in designing it manually or using techniques such as Neural Architecture Search (Elsken et al., 2019).

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this paper, we take a fundamentally different approach to address model heterogeneity and propose an algorithm that is motivated from theoretical results in the *Domain Adaptation* (DA) literature. In our case since the client models are different, we separate the client model architecture as a feature extractor that projects the input data into a latent space (common for all clients) and a classifier that acts on the latent space. We summarize our contributions below: (i) We show the theoretical generalization guarantees for model homogeneous and model heterogeneous FL, highlighting the difficulty in the latter case. To the best of our knowledge, this is the first theoretical generalization error bound for the model heterogeneous FL setup, which justifies *the importance of aligning the latent space distributions across the clients* and *the advantage of participating in FL*. (ii) We propose a simple algorithm (Fed-CMA) to align the latent space conditional distributions and the classification weights across all clients in a Federated way. We show convergence guarantees for the proposed algorithm for general non-convex problems. (iii) We provide a detailed evaluation under different FL settings and datasets and observe that Fed-CMA outperforms the considered baselines. Additionally, Fed-CMA has reduced communication complexity as compared to other model heterogeneous FL algorithms.

Work	MH	MP	PD	RM
Zhang et al. (2021)	✓	✓	✓	×
Lin et al. (2020) Cho et al. (2022)	✓	×	✓	×
Diao et al. (2020) Li et al. (2021) Yao et al. (2021)	✓	×	×	✓
Litany et al. (2022)	✓	✓	×	✓
Zhu et al. (2021) **Ours**	✓	✓	×	×

Table 1: MH: Model Heterogeneity, MP: Model Privacy, PD: Public Datset, RM: Restricted Model space.

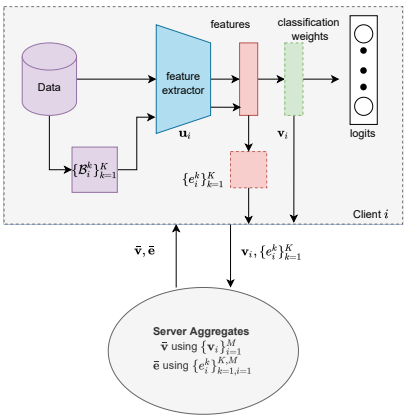


Figure 1: Illustration of Fed-CMA .

2 Model Heterogeneous Federated Learning Formulation

Consider a K-class classification task, $\mathcal{T} := \langle \mathcal{D}, c \rangle$, where $\mathcal{D}$ is the distribution on $\mathcal{X} \subset \mathbb{R}^d$ and c is the ground truth labeling function that maps $\mathcal{X}$ to $\mathcal{Y} := \{1, \cdots, K\}$. We consider the model heterogeneous FL problem with M workers where each worker has a task $\mathcal{T}_i := \langle \mathcal{D}_i, c \rangle \; \forall i \in [M]$. Each client has $D_i := \{x_j, y_j\}_j^{N_i}$, such that $x_j \sim \mathcal{D}_i$, $y_j = c(x_j)$. Let $p_i \in [0, 1]$ such that, $N_i = p_i N$, where $N = \sum_{i=1}^M N_i$. Each client has a model $\mathbf{w}_i$ in a hypothesis class $\mathcal{W}_i \subset \mathbb{R}^{d_i}$ (where d_i is the dimensionality of the model space $\mathcal{W}_i$) and a function $f_i : \mathcal{W}_i \times \mathcal{X} \to \mathcal{Y}$ which is used to make a prediction on a given data point. We denote $\mathbf{w}_{(i,t)}$ as the weights at client i at time t. We simply write $\mathbf{w}_i$ when time is clear from context and $\mathbf{w}$ when worker and time are clear from context. Given a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, we define the loss function $l_i : \mathcal{W}_i \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ for each worker as $l_i(\mathbf{w}_i, x, y) = l\big(f_i(\mathbf{w}_i, x), y\big)$. Given a finite dataset $D_i = \{(x_j, y_j)\}_{j=1}^{N_i}$, we define the population loss L_i and the empirical loss $\widehat{L}_i$ (the subscript i refers to $\mathcal{D}_i$ at the worker i) as $L_i(\mathbf{w}_i) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\big[l_i(\mathbf{w}_i, x, y)\big]; \widehat{L}_i(\mathbf{w}, D_i) = \frac{1}{N_i}\sum_{j=1}^{N_i} l_i\Big(\mathbf{w}, x_j, y_j\Big)$. The objective of model heterogeneous FL is to simultaneously learn $\{\mathbf{w}_i^*\}_{i=1}^M$ such that

$$(\mathbf{w}_1^*, \mathbf{w}_2^*, \cdots, \mathbf{w}_M^*) = \arg \min_{\{\mathbf{w}_i\}_{i=1}^M} \sum_{i=1}^M \alpha_i L_i(\mathbf{w}_i). \tag{1}$$

where $\alpha_i \geq 0$, $\sum_{i=1}^M \alpha_i = 1$. Popular choices for α_i are $\frac{1}{M}$ or p_i. The challenge in solving (1) is because $\{\mathbf{w}_i\}_{i=1}^M$ share different network architectures which means that existing algorithms on parameter aggregation (such as FedAvg, FedProx) can not be used. Under model heterogeneity, we only assume a common latent space ($\mathcal{Z} \subseteq \mathbb{R}^{d_e}$) for all workers. In other words, each worker i has a model $\mathbf{w}_i = (\mathbf{u}_i, \mathbf{v}_i) \in \mathcal{W}_i$ where $\mathbf{u}_i \in \mathcal{U}_i$, $\mathbf{v}_i \in \mathcal{V}$. We define the prediction function

as $f_i(\mathbf{w}_i, x) := h\big(\mathbf{v}_i, g_i(\mathbf{u}_i, x)\big)$ where $g_i(\mathbf{u}_i, \cdot) : \mathcal{X} \to \mathcal{Z}$ projects $\mathcal{X}$ to the latent space and $h(\mathbf{v}_i, \cdot) : \mathcal{Z} \to \mathcal{Y}$ makes the prediction. In general, $\dim(\mathcal{U}_i) \gg \dim(\mathcal{V})$. For example, consider a neural network where $\mathbf{u}_i$ corresponds to the weights of the feature extractor and $\mathbf{v}_i$ corresponds to the weights of the classification layer (see Fig 1). Pillutla et al. (2022) and Liang et al. (2020) also consider a similar setup for personalizing the input representation learning layers while sharing the classification layer with the server.

3 Generalization Result for Model Heterogeneous FL

Let us first consider the case where all the client models are homogeneous, and the client weights can be shared. The weighted data distribution is denoted as $\mathcal{D}_{\boldsymbol{\alpha}} = \sum_{i=1}^{M} \alpha_i \mathcal{D}_i$. Let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_M)$ be the domain weights for each user such that $\sum_{j=1}^{M} \alpha_i = 1$, and let j be the target domain (which can be one of the clients). Let the empirical risk be defined as $\widehat{\epsilon}_i(\mathbf{w}_i) := \frac{1}{N_i} \sum_{(x,y)\in D_i} |f(\mathbf{w}_i, x) - y|$ and the $\boldsymbol{\alpha}$ empirical weighted risk as $\widehat{\epsilon}_{\boldsymbol{\alpha}}(\mathbf{w}) = \sum_{i=1}^{M} \alpha_i \widehat{\epsilon}_i(\mathbf{w})$ and the true weighted risk $\epsilon_{\boldsymbol{\alpha}}$ is defined similarly. Theorem 2 of Peng et al. (2019) shows a generalization bound for Federated Learning as follows: with probability at least $1 - \delta$, for $\mathbf{w} = \sum_{i=1}^{M} \alpha_i \mathbf{w}_i$,

$$\epsilon_j(\mathbf{w}_j) \leq \widehat{\epsilon}_{\boldsymbol{\alpha}}\Big(\sum_{i=1}^{M} \alpha_i \mathbf{w}_i\Big) + \sum_{i=1}^{M} \alpha_i \Big(\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_j)\Big) + \mathcal{O}\Big(\frac{1}{\sqrt{N}}\Big). \tag{2}$$

Here, $\mathcal{O}(\cdot)$ hides logarithmic terms and function class complexities that are constant. If the target domain only used the data available locally, from standard results in generalization, we know that the user's generalization error decays as $\mathcal{O}\big(\sqrt{1/N_j}\big)$. The result shows the advantage of going federated by showing $\sqrt{1/N}$ dependence since $N > N_j$. However, this result has two short-comings. Firstly, they assume that the weighted dataset $D_{\boldsymbol{\alpha}}$ is present in a centralized location which is not practical for FL setup. This is key to their proof to show $\sqrt{1/N}$ dependence. Secondly, their result only holds of model homogeneous FL, since the generalization error is with respect to the weighted model $\sum_{i=1} \alpha_i \mathbf{w}_i$, which is clearly not possible in model heterogeneous (or model private) FL. In our setup, only the final classification layer model space is common among the users. Therefore, we need decouple the representation layers from the classification layer to show the generalization bound. Now, we show a generalization result that addresses both these concerns.

We present here a trick to show the desired generalization bound for the general non-convex FL problems. We consider the following setup: (i) Train any FL algorithm for T time steps and let the feature extractor part of the global model at the end of training be denoted by $\mathbf{u}$, (ii) fix the representation learning weights of all the clients to $\mathbf{u}$ and freeze the weights, (iii) continue training the final layer weights using FL (along with an l2 regularizer). This *freezing the representation weights* and *continuing in the latent space* (see Appendix B.1) enables us to extend the generalization result to the model heterogeneous case.

Let us first show the result for the model homogeneous case. Given a representation function $\mathbf{u}$, denote the induced distribution of the ith client as $\tilde{\mathcal{D}}_i(\mathbf{u})$ and the induced labeling function as $\tilde{c}_{\mathbf{u}}$. The empirical risk of the ith client induced by $(\mathbf{u}, \mathbf{v})$ are defined as $\widehat{\epsilon}_{i,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}}) := \frac{1}{N_i} \sum_{x\in D_i} |h(\mathbf{v}, g(\mathbf{u}, x)) - \tilde{c}_{\mathbf{u}}(g(\mathbf{u}, x))|$ and the true risk $\epsilon_{i,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}})$ is the expectation of $\widehat{\epsilon}_{i,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}})$ taken with respect to D_i. The domain weighted empirical risk is defined similarly as $\widehat{\epsilon}_{\boldsymbol{\alpha}}(\mathbf{v}) = \sum_{i=1}^{M} \alpha_i \widehat{\epsilon}_{i,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}})$ and the true weighted risk is defined similarly as $\epsilon_{\boldsymbol{\alpha}}(\mathbf{v})$. We denote the mixture of the induced client distributions defined as $\tilde{\mathcal{D}}_{\boldsymbol{\alpha}} = \sum_{i=1}^{M} \alpha_i \tilde{\mathcal{D}}_i(\mathbf{u})$. Let the target distribution be given by $\mathcal{D}_j$ and the induced distribution is $\tilde{\mathcal{D}}_j(\mathbf{u})$. We now state the generalization bound.

Theorem 1 (Model Homogeneity)**.** *Given* $\mathbf{u}$*, let* $\widehat{\mathbf{v}} = \arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{\epsilon}_{\boldsymbol{\alpha}}(\mathbf{v})$ *and* $\mathbf{v}_j^* = \arg\min_{\mathbf{v}\in\mathcal{V}} \epsilon_{j,\mathbf{u}}(\mathbf{v}, \tilde{c}_{\mathbf{u}})$*. Then for any* $\delta > 0$*, w.p.* $> 1 - \delta$*, we have*

$$\epsilon_{T,\mathbf{u}}(\widehat{\mathbf{v}}) - \epsilon_{T,\mathbf{u}}(\mathbf{v}_j^*) \leq 4\sqrt{\sum_{i=1}^{M} \frac{\alpha_i^2}{p_i}} \mathcal{O}\Big(\sqrt{\frac{1}{2N}}\Big) + \sum_{i=1}^{M} \alpha_i d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_i(\mathbf{u}), \tilde{\mathcal{D}}_j(\mathbf{u})\big).$$

Let $\alpha_i = p_i$, then the generalization error decays as $\mathcal{O}\big(\sqrt{1/N}\big)$. Therefore, the dependence on the number of data points shows that the client attains better generalization error by participating

in the FL setup provided the divergence term is small. Note that, as compared to (2), we have divergence between the induced distributions $\tilde{\mathcal{D}}_i(\mathbf{u})$ due to the representation weights $\mathbf{u}$ as compared to $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_j)$ which is fixed for the given client datasets. The divergence term in Theorem 1 can be minimized by learning invariant feature representations $\mathbf{u}$ which has been the topic of interest in domain invariant representation learning (Long et al., 2015; Zhao et al., 2019).

For the model heterogeneous case (since there is no global model), let $\mathbf{u}_i$ be the frozen representation layer weights of the ith user. We then define the induced distributions $\tilde{\mathcal{D}}_i(\mathbf{u}_i)$, labeling functions $\tilde{c}_{\mathbf{u}_i}$, empirical risk $\widehat{\epsilon}_{i,\mathbf{u}_i}(\mathbf{v}_i, \tilde{c}_{\mathbf{u}_i})$ with respect to $\mathbf{u}_i$ for every client. The weighted empirical risk is then $\widehat{\epsilon}_{\boldsymbol{\alpha}}(\mathbf{v}) = \sum_{i=1}^{M} \alpha_i \widehat{\epsilon}_{i,\mathbf{u}_i}(\mathbf{v}, \tilde{c}_{\mathbf{u}_i})$. A major difference is that, the induced labeling functions $\tilde{c}_{\mathbf{u}_i}$ are different for every client whereas in Theorem 1, it is same for all clients due to shared $\mathbf{u}$. Let the target distribution be given by $\mathcal{D}_j$ and the representation $\mathbf{u}_j$ that induces $\tilde{\mathcal{D}}_j(\mathbf{u}_j)$ and $\tilde{c}_{\mathbf{u}_j}$. We now state the generalization bound.

Theorem 2 (Model Heterogeneity)**.** *Let $\widehat{\mathbf{v}} = \arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{\epsilon}_{\boldsymbol{\alpha}}(h)$ be the output of continuing in the latent space dataset and $\mathbf{v}_j^* = \arg\min_{\mathbf{v}\in\mathcal{V}} \epsilon_{T,\mathbf{u}_j}(\mathbf{v}, \tilde{c}_{\mathbf{u}_j})$ be the minimizer of the true target risk. Then for any $\delta > 0$, w.p. $> 1-\delta$, we have*

$$\epsilon_{T,\mathbf{u}_j}(\widehat{\mathbf{v}}) - \epsilon_{T,\mathbf{u}_j}(\mathbf{v}_j^*) \leq 4\sqrt{\sum_{i=1}^{M} \frac{\alpha_i^2}{p_i}} \mathcal{O}\Big(\sqrt{\frac{1}{N}}\Big) + \underbrace{2\Big(\lambda_\alpha + \frac{1}{2}\sum_{i=1}^{M} \alpha_i d_{\mathcal{H}\Delta\mathcal{H}}(\tilde{\mathcal{D}}_i(\mathbf{u}_i), \tilde{\mathcal{D}}_T(\mathbf{u}_j))\Big)}_{B},$$

where $\lambda_\alpha = \sum_{i=1}^{M} \alpha_i \min\Big\{\mathbb{E}_{z\in\tilde{\mathcal{D}}_i(\mathbf{u}_i)}\big[|\tilde{c}_{\mathbf{u}_i}(z) - \tilde{c}_{\mathbf{u}_j}(z)|\big], \mathbb{E}_{z\in\tilde{\mathcal{D}}_j(\mathbf{u}_j)(z)}\big[|\tilde{c}_{\mathbf{u}_i}(z) - \tilde{c}_{\mathbf{u}_j}(z)|\big]\Big\}$.

The bound is similar to that of Theorem 1 with two differences: (i) the addition of the $\lambda_{\boldsymbol{\alpha}}$ term and (ii) the divergence is computed between the induced distributions on private representations $\mathbf{u}_i$ of the clients. These two differences are inherently due to the presence of model heterogeneity where the private representations $\mathbf{u}_i$ belong to different model spaces.

Role of the representations $\{\mathbf{u}_i\}$: The term B is non-zero when the data distributions are non-iid. If the given representations $\{\mathbf{u}_j\}$ are such that in the latent space $\mathcal{Z}$, the induced distribution of the target $\tilde{\mathcal{D}}_j(\mathbf{u}_j)$ is close to the induced $\tilde{\mathcal{D}}_i(\mathbf{u}_i)$, then $d_{\mathcal{H}\Delta\mathcal{H}}(\tilde{\mathcal{D}}_i(\mathbf{u}_i), \tilde{\mathcal{D}}_j(\mathbf{u}_j))$ is small. The λ_α term, determines the closeness of $\tilde{c}_{\mathbf{u}_j}$ with every other $\tilde{c}_{\mathbf{u}_i}$. Observe that in the model homogeneous case where $\mathbf{u}$ is shared among the clients, $\lambda_{\boldsymbol{\alpha}} = 0$ since all the induced labeling functions are equal to $\tilde{c}_{\mathbf{u}}$. Even when the representations are shared and the $d_{\mathcal{H}\Delta\mathcal{H}}$ is small, this is not guaranteed to be small (Zhao et al., 2019). The goal of representation learning should therefore be to simultaneously reduce the divergence between the marginal distributions and the distance between the labeling functions. While the divergence can be minimized by minimizing an IPM (such as MMD), the distance between the induced labeling functions is difficult to measure. Therefore, in Fed-CMA , we propose to minimize the divergence between the class conditional latent space distributions to bring the induced labeling functions closer.

Comparison with existing generalization bounds for FL: The generalization bound of Cho et al. (2022) does not explain the benefits of participating in the FL setup as they show $\mathcal{O}(\sqrt{1/N_j})$ dependence instead of $\mathcal{O}(\sqrt{1/N})$. The bounds in Zhu et al. (2021); Zhao et al. (2018); Peng et al. (2019); Cho et al. (2022) are relative to the empirical error on the sources whereas our bound is more informative since we show $\mathcal{O}(\sqrt{1/N})$ dependence relative to the optimal error on the target task $\epsilon_{T,\mathbf{u}_j}(\mathbf{v}_j^*)$. While FL research primarily focuses on improving the convergence rate by addressing issues such as variance reduction or client-drift (Karimireddy et al., 2020), the role of the feature extractors in reducing the degree of heterogeneity in the latent space received little attention. Our results provide these fundamental insights. To the best of our knowledge, our work here is the first to show generalization bounds for model heterogeneous FL that provide useful insights about aligning the distributions in the latent space .

4 Conditional Moment Alignment

We propose a simple algorithm that aligns the latent space conditional distributions to minimize the $d_{\mathcal{H}\Delta\mathcal{H}}$ term. We do this by aligning only the first order moments of the class conditional mean

Algorithm 1 Fed-CMA

At Server:
for $t = 1, \cdots, T$ **do**
  Collect mean embeddings $\{e^k_{(i,t)}\}_{k=1}^K$ and classification layer weights $\mathbf{v}_{(i,t)}$ for all $i \in [M]$
  Update $\bar{\mathbf{e}}$ and $\bar{\mathbf{v}}$ using (5)
  Broadcast $\{e^k_{(t)}\}_{k=1}^K$ and $\bar{\mathbf{v}}_{(t)}$ to the workers.
end for
At Worker i**:**
for $t = 1, \cdots, T$ **do**
  Receive $\{e^k_{(t)}\}_{k=1}^K$ and $\bar{\mathbf{v}}_{(t)}$ from the server.
  Pick random minibatch $\xi_{(t)}$ from D_i and compute $\nabla \widehat{\Phi}_i$.
  Update $\mathbf{w}_{(i,t)}$ according to (4)
  Compute $\{e^k_{(i,t)}\}_{k=1}^K$ using $e^k_i = \frac{1}{b^k_i} \sum_{x \in \mathcal{B}^k_i} g_i(\mathbf{u}_{(i,t)}, x)$ for $k \in [K]$.
  Share with server: mean embeddings $\{e^k_{(i,t)}\}_{k=1}^K$ and classification layer weights $\mathbf{v}_{(i,t)}$.
end for

embeddings of each client and maintain a global variable that tracks the weighted average of the individual mean embeddings. Let p^k_i denote the fraction of data points of label k available at client i out of all data points of label k at all clients. At the beginning of the training, each client i samples a fixed i.i.d minibatch $\mathcal{B}^k_i \sim \mathcal{D}_i$ for each kth label in the dataset D_i with b^k_i points. The size of b^k_i can be much smaller than the size of the original dataset N_i. This minibatch is used throughout the training. The class conditional mean embedding vector $e^k_i \in \mathbb{R}^{d^e}$ is computed as $e^k_i = \frac{1}{b^k_i} \sum_{x \in \mathcal{B}^k_i} g_i(\mathbf{u}_i, x)$. We abuse the notation slightly, by using $e^k_i(\mathbf{u})$ when the parameter $\mathbf{u}$ is of interest. These vectors are the class conditional empirical means of each class available at each client.

We maintain a global variable $\mathbf{e} = \{e^k\}_{k=1}^K$ that stores a weighted average of the means of all the clients. This global variable is used to achieve consensus among the clients and align the client distributions in the latent space. Similarly, we maintain a global variable $\bar{\mathbf{v}}$ for the classification layer weights. The population local loss at worker i is

$$\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}) = L_i(\mathbf{w}_i) + \frac{\lambda_1}{2} p_i \left\| \mathbf{v}_i - \bar{\mathbf{v}} \right\|_2^2 + \frac{\lambda_2}{2} \sum_{k=1}^K \frac{p^k_i}{b^k_i} \sum_{x \in \mathcal{B}^k_i} \left\| g_i(\mathbf{u}_i, x) - e^k \right\|_2^2. \tag{3}$$

Note that, the computation here is just linear in the number of data points as compared to quadratic in the case of MMD. Let $\mathbf{w}_{(i,t)} := (\mathbf{u}_{(i,t)}, \mathbf{v}_{(i,t)})$ be the parameters at ith worker and $\eta_{(t)}$ be the learning rate at time t. We define $\mathbf{e}_t = \{e^k_{(t)}\}_{k=1}^K$ and $\bar{\mathbf{v}}_{(t)}$. Note that we use (t) to highlight that the subscript refers to the time. In Fed-CMA , we perform the following updates on $\mathbf{w}_{(i,t)}, \bar{\mathbf{v}}_{(t)}, \mathbf{e}_{(t)}$ simultaneously,

$$\text{(Local update)} \left\{ \mathbf{w}_{(i,t+1)} \quad = \mathbf{w}_{(i,t)} - \eta_{(t)} \nabla_{\mathbf{w}_i} \widehat{\Phi}_i(\mathbf{w}_{(i,t)}, \bar{\mathbf{v}}_{(t)}, \mathbf{e}_{(t)}); \right. \tag{4}$$

$$\text{(Global update)} \begin{cases} \bar{\mathbf{v}}_{(t+1)} & = \bar{\mathbf{v}}_{(t)}(1 - \lambda_1 \eta_{(t)}) + \lambda_1 \eta_{(t)} \left(\sum_{i=1}^M p_i \mathbf{v}_{(i,t)} \right); \\ e^k_{(t+1)} & = e^k_{(t)}(1 - \lambda_2 \eta_{(t)}) + \lambda_2 \eta_{(t)} \left(\sum_{i=1}^M p^k_i e^k_i \left(\mathbf{u}_{(i,t)} \right) \right), \end{cases} \tag{5}$$

where the empirical gradient $\widehat{\Phi}_i$ is computed using a minibatch ξ_i that is drawn i.i.d. from D_i and $0 < \lambda_1 \eta_{(t)} \leq 1$ and $0 < \lambda_2 \eta_{(t)} \leq 1$. We provide a summary of Fed-CMA in Algorithm 1. We study the convergence properties of this algorithm in Appendix B.

5 Numerical Simulations

It is difficult to compare the performance of Fed-CMA with earlier works such as Fed-ET (Cho et al., 2022), FedDF (Lin et al., 2020) and KT-pFL(Zhang et al., 2021) since they utilize additional information in the form of a public dataset that is used for knowledge distillation. Moreover, these

works may involve sharing the parameters with the server. Therefore, we do not provide a comparison with these works. Without using a public dataset, FedGen trains a generator at the server using knowledge distillation. This generator is shared with the clients which use the generated data as an augmented data to improve their training. While the main results in Zhu et al. (2021) require sharing the model weights with the server (the server aggregates the weights), FedGen also accommodates limited parameter sharing (such as sharing only the classification layer weights $\mathbf{v}_i$). Therefore, in this section, we compare Fed-CMA with FedGen under the limited parameter setting. Additionally, we also provide a comparison against local training and against FedAvg-partial which simply shares the final layer weights in a FedAvg fashion (called FedAvgSim in Pillutla et al. (2022)).

We perform evaluation on MNIST, EMNIST and FEMNIST (Caldas et al., 2018) datasets. From Table 2, we observe that the proposed algorithm outperforms local training, FedAvg and FedGen(Zhu et al., 2021). For MNIST and EMNIST, we consider $M = 20$ clients and to simulate **data heterogeneity**, we follow Hsu et al. (2019) where a Dirichlet distribution $\text{Dir}_M(\alpha)$ is used to partition among the M clients. For smaller values of α, the data is more heterogeneous and for larger values of α the data is more homogeneous across the clients. For the MNIST and EMNIST datasets, we consider full client availability. In the FEMNIST experiment, we consider around 25% of the FEMNIST dataset and contains 923 clients. At each communication round, roughly 50% of the clients are sampled at random.

From the results in Table 2, we observe that Fed-CMA outperforms all the baselines in the limited parameter setting. The improvement offered by Fed-CMA over the other baselines is much higher in the high data heterogeneity (small α) setting and the improvements diminish as α increases. This shows that in more practical settings, Fed-CMA can provide much needed gains to justify participating in the FL setup. We provide more experiments with detailed ablations in Appendix E.

dataset	α	Local	FedAvg	FedGen	FedCMA
MNIST	0.02	76.28	76.14	75.95	**79.22**
	0.03	59.89	59.82	60.08	**64.03**
	0.05	61.36	61.26	60.82	**62.57**
	0.10	62.27	62.26	62.84	**63.94**
EMNIST	0.02	50.2	49.77	49.93	**51.0**
	0.03	47.95	47.46	47.5	**49.26**
	0.05	52.73	52.2	52.48	**53.96**
	0.10	**47.31**	46.84	46.35	46.5
FEMNIST	-	63.20	64.30	-	**65.60**

Table 2: Comparison on MNIST, EMNIST and FEMNIST datasets with data heterogeneity and limited parameter sharing

6 Conclusion

In this work, we focused on the model heterogeneous FL setup without assuming a public dataset, without imposing restrictions on the choice of the model architectures and keeping the model architectures private. We proposed Fed-CMA based on conditional distribution alignment in the latent space in a federated way. We prove the convergence and generalization properties of the algorithm with emphasis on the role of the learned representations. These insights are helpful in designing better FL algorithms and the proposed algorithm can be used along with existing FL algorithms in the weight sharing setup, and/or use an adversarial loss to improve the performance. Moreover, our work opens up several intriguing questions: (i) What level of model heterogeneity can be tolerated to achieve a reasonable improvement in the FL setup?, (ii) What is the role of the latent space in tolerating this model heterogeneity?

References

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, 79(1):151–175.

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. (2016). Domain separation networks. *Advances in neural information processing systems*, 29.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. (2018). Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*.

Cho, Y. J., Manoel, A., Joshi, G., Sim, R., and Dimitriadis, D. (2022). Heterogeneous ensemble knowledge transfer for training large models in federated learning. *arXiv preprint arXiv:2204.12703*.

Diao, E., Ding, J., and Tarokh, V. (2020). Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. (2006). A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19.

Hsu, T.-M. H., Qi, H., and Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.

Johansson, F. D., Sontag, D., and Ranganath, R. (2019). Support and invertibility in domain-invariant representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 527–536. PMLR.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Li, A., Sun, J., Li, P., Pu, Y., Li, H., and Chen, Y. (2021). Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 420–437.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. (2019). On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.

Liang, P. P., Liu, T., Ziyin, L., Allen, N. B., Auerbach, R. P., Brent, D., Salakhutdinov, R., and Morency, L.-P. (2020). Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. (2020). Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363.

Litany, O., Maron, H., Acuna, D., Kautz, J., Chechik, G., and Fidler, S. (2022). Federated learning with heterogeneous architectures using graph hypernetworks. *arXiv preprint arXiv:2201.08459*.

Long, M., Cao, Y., Wang, J., and Jordan, M. (2015). Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR.

Long, M., Wang, J., Ding, G., Sun, J., and Yu, P. S. (2013). Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Peng, X., Huang, Z., Zhu, Y., and Saenko, K. (2019). Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*.

Pillutla, K., Malik, K., Mohamed, A.-R., Rabbat, M., Sanjabi, M., and Xiao, L. (2022). Federated learning with partial model personalization. In *International Conference on Machine Learning*, pages 17716–17758. PMLR.

Shui, C., Li, Z., Li, J., Gagné, C., Ling, C. X., and Wang, B. (2021). Aggregating from multiple target-shifted sources. In *International Conference on Machine Learning*, pages 9638–9648. PMLR.

Tachet des Combes, R., Zhao, H., Wang, Y.-X., and Gordon, G. J. (2020). Domain adaptation with conditional distribution matching and generalized label shift. *Advances in Neural Information Processing Systems*, 33:19276–19289.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176.

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. (2014). Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Yao, D., Pan, W., Wan, Y., Jin, H., and Sun, L. (2021). Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization. *arXiv preprint arXiv:2111.14655*.

Zellinger, W., Grubinger, T., Lughofer, E., Natschläger, T., and Saminger-Platz, S. (2017). Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811*.

Zhang, J., Guo, S., Ma, X., Wang, H., Xu, W., and Wu, F. (2021). Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems*, 34.

Zhao, H., Des Combes, R. T., Zhang, K., and Gordon, G. (2019). On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532. PMLR.

Zhao, H., Zhang, S., Wu, G., Moura, J. M., Costeira, J. P., and Gordon, G. J. (2018). Adversarial multiple source domain adaptation. *Advances in neural information processing systems*, 31.

Zhu, Z., Hong, J., and Zhou, J. (2021). Data-free knowledge distillation for heterogeneous federated learning. *arXiv preprint arXiv:2105.10056*.

A Additional Preliminaries

A.1 Domain Adaptation

In DA, a classifier is trained on a sufficiently large source dataset (task) such that it is expected to perform well on a target dataset (task) with few or no labeled data points. For the purpose of understanding the DA perspective and studying the generalization, we shall consider a binary classification task. Let $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} = [0, 1]$, which is construed as the probability of having the label 0. Let us denote the random variables in $\mathcal{X}, \mathcal{Y}$ as X and Y respectively. We use the terms task, domain and client interchangeably from hereon. For this discussion, we denote the hypothesis class $\mathcal{W}$ as $\mathcal{H}$. We define the disagreement between two hypotheses $\mathbf{w}, \mathbf{w}' \in \mathcal{H} \subseteq \{0, 1\}^{\mathcal{X}}$ as $\epsilon_i(\mathbf{w}, \mathbf{w}') = \mathbb{E}_{x \in \mathcal{D}_i}\big[|f(\mathbf{w}, x) - f(\mathbf{w}', x)|\big]$. Suppose that the target labeling functions $\{c_i : \mathcal{X} \to \mathcal{Y}\}$ are different

for every client, then the error of a hypothesis $\mathbf{w} \in \mathcal{H}$ on a client i is as $\epsilon_i(\mathbf{w}) := \epsilon_i(\mathbf{w}, c_i)$. The error of a hypothesis on clients i, j can be related as (Zhao et al., 2019)

$$\epsilon_j(\mathbf{w}) \leq \epsilon_i(\mathbf{w}) + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_j) + \min\left\{\mathbb{E}_{x\in\mathcal{D}_i}\left[|c_i(x) - c_j(x)|\right], \mathbb{E}_{x\in\mathcal{D}_j}\left[|c_i(x) - c_j(x)|\right]\right\}, \quad (6)$$

where $d_{\mathcal{H}\Delta\mathcal{H}}$ denotes the divergence between the two distributions $\mathcal{D}_i, \mathcal{D}_j$. Ben-David et al. (2010) describe a way to empirically measure $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_j)$ in terms of the accuracy of a classifier that discriminates between the two datasets. If the divergence is small, it is hard to discriminate between the two data distributions. This shows that, if the labeling functions c_i, c_j are close, and $d_{\mathcal{H}\Delta\mathcal{H}}$ is small, then a hypothesis trained on one client i (source) can perform well on another client j (target). Let $q_i(X, Y)$ denote the joint distribution of X, Y for client i. Suppose that the labeling functions $\{c_i\}$ are the same for all the clients, then $q_i(Y|X) = q_j(Y|X)$ but the marginals of X are not equal. This is called the *covariate shift* assumption (Johansson et al., 2019) which we consider in this work. Under this assumption, the last term in (6) is zero and only $d_{\mathcal{H}\Delta\mathcal{H}}$ needs to be small. However, given two tasks, the divergence between the distributions in $\mathcal{X}$ is fixed. This motivated the learning of *domain invariant representations* such that the divergence of distributions in the representation space (or latent space) is minimized. Broadly, these methods take the following approaches. **(i) Minimizing an integral probability metric IPM):** Along with the classification loss on source data, the network is explicitly trained to minimize an IPM such as Maximum Mean Discrepancy (MMD (Gretton et al., 2006)) (Long et al., 2013; Tzeng et al., 2014; Long et al., 2015) or a Central Moment Discrepancy (CMD) (Zellinger et al., 2017) between the source and target datasets at the representation layer. **(ii) Adversarial methods:** Ganin et al. (2016); Bousmalis et al. (2016); Tzeng et al. (2017) train the network by minimizing the label classification loss and maximizing a domain classification loss.

Let $\mathcal{Z} \subset \mathbb{R}^{d_e}$ is a latent space, $g(\mathbf{u}, \cdot) : \mathcal{X} \to \mathcal{Z}$ is a representation function where $\mathbf{u} \in \mathcal{U}$ and Z is a random variable in $\mathcal{Z}$. Given a task $\langle \mathcal{D}, c\rangle$, $\mathbf{u}$ induces the distribution $\tilde{\mathcal{D}}(\mathbf{u})$ on $\mathcal{Z}$ and the labeling function $\tilde{c}_{\mathbf{u}} : \mathcal{Z} \to \mathcal{Y}$ such that $\mathbb{E}_{z\in\tilde{\mathcal{D}}(\mathbf{u})}\left[1_B(z)\right] = \mathbb{E}_{x\in\mathcal{D}}\left[1_B(g(\mathbf{u}, x))\right]$, where $B \subset \mathcal{Z}$ is a measurable set, 1_B is an indicator fuction over B, and $\tilde{c}_{\mathbf{u}}(z) := \mathbb{E}_{x\in\mathcal{D}}\left[c(x)|g(\mathbf{u},x) = z\right]$, $\forall z \in \mathcal{Z}$. Let $q_i(Z, Y)$ be the joint distribution at client i using a representation $\mathbf{u}_i$. Under this setup, the induced labeling functions in the latent space may not be equal even when the labeling functions in the input space are the same (Zhao et al., 2019) and several works followed to address this shortcoming by assuming a *generalized label shift* condition (Tachet des Combes et al., 2020; Shui et al., 2021) where $q_i(Z|Y) = q_j(Z|Y)$. In this work, we shall adopt these insights to solve (1). In contrast with DA (even the multi-source) setting, we have the following challenges: **(i) restriction on data sharing and model sharing, (ii) learning simultaneously for all clients**. In classic multi-source DA, the goal is to train a classifier on all sources so that it can generalize well on the target task. In our Federated setting, every client simultaneously solves a multi-source DA problem viewing itself as the target. On the other hand, each client has a labelled dataset unlike in unsupervised DA, where the target has no labeled data and one might need to estimate label distribution ratios.

A.2 MMD:

Define a feature map $\phi : \mathcal{X} \to \mathcal{H}_k$ where $\mathcal{H}_k$ is a reproducing kernel Hilbert space (RKHS) endowed with a characteristic kernel $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}')\rangle_{\mathcal{H}_k}$. The Maximum Mean Discrepancy (MMD) between two distributions $\mathcal{D}, \mathcal{D}'$ is defined as $\text{MMD}(\mathcal{D}, \mathcal{D}') = \left\|\mathbb{E}_{x\sim\mathcal{D}}[\phi(x)] - \mathbb{E}_{x'\sim\mathcal{D}'}[\phi(x')]\right\|_{\mathcal{H}_k}$. The empirical estimate of the MMD given two datasets D, D' of size N each is given by $\text{MMD}(D, D') = \frac{1}{N}\sum_{x_i,x_j\in D} k(x_i, x_j) + \frac{1}{N}\sum_{x'_i,x'_j\in D'} k(x'_i, x'_j) - \frac{2}{N}\sum_{x_i\in D, x'_j\in D'} k(x_i, x'_j)$. When ϕ is an identity map, minimizing the MMD is equivalent to minimizing the first order moments and when ϕ is a gaussian kernel, then it is equivalent to minimizing all moments. In this work, we consider minimizing the distance between the first order moments (and the algorithm/analysis can be easily extended to a gaussian kernel case).

B Convergence Results

We wish to study the convergence of Fed-CMA and to understand if and when it is beneficial for the clients to participate in the FL setup. Arguably, the clients can benefit if the degree of heterogeneity between the clients' data is small and the number of combined data points at all clients is much

larger than the local data points, resulting in a better generalization. We wish to bound the error relative to the optimal error of the target task unlike existing FL generalization bounds. To derive these insights, we need an empirical minimizer of the combined error of the sources (Ben-David et al., 2010). Since we use neural networks, even in the simplest case of a 2 layer network, we end up with a non-convex objective where we are not guaranteed convergence to the global optima. To overcome this, we propose a simple trick to freeze the feature weights (after training for a sufficient number of iterations), and then continue performing FL in the latent space. This enables us to get the required bound and as we will see, this bound is dependent on the quality of the representations.

Let us denote $\mathbf{a} = (\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})$ and we can write the global objective as

$$\arg\min_{\mathbf{a}} \Phi(\mathbf{a}) := \arg \min_{\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \bar{\mathbf{e}}} \sum_{i=1}^{M} \Phi(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}). \tag{7}$$

Unless otherwise specified, $\|\cdot\|$ is $\|\cdot\|_2$ for vectors and $\|\cdot\|_{\mathrm{F}}$ for matrices. When we consider $\mathbf{a}$, we assume that all $\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}$ are vectorized and concatenated such that $\mathbf{a} \in \mathbb{R}^{Md_i+2(d_e+1)K}$. We place the following assumptions on the loss functions and embedding maps.

Assumption 1. *The local losses $\{L_i(\mathbf{w}_i)\}_{i=1}^N$ are β_l Lipschitz continuous, and β_s smooth in $\mathbf{w}_i$. Note that $\|\mathbf{w}_i\|^2 = \|\mathbf{u}_i, \mathbf{v}_i\|^2 = \|\mathbf{u}_i\| + \|\mathbf{v}_i\|$. The losses L_i are lower bounded uniformly by a scalar $L_{\inf}$.*

Assumption 2. *The function $g_i(\mathbf{u}_i, x)$ is β_e Lipschitz continuous and β_g smooth $\forall i \in [M]$ with respect to $\mathbf{u}_i$. Moreover, we assume that $g_i(\mathbf{u}_i, x) \in [0,1]^{d_e}$.*

Assumption 3. *The empirical gradient is an unbiased gradient of the population loss $\mathbb{E}[\nabla_{\mathbf{a}} \widehat{\Phi}(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})] = \nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})$. The variance of the stochastic gradient is bounded, i.e., $\mathbb{E}\left[\left\|\nabla_{\mathbf{a}} \widehat{\Phi}(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}) - \nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})\right\|^2\right] \leq G_1 \left\|\nabla_{\mathbf{a}} \Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})\right\|^2 + G_2^2$.*

Assumption 1 is commonly used in analyzing non-convex SGD. Assumption 2 implies that the output of $g_i(\mathbf{u}_i, x)$ is bounded and this is easily satisfied in deep neural networks where the activation function is a sigmoid function. While the boundedness of the activation function is not necessary, it makes some parts of the proof simpler. The more general ReLU activation can be used by making some changes to the loss function; more details are provided in the discussion after Lemma 1. Assumption 3 follows from assuming that the local dataset at each client i is drawn i.i.d from $\mathcal{D}_i$. For simplicity, we consider only one local update (4) before computing the global update (5). We now state the convergence theorem.

Theorem 3. *Let Assumptions 1, 2, and 3 hold and run the algorithm with T timesteps. If we chose a constant learning rate $\eta = \sqrt{\frac{2C_1}{TC_0G_2^2}}$ that satisfies $0 < \eta \leq \min\left\{\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \frac{1}{C_0(G_1+1)}\right\}$, where $\mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \mathbb{E}[\Phi(\mathbf{a}_{(T)})] \leq \mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \Phi^* \leq C_1$, and C_0 is a problem specific constant, then we have $\min_{t:1,\cdots,T} \mathbb{E}\left[\|\nabla_{\mathbf{a}} \Phi(\mathbf{a}_{(t)})\|^2\right] \leq \sqrt{\frac{2C_1C_0G_2^2}{T}}$.*

We defer the proof to Appendix. Note that here the algorithm converges to a first order stationary point since we are dealing with smooth and non-convex functions. We now restrict the training only to the latent space to obtain convergence to the global optimum which we will then use to prove the generalization result.

B.1 Continuing FL in the latent space

Let $\left(\{(\mathbf{u}_i^*, \mathbf{v}_i^*)\}_{i=1}^M, \bar{\mathbf{v}}^*, \mathbf{e}^*\right)$ be the output from solving (7). We now freeze the representation functions $\{\mathbf{u}_i^*\}_{i=1}^M$ for all the clients. For every dataset $D_i = \{x_j, y_j\}_{j=1}^{N_i}$, we define a projected dataset $\tilde{D}_i = \{(z_j, y_j) | z_j = g_i(\mathbf{u}_i^*, x_j)\ \forall (x_j, y_j) \in D_i\}$. We now define the local objective as $\arg\min F_i(\mathbf{v}) := \frac{1}{N_i} \sum_{(x,y) \in D_i; z = g_i(\mathbf{u}_i^*, x)} l(\mathbf{v}, z, y)$. The global objective is then defined as

$$\arg\min_{\mathbf{v}} F(\mathbf{v}) := \sum_{i=1}^{M} p_i F_i(\mathbf{v}). \tag{8}$$

To solve (8), we initialize with $\mathbf{v}_{(i,0)} = \bar{\mathbf{v}}^*$, follow the FedAvg algorithm (Li et al., 2019) and update the local weights $\mathbf{v}_{(i,t)}$ at $i-$th worker and the global weights $\mathbf{v}$ as

$$\mathbf{v}_{(i,t+\tau+1)} = \mathbf{v}_{(i,t+\tau)} - \eta_{(t+\tau)} \nabla \widehat{F}_i(\mathbf{v}_{(i,t+\tau)}, \xi_{(i,t+\tau)}); \qquad \mathbf{v}_{(t+E)} = \sum_{i=1}^{M} p_i \mathbf{v}_{(i,t+E)}, \tag{9}$$

where τ is the local iteration number, $\nabla \widehat{F}_i(\mathbf{v}_{(i,t+\tau)}, \xi_{(i,t+\tau)})$ is the stochastic gradient at time $t+\tau$ by sampling a local minibatch $\xi_{(i,t+\tau)}$ and $\eta_{(t+\tau)}$ is the learning rate. After every E local updates, the server aggregates the local weights and for simplicity, we assume full device participation in updating the weights. After the global update, the server assigns $\mathbf{v}_{(i,t+E)} = \mathbf{v}_{(t+E)}$ to every client. Before stating the convergence theorem, we make the following assumptions.

Assumption 4. *The local loss functions F_i are all $L-$smooth and μ strongly convex for all $i \in [M]$.*

Assumption 5. *The second moment of the stochastic gradients of the $i-th$ worker at time t is bounded as: $\mathbb{E}\left[\left\|\nabla \widehat{F}_i(\mathbf{v}_{(i,t)}, \xi_{(i,t)}) - \nabla F_i(\mathbf{v}_{(i,t)})\right\|^2\right] \leq \sigma_i^2$, for all $i \in [M]$.*

Assumption 6. *The expected squared norm of the stochastic gradients is uniformly bounded as $\mathbb{E}\left[\left\|\nabla \widehat{F}_i(\mathbf{v}_{(i,t)}, \xi_{(i,t)})\right\|^2\right] \leq G^2$ for all $i \in [M]$.*

The problem we solve in (8) is convex since the cross entropy loss is convex and we are only optimizing the final layer (classification) weights $\mathbf{v}$ (keeping the $\mathbf{u}_i^*$ fixed). Moreover, we can assume that each local objective uses an L2 regularization term, thus satisfying strong convexity. The Assumptions 4, 5, 6 are standard for analyzing FedAvg for non-iid scenarios (Li et al., 2019).

Definition 1 ((Li et al., 2019))**.** *Let F^*, F_i^* be the minimum values of F and F_i respectively. The degree of non-iid (heterogeneity) is defined as $\Gamma = F^* - \sum_{i=1}^{M} p_i F_i^*$.*

When the data $\tilde{D}_i$ are all i.i.d, then the degree of heterogeneity Γ goes to zero as the sample size increases. Here, the degree of heterogeneity between $\{\tilde{D}_i\}_{i=1}^M$ is determined by the degree of heterogeneity between $\{D_i\}_{i=1}^M$ and the learned $\{\mathbf{u}_i^*\}_{i=1}^M$. We now state the convergence theorem.

Theorem 4 (Theorem 1 (Li et al., 2019))**.** *Let Assumptions 4 to 6 hold and L, μ, σ_i, G be defined therein. Choose $\kappa = \frac{L}{\mu}$, $\gamma = \max\{8\kappa, E\}$ and the learning rate $\eta_{(t)} = \frac{2}{\mu(\gamma+t)}$. Then solving (9) with full client participation satisfies $\mathbb{E}[F(\mathbf{v}_{(T)})] - F^* \leq \frac{\kappa}{\gamma+T-1}\left(\frac{2B}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}[\|\bar{\mathbf{v}}^* - \mathbf{v}^*\|^2]\right)$, where $B = \sum_{i=1}^{M} p_i^2 \sigma_i^2 + 6L\Gamma + 8(E-1)^2 G^2$, $\mathbf{v}^*$ is the unique minimizer of (8).*

Theorem 4 states that applying FedAvg to the problem (8), the resultant iterate $\mathbf{v}_{(T)}$ convergences with a rate of $\mathcal{O}(\frac{1}{T})$. The faster rate compared to Theorem 3 is due to the simplification of the problem to the strongly convex case by freezing the $\{\mathbf{u}_i\}_{i=1}^M$. Moreover, unlike Theorem 3, where the algorithm converges to a local optima, here the convergence is to the global optimum of (8). However, (8) itself depends on the frozen representations at which the latent space training took place. The rate of convergence slows with higher values of Γ, thus implying that the frozen $\{\mathbf{u}_i^*\}$ from solving (7) also play a role in the convergence of $\mathbf{v}_{(T)}$. Also, observe that the convergence rate depends on $\|\bar{\mathbf{v}}^* - \mathbf{v}^*\|^2$. Suppose, the freezing is done when $\mathbf{u}_i$ are far away from minimizing (7), then $\bar{\mathbf{v}}^*$ is also far away from the optimal point and therefore it slows down convergence. Note that, while this dependence on $\mathbf{u}_i^*$ is not desirable from the convergence perspective, we want to highlight that we are dealing with model heterogeneity and this decoupling of $\mathbf{u}_i, \mathbf{v}_i$ is required to show a reasonable generalization result.

C Proof of Convergence Result

We now state some useful lemmas towards showing the convergence of the proposed algorithm. Unless otherwise specified, $\|\cdot\|$ is $\|\cdot\|_2$ for vectors and $\|\cdot\|_{\mathrm{F}}$ for matrices. When we consider $\mathbf{a}$, we assume that all $\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}$ are vectorized and concatenated such that $\mathbf{a} \in \mathbb{R}^{Md_i + 2(d_e+1)K}$.

Lemma 1. *When Assumptions 1,2 are satisfied, the function $\Phi(\mathbf{a})$ is C_0-smooth function wrt $\mathbf{a}$ with $C_0 = \left(\beta_s + \lambda_2 K \beta_e(\beta_e+1) + \lambda_2 K \beta_g \sqrt{d_e} + 4\lambda_1 + \sqrt{K}\lambda_2(1+\beta_e)\right)$.*

Proof. We will show that $\Phi(\mathbf{a})$ is smooth wrt $\mathbf{a}$ by showing that $\nabla_{\mathbf{a}}\Phi$ is Lipschitz, i.e. $\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}) - \nabla_{\mathbf{a}}\Phi(\mathbf{a}')\| \leq C_0\|\mathbf{a}-\mathbf{a}'\|$. Consider $\mathbf{a} = (\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e})$ and $\mathbf{a}' = \{\mathbf{w}_i'\}_{i=1}^M, \bar{\mathbf{v}}', \mathbf{e}'$. To show this, we need to show that $\nabla\Phi$ is Lipschitz with respect to $\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}$ and $\mathbf{e}$.

Firstly, observe that $\nabla_{e^k}\Phi(\{\mathbf{w}_i\}_{i=1}^M, \bar{\mathbf{v}}, \mathbf{e}) = \lambda_2 \sum_{i=1}^M \frac{p_i^k}{b_i^k} \sum_{x\in\mathcal{B}_i^k}(g_i(\mathbf{u}_i, x) - e^k)$. Clearly,

$$\begin{aligned}\|\nabla_{e^k}\Phi(\mathbf{a}) - \nabla_{e^k}\Phi(\mathbf{a}')\| &\leq \lambda_2\|\mathbf{e}-\mathbf{e}'\| + \sum_{i=1}^M \sum_{x\in\mathcal{B}_i^k} \frac{\lambda_2 p_i^k}{b_i^k}\|g_i(\mathbf{u}_i, x) - g_i(\mathbf{u}_i', x)\| \\ &\leq \lambda_2(\|\mathbf{e}-\mathbf{e}'\| + \beta_e\|\mathbf{u}_i - \mathbf{u}_i'\|) \leq \lambda_2(1+\beta_e)\|\mathbf{a}-\mathbf{a}'\|.\end{aligned}$$

where the second inequality follows from the Lipschitz property of g_i in Assumption 2, and since $\|e^k - e^{'k}\| \leq \|\mathbf{e}-\mathbf{e}'\| \leq \|\mathbf{a}-\mathbf{a}'\|$ and $\|\mathbf{u}_i - \mathbf{u}_i'\| \leq \|\mathbf{a}-\mathbf{a}'\|$. Therefore, we can write

$$\|\nabla_{\mathbf{e}}\Phi(\mathbf{a}) - \nabla_{\mathbf{e}}\Phi(\mathbf{a}')\|^2 = \sum_{k=1}^K \|\nabla_{e^k}\Phi(\mathbf{a}) - \nabla_{e^k}\Phi(\mathbf{a}')\|^2 \leq K\lambda_2^2(1+\beta_e)^2\|\mathbf{a}-\mathbf{a}'\|^2. \tag{10}$$

Therefore, Φ is smooth in $\mathbf{e}$ with parameter $\lambda_2(1+\beta_e)\sqrt{K}$. Similarly, consider $\nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}) = \sum_{i=1}^M p_i\lambda_1(\bar{\mathbf{v}} - \mathbf{v}_i)$. Therefore,

$$\|\nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}) - \nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}')\| \leq \lambda_1\|\bar{\mathbf{v}} - \bar{\mathbf{v}}'\| + \lambda_1\sum_{i=1}^M p_i\|\mathbf{v}_i - \mathbf{v}_i'\| \leq 2\lambda_1\|\mathbf{a}-\mathbf{a}'\|, \tag{11}$$

since $\|\bar{\mathbf{v}} - \bar{\mathbf{v}}'\| \leq \|\mathbf{a}-\mathbf{a}'\|$ and $\|\mathbf{v}_i - \mathbf{v}_i'\| \leq \|\mathbf{a}-\mathbf{a}'\|$. Therefore Φ is $2\lambda_1$ smooth in $\bar{\mathbf{v}}$.

Now, let us show that Φ is smooth in every $\mathbf{w}_i$. Clearly, $\nabla_{\mathbf{w}_i}\Phi(\mathbf{a}) = \nabla_{\mathbf{w}_i}\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e})$. Therefore, let us consider smoothness of Φ_i with respect to $\mathbf{w}_i$. Since $\mathbf{w}_i = (\mathbf{u}_i, \mathbf{v}_i)$, note that

$$\begin{aligned}\nabla_{\mathbf{w}_i}\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}_i, \mathbf{e}) &= p_i\nabla_{\mathbf{w}_i}L_i(\mathbf{w}_i) + \nabla_{\mathbf{w}_i}\Big(p_i\frac{\lambda_1}{2}\|\mathbf{v}_i - \bar{\mathbf{v}}\|^2 + \frac{\lambda_2}{2}\sum_{k=1}^K\sum_{x\in\mathcal{B}_i^k}\frac{p_i^k}{b_i^k}\|g_i(\mathbf{u}_i, x) - e^k\|^2\Big) \\ &= p_i\nabla_{\mathbf{w}_i}L_i(\mathbf{w}_i) + \begin{bmatrix}\nabla_{\mathbf{u}_i}\Big(\frac{\lambda_2}{2}\sum_{k=1}^K\sum_{x\in\mathcal{B}_i^k}\frac{p_i^k}{b_i^k}\|g_i(\mathbf{u}_i, x) - e^k\|^2\Big) \\ p_i\nabla_{\mathbf{v}_i}\big(\frac{\lambda_1}{2}\|\mathbf{v}_i - \bar{\mathbf{v}}\|^2\big)\end{bmatrix}.\end{aligned} \tag{12}$$

We know from the smoothness of L_i from Assumption 1 that $\|\nabla_{\mathbf{w}_i}L_i(\mathbf{w}_i) - \nabla_{\mathbf{w}_i}L_i(\mathbf{w}_i')\| \leq \beta_s\|\mathbf{w}_i - \mathbf{w}_i'\|$. Also, it is easy to note that $\|\nabla_{\mathbf{v}_i}\big(\frac{\lambda_1}{2}\|\mathbf{v}_i - \bar{\mathbf{v}}\|^2\big) - \nabla_{\mathbf{v}_i}\big(\frac{\lambda_1}{2}\|\mathbf{v}_i' - \bar{\mathbf{v}}'\|^2\big)\| \leq (\lambda_1\|\mathbf{v}_i - \mathbf{v}_i'\| + \|\bar{\mathbf{v}} - \bar{\mathbf{v}}'\|) \leq 2\lambda_1\|\mathbf{a}-\mathbf{a}'\|$.

Now consider the term

$$\nabla_{\mathbf{u}_i}\Big(\frac{\lambda_2}{2}\sum_{k=1}^K\sum_{x\in\mathcal{B}_i^k}\frac{p_i^k}{b_i^k}\|g_i(\mathbf{u}_i, x) - e^k\|^2\Big) = \sum_{k=1}^K\sum_{x\in\mathcal{B}_i^k}\frac{\lambda_2 p_i^k}{b_i^k}\nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i, x) - e^k).$$

We can write

$$\begin{aligned}&\Big\|\nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i, x) - e^k) - \nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i', x)^T(g_i(\mathbf{u}_i', x) - e^{'k})\Big\| \\ &\overset{(i)}{\leq} \Big\|\nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i, x) - e^k) - \nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i', x) - e^{'k})\Big\| \\ &\qquad + \Big\|\nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i, x)^T(g_i(\mathbf{u}_i', x) - e^{'k}) - \nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i', x)^T(g_i(\mathbf{u}_i', x) - e^{'k})\Big\| \\ &\overset{(ii)}{\leq} \|\nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i, x)\|\|(g_i(\mathbf{u}_i, x) - e^k) - (g_i(\mathbf{u}_i', x) - e^{'k})\| \\ &\qquad + \|\nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i, x)^T - \nabla_{\mathbf{u}_i}g_i(\mathbf{u}_i', x)\|\|(g_i(\mathbf{u}_i', x) - e^{'k})\| \\ &\overset{(iii)}{\leq} \beta_e(\beta_e\|\mathbf{u}_i - \mathbf{u}_i'\| + \|\mathbf{e}-\mathbf{e}'\|) + \beta_g\|\mathbf{u}_i - \mathbf{u}_i'\|\sqrt{d_e} \\ &\overset{(iv)}{\leq} \Big(\beta_e(\beta_e+1) + \beta_g\sqrt{d_e}\Big)\|\mathbf{a}-\mathbf{a}'\|.\end{aligned} \tag{13}$$

where (i) follows from adding and subtracting a term and applying triangle inequality, (ii) follows from submultiplicativity of the norm, (iii) follows from the smoothness and Lipschitz properties of g_i from Assumption 2 and that $g_i(\mathbf{u}_i, x), e^k \in [0,1]^{d_e}$, and (iv) follows because $\|\mathbf{u}_i - \mathbf{u}_i'\| \leq \|\mathbf{a} - \mathbf{a}'\|$. Substituting these smoothness results in (12), we get

$$\begin{aligned}
&\left\|\nabla_{\mathbf{w}_i}\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}) - \nabla_{\mathbf{w}_i}\Phi_i(\mathbf{w}_i', \bar{\mathbf{v}}', \mathbf{e}')\right\| \\
&\leq \beta_s\|\mathbf{a}_i - \mathbf{a}_i'\| + \lambda_2 K\Big(\beta_e(\beta_e+1) + \beta_g\sqrt{d_e}\Big)\|\mathbf{a} - \mathbf{a}'\| + 2\lambda_1\|\mathbf{a} - \mathbf{a}'\| \\
&\leq (\beta_s + \lambda_2 K\beta_e(\beta_e+1) + \lambda_2 K\beta_g\sqrt{d_e} + 2\lambda_1)\|\mathbf{a} - \mathbf{a}'\|.
\end{aligned} \tag{14}$$

Therefore, finally from (10), (11), (14), we have

$$\begin{aligned}
&\left\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}) - \nabla_{\mathbf{a}}\Phi(\mathbf{a}')\right\| \\
&\leq \sum_{i=1}^{M}\left\|\nabla_{\mathbf{w}_i}\Phi(\mathbf{a}) - \nabla_{\mathbf{w}_i}\Phi(\mathbf{a}')\right\| + \|\nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}) - \nabla_{\bar{\mathbf{v}}}\Phi(\mathbf{a}')\| + \|\nabla_{\mathbf{e}}\Phi(\mathbf{a}) - \nabla_{\mathbf{e}}\Phi(\mathbf{a}')\| \\
&\leq \Big((\beta_s + \lambda_2 K\beta_e(\beta_e+1) + \lambda_2 K\beta_g\sqrt{d_e} + 2\lambda_1) + 2\lambda_1 + \sqrt{K}\lambda_2(1+\beta_e)\Big)\|\mathbf{a} - \mathbf{a}'\|.
\end{aligned} \tag{15}$$

Hence, Φ is C_0-smooth in $\mathbf{a}$. □

Note that, we use boundedness of $g_i(\mathbf{u}_i, x), e^k$ in (13)(iii) to show that this term is smooth. However, it is not necessary to assume that $g_i(\mathbf{u}_i, x)$ is bounded. Instead, we can modify the loss as

$$\Phi_i(\mathbf{w}_i, \bar{\mathbf{v}}, \mathbf{e}) = L_i(\mathbf{w}_i) + \frac{\lambda_1}{2}p_i\left\|\mathbf{v}_i - \bar{\mathbf{v}}\right\|_2^2 + \frac{\lambda_2}{2}\sum_{k=1}^{K}\frac{p_i^k}{b_i^k}\sum_{x\in\mathcal{B}_i^k}\left\|\sigma\big(g_i(\mathbf{u}_i, x)\big) - e^k\right\|_2^2, \tag{16}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the element wise sigmoid function. The update equation in (5) then becomes

$$e_{(t+1)}^k = e_{(t)}^k(1 - \lambda_2\eta_{(t)}) + \lambda_2\eta_{(t)}\left(\sum_{i=1}^{M}\frac{p_i^k}{b_i^k}\sum_{x\in\mathcal{B}_i^k}\sigma\big(g_i(\mathbf{u}_{(i,t)}, x)\big)\right).$$

The smoothness result in (13) follows with a little more effort.

Lemma 2. *When Assumptions 1, 2, 3 are satisfied, we have*

$$\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})|\mathcal{F}_t\Big] \leq \Phi(\mathbf{a}_{(t)}) - \left(\eta_{(t)} - \frac{\eta_{(t)}^2 C_0(G_1+1)}{2}\right)\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + \frac{\eta_{(t)}^2 C_0 G_2^2}{2} \tag{17}$$

where $\mathcal{F}_t$ is a filtration accounting for the randomness (in sampling the mini-batches) until time t.

Proof. Using the smoothness property of Φ from Lemma 1, we can write

$$\Phi\big(\mathbf{a}_{(t+1)}\big) \leq \Phi\big(\mathbf{a}_{(t)}\big) + \nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})^T\big(\mathbf{a}_{(t+1)} - \mathbf{a}_{(t)}\big) + \frac{C_0}{2}\|\mathbf{a}_{(t+1)} - \mathbf{a}_{(t)}\|^2$$

Taking expectation with respect to randomness at time t, we get

$$\begin{aligned}
\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})|\mathcal{F}_t\Big] &\leq \Phi(\mathbf{a}_{(t)}) + \mathbb{E}\Big[\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})^T\big(\mathbf{a}_{(t+1)} - \mathbf{a}_{(t)}\big)|\mathcal{F}_t\Big] + \frac{C_0}{2}\mathbb{E}\Big[\|\mathbf{a}_{(t+1)} - \mathbf{a}_{(t)}\|^2|\mathcal{F}_t\Big] \\
\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})|\mathcal{F}_t\Big] &\leq \Phi(\mathbf{a}_{(t)}) - \eta_{(t)}\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})^T\mathbb{E}\Big[\nabla_{\mathbf{a}}\widehat{\Phi}(\mathbf{a}_{(t)})|\mathcal{F}_t\Big] + \frac{\eta_{(t)}^2 C_0}{2}\mathbb{E}\Big[\|\nabla_{\mathbf{a}}\widehat{\Phi}(\mathbf{a}_{(t)})\|^2|\mathcal{F}_t\Big] \\
\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})|\mathcal{F}_t\Big] &\leq \Phi(\mathbf{a}_{(t)}) - \eta_{(t)}\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + \frac{\eta_{(t)}^2 C_0}{2}\Big(\mathbb{E}\Big[\|\nabla_{\mathbf{a}}\widehat{\Phi}(\mathbf{a}_{(t)}) - \nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2|\mathcal{F}_t\Big] \\
&\qquad\qquad + \|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2\Big) \\
\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})|\mathcal{F}_t\Big] &\leq \Phi(\mathbf{a}_{(t)}) - \eta_{(t)}\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + \frac{\eta_{(t)}^2 C_0}{2}\Big((G_1+1)\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + G_2\Big) \\
\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})|\mathcal{F}_t\Big] &\leq \Phi(\mathbf{a}_{(t)}) - \left(\eta_{(t)} - \frac{\eta_{(t)}^2 C_0(G_1+1)}{2}\right)\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2 + \frac{\eta_{(t)}^2 C_0 G_2^2}{2}
\end{aligned} \tag{18}$$

where the second inequality follows from the update rule and the third inequality follows from Assumption 3. □

We now proceed to prove Theorem 3. We state the theorem here for completeness.

Theorem 5. *Let Assumptions 1, 2, and 3 hold and run the algorithm with T timesteps. If we chose a constant learning rate $\eta = \sqrt{\frac{2C_1}{TC_0G_2^2}}$ that satisfies $0 < \eta \leq \min\left\{\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \frac{1}{C_0(G_1+1)}\right\}$, where $\mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \mathbb{E}[\Phi(\mathbf{a}_{(T)})] \leq \mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \Phi^* \leq C_1$, and C_0 is as defined in Lemma 1, then we have $\min_{t:1,\cdots,T}\mathbb{E}\big[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2\big] \leq \sqrt{\frac{2C_1C_0G_2^2}{T}}$.*

Proof. Taking expectation on both sides of (18) from Lemma 2 and since $\frac{\eta C_0(G_1+1)}{2} \leq \frac{1}{2}$, we have

$$\mathbb{E}\Big[\Phi(\mathbf{a}_{(t+1)})\Big] \leq \mathbb{E}\Big[\Phi(\mathbf{a}_{(t)})\Big] - \frac{\eta}{2}\mathbb{E}\Big[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2\Big] + \frac{\eta_{(t)}^2 C_0 G_2^2}{2}.$$

By summation over $t = 1$ to T followed by telescoping the sum and rearranging, we can write

$$\sum_{t=1}^{T}\frac{\eta}{2}\mathbb{E}[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2] \leq \mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \mathbb{E}[\Phi(\mathbf{a}_{(T)})] + \sum_{t=1}^{T}\eta^2\Big(\frac{C_0G_2^2}{2}\Big).$$

Choosing a constant learning rate η and dividing both sides by T, we get

$$\sum_{t=1}^{T}\frac{\mathbb{E}[\|\nabla_{\mathbf{a}}\Phi(\mathbf{a}_{(t)})\|^2]}{T} \leq \frac{2(\mathbb{E}[\Phi(\mathbf{a}_{(1)})]) - \mathbb{E}[\Phi(\mathbf{a}_{(T)})]}{\eta T} + \eta C_0 G_2^2$$

Now, if $\mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \mathbb{E}[\Phi(\mathbf{a}_{(T)})] \leq \mathbb{E}[\Phi(\mathbf{a}_{(1)})] - \Phi^* \leq C_1$, we can choose $\eta = \sqrt{\frac{2C_1}{TC_0G_2^2}}$ which results in convergence to a first order stationary point with a rate of $\frac{1}{\sqrt{T}}$. □

D Generalization Result

D.1 Preliminaries

We shall first define the following quantities, state some usefull lemmas and then proceed to proving Theorem 2. The following definition is used to measure the divergence between two distributions. For this section, we abuse the notation slightly by writing $h(z)$ instead of writing $h(\mathbf{v}, z)$. When h, h' are discussed, it is assumed that two different $\mathbf{v}, \mathbf{v}'$ are being considered. To prove the main result, we use $\mathbf{v}, \mathbf{v}'$ to remain consistent with the main paper. We now provide some useful definitions.

Definition 2 ($\mathcal{H}$-divergence, Kifer et al. (2004))**.** *Let $\mathcal{H} = \{h : \mathcal{Z} \to \mathcal{Y}\}$ be the hypothesis class. Let $\mathcal{A}_\mathcal{H}$ be the set of subsets of $\mathcal{Z}$ that are the support of some hypothesis in $\mathcal{H}$, i.e., $\mathcal{A}_\mathcal{H} = \{B \subset \mathcal{Z} : \forall z \in B, h(z) = 1\}$. Given two distributions $\mathcal{D}, \mathcal{D}'$ on $\mathcal{Z}$, we define*

$$d_\mathcal{H}(\mathcal{D}, \mathcal{D}') = 2 \sup_{\mathcal{A} \in \mathcal{A}_\mathcal{H}} |\mathbb{P}_\mathcal{D}[\mathcal{A}] - \mathbb{P}_{\mathcal{D}'}[\mathcal{A}]| = 2 \sup_{h \in \mathcal{H}} |\mathbb{P}_\mathcal{D}[\mathbb{I}(h)] - \mathbb{P}_{\mathcal{D}'}[\mathbb{I}(h)]|.$$

where $\mathbb{I}$ is the indicator function.

From the above definition for $\mathcal{H}-$ divergence, we can similarly write the divergence between two induced distributions $\tilde{\mathcal{D}}_1(\mathbf{u}_1), \tilde{\mathcal{D}}_2(\mathbf{u}_2)$ induced by two different representation functions $\mathbf{u}_1, \mathbf{u}_2$ respectively as $d_\mathcal{H}(\tilde{\mathcal{D}}_1(\mathbf{u}_1), \tilde{\mathcal{D}}_2(\mathbf{u}_2)) = 2 \sup_{\mathcal{A} \in \mathcal{A}_\mathcal{H}} |\mathbb{P}_{\tilde{\mathcal{D}}_1(\mathbf{u}_1)}[\mathcal{A}] - \mathbb{P}_{\tilde{\mathcal{D}}_2(\mathbf{u}_2)}[\mathcal{A}]|$.

Definition 3 (Symmetric Hypothesis space)**.** *Given a hypothesis space $\mathcal{H}$, we define the symmetric hypothesis space $\mathcal{H}\Delta\mathcal{H}$ as follows*

$$g \in \mathcal{H}\Delta\mathcal{H} \iff g(z) = h(z) \oplus h'(x), \text{ for some} h, h' \in \mathcal{H}.$$

The indicator function of such g is the set of all $z \in \mathcal{Z}$ where h, h' disagree.

Lemma 3. *For $h, h' \in \mathcal{H}$, given two distributions $\mathcal{D}_1, \mathcal{D}_2$ and the respective representation functions $\mathbf{u}_1, \mathbf{u}_2$, we have $|\epsilon_{1,\mathbf{u}_1}(h, h') - \epsilon_{2,\mathbf{u}_2}(h, h')| \leq \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_1(\mathbf{u}_1), \tilde{\mathcal{D}}_2(\mathbf{u}_2)\big)$.*

Proof. Recall that $\epsilon_{1,\mathbf{u}_1}(h, h') = \mathbb{E}_{z \sim \tilde{\mathcal{D}}_1(\mathbf{u}_1)}\big[|h(z) - h'(z)|\big]$. Therefore,

$$\begin{aligned}
|\epsilon_{1,\mathbf{u}_1}(h, h') - \epsilon_{2,\mathbf{u}_2}(h, h')| &= \left| \mathbb{E}_{z \sim \tilde{\mathcal{D}}_1(\mathbf{u}_1)}\Big[|h(z) - h'(z)\Big] - \mathbb{E}_{z \sim \tilde{\mathcal{D}}_2(\mathbf{u}_2)}\Big[|h(z) - h'(z)\Big] \right| \\
&= \left| \mathbb{P}_{\tilde{\mathcal{D}}_1(\mathbf{u}_1)}\Big[\mathbb{I}_{\{z : h(z) \neq h'(z)\}}\Big] - \mathbb{P}_{\tilde{\mathcal{D}}_2(\mathbf{u}_2)}\Big[\mathbb{I}_{\{z : h(z) \neq h'(z)\}}\Big] \right| \\
&\leq \sup_{g \in \mathcal{H}\Delta\mathcal{H}} \left| \mathbb{P}_{\tilde{\mathcal{D}}_1(\mathbf{u}_1)}\Big[\mathbb{I}(g)\Big] - \mathbb{P}_{\tilde{\mathcal{D}}_2(\mathbf{u}_2)}\Big[\mathbb{I}(g)\Big] \right| \\
&= \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\tilde{\mathcal{D}}_1(\mathbf{u}_1), \tilde{\mathcal{D}}_2(\mathbf{u}_2))
\end{aligned}$$

The first equality follows from the definition and the second equality follows from the definition of $h \in \mathcal{H}$. The inequality follows from the definition of the symmetric difference hypothesis space and taking a supremum. □

Lemma 4 ((Ben-David et al., 2010))**.** *For each $j \in \{1, \cdots, M\}$, let D_j be a labeled sample of size $\beta_j N$ generated by drawing $\beta_j N$ points from $\mathcal{D}_j$ and labeling them according to c. For any fixed weight vector α, let $\hat{\epsilon}_\alpha(h)$ be the empirical α-weighted error of some fixed hypothesis h on this sample, and let $\epsilon_\alpha(h)$ be the true $\alpha-$weighted error. Then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$\mathbb{P}\Big[|\hat{\epsilon}_\alpha(h) - \epsilon_\alpha(h)| \geq \epsilon\Big] \leq 2 \exp\left(\frac{-2N\epsilon^2}{\sum_{j=1}^N \frac{\alpha_j^2}{\beta_j}} \right)$$

The following result provides a way to empirically estimate the $d_{\mathcal{H}\Delta\mathcal{H}}$ divergence based on the data samples.

Lemma 5 (Kifer et al. (2004),Ben-David et al. (2010))**.** *Let $\mathcal{H}$ be a hypothesis space on $\mathcal{X}$ with VC dimension $d_\mathcal{H}$. If $\mathcal{U}$ and $\mathcal{U}'$ are samples of size n from $\mathcal{D}$ and $\mathcal{D}'$ respectively. Then, for any $\delta \in (0, 1)$ with probability at least $1 - \delta$,*

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}, \mathcal{D}') \leq 2\Big(1 - \min_{h \in \mathcal{H}} \Big[\frac{1}{n} \sum_{x : h(x) = 0} \mathbb{I}[x \in \mathcal{U}] + \frac{1}{n} \sum_{x : h(x) = 1} \mathbb{I}[x \in \mathcal{U}']\Big]\Big) + 4\sqrt{\frac{d \log(2n) + \log(\frac{2}{\delta})}{n}}$$

We train a classifier to identify the distribution to which the data points belong. The divergence term is large, when the learnt function is able to classify properly between the distributions, and the divergence term is small when we are unable to learn a hypothesis that can distinguish between the distributions. (Long et al., 2015) provides a relation between the $d_{\mathcal{H}\Delta\mathcal{H}}$ divergence and two sample testing using MMD in Theorem 1 (eq 11). However, the bound appears vacuous, since the risk is bounded by 1 but there is a term with value 2 in eq 11.

D.2 Proof of main result

Theorem 1 is a special case of Theorem 2 where $\mathbf{u}_i = \mathbf{u},\ \forall i \in [M]$. Therefore, we only show the proof for the model heterogeneous case. We now state the Theorem 2 here for completeness.

Theorem 6. *Let $\widehat{\mathbf{v}} = \arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{\epsilon}_{\boldsymbol{\alpha}}(h)$ be the output of FedAvg on the latent space dataset and $\mathbf{v}_T^* = \arg\min_{\mathbf{v}\in\mathcal{V}} \epsilon_{T,\mathbf{u}_T}(\mathbf{v}, \tilde{c}_{\mathbf{u}_T})$ be the minimizer of the true target risk. Then for any $\delta > 0$, w.p. $> 1-\delta$, we have*

$$\epsilon_{T,\mathbf{u}_T}(\widehat{\mathbf{v}}) - \epsilon_{T,\mathbf{u}_T}(\mathbf{v}_T^*) \leq 4\sqrt{\sum_{i=1}^{M} \frac{\alpha_i^2}{p_i}}\sqrt{\frac{2d_{VC}\log(2(N+1)) + \log(\frac{8}{\delta})}{N}}\Big)$$
$$+ 2\Big(\lambda_\alpha + \frac{1}{2}\sum_{i=1}^{M} \alpha_i d_{\mathcal{H}\Delta\mathcal{H}}(\tilde{\mathcal{D}}_i(\mathbf{u}_i), \tilde{\mathcal{D}}_T(\mathbf{u}_T))\Big),$$

where $\lambda_\alpha = \sum_{i=1}^{M} \alpha_i \min\Big\{\mathbb{E}_{z\in\tilde{\mathcal{D}}_i(\mathbf{u}_i)}\big[|\tilde{c}_{\mathbf{u}_i}(z) - \tilde{c}_{\mathbf{u}_T}(z)|\big], \mathbb{E}_{z\in\tilde{\mathcal{D}}_T(\mathbf{u}_T)(z)}\big[|\tilde{c}_{\mathbf{u}_i}(z) - \tilde{c}_{\mathbf{u}_T}(z)|\big]\Big\}$.

Proof. The proof follows the procedure of Theorem 4 in Ben-David et al. (2010) with some changes. Note that, we have induced distributions $\mathcal{D}_i(\mathbf{u}_i)$ instead of the actual distributions $\mathcal{D}_i$. We write the result in terms of the induced distributions using Lemma 3 as follows. For every $i \in [M]$, we have

$$\epsilon_{T,\mathbf{u}_T}(\mathbf{v}, \tilde{c}_{\mathbf{u}_T}) \leq \epsilon_{i,\mathbf{u}_i}(\mathbf{v}, \tilde{c}_{\mathbf{u}_T}) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big)$$
$$\leq \epsilon_{i,\mathbf{u}_i}(h, \tilde{c}_{\mathbf{u}_i}) + \epsilon_{i,\mathbf{u}_i}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_T}) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big),$$

where the first inequality follows from Lemma 3 and second inequality follows from triangle inequality. Similarly, we can write,

$$\epsilon_{T,\mathbf{u}_T}(h, \tilde{c}_{\mathbf{u}_T}) \leq \epsilon_{i,\mathbf{u}_i}(h, \tilde{c}_{\mathbf{u}_i}) + \epsilon_{T,\mathbf{u}_T}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_T}) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big).$$

Therefore, combining both the equations, we have

$$\epsilon_{T,\mathbf{u}_T}(h) \leq \epsilon_{i,\mathbf{u}_i}(h) + \min\{\epsilon_{T,\mathbf{u}_T}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_T}), \epsilon_{i,\mathbf{u}_i}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_T})\} + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big). \quad (19)$$

This is similar to (6) (Zhao et al., 2019), but we have the induced distributions and the induced labeling functions with respect to separate representations $\{\mathbf{u}_i\}_{i=1}^M, \mathbf{u}_T$. This gives us the freedom to interpret the bounds in terms of the private representation functions of every client.

Now, we have

$$\big|\epsilon_{T,\mathbf{u}_T}(\mathbf{v}) - \epsilon_{\boldsymbol{\alpha}}(\mathbf{v})\big| = \big|\epsilon_{T,\mathbf{u}_T}(\mathbf{v}) - \sum_{i=1}^{M} \alpha_i \epsilon_{i,\mathbf{u}_i}(\mathbf{v})\big|$$
$$\leq \sum_{i=1}^{M} \alpha_i \big|\epsilon_{T,\mathbf{u}_T}(h) - \epsilon_{i,\mathbf{u}_i}(\mathbf{v})\big|,$$

and substituting (19), we get

$$\big|\epsilon_{T,\mathbf{u}_T}(\mathbf{v}) - \epsilon_{\boldsymbol{\alpha}}(\mathbf{v})\big| \leq \sum_{i=1}^{M} \alpha_i \min\{\epsilon_{T,\mathbf{u}_T}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_T}), \epsilon_{i,\mathbf{u}_i}(\tilde{c}_{\mathbf{u}_i}, \tilde{c}_{\mathbf{u}_T})\}$$
$$+ \frac{1}{2}\sum_{i=1}^{M} d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big) \quad (20)$$
$$= \lambda_\alpha + \frac{1}{2}\sum_{i=1}^{M} d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big).$$

Now, substituting $\widehat{\mathbf{v}} = \arg\min_{\mathbf{v}\in\mathcal{V}} \widehat{\epsilon}_{\boldsymbol{\alpha}}(\mathbf{v})$ in the above equation, we get

$$\epsilon_{T,\mathbf{u}_T}(\widehat{\mathbf{v}}) \leq \epsilon_{\boldsymbol{\alpha}}(\widehat{\mathbf{v}}) + \lambda_\alpha + \frac{1}{2}\sum_{i=1}^{M} d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big),$$

and now applying Lemma 4, we get for $\delta \in (0,1)$, with probability $> 1-\delta/2$,

$$\epsilon_{T,\mathbf{u}_T}(\widehat{\mathbf{v}}) \leq \widehat{\epsilon}_{\boldsymbol{\alpha}}(\widehat{\mathbf{v}}) + 2\sqrt{\sum_{j=1}^{M}\frac{\alpha_j^2}{p_j}\Big(\frac{2d_{VC}\log(2(N+1)) + \log(\frac{8}{\delta})}{N}\Big)} + \lambda_\alpha + \frac{1}{2}\sum_{i=1}^{M} d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big).$$

Since $\widehat{\mathbf{v}}$ is the empirical minimizer of $\widehat{\epsilon}_{\boldsymbol{\alpha}}$, we have $\widehat{\epsilon}_{\boldsymbol{\alpha}}(\widehat{\mathbf{v}}) \leq \widehat{\epsilon}_{\boldsymbol{\alpha}}(\mathbf{v}_T^*)$, then with probability $> 1-\delta/2$,

$$\epsilon_{T,\mathbf{u}_T}(\widehat{\mathbf{v}}) \leq \widehat{\epsilon}_{\boldsymbol{\alpha}}(\mathbf{v}_T^*) + 2\sqrt{\sum_{j=1}^{M}\frac{\alpha_j^2}{p_j}\Big(\frac{2d_{VC}\log(2(N+1)) + \log(\frac{8}{\delta})}{N}\Big)} + \lambda_\alpha + \frac{1}{2}\sum_{i=1}^{M} d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big).$$

Further, applying Lemma 4 again to $\widehat{\epsilon}_{\boldsymbol{\alpha}}(\mathbf{v}_T^*)$, with probability $> 1-\delta$,

$$\epsilon_{T,\mathbf{u}_T}(\widehat{\mathbf{v}}) \leq \epsilon_{\boldsymbol{\alpha}}(\mathbf{v}_T^*) + 4\sqrt{\sum_{j=1}^{M}\frac{\alpha_j^2}{p_j}\Big(\frac{2d_{VC}\log(2(N+1)) + \log(\frac{8}{\delta})}{N}\Big)} + \lambda_\alpha + \frac{1}{2}\sum_{i=1}^{M} d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big).$$

Finally, substituing $\mathbf{v}_T^*$ in (20), we get with probability $> 1-\delta$,

$$\epsilon_{T,\mathbf{u}_T}(\widehat{\mathbf{v}}) \leq \epsilon_{T,\mathbf{u}_T}(\mathbf{v}_T^*) + 4\sqrt{\sum_{j=1}^{M}\frac{\alpha_j^2}{p_j}\Big(\frac{2d_{VC}\log(2(N+1)) + \log(\frac{8}{\delta})}{N}\Big)} + 2\lambda_\alpha + \sum_{i=1}^{M} d_{\mathcal{H}\Delta\mathcal{H}}\big(\tilde{\mathcal{D}}_T(\mathbf{u}_T), \tilde{\mathcal{D}}_i(\mathbf{u}_i)\big).$$

The proof is complete. □

E More Numeric Results

In this section, we explain the experimental setup and provide additional analysis to understand the algorithm. Specifically, we aim to answer the following questions.

1. Is there an advantage of minimizing MMD on the conditional distributions instead of the marginal distributions? See Section E.4.
2. Is communicating both $\bar{\mathbf{v}}$ and $\mathbf{e}$ with the server important for the improved performance? What happens if only one of them is communicated? See Section E.5.
3. How does the dimension d_e of the embedding space affect the algorithm? See Section E.6.
4. How can new clients participate in the federated setup? See Section E.8.

Before we address these questions, we first provide additional results on CIFAR-10 dataset in Section E.1, describe the experimental setup and the choice of the hyperparameters in Section E.1 and E.3. We will revisit some of these questions again for the synthetic data experiments and provide a more thorough summary in Section E.7.

E.1 CIFAR-10 experiments

To simulate **data heterogeneity**, we follow Hsu et al. (2019) where a Dirichlet distribution $\text{Dir}_M(\alpha)$ is used to partition the CIFAR-10 dataset (Krizhevsky et al., 2009) among M clients for $M \in \{10, 20, 50, 100\}$. Larger values of α indicate a more homogeneous distribution (Fig 2). Each client's test set is comprised of the original test data points for all the classes in the client's training set. To simulate **model heterogeneity,** we consider the four networks: `CNN1`: `Conv2D(3,6,5)` − `Conv2D(6,16,5)` − `Lin(400,120)` − `Lin(120,84)` − `Lin(84,10)`, `CNN2`: `Conv2D(3,6,5)` − `Conv2D(6,16,5)` − `Lin(400,84)` − `Lin(84,10)`, `MLP1`: `Lin(924,512)` − `Lin(512,84)` − `Lin(84,10)`, `MLP2`: `Lin(924,256)` − `Lin(256,84)` − `Lin(84,10)`. In all the networks, we use `sigmoid` in the last but one layer to keep the embeddings bounded and `ReLU` activation for all other layers. We consider two setups: (i) **Setup I** clients use either `CNN1` or `CNN2` and (ii) **Setup II** clients use any of the four networks at random. For each setup, we run Fed-CMA and local training in which the clients do not communicate. We compute the average of the local classification accuracy of all the clients (details in Appendix). We repeat the experiments for three seeds and present the summary in Table 3. We can observe that for different levels of data heterogeneity, Fed-CMA outperforms local training. The improvement achieved by Fed-CMA in Setup II is less due to higher model heterogeneity since the capacity of the function class of CNNs and MLPs are different and therefore the alignment of distributions in the latent space shared between these networks is a tougher task than when the models consisted only of CNNs. In Figure 5a, we plot the histogram of improvement in classification accuracy of every client (compared to local training) for Setup I with 100 clients and $\alpha = 0.1$. We observe that almost every client benefits from participating in the FL setup as compared to training locally. **Number of clients:** In Figure 5b, we measure the norm of the updates in (5) to understand how fast a consensus is reached on $\mathbf{e}$ and $\bar{\mathbf{v}}$ and we observe that as the number of clients increases, the norm of the updates converges faster and to a smaller value showing the advantage of having more clients in the setup. **Communication:** In KD based approaches (such as KT-pFL (Zhang et al., 2021) that communicate logits), the number of parameters communicated per client per communication round is $K \times |P|$ where P is the public dataset (KT-pFL uses $|P| = 3000$); for weight aggregation it is $\approx dim(\mathcal{H}_i)$; and for Fed-CMA it is $2K \times d_e$. In general, d_e is much smaller than $dim(\mathcal{H}_i)$ and is comparable or smaller than $|P|$, hence very little communication overhead for Fed-CMA . We illustrate this with the example of `MLP1` in Table 4.

E.2 Experimental Setup

We split the CIFAR-10 dataset as discussed earlier. While allotting the train dataset to each user, we allot 90% of the data as train data and 10% of the data as validation data randomly uniformly. Note that the train data and validation data have the same distribution which is different than the test distribution (since in the test distribution all labels have equal number of data points).

		Setup I		Setup II	
M	α	Local	Fed-CMA	Local	Fed-CMA
10	0.05	51.9 (0.6)	**54.1** (0.8)	39.3 (12.6)	**40.3** (13.0)
10	0.10	50.3 (0.2)	**52.9** (0.5)	38.3 (12.6)	**39.1** (13.3)
20	0.05	50.5 (0.1)	**53.9** (0.9)	47.8 (0.9)	**48.7** (1.1)
20	0.10	40.3 (1.9)	**45.7** (1.9)	36.5 (1.2)	**39.6** (0.9)
50	0.05	56.0 (0.4)	**59.8** (0.2)	52.8 (0.7)	**55.6** (0.8)
50	0.10	42.1 (0.3)	**47.5** (0.4)	38.0 (1.0)	**40.9** (0.9)
100	0.05	58.8 (0.4)	**63.1** (0.3)	56.7 (0.3)	**59.4** (0.4)
100	0.10	43.3 (0.1)	**48.0** (0.1)	41.0 (0.4)	**44.3** (0.5)

Table 3: Summary of the results on CIFAR-10 dataset. Setup I: the clients models $\in$ {`LeNet1`, `LeNet2`}. Setup II: the clients models $\in$ {`LeNet1`, `LeNet2`, `TwoLayer1`, `TwoLayer2`}

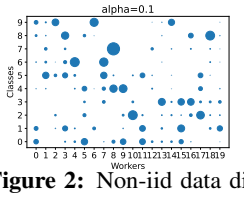


Figure 2: Non-iid data distribution of CIFAR-10 for 20 workers, $\alpha = 0.1$.

Comm Var	#Parameters
$\mathbf{w}_i$	5.6×10^5
Logits	3×10^4
$\bar{\mathbf{v}}, \mathbf{e}$	$\mathbf{1.69 \times 10^3}$

Table 4: Communication complexity

To choose the hyperparameters, we selected the setup ($M = 20, \alpha = 0.1, \texttt{seed} = 0$) and used the model heterogeneous Setup I. We can this as the `hyperparam` setup. We fixed the batch size for the experiments to be 256. We chose the learning rate as 0.01 by performing a grid search over $[0.001, 0.01, 0.1]$ during Local only training and selected the value with the best validation accuracy. These hyperparameters are then fixed for all different setups (varying M, α, `seed` and for Setup I, II). We chose $\mathcal{B}_i^k$ at the beginning by randomly sampling a mini-batch of size 256 from the training dataset of each worker. Then we split the chosen minibatch for every class k. We chose the values λ_1, λ_2 through hyperparameter selection by training Fed-CMA for the `hyperparam` setup. We performed grid search over the values $\lambda_1 \in [0.5, 1, 5]$ and $\lambda_2 \in [0.5, 1, 5]$. We chose a different learning rate for the user model parameters ($\eta = 0.01$) and the global variables $\bar{\mathbf{v}}, \mathbf{e}$. The learning rate of λ_1, denoted by η_{λ_1} is fixed to be 0.2 and the learning rate of λ_2, denoted by η_{λ_2} is fixed to be 0.18. These values were chosen such that $\eta_{\lambda_1}\lambda_1 \in [0, 1]$ and $\eta_{\lambda_2}\lambda_2 \in [0, 1]$.. The values for λ_1 and λ_2 are selected such that the average validation accuracy across all the clients is higher (good classification accuracy) and simultaneously $\|\mathbf{e}_{t+1} - \mathbf{e}_t\|, \|\bar{\mathbf{v}}_{t+1} - \bar{\mathbf{v}}_t\|$ are small (reach consensus on $\mathbf{e}, \bar{\mathbf{v}}$). The selected values $\lambda_1 = 5.0$, $\lambda_2 = 1.0$ are used throughout all the CIFAR-10 experiments in this paper.

In the experiments, we assume full user availability (i.e., every user participates at every global update), however each user runs a different number of local updates due to the difference in the size of the datasets. This simulates user heterogeneity in terms of the number of local updates at every global update. In the implementation, instead of weighing the local losses L_i with p_i, we give an equal weightage of $\frac{1}{M}$ to avoid changing the hyperparameters for every different choice of $M, \alpha, \texttt{seed}$. The pseudo code for the algorithm is provided in Alg 2. We implemented Fed-CMA using Pytorch and can be easily used with other existing algorithms. We used an Nvidia Quadro RTX 6000 gpu for the simulations, and the GPU RAM required to run any experiment is less than 2GB and can be run on even smaller GPUs. We simulate the distributed setup regarding dataset availability, but run the user updates sequentially so that the experiments can finish on a single GPU without requiring a cluster (since we run for different numbers of users upto 100). We include the data splits and the code in the supplementary material submission.

Please note that, we use this experimental setup and use the CIFAR-10 dataset in all the experiments, unless otherwise specified. For all other ablations in the next sections, we use the setting with $M = 20, \alpha = 0.1, \texttt{seed} = 0$, and Setup I model heterogeneity.

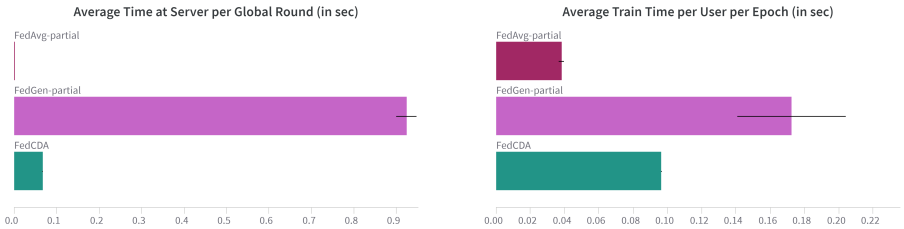
E.3 MNIST datasets

It is difficult to compare the performance of Fed-CMA with Fed-ET (Cho et al., 2022), FedDF (Lin et al., 2020) and KT-pFL(Zhang et al., 2021) since they utilize additional information in the form of a public dataset that is used for knowledge distillation. Moreover, some of these works may involve sharing the parameters with the server. Therefore, we do not provide a comparison with these works.

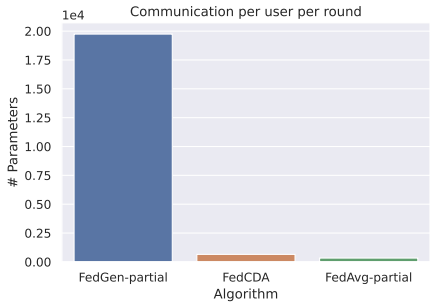
Without using a public dataset, FedGen trains a generator at the server using knowledge distillation. This generator is shared with the clients which use the generated data as an augmented data to improve their training. While the main results in Zhu et al. (2021) require sharing the model weights with the server (the server aggregates the weights), FedGen also accommodates limited parameter sharing (such as sharing only the classification layer weights $\mathbf{v}_i$). Therefore, in this section, we compare Fed-CMA with FedGen under the limited parameter setting.

For this experiment, we use the MNIST dataset and follow Zhu et al. (2021) to split the data in a non-iid way among 20 users using the Dirichlet allocation method for varying values of $\alpha \in [0.02, 0.03, 0.05, 0.1]$. The rest of the setup is similar to Zhu et al. (2021). We follow the same architecture used in FedGen here: `Conv2d(6,3,2)` $-$ `BatchNorm2d()` $-$ `ReLU` $-$ `Conv2d(16,3,2)` $-$ `BatchNorm2d()` $-$ `ReLU` $-$ `Flatten` $-$ `Linear(784,32)` $-$ `Linear(32,10)`. Here the latent space dimension is 32. Note that all the users have the same architecture in this experiment (following FedGen) but the users do not share the feature extractor weights and can only share the classification layer (i.e.,`Linear(32,10)`). We use the same setup for FedGen and Fed-CMA , and re-run the FedGen code available at https://github.com/zhuangdizhu/FedGen.

We denote the algorithm as FedGen-partial in the limited parameter sharing setting. Additionally, we also make a comparison with FedAvg in this setting (denoted by FedAvg-partial) and with local training. We run every algorithm for 400 global iterations and three random seeds. We present the



(a) Local Training **(b)** Fed ($\bar{\mathbf{v}}$)



(c) Communication Complexity

Figure 3: Comparison of runtimes and communication complexity of Fed-CMA , FedAvg-partial and FedGen-partial.

results of our experiments in Table 2. We can observe that, for smaller values of α (when the datasets are more heterogeneous), Fed-CMA outperforms FedGen-partial by upto 5%.

We also provide a comparison of the runtimes and communication complexity for the algorithms in Fig 3. FedGen-partial takes the most time at both the server and user, because it involves training a generator. Fed-CMA has only a small runtime overhead as compared to FedAvg since we compute the class conditional mean embeddings and update the global variables. FedGen-partial needs to communicate the generator between the users and the server, whereas in Fed-CMA we only need to communicate the final layer weights and the mean embeddings resulting in an improved communication complexity.

E.4 Conditional Distribution Alignment vs Marginal Distribution Alignment

In this section, we shall examine the importance of performing conditional distribution alignment as compared to just aligning the marginal distributions of the representations. For this experiment, we consider the setting with 20 users, $\alpha = 0.1$, and use the Setup I model heterogeneity. The rest of the experiment setup is the same as described in Section E.1, and we run only for one seed. From Table 5, we can observe that minimizing MMD on marginal distributions does not improve the performance, whereas minimizing MMD on conditional distributions provides a boost in performance.

	#Clients	α	Average Accuracy
Conditional (Fed-CMA)	20	0.1	**47.16**
Marginal	20	0.1	39.0
Local	20	0.1	41.6

Table 5: Comparison of conditional distribution alignment and marginal alignment

E.5 Importance of $\bar{\mathbf{v}}$ and $\mathbf{e}$

In this Section, we perform an ablation on sharing $\bar{\mathbf{v}}$ and $\mathbf{e}$ in Fed-CMA . We follow the same experiment setup described in Section E.4. In the algorithm where $\bar{\mathbf{v}}$ is not shared (only $\mathbf{e}$ is shared),

we only minimize the $\frac{\lambda_2}{2}\sum_{k=1}^{K}\frac{p_i^k}{b_i^k}\sum_{x\in\mathcal{B}_i^k}\left\|g_i(\mathbf{u}_i,x)-e^k\right\|_2^2$ along with L_i at each client. Note that $\bar{\mathbf{v}}$ is not updated at the global level since it does not play any role. We denote this algorithm by Fed-CMA ($\mathbf{e}$). Similarly, in the algorithm where $\mathbf{e}$ is not shared (we only share $\bar{\mathbf{v}}$), we only minimize $\lambda_1\|\bar{\mathbf{v}}-\mathbf{v}_i\|$ at each client. We denote this algorithm Fed($\bar{\mathbf{v}}$) since we are only sharing the final classification layer weights. We compare these two algorithms with Fed-CMA and Local training and present the results in Table 6. We can observe that these two algorithms perform poorly compared to Local training only. This is possible since, in Fed ($\bar{\mathbf{v}}$), the latent space distributions of all the clients are not homogenized/aligned, therefore learning a classifier for these heterogeneous distributions does not result in a good classifier and the clients are better of not participating in the FL algorithm. In the case of Fed-CMA ($\mathbf{e}$), the latent space distributions are aligned, however the final classification layer weights are not shared, therefore the clients do not have a chance to improve their decision boundaries by using the classification weights of the other clients. Note that in this case, the performance is slightly affected compared to Local only training, and this may be avoided by a better choice of the hyperparameters.

	#Clients	α	Average Accuracy
Fed-CMA	20	0.1	**47.16**
Fed-CMA ($\mathbf{e}$)	20	0.1	39.56
Fed ($\bar{\mathbf{v}}$)	20	0.1	35.03
Local	20	0.1	41.6

Table 6: Ablation on sharing only $\bar{\mathbf{v}}$ and only $\mathbf{e}$.

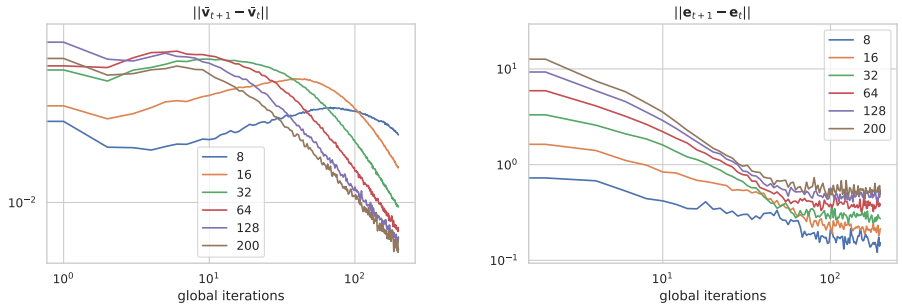
E.6 Dimension of the embedding space

The latent space plays an important role in aligning the conditional distributions and subsequently learning the classification weights. Therefore, it is important to understand what impact the latent space has on the algorithm, especially since we are not sharing the feature extractor weights and the data is heterogeneous at all clients. We vary the dimension d_e of the latent space and evaluate Fed-CMA and Local training and present the results in Table 7.

We also plot the norms of the updates to the global variables for different values of the latent space dimension d_e, i.e., we plot $\|\bar{\mathbf{v}}_{t+1}-\bar{\mathbf{v}}_t\|$ and $\|\mathbf{e}_{t+1}-\mathbf{e}_t\|$. We observe from Fig 4, $\bar{\mathbf{v}}$ and $\mathbf{e}$ converge differently for different values of d_e. **Norm of the classifier update $\bar{\mathbf{v}}$**: For higher values of d_e, the convergence is faster for $\bar{\mathbf{v}}$. In other words, a richer space enables a faster convergence of the classification weights. **Norm of the mean embeddings update $\mathbf{e}$:** For higher values of d_e, the convergence is slower for $\mathbf{e}$. In other words, it is easier to obtain a consensus on the mean embeddings when the latent space is small. While a small latent space (small d_e) makes it easier to push the mean embeddings closer, it may not lead to a good classification boundary since the complexity of the hypothesis space is reduced, therefore the convergence of the classification weights is slowed down. On the other hand, for a relatively large d_e, the complexity of the hypothesis space is increased and the quality of the decision boundaries increase making the convergence of $\bar{\mathbf{v}}$ faster. However, since the latent space is now relatively larger, it becomes harder to bring the mean embeddings close to each other, slowing down the convergence of $\mathbf{e}$. This tradeoff in the convergence of the global variables shows us that choosing the latent space dimension is very important to the success of the algorithm. Note that, the performance of Fed-CMA is relatively stable across all the different values of d_e chosen as compared to that of Local training only.

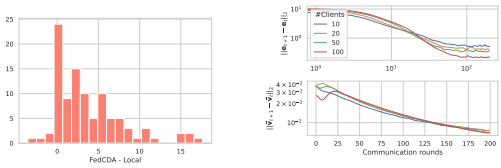
Algorithm	8	16	32	64	84	128	200
Fed-CMA	43.7	46.21	47.18	46.72	47.17	45.29	45.57
Local	36.69	37.72	40.37	41.22	41.61	41.93	41.84

Table 7: Varying the dimension d_e of embedding space



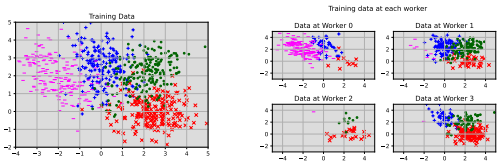
(a) **(b)**

Figure 4: Norm of the updates to $\bar{\mathbf{v}}$ and $\mathbf{e}$ for different values of the dimension of the latent space d_e.



(a) **(b)**

Figure 5: (a) Histogram of per worker improvements for Setup I, (b) Norm of the global updates



(a) **(b)**

Figure 6: (a) Synthetic data at all the workers, (b) Data at every worker.

E.7 Synthetic Data Experiments

We solve a 4-class classification problem, where we generate a synthetic dataset in $\mathbb{R}^2$ from a mixture of four Gaussian distributions and distribute them in a non-iid way among 4 clients following $\text{Dir}_4(0.1)$ (see Fig 6b, details are deferred to Appendix). We use `Linear(2,2)` $\rightarrow$ `Linear(2,2)` $\rightarrow$ `Linear(2,4)` at every client for simplicity. After training, we plot the decision boundaries in the original and latent spaces (see Fig 7a,7b). We see that the distributions in latent space and latent space decision boundaries are aligned for Fed-CMA resulting in much better decision boundary in the original space, even for workers with scarce data. We empirically approximate $d_{\mathcal{H}\Delta\mathcal{H}}$ by training a classifier to discriminate between the latent space data of the four clients (motivated from (Ben-David et al., 2010)). As we observe from Fig 9, as the training progresses the accuracy reaches that of a random classifier which means the latent space data among the workers are more aligned and it gets difficult to tell from which client the data came from.

We generated the data in Fig 6a as follows. We first generated four 2D Gaussian distributions with $\mu_1 = (0, 2.5), \Sigma_1 = [(0.5, 0), (0, 1)]$, $\mu_2 = (2.5, 0), \Sigma_2 = [(1, 0), (0, 0.5)]$, $\mu_3 = (-2, 2), \Sigma_3 = [(1, -0.5), (-0.5, 1)]$ and $\mu_4 = (2, 2), \Sigma_4 = [(0.5, 1), (1, 0.5)]$. We consider each Gaussian distribution to be representing a class in a 4 class classification problem. The four classes are represented using the symbols [`+`,`x`,`-`,`.`]. The data is then split across the four workers using Dirichlet allocation as shown in Fig 6b. Note that, not all workers have the data of all four labels. In some extreme cases, some workers may posses very few data points of a particular class (see class - for Workers 2 & 3).

We train the algorithms for 3000 epochs. For every epoch, each worker has a different number of iterations since the number of data points at every worker is different. At the end of the epoch, the workers communicate with the server. We chose a learning rate of 0.1 and batch size of 64 for the synthetic experiments. We chose $\mathcal{B}_i^k$ at the beginning by randomly sampling a mini-batch of size 256 from the training dataset of each worker. Then we split the chosen minibatch for every class k. We chose $\lambda_1 = 0.01, \lambda_2 = 1.0$ for Fed-CMA . The learning rates for $\bar{\mathbf{v}}, \mathbf{e}$ are chosen separately as $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}$ respectively. We also train Fed-CMA ($\mathbf{e}$) and Fed ($\bar{\mathbf{v}}$) similar to Section E.5 and present the results in Table 8.

Similar to the observations in Section E.5, we can see that sharing both $\mathbf{e}$ and $\bar{\mathbf{v}}$ is important to get a significant boost in performance. In this case however, Fed ($\bar{\mathbf{v}}$) and Fed-CMA ($\mathbf{e}$) perform better than Local training, but significantly worse than Fed-CMA . In Figures 10 and 11, we illustrate the

Figure 7

(a) Local Training **(b)** Fed-CMA

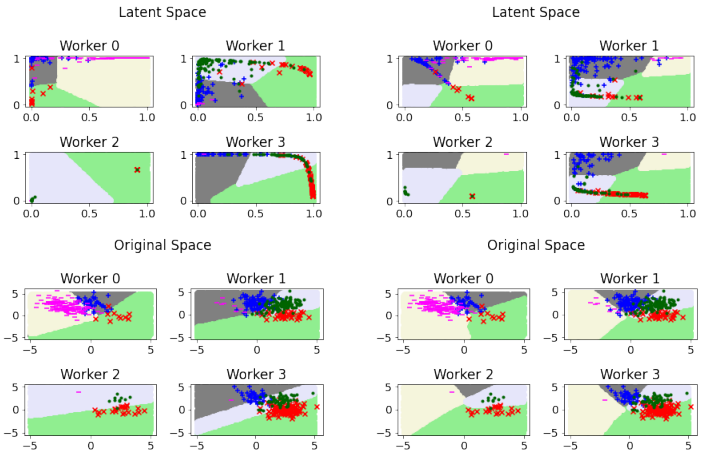
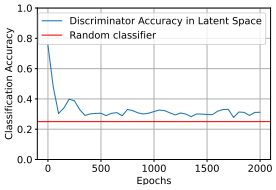


Figure 8: Illustration of the latent space distributions and decision boundaries for the synthetic dataset.

Figure 9: Accuracy on synthetic dataset (top). Discriminator accuracy: smaller the better (bottom).

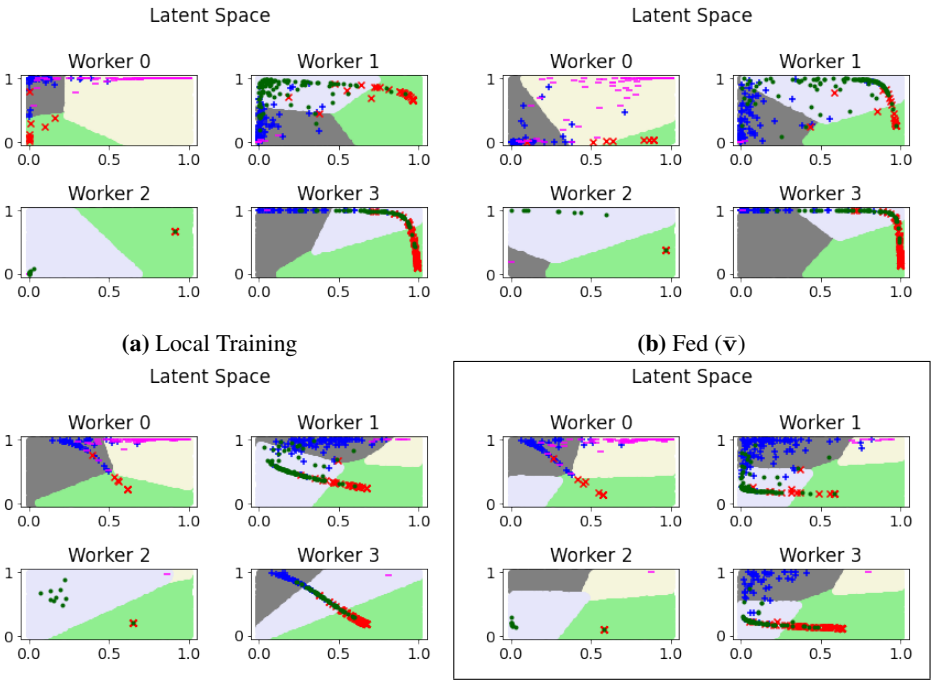
	Accuracy
Local	51.4±7.8
Fed-CMA	**59.1±6.1**



Num Workers	Local	Fed ($\bar{\mathbf{v}}$)	Fed-CMA ($\mathbf{e}$)	Fed-CMA
4	53.15 (5.97)	52.65 (5.39)	55.58 (4.95)	**58.44** (7.6)

Table 8: Summary of Synthetic data experiments

decision boundaries learnt by the respective clients. From Fig 10a and 10b, we can see that the latent space distributions of the clients are not aligned properly. In Fig 10c, the latent space distributions are aligned, however, since $\bar{\mathbf{v}}$ is not shared and the decision boundaries of every client are different and are overfit to their local training data. In Fig 10d, the decision boundaries also align well along with the latent space distributions, thus resulting in a much better overall decision boundary in the input space (see Fig 11d). Observe that, for workers 2 & 3, even with very little data for the - class, the corresponding decision boundary is well formed and aligns with that of worker 0 who has an abundance of - class. Note that, for Fed ($\bar{\mathbf{v}}$), despite the latent space decision boundaries aligning roughly, the input space decision boundaries are poorly formed since the latent space distributions are not aligned. We also plot the mean embeddings in the latent space for all the clients and all classes in Figure 12. We can observe that, for Fed-CMA and Fed-CMA ($\mathbf{e}$) the mean embeddings for all clients form a nice cluster for every class, and are well separated from that of the other classes. For Local training and Fed ($\bar{\mathbf{v}}$), the mean embeddings are scattered around haphazardly.



(a) Local Training **(b)** Fed ($\bar{\mathbf{v}}$)

(c) Fed-CMA ($\mathbf{e}$) **(d)** Fed-CMA

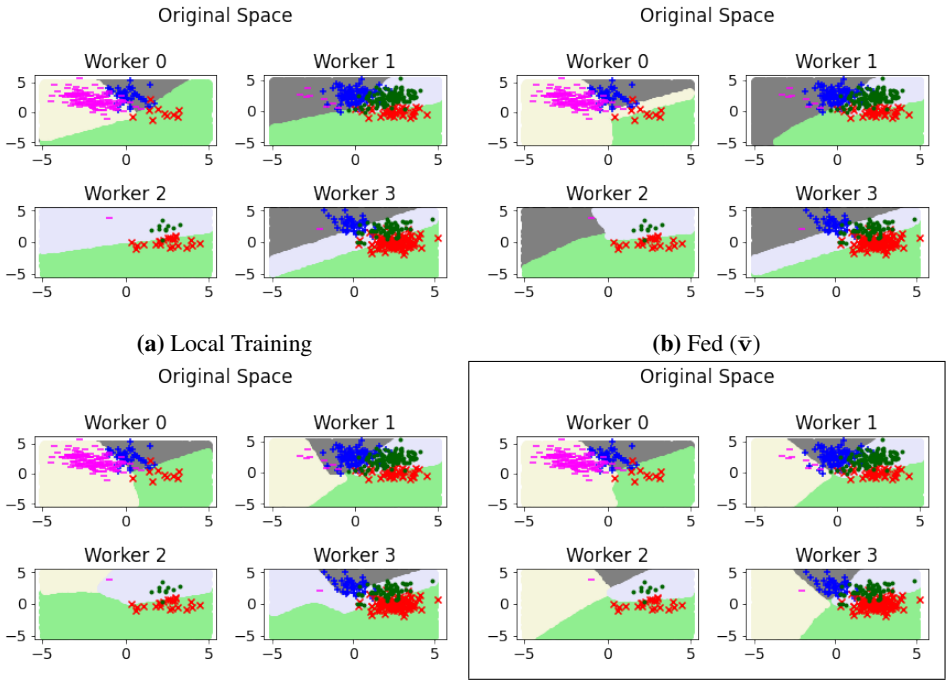
Figure 10: Visualizing decision boundaries in latent space

E.8 Joining New Clients

The proposed method is scalable. It is straight forward for a new client to enter the federation. For every new client that is added, the server first communicates the global mean embeddings and the global final layer weights to the new client. The client then trains locally using the proposed algorithm until the local mean embeddings are ϵ away from the global mean embeddings (where ϵ is determined by the server based on the global training statistics). After this point, the client is free to participate in the federation. For details, see Algorithm 2.

Algorithm 2 Fed-CMA with new clients

Time steps T elapsed with existing clients $[M]$.
New Client $M+1$ joins the federation.
Given: Mean embeddings $\{e^k_{(T)}\}_{k=1}^K$ and final classification layer weights $\bar{\mathbf{v}}_{(T)}$
Server: Broadcast $\{e^k_{(T)}\}_{k=1}^K$ and $\bar{\mathbf{v}}_{(T)}$ to client $M+1$. Send threshold parameters ϵ_1 and ϵ_2.
At new client $M+1$**:**
for $t = 1, \cdots, T$ **do**
    Pick random minibatch $\xi_{(t)}$ from D_{M+1} and compute $\nabla\widehat{\Phi}_{M+1}$.
    Update $\mathbf{w}_{(M+1,t)}$ according to (4)
    Compute $\{e^k_{(M+1,t)}\}_{k=1}^K$ using $e^k_{M+1} = \frac{1}{b^k_{M+1}} \sum_{x \in \mathcal{B}^k_{M+1}} g_{M+1}(\mathbf{u}_{(M+1,t)}, x)$ for $k \in [K]$.
    Continue until: $\sum_{k=1}^K \|e^k_{(M+1,t)} - e^k_{(T)}\|_2 \leq \epsilon_1$ and $\|\mathbf{v}_{(M+1,t)} - \bar{\mathbf{v}}_{(T)}\|_2 \leq \epsilon_2$.
end for
Join Federation and share with server: mean embeddings $\{e^k_{(M+1,t)}\}_{k=1}^K$ and classification layer weights $\mathbf{v}_{(M+1,t)}$.



(a) Local Training **(b)** Fed ($\bar{\mathbf{v}}$)

(c) Fed-CMA ($\mathbf{e}$) **(d)** Fed-CMA

Figure 11: Visualizing decision boundaries in input space

Num Workers	Local	FedCDA
20	32.1 (1.7)	37.5 (0.9)
50	18.0 (1.5)	26.9 (1.5)
100	25.5 (0.1)	34.8 (0.2)

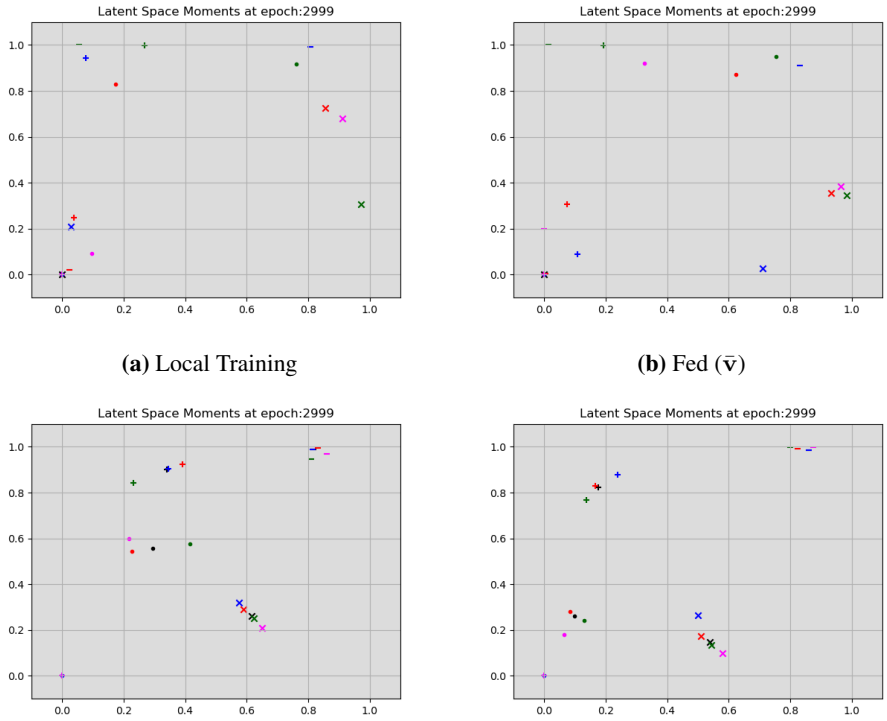
Table 9: Adding New Clients

E.9 Ablations: Data Heterogeneity only or Model Heterogeneity only

In this section, we present results for CIFAR10 dataset when there is (i) data heterogeneity and model homogeneity (we still share only the last layer weights), (ii) data homogeneity and model heterogeneity.

α	Local	FedAvg	FedGen	FedCDA
0.02	76.28 (0.39)	81.44 (12.36)	**93.23** (0.11)	92.03 (1.12)
0.03	59.89 (0.26)	81.27 (6.46)	93.23 (2.13)	**94.4** (0.1)
0.05	61.36 (0.71)	87.86 (3.71)	**94.4** (0.34)	93.91 (0.11)
0.10	62.27 (0.48)	89.7 (3.77)	**95.69** (0.1)	94.64 (0.17)

Table 10: Comparison on MNIST dataset with model homogeneity and full parameter sharing



(a) Local Training

(b) Fed ($\bar{\mathbf{v}}$)

(c) Fed-CMA ($\mathbf{e}$)

(d) Fed-CMA

Figure 12: Visualizing mean embeddings $\mathbf{e}$ in latent space

Num Workers	alpha	Local	FedCDA
10	0.05	50.4	**53.9**
10	0.10	50.4	**53.1**
20	0.05	50.8	**54.5**
20	0.10	41.1	**42.0**
50	0.05	55.4	**59.2**
50	0.10	41.7	**47.0**
100	0.05	58.0	**63.3**
100	0.10	42.2	**47.3**

Table 11: Data Heterogeneity and Model Homogeneity

Num Workers	Local	FedCDA
10	38.5	**43.5**
20	32.2	**36.8**
50	25.7	**28.7**
100	21.2	**23.6**

Table 12: Comparison on CIFAR10 with i.i.d data and model heterogeneity