Architecture Agnostic Federated Learning for Neural Networks

Disha Makhija [1] **Xing Han** [1] **Nhat Ho** [1] **Joydeep Ghosh** [1]

Abstract

With growing concerns regarding data privacy and rapid increase in data volume, Federated Learning (FL) has become an important learning paradigm. However, jointly learning a deep neural network model in a FL setting proves to be a non-trivial task because of the complexities associated with the neural networks, such as varied architectures across clients, permutation invariance of the neurons, and presence of non-linear transformations in each layer. This work introduces a novel framework, *Federated Heterogeneous Neural Networks* (FedHeNN), that allows each client to build a personalised model without enforcing a common architecture across clients. This allows each client to optimize with respect to local data and compute constraints, while still benefiting from the learnings of other (potentially more powerful) clients. The key idea of FedHeNN is to use the instance-level representations obtained from peer clients to guide the simultaneous training on each client. The extensive experimental results demonstrate that the FedHeNN framework is capable of learning better performing models on clients in both the settings of homogeneous and heterogeneous architectures across clients.

1. Introduction

Distributed machine learning has been an important field of study for long and is becoming more and more important with time (Park & Kargupta, 2003). Federated Learning is a type of distributed machine learning setting that consists of many end devices or silo organisations (*clients*) which have access to the data stored locally and a global server which can orchestrate the learning without accessing all the data. With the ever so rapidly growing amount of data and the concerns around data privacy, federated learning has emerged as a very promising direction, as it allows learning of a global model using the data present at each client but without explicitly sharing the data outside the client devices, thus helping in ensuring data privacy and also reducing the cost of centralised training and storage.

FedAvg (McMahan et al., 2017) is the de-facto federated learning algorithm where each client performs SGD steps towards training its local model using its own data and compute resources. The client models are then periodically shared with the server and the server aggregates the client models to create a global model which is sent back to the clients. However, the solution obtained from FedAvg has been shown to diverge in presence of statistical heterogeneity across clients (Li et al., 2020). Over the years several modifications have been proposed to the original algorithm addressing different aspects like data heterogeneity, availability of clients for training, modifying the aggregation mechanism, optimizing communication costs, personalised client models etc. (Li et al., 2020), (Li et al., 2021b), (Karimireddy et al., 2020), (Eichner et al., 2019), (Smith et al., 2017a), (Wang et al., 2020a), (Collins et al., 2021). In all of these methods the global model parameters are obtained by appropriately aggregating the local model parameters.

Yet in most practical settings where the clients are heterogeneous and differ a lot in their compute resources and data distributions, these methods may face significant challenges. Several real-world FL scenarios require training over end devices which have very different hardware. In such cases, for the above methods, the clients that are incapable of training large models will never be able to take part in the training process. Similarly, if the common model architecture is kept small to accommodate all the clients, some clients will be under-utilised. These negative effects might become more prominent in the cross-silo FL setting where the total number of clients is even smaller (typically less than 100). An illustrative experiment shown in Figure 1 demonstrates the drop in accuracy of the global methods like FedAvg and FedProx when a few clients are left out of the training.

In this work, we propose a systematic framework for architecture agnostic federated learning called FedHeNN. This framework is able to overcome the aforementioned challenges of client heterogeneity by allowing each client to build a personalised model without any constraint on the

[1]The University of Texas at Austin, Austin, Texas, USA. Correspondence to: Disha Makhija <disham@utexas.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

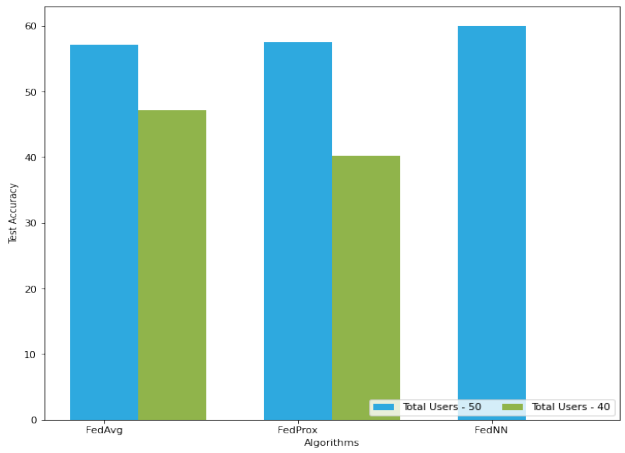

Figure 1. When a fraction of clients cannot afford to build full-blown models, FedAvg and FedProx need to drop the clients altogether resulting in poorer average performance versus FedHeNN which can accommodate clients with heterogeneous architectures.

model architecture including number and size of the hidden layers, activation functions, etc. The learning across clients is transferred by grounding the representations being learnt at each client through a proximal term. Specifically, the optimisation objective at each client comprises of two terms - the task loss, and a proximal term that pulls the final learned representations of the models together. This also makes the neural networks learn more robust representations on a wide range of data leading to better performance. We test our framework on a suite of federated learning datasets in both the settings of homogeneous and heterogeneous model architectures across clients.

Our Contributions are summarized as follows :

1. Our primary contribution is FedHeNN, a new framework for training deep neural networks in federated learning settings. We identify the shortcomings in joint learning of neural-networks and circumvent those by grounding the representations being learnt at each client through a proximal term. We empirically show that having the proximal term on the representations can deliver superior performance.

2. To allow transfer of learning across architectures in different output spaces, we suggest the use of a kernel based distance metric called Centered Kernel Alignment (CKA). The use of kernel based distance metric provides FedHeNN greater flexibility.

3. Additionally, the structure of FedHeNN allows itself to be extended to the setting where different clients have different model architectures. Thus, we propose a solution that is architecture agnostic and has performance that is better or comparable to the existing methods that operate in homogeneous architecture settings.

The rest of the paper is organised as follows. Section 2 provides a background on Federated Learning and related developments. In Section 3, we go over the preliminaries and then propose our framework, FedHeNN. We provide a thorough experimental evaluation of FedHeNN on different types of FL datasets in Section 4 and conclude in Section 5.

2. Related Work

Distributed learning algorithms were extensively studied in the data mining community in the early 2000s under topics such as distributed (and privacy preserving) data mining (Lindell & Pinkas, 2000), (Merugu & Ghosh, Nov, 2003), (Merugu & Ghosh, 2005), (Park & Kargupta, 2003), (Ghosh & Tumer, 2000). This topic was re-framed as "Federated Learning" in an influential paper (McMahan et al., 2017) that introduced the FedAvg algorithm, and has rapidly gained new adherents since then.

The framework proposed by FedAvg (McMahan et al., 2017) has been the standard solution for Federated Learning since 2017. However when the data across clients is non-iid, averaging the local optima of the multiple clients to obtain the global solution may lead to divergence in the optimization. To solve this, FedProx (Li et al., 2020) proposes to modify the local training objective by adding a proximal term which penalizes the distance between the current global model weights and the local model weights thus preventing each local update from moving far away. Similarly, SCAFFOLD (Karimireddy et al., 2020) introduces control-variates to correct the local updates. FedPD (Zhang et al., 2021), FedSplit (Pathak & Wainwright, 2020), and FedDyn (Acar et al., 2021) are other important works that study the problem of finding better fixed points.

(Lin et al., 2020) shows that simply averaging the local models learnt on local distributions to obtain a global model may not be ideal. FedBE (Chen & Chao, 2021) also provides evidence that the best performing aggregate model need not necessarily be the average of the local models. Another important aspect of Federated Learning is thus to appropriately aggregate the local models. PFNM (Yurochkin et al., 2019) and FedMA (Wang et al., 2020a) show that neurons in each layer are permutation invariant and these works perform a layer by layer matching of neurons and then aggregate of the matched neurons. (Singh & Jaggi, 2020) use an Optimal-Transport based distance to perform the matching of neurons before aggregation. Different from above, FedDF (Lin et al., 2020) uses additional data samples to distill knowledge from clients' local models. FedNova (Wang et al., 2020b) allows each client to perform variable amounts of work and calculates a regularized average for the aggregate.

Several works focus on creating local personalized models instead of a single global model to cater to the heterogeneity of the data distributions across clients. In the literature, Personalised FL has been solved using various different ways like keeping the task specific component of the clients local (Collins et al., 2021), clustering the clients (Sattler et al., 2021), (Ghosh et al., 2020), using meta-learning (Fallah et al., 2020), (Beaussart et al., 2021), (Jiang et al., 2019), (Khodak et al., 2019), multi-task learning (Smith et al., 2017b), (Li et al., 2021b), (Smith et al., 2017a) and transfer-learning (Yu et al., 2020).

(Hao et al., 2021), (Goetz & Tewari, 2020), (Luo et al., 2021) augment the data distribution by creating additional data samples and using them for learning. MOON (Li et al., 2021a) uses a contrastive loss to bring the representations of objects closer and utilises it to correct the local training.

The need for heterogeneous client models was recently highlighted in FedProto (Tan et al., 2022). Our work is different from them as our formulation directly works on representations of data instances as opposed to class specific prototypes being used in FedProto for learning and inference. We think that having only one prototype per class could be limiting in case of multi-modal classes.

3. FedHeNN Methodology

In an effort to enhance FL with heterogeneous clients, we propose a new framework, *Federated Heterogeneous Neural Networks* (FedHeNN). FedHeNN provides a systematic way to achieve the goal of joint learning of deep neural networks varying in architecture and output space across distributed clients. In this section, we will first cover the preliminaries and then go to the proposed method. The algorithms for homogeneous and heterogeneous settings are described in Algorithm 1 and 2 respectively.

Preliminaries We first go over the key concepts of the federated learning methods and then elaborate on our proposed method. Consider a set of clients $i \in |N|$, traditional FL algorithm learns a single global model $\hat{\mathcal{W}}$ by using the model parameters learned on individual clients: $\hat{\mathcal{W}} = g(\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_N)$, where $\mathcal{W}_i$ are the local parameters obtained at the i^{th} client and g is the aggregation function. For a single-layer model, $\mathcal{W} = \mathbf{w}$ is the vector of weights. For an m-layer model $\mathcal{W} = (W^1, W^2, \ldots, W^m)$ is the collection containing weights at each layer. We assume the bias terms are incorporated in the weights.

3.1. FedAvg

FedAvg learns each client's local parameters by solving

$$\min_{\mathcal{W}_i} \mathcal{L}_i = \mathcal{F}(\mathcal{W}_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(y_j, \hat{y}_j; \mathcal{W}_i). \tag{1}$$

where $\ell(.)$ is the loss function. An element-wise average is then performed on the weight matrices of the clients for getting an aggregated model $\hat{\mathcal{W}}$ at the server. In case of a multi-layer network, the average is taken layer-wise and the parameters of each client are weighted based on the number of samples present at the client. For the l^{th} layer, we have

$$\hat{\mathcal{W}}^l = \frac{1}{\sum_k n_k} \sum_{i=1}^{N} n_i \mathcal{W}_i^l.$$

While this aggregation mechanism is shown to have good empirical results, this method of learning involving element-wise averaging might give sub-optimal results because of various reasons like permutation invariance property of the neurons, different data distributions across clients etc. Besides, FedAvg imposes a hard constraint on clients to train the models with the exact same architectures. In practice, it is highly likely that different clients may not be able to train the models of same capacity.

We identify that apart from solving for sub-problems like aggregation mechanism or data heterogeneity, we also need to make sure that the networks on clients are being trained towards the same goal. In order to do so we suggest to modify the objective function being optimized at each client.

We consider a neural network to be composed of two components - the representation learning component that maps the input to a k-dimensional representation vector and the task learning component which learns the prediction function. The initial layers of the neural network are considered as the representation learning component whose output is the representation vector denoted by $\Phi(x; \mathcal{W})$ for a data instance x under the model with parameters $\mathcal{W}$, while the last layer is considered to be the task-specific layer and the output of which is the prediction corresponding to data instances denoted by $\hat{y}$, as depicted in Figure 2. The details of the method are explained in the next sub-sections. Additionally, we explore two different settings for the FedHeNN framework - homogeneous and heterogeneous setting, which we define below and elaborate on in the next sub-section.

Definition 3.1. We refer to a FL setting as **homogeneous** when each of the client in the network has the exact same architecture. A FL setting is said to be **heterogeneous** when each client in the network can define its own architecture.

3.2. FedHeNN for Homogeneous Clients

For the homogeneous federated learning setting, we propose to modify the loss function on each client to additionally incorporate a proximal term on the representations. In particular, instead of just minimising the loss function $\mathcal{F}(\mathcal{W}_i)$, each client minimises the following objective -

$$\min_{\mathcal{W}_i} \mathcal{L}_i = \mathcal{F}(\mathcal{W}_i) + \eta \, \mathrm{d}(\Phi_i(\mathbf{X}; \mathcal{W}_i), \Phi_{\text{global}}(\mathbf{X}; \hat{\mathcal{W}}(t-1))). \tag{2}$$

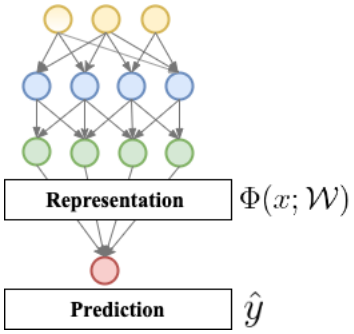

Figure 2. Depiction of a Neural Network consisting of two components - representation learning part and prediction function.

where $\mathrm{d}(.,.)$ is some distance metric, $\Phi_i(\mathbf{X}; \mathcal{W}_i)$ is the representation learned on the i^{th} client for the set of data points $\mathbf{X}$, and the $\Phi_{\text{global}}(\mathbf{X}; \hat{\mathcal{W}}(t-1))$ is the representation learned by the global model communicated at the previous round $t-1$. The $\hat{\mathcal{W}}$ is learned using the FedAvg algorithm. Because we are ensuring the similarities between the representations we do not explicitly need to address the aggregation mechanism. The proximal term helps in bringing the representations of the local and global model together and helps in correcting the learning on each client.

3.3. FedHeNN for Heterogeneous Clients

In certain settings it might not be practical for all clients to train the neural network models of same complexity. We identify that it is non-trivial to use existing algorithms to perform joint training in such settings.

Our framework allows one to perform federated learning across heterogeneous clients by collecting and aggregating the representations learnt on the local clients and pulling those representations closer. The framework is inspired by the same intuition that there is an underlying lower dimensional representation of the data that the clients can together uncover. To achieve this, we let each client train its own model but pull the representations learnt by different clients closer by adding a proximal term to the client's loss function. The proximal term measures the distance between the representations learnt by different local models. Specifically, if $\Phi_i(\mathbf{X}; \mathcal{W}_i)$ is the representation matrix obtained on the i^{th} client for data $\mathbf{X}$, then the proximal term measuring the distances between client representations for the i^{th} client can be written as -

$$\mathrm{d}(\Phi_i(\mathbf{X}; \mathcal{W}_i), \bar{\Phi}(\mathbf{X}; (t-1))),$$

where

$$\bar{\Phi}(\mathbf{X}; (t-1)) = \sum_{j=1}^{N} w_j \Phi_j(\mathbf{X}; \mathcal{W}_j(t-1)).$$

The contribution of each client j, w_j could be set to reflect the capacity or the strength of the model on the j^{th} client. At each iteration the server generates a set of unlabelled data instances called *Representation Alignment Dataset* (RAD) denoted by $\mathbf{X}$ and uses it to align the representations across clients. As before, the clients share the local weights with the server. Instead of aggregating the weights, the server uses each client's weights to generate representations for the RAD and aggregates the representations over clients. The server then distributes the aggregated representations and the RAD to each client. The clients on the next iteration learn a model that minimizes the loss that is a combination of the task loss and distance to the aggregated representations.

Specifically, at iteration t each client is trying to minimize

$$\min_{\mathcal{W}_i} \mathcal{L}_i = \mathcal{F}(\mathcal{W}_i) + \eta\, \mathrm{d}(\Phi_i(\mathbf{X}; \mathcal{W}_i), \bar{\Phi}(\mathbf{X}; (t-1))). \quad (3)$$

This method helps us utilise the training from local clients but without explicitly generating a global aggregated model. The privacy of the clients data is also preserved as the clients do not need to share their local data. Thus the clients can keep on using the local personalised models but utilise the training from other peer clients.

Later when we describe the distance function $\mathrm{d}(.,.)$ we will see that our method is robust to having output representations of clients in different spaces, i.e., $\Phi_i(x; \mathcal{W}_i)$'s, $\forall i$, need not be aligned with each other.

3.4. The Proximal Term

Here we give a detailed explanation about the proximal term or the distance function being used in our objective formulation in equations (2) and (3).

As mentioned earlier, we consider a deep neural network model training to be performing two high level steps - learning to map the inputs of different modalities to a k-dimensional representation, and then learning a prediction function. We consider all but the last layer of the neural network as the representation learning module and its output as the learned representation and denote it by $\Phi(.;.)$. So in order to match the representation learning part of the network across all clients we compare the outputs of the representation learning module of the networks on the same set of instances. In specific, we use a set of input examples $x_l \in |L|$ encoded in matrix $\mathbf{X}$ a.k.a. RAD, pass them through all the local networks and capture the outputs of the representation learning module of the networks in matrices $A_i \in \mathbb{R}^{L \times d_i}$, where $A_i(l,:) = \Phi_i(x_l; \mathcal{W}_i)$. A_i stands for the activation (representation) matrix of the i^{th} client, L is the size of RAD and d_i is the output dimension of the representation. Note that our method can work with representations of different dimensions on different clients, i.e., d_i can vary with i. At each communication round, we

randomly select only a sample of the instances to be a part of the alignment dataset keeping $L << n_i, \forall i$.

After obtaining the activation matrices A_i's, we need a way to compare the representations. For this, we suggest using the representation distance matrices (RDMs). An RDM uses the distance between the instances to capture the characteristics of the representational space. Representational distance learning (McClure & Kriegeskorte, 2016) has also been used for knowledge distillation in the past.

To measure the distance, we use a distance metric proposed specifically for neural network representations called Centered Kernel Alignment(CKA) (Kornblith et al., 2019). CKA was originally proposed to compare the representations obtained from different neural networks to determine equivalence relationships between hidden layers of neural networks. CKA takes into input the activation (representation) matrices and outputs a similarity score between 1 (identical representations) and 0 (not similar at all). Some of the useful properties of CKA include invariance to invertible linear transformation, orthogonal transformation and isotropic scaling.

The other common distance metrics used in the literature for comparing representations from neural networks (Kornblith et al., 2019; Ding et al., 2021) are based on Canonical Correlation Analysis(CCA) and Procrustes distance. The CCA based distance requires computing eigenvectors of a $n \times k$ matrix formed by k-dimensional representations of n instances. But usage of eigenvector based function in the loss causes numerical instabilities while training through backpropagation thus making all CCA based distances unusable for training neural networks. The Procrustes based distance is only suitable for matrices with same dimensions and thus works only in the homogeneous settings. We did experiment with using orthogonal Procrustes distance for homogeneous settings in our method and found that the obtained performance is comparable with respect to CKA, for example, on CIFAR10, CKA obtains $94.7 \pm 1.1\%$ accuracy and orthogonal Procrustes obtains $93.7 \pm 1.7\%$. Also, the experiments shown in the paper that proposed CKA (Kornblith et al., 2019) show the superiority of the CKA based method for comparing representations across different neural networks. Moreover, because CKA is a kernel based metric it provides a way to compare the representations obtained from networks of different widths thus providing greater flexibility for heterogeneous settings. The kernel based metric is able to characterize the representation space via the pairwise similarity between instances and thus helps in diminishing the effects of permutation invariance. CKA is also shown to be invariant to changes in initializations of the network. Because of all these properties, CKA extends itself naturally for use as a distance metric in our method.

Let $A_i \in \mathbb{R}^{L \times d_i}$ and $A_j \in \mathbb{R}^{L \times d_j}$ be the representation matrices for clients i and j obtained for the RAD. The distance between A_i and A_j is obtained by first computing the kernel matrices K_i and K_j for any choice of kernel $\mathcal{K}$:

$$K_i(p,q) = \mathcal{K}(A_i(p,:), A_i(q,:)),$$

$$K_j(p,q) = \mathcal{K}(A_j(p,:), A_j(q,:)),$$

where $K_i \in \mathbb{R}^{L \times L}$ and $K_j \in \mathbb{R}^{L \times L}$ are the representational distance matrices then similarity between K_i and K_j is given under CKA by -

$$\text{CKA}(K_i, K_j) = \frac{\text{HSIC}(K_i, K_j)}{\sqrt{\text{HSIC}(K_i, K_i)\text{HSIC}(K_j, K_j)}}.$$

where the estimator for Hilbert-Schmidt Independence Criterion (HSIC) could be written as -

$$HSIC(K_i, K_j) = \frac{1}{(L-1)^2}\text{tr}(K_i H K_j H) \quad .$$

with H as the centering matrix

$$H = I_L - \frac{1}{L}\mathbf{1}\mathbf{1}^T.$$

We use a linear and an RBF kernel for computing distances. The Linear CKA can be simply written as -

$$\text{Linear CKA}(K_i, K_j) = \text{Linear CKA}(A_i A_i^T, A_j A_j^T) = \frac{||A_j^T A_i||_F^2}{||A_i^T A_i||_F ||A_j^T A_j||_F}. \tag{4}$$

The local learning objectives for our method in homogeneous and heterogeneous settings use CKA based dissimilarity and thus respectively become

$$\min_{\mathcal{W}_i} \mathcal{L}_i = \mathcal{F}(\mathcal{W}_i) + \eta \, \text{d}_{\text{CKA}}(K_i, K_{\text{global}}(t-1)). \tag{5}$$

where $K_{\text{global}}(t-1)$ is the representation distance matrix obtained over the instances from the global model at previous iteration, and

$$\min_{\mathcal{W}_i} \mathcal{L}_i = \mathcal{F}(\mathcal{W}_i) + \eta \, \text{d}_{\text{CKA}}\left(K_i, \bar{K}(t-1)\right), \tag{6}$$

where we define

$$\bar{K}(t-1) = \sum_{j=1}^{N} w_j K_j(t-1).$$

For $\eta = 0$, the objective function corresponding to homogeneous FedHeNN, in equation (5), becomes the FedAvg algorithm. For the heterogeneous FedHeNN, the objective function given in equation (6) boils down to individual

clients training their own local models in isolation using only the local data. On the other hand, when $\eta \to \infty$, the framework will try to obtain identical representations from all the local models without caring about the prediction task. In homogeneous FL setting for FedHeNN with linear CKA, after sufficiently large number of iterations t', the framework will make all the models identical. The following lemma provides an intuitive explanation on the loss function.

Lemma 3.2. *Given a homogeneous FL setting with clients training linear models and $\eta \to \infty$, then after sufficiently large number of iterations t', for the FedHeNN framework with Linear CKA, we have $\mathcal{W}_i = \hat{\mathcal{W}}$ for all i.*

Proof. The optimization problem at each client i is

$$\min_{\mathcal{W}_i} \mathcal{L}_i = \mathcal{F}(\mathcal{W}_i) + \eta \, \mathrm{d}_{\text{CKA}}(K_i, K_{\text{global}}(t-1)).$$

When $\eta \to \infty$, the problem becomes

$$\max_{\mathcal{W}_i} \text{CKA}(K_i, K_{\text{global}}(t-1)).$$

For linear CKA, we have

$$\text{CKA}(K_i, K_{\text{global}}) = \frac{||A_{\text{global}}^T A_i||_F^2}{||A_i^T A_i||_F ||A_{\text{global}}^T A_{\text{global}}||_F}.$$

If we assume that each client has a linear network, then $A_i(\mathbf{X}) \in \mathbb{R}^{L \times d_i} = \Phi_i(\mathbf{X}; \mathcal{W}_i) = (\mathbf{X} W_i^{(1)})$. Then, after sufficiently large number of iterations t', i.e., when each client has seen numerous RADs, optimizing equation (5) for $\eta \to \infty$ will lead to

$$\mathcal{W}_i = \hat{\mathcal{W}} \quad \forall i.$$

As a consequence, we reach the conclusion of the lemma. □

Algorithm 1 FedHeNN Algorithm for Homogeneous clients

Input: number of clients N, number of communication rounds T, number of local epochs E, parameter η_0
Output: Final model $\hat{\mathcal{W}}(T)$
At Server -
Initialize $\hat{\mathcal{W}}(0)$
for $t = 1$ **to** T **do**
  $\eta = \eta_0 \times f(t)$
  Generate RAD, $\mathbf{X}$, by random sampling
  Select a subset of clients $\mathcal{N}_t$
  for each selected client $i \in \mathcal{N}_t$ **do**
    $\mathcal{W}_i(t) =$ **LocalTraining**$(\hat{\mathcal{W}}(t-1), \mathbf{X}, \eta)$
  end for
  $\hat{\mathcal{W}}(t) = \frac{1}{\sum_{j \in \mathcal{N}_t} n_j} \sum_{i \in \mathcal{N}_t} n_i \mathcal{W}_i(t)$
end for
$\hat{\mathcal{W}}(T) = \frac{1}{\sum_{j \in |N|} n_j} \sum_{i \in |N|} n_i \mathcal{W}_i(t)$
Return $\hat{\mathcal{W}}(T)$
LocalTraining$(\hat{\mathcal{W}}(t-1), \mathbf{X}, \eta)$
Initialize $\mathcal{W}_i(t)$ with $\hat{\mathcal{W}}(t-1)$
$A_{\text{global}}(\mathbf{X}) = \Phi_{\text{global}}(\mathbf{X}; \hat{\mathcal{W}}(t-1))$
$K_{\text{global}}(t-1) = \mathcal{K}(A_{\text{global}}(\mathbf{X}), A_{\text{global}}(\mathbf{X}))$
for each local epoch **do**
  $A_i(\mathbf{X}) = \Phi_i(\mathbf{X}; \mathcal{W}_i(t))$
  $K_i = \mathcal{K}(A_i(\mathbf{X}), A_i(\mathbf{X}))$
  Update $\mathcal{W}_i(t)$ using SGD for loss in equation (5)
end for
Return $\mathcal{W}_i(t)$ to the server

4. Experiments

We now present the effectiveness of the FedHeNN framework using empirical results on different datasets and models. We simulate statistically heterogeneous and system heterogeneous FL settings by manipulating the data partitions and model architectures across clients. We also discuss the effects of other variables like the size of RAD, η and the choice of kernel on the performance of FedHeNN.

4.1. Experimental Details

We evaluate FedHeNN in different settings involving varied models, tasks, heterogeneity levels, and datasets. We start with descriptions of these settings followed by our results.

Datasets We consider two different high level tasks, image classification and text classification, and use datasets corresponding to these from the popular federated learning benchmark LEAF (Caldas et al., 2019). For the image classification task, we use CIFAR-10 and CIFAR-100 datasets that contain colored images in 10 and 100 classes respectively. And for the text classification task we use a binary classification dataset called Sentiment140. We partition the entire data to generate non-iid samples on each client and then split those into training and test sets at the client site.

Baselines We compare our method against three different baselines - FedAvg, FedProx and FedRep. The FedAvg and FedProx algorithms learn a centralised global model thus the reported performance metric is of the global model. On the contrary, the FedRep method learns personalised models for each client. Therefore, FedRep's performance is reported for the personalised models. For FedHeNN, we report the performance of local models for both homogeneous and heterogeneous settings and that of the global model for the homogeneous setting. As for FedProto, it is demonstrated in the paper that the performance gap between FedProto and FedAvg decreases when we have more samples per class. In our settings, since we do not restrict the number of samples per class and also beat the FedAvg algorithm by a significant

Algorithm 2 FedHeNN Algorithm for Heterogeneous clients

Input: number of clients N, number of communication rounds T, number of local epochs E, parameter η_0, weight vector for clients $[w_1, w_2, \dots w_N]$
Output: Final set of personalised models $\mathcal{W}_1(T), \mathcal{W}_2(T) \dots \mathcal{W}_N(T)$
At Server -
Initialize $\mathcal{W}_1(0), \mathcal{W}_2(0) \dots \mathcal{W}_N(0)$
for $t = 1$ **to** T **do**
  $\eta = \eta_0 \times f(t)$
  Generate RAD, $\mathbf{X}$, by random sampling
  for each client j **do**
    $A_j(\mathbf{X}) = \Phi_j(\mathbf{X}; \mathcal{W}_j(t-1))$
    $K_j = \mathcal{K}(A_j(\mathbf{X}), A_j(\mathbf{X}))$
  end for
  $\bar{K}(t-1) = \sum_{j=1}^{N} w_j K_j$
  Select a subset of clients $\mathcal{N}_t$
  for each selected client $i \in \mathcal{N}_t$ **do**
    $\mathcal{W}_i(t) =$ **LocalTraining**$(\mathcal{W}_i(t-1), \bar{K}(t-1), \mathbf{X}, \eta)$
  end for
end for
Return $\mathcal{W}_1(T), \mathcal{W}_2(T) \dots \mathcal{W}_N(T)$
LocalTraining$(\mathcal{W}_i(t-1), \bar{K}(t-1), \mathbf{X}, \eta)$
Initialize $\mathcal{W}_i(t)$ with $(\mathcal{W}_i(t-1))$
for each local epoch **do**
  $A_i(\mathbf{X}) = \Phi_i(\mathbf{X}; \mathcal{W}_i(t))$
  $K_i = \mathcal{K}(A_i(\mathbf{X}), A_i(\mathbf{X}))$
  Update $\mathcal{W}_i(t)$ using SGD for loss in equation (6)
end for
Return $\mathcal{W}_i(t)$ to the server

margin, we do not directly compare with FedProto.

Evaluation Metric and other Parameters We use the average test accuracy obtained on the clients' test datasets as the evaluation criterion. For global models the test accuracy is computed by evaluating the global model on the local test datasets and for the local models test accuracy is computed by testing personalised models on the local test datasets. For hyperparameter tuning of methods, we utilise a global validation dataset which is not shared with any of the clients.

The hyperparameter η that controls the contribution of representation similarity in the objective function is kept as a function of t (the number of communication round). This is because the initial representations obtained from insufficiently trained models are not accurate and keeping a high η in the initial rounds may mislead the training. The base value of η_0 is tuned as a hyperparameter and the best performance is obtained by keeping $\eta_0 = 0.001$ for CIFAR-10 and CIFAR-100, and $\eta_0 = 0.01$ for Sentiment140.

The size of RAD is an important parameter. The reported performance is obtained by keeping this size constant at 5000 which is much smaller than the size of training or test sets and doesn't increase the memory footprint drastically.

Implementation For the FL simulations, we keep the non-iid data distribution across clients such that each client will have access to data of only certain classes, for example, with 5 classes per client, i^{th} client might have access to data for classes $\{2, 3, 4, 5, 8\}$ and client j might have $\{1, 4, 5, 6, 7\}$. We also vary this number of classes to change the heterogeneity level across clients. The robustness and scalability of our method is tested by increasing the number of clients participating in training from 100 to 500 for CIFAR-10 dataset. The total number of communication rounds is kept constant at 200 for all algorithms and at each round only 10% of the clients are sampled and updated. We find that increasing the number of local epochs on clients doesn't worsen the performance of the client models for our method, so the number of local epochs is set to 20. For the heterogeneous FedHeNN, each entry of the weight vector for aggregating the representations $\mathbf{w}$ is set to $\frac{1}{N}$. In each local update, we use SGD with momentum for training.

For the homogeneous FedHeNN for CIFAR datasets, we use a CNN model with 2 convolutional layers with each convolutional layer followed by a max-pooling layer followed by 3 fully-connected layers at the end. For the heterogeneous FedHeNN, for each client we uniformly randomly sample from a set of 5 different CNNs obtained by varying the architecture size in between the simplest one that contains 1 convolutional and 1 fully connected layer and the most complex one containing 3 convolutional and 3 fully connected layers. For the Sentiment140 dataset, we use either a 1-layer or a 2-layer LSTM followed by 2 fully connected layers.

CKA For the CKA distance metric, we evaluate the performances by using a linear kernel as well as an RBF kernel. For the RBF kernel, we try various values for σ but as we will show later, the performance obtained using RBF kernel is not very different from that of the linear kernel.

4.2. Results

The performance of FedHeNN and baselines under various settings is reported in Table 1 and Table 2. The global model performances of the FedHeNN global model and related baselines is reported in Table 1. We observe that the FedHeNN global model outperforms the FedAvg and FedProx algorithms. The results for comparisons of the personalised models are reported in Table 2 and the results demonstrate that the FedHeNN's performance is better than the baselines. We observe that the homogeneous FedHeNN has a higher gain over the baselines than the heterogeneous FedHeNN, which is expected because of the varying capacity of the local models in the heterogeneous setting.

Table 1. Average test accuracy of FedHeNN computed for the common global model as compared to the baselines with global models.

DATA SET(SETTING)	FEDAVG	FEDPROX	FEDHENN GLOBAL
CIFAR-10(100 CLIENTS, 2 CLS/CLIENT)	44.29 ± 0.5	53.8 ± 2.3	68.8 ± 2.1
CIFAR-10(100 CLIENTS, 5 CLS/CLIENT)	58.14 ± 0.7	63.3 ± 2.0	70.19 ± 2.0
CIFAR-10(500 CLIENTS, 2 CLS/CLIENT)	42.7 ± 0.4	50.46 ± 1.4	65.4 ± 0.8
CIFAR-10(500 CLIENTS, 5 CLS/CLIENT)	56.8 ± 0.5	55.2 ± 1.2	64.7 ± 0.7
CIFAR-100(100 CLIENTS, 20 CLS/CLIENT)	28.6 ± 0.8	27.3 ± 1.1	44.2 ± 0.7
SENTIMENT140(100 CLIENTS, 2 CLS/CLIENT)	52.6 ± 0.4	52.7 ± 1.0	52.7 ± 0.01

Table 2. Average test accuracy of FedHeNN computed for the personalised models as compared to the baselines with personalised models.

DATA SET(SETTING)	FEDREP	FEDHENN HOMO	FEDHENN HETERO
CIFAR-10(100 CLIENTS, 2 CLS/CLIENT)	85.7 ± 0.4	94.7 ± 1.1	88.9 ± 0.35
CIFAR-10(100 CLIENTS, 5 CLS/CLIENT)	72.4 ± 1.2	84.37 ± 1.5	73.01 ± 0.3
CIFAR-10(500 CLIENTS, 2 CLS/CLIENT)	78.9 ± 0.6	86.5 ± 0.9	82.02 ± 0.8
CIFAR-10(500 CLIENTS, 5 CLS/CLIENT)	58.14 ± 0.21	73.32 ± 1.23	61.74 ± 0.6
CIFAR-100(100 CLIENTS, 20 CLS/CLIENT)	38.85 ± 0.9	62.89 ± 0.8	43.36 ± 0.2
SENTIMENT140(100 CLIENTS, 2 CLS/CLIENT)	69.8 ± 0.4	72.6 ± 0.3	71.5 ± 0.5

Table 3. Test Accuracy for Linear vs RBF Kernel compared for homogeneous FedHeNN on CIFAR-10 dataset.

DATASET	LINEAR CKA	RBF CKA
CIFAR-10	94.47	93.03
CIFAR-100	84.37	83.03
SENTIMENT140	72.6	72.8

Linear vs RBF Kernel For computing the CKA based distances, we try using both the Linear as well as RBF kernel. Based on the empirical analysis of FedHeNN, it is observed that both the linear and RBF kernels give comparable performances as shown in Table 3.

Effect of Local epochs We also analyse the effect of varying local epochs in FedHeNN. In FedAvg, increasing the number of local epochs has an adverse effect on the performance of the model. On the other hand, no such effect was observed for FedHeNN owing to the presence of the proximal term. We keep the number of local epochs for FedHeNN to be as high as 20.

Sensitivity to Changing Amount of Data We have empirically shown that FedHeNN is able to accommodate the clients with lower compute resources in an effective way. In order to show that the FedHeNN can also work with the clients with smaller data footprint, we do an experiment in which we randomly take a fraction of clients and reduce the data on those clients by 50%. We show the results of the experiment in Figure 3 where x-axis has the fraction of clients picked for shrinking the data and y-axis is the average test accuracy of all clients obtained when the framework is trained on the reduced dataset. We notice that even with the decreasing data size on the clients' ends FedHeNN is

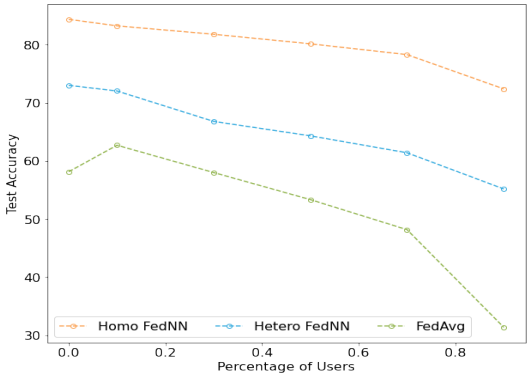

Figure 3. Changes in test accuracy when the data on a fraction of clients is reduced by 50% shown for CIFAR-10 dataset.

able to maintain graceful performance. This effect could be attributed to the fact that even though the number of instances to train the prediction function is reduced, the representation learning component is still robust.

5. Discussion

In this paper, we present a new method for enhancing learning in Federated Learning by introducing a systematic framework called FedHeNN. The FedHeNN framework is unique because it allows the clients with heterogeneous architectures to participate in the joint learning process helping boost the performance. This could be a huge practical advancement as now the individual client devices or organisations with variable amount of resources can equally contribute and learn from each other. The empirical results indicate that FedHeNN is able to achieve better performing results

while also being more inclusive. For future work, we would work on determining the solution characteristics and the convergence guarantees of both the FedHeNN algorithms. There are a few natural research directions arising from this work. First, it is of interest to extend the FedHeNN to the Transformer architectures (Vaswani et al., 2017; Nguyen et al., 2022a;b). Second, we can use generative models to generate the RAD, the data being used for aligning representations to relax the assumption of presence of additional data on the server.

Acknowledgements

This work was supported by Office of Naval Research(ONR) [N00014-19-1-2625]. We thank Dr. Ravi Srinivasan and Dr. Alex Liu for their valuable suggestions. NH gratefully acknowledges support from the NSF [IFML 2019844] award and research gifts by the NSF AI Institute for Foundations of Machine Learning.

References

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

Beaussart, M., Grimberg, F., Hartley, M., and Jaggi, M. WAFFLE: weighted averaging for personalized federated learning. *CoRR*, abs/2110.06978, 2021.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., and Talwalkar, A. Leaf: A benchmark for federated settings, 2019.

Chen, H.-Y. and Chao, W.-L. Fed{be}: Making bayesian model ensemble applicable to federated learning. In *International Conference on Learning Representations*, 2021.

Collins, L., Hassani, H., Mokhtari, A., and Shakkottai, S. Exploiting shared representations for personalized federated learning. *arXiv preprint arXiv:2102.07078*, 2021.

Ding, F., Denain, J.-S., and Steinhardt, J. Grounding representation similarity through statistical testing. In *Advances in Neural Information Processing Systems*, 2021.

Eichner, H., Koren, T., McMahan, B., Srebro, N., and Talwar, K. Semi-Cyclic Stochastic Gradient Descent. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1764–1773. PMLR, 2019.

Fallah, A., Mokhtari, A., and Ozdaglar, A. E. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. An efficient framework for clustered federated learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Ghosh, J. and Tumer, K. Robust order statistics based ensembles for distributed data mining. In Kargupta, H. and Chan, P. (eds.), *Advances in Distributed and Parallel Knowledge Discovery*, pp. 185–210. AAAI Press, 2000.

Goetz, J. and Tewari, A. Federated learning via synthetic data. *CoRR*, abs/2008.04489, 2020.

Hao, W., El-Khamy, M., Lee, J., Zhang, J., Liang, K. J., Chen, C., and Carin, L. Towards fair federated learning with zero-shot data augmentation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*, pp. 3310–3319. Computer Vision Foundation / IEEE, 2021.

Jiang, Y., Konečný, J., Rush, K., and Kannan, S. Improving federated learning personalization via model agnostic meta learning. *CoRR*, abs/1909.12488, 2019.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. SCAFFOLD: Stochastic controlled averaging for federated learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143. PMLR, 13–18 Jul 2020.

Khodak, M., Balcan, M., and Talwalkar, A. S. Adaptive gradient-based meta-learning methods. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 5915–5926, 2019.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 09–15 Jun 2019.

Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021a.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In Dhillon, I. S., Papailiopoulos, D. S., and Sze, V. (eds.), *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020.

Li, T., Hu, S., Beirami, A., and Smith, V. Ditto: Fair and robust federated learning through personalization. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6357–6368. PMLR, 2021b.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2351–2363. Curran Associates, Inc., 2020.

Lindell, Y. and Pinkas, B. Privacy preserving data mining. *LNCS*, 1880:36–77, 2000.

Luo, M., Chen, F., Hu, D., Zhang, Y., Liang, J., and Feng, J. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data, 2021.

McClure, P. and Kriegeskorte, N. Representational distance learning for deep neural networks. *Frontiers in Computational Neuroscience*, 10:131, 2016. ISSN 1662-5188. doi: 10.3389/fncom.2016.00131.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 20–22 Apr 2017.

Merugu, S. and Ghosh, J. A distributed learning framework for heterogeneous data sources. In *Proc. KDD*, pp. 208–217, 2005.

Merugu, S. and Ghosh, J. Privacy perserving distributed clustering using generative models. In *Proc. ICDM*, pp. 211–218, Nov, 2003.

Nguyen, T., Nguyen, T., Le, D., Nguyen, D., Anh, T., Baraniuk, R., Ho, N., and Osher, S. Improving Transformers with Probabilistic Attention Keys. In *International Conference on Machine Learning (ICML)*, 2022a.

Nguyen, T., Pham, M., Nguyen, T., Nguyen, K., Osher, S., and Ho, N. Transformer with Fourier integral attentions. *arXiv preprint arXiv:2206.00206*, 2022b.

Park, B.-Y. and Kargupta, H. Distributed data mining. In Ye, N. (ed.), *The Handbook of Data Mining*, pp. 341–364. Lawrence Erlbaum Assoc., 2003.

Pathak, R. and Wainwright, M. J. Fedsplit: an algorithmic framework for fast federated optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7057–7066. Curran Associates, Inc., 2020.

Sattler, F., Müller, K., and Samek, W. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Trans. Neural Networks Learn. Syst.*, 32(8):3710–3722, 2021.

Singh, S. P. and Jaggi, M. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33, 2020.

Smith, V., Chiang, C., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4424–4434, 2017a.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017b.

Tan, Y., Long, G., Liu, L., Zhou, T., Lu, Q., Jiang, J., and Zhang, C. Fedproto: Federated prototype learning across heterogeneous clients. In *AAAI Conference on Artificial Intelligence*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020a.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances*

in Neural Information Processing Systems, volume 33, pp. 7611–7623. Curran Associates, Inc., 2020b.

Yu, T., Bagdasaryan, E., and Shmatikov, V. Salvaging federated learning by local adaptation. *CoRR*, abs/2002.04758, 2020.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7252–7261, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Zhang, X., Hong, M., Dhople, S., Yin, W., and Liu, Y. Fedpd: A federated learning framework with adaptivity to non-iid data. *IEEE Transactions on Signal Processing*, 69:6055–6070, 2021. doi: 10.1109/TSP.2021.3115952.