In Federated learning (FL), multiple clients collaborate to learn a shared model through a central server while they keep data decentralized. Personalized federated learning (PFL) further extends FL by learning personalized models per client. In both FL and PFL, all clients participate in the training process and their labeled data is used for training. However, in reality, novel clients may wish to join a prediction service after it has been deployed, obtaining predictions for their own unlabeled data.

Here, we introduce a new learning setup, Inference-Time Unlabeled PFL (ITU-PFL), where a system trained on a set of clients, needs to be later applied to novel unlabeled clients at inference time. We propose a novel approach to this problem, ITUFL-HN, which uses a hypernetwork to produce a new model for the late-to-the-party client. Specifically, we train an encoder network that learns a representation for a client given its unlabeled data. That client representation is fed to a hypernetwork that generates a personalized model for that client. Evaluated on five benchmark datasets, we find that ITUFL-HN generalizes better than current FL and PFL methods, especially when the novel client has a large domain shift from training clients. We also analyzed the generalization error for novel clients, and showed analytically and experimentally how they can apply differential privacy to their data.

1. Introduction

Federated learning (FL) is the task of learning a model over multiple disjoint local datasets, while keep data decentralized (McMahan et al., 2017a). Personalized Federated learning (PFL) (Zhao et al., 2018) extends FL to the case where the data distribution varies across clients. PFL has numerous applications from a smartphone application wishes to improve text prediction without uploading user-sensitive data, to the case when a consortium of hospitals wishes to train a joint model while preserving the privacy of their patients. Current methods assume that all clients participate in the training process and that their data is labeled (Shamsian et al., 2021; Bui et al., 2019; Hsu et al., 2020; Zhu et al., 2020; Yang et al., 2020; Hanzely & Richtárik, 2020), so once a model is trained, novel client can not collaborate with the existing clients without retrain the system.

When a novel client joins with its labeled data, there are various strategies for adapting the model to the novel client. For instance, a FL model can be fine-tuned using those labels (Fan & Huang, 2021). For PFL, it is less clear which personalized model should be fine-tuned. A different approach was proposed by Shamsian et al. (2021). They train a hypernetwork that generates a personalized model for each client, given a descriptor of that client. When a novel client joins with its labeled data, there are various strategies for adapting the model to the novel client. For instance, a FL model can be fine-tuned using those labels (Fan & Huang, 2021). For PFL, it is less clear which personalized model should be fine-tuned. A different approach was proposed by Shamsian et al. (2021). They train a hypernetwork that generates a personalized model for each client, given a descriptor of that client. To generalize to a novel client with labeled data, they find a descriptor for that client by fine-tuning it using the labeled data through the hypernetwork. While all those approaches are useful, they cannot handle novel clients that have no labeled data.

Here, we define a novel problem, performing federated learning on novel clients with unlabeled data that are only available at inference time. We call this setup ITU-PFL for Inference-Time Unlabeled Personalized Federated Learning. We propose a novel approach to this problem, called ITUFL-HN. During training, our architecture learns a space
of personalized models, one for each client, together with an encoder that maps each client to a point in that client space. All personalized models are learned jointly through a hypernetwork, allowing to combine personalized data effectively (Figure 1). At inference time, a novel client can locally compute its own descriptor using the client encoder. Then, it sends the descriptor to the server as input to the hypernetwork and obtains its personalized model.

A key question remains for this approach to succeed: How to compute a descriptor of a novel unlabeled client? A key idea is to define a client encoder that maps a dataset into a dense descriptor, and tune its parameters jointly with the main model during training. We explore properties that this encoder should hold. First, its architecture should implement a function that is invariant to the order of its inputs (Zaheer et al., 2017). Furthermore, if objects in the data adhere to their own symmetries, like images, graphs, or point clouds, the encoder architecture should be invariant to these symmetries as well (Maron et al., 2020). To the best of our knowledge, this is the first paper that discusses learning such invariant descriptors of datasets or clients.

FL was motivated by privacy, but was shown vulnerable to differential attacks (Xie et al., 2018; Augenstein et al., 2019; Melis et al., 2019; Zhu & Philip, 2019; Hao et al., 2019; Truex et al., 2019; Mothukuri et al., 2021; Hitaj et al., 2017). In our setup of ITU-PFL, novel clients do not contribute their data to training, so they have better control over their data privacy. We show theoretically how differential privacy (DP) can be applied effectively to a novel client and then experimentally measured how applying DP to the client affects the accuracy of its personalized model.

This paper makes the following contributions: (1) A new learning setup, ITU-PFL, learning a personalized model to novel unlabeled clients at inference time. (2) A new approach, learning a space of models and an encoder that maps a client to that space, and an architecture ITUFL-HN based on hypernetworks. (3) A generalization bound based on multi-task learning (MTL) and domain adaptation (DA), and analysis of differential privacy for the novel client. (4) Evaluation on 5 benchmark datasets, CIFAR10, CIFAR100, iNaturalist, Landmarks, Yahoo answers showing that ITUFL-HN performs better or equally well as baselines.

2. Related work

Federated learning (FL). In FL, clients collaboratively solve a learning task. The key motivation for individual clients to participate in FL is to leverage the shared pool of knowledge from other clients in the federation. Individual clients often face data constraints such as data scarcity, low data quality, and unseen classes that limit their capacity to train well performing local models. FedAvg (McMahan et al., 2017a), is an early but effective FL approach that updates models locally and averages them into a global model. Several optimization methods have been proposed for improving convergence in FL (Sahu et al., 2018b; Lin et al., 2018; Stich, 2018; Wang & Joshi, 2018). Other approaches focus on preserving client privacy (Duchi et al., 2014; McMahan et al., 2017b; Agarwal et al., 2018; Zhu et al., 2020), improving robustness to statistical diversity (Haddadpour & Mahdavi, 2019; Hanzely & Richtárik, 2020; Hsu et al., 2019b; Karimireddy et al., 2020; Zhao et al., 2018; Zhou & Cong, 2017), and reducing communication cost (Reisizadeh et al., 2020; Dai et al., 2019). These methods aim to learn a global model across clients, which limits their ability to deal with heterogeneous clients.

Personalized federated learning (PFL) aims to handle data heterogeneity across clients. Following (Tan et al., 2021), we divide PFL methods into data-based and model-based approaches.

Data-based PFL approaches aim to smooth the statistical heterogeneity of data among different clients. This can be achieved by normalizing the data (Duan et al., 2021), or by using a client selection mechanisms that enable sampling from a more homogeneous data distribution. Wang et al. (2020b) selects a subset of participating clients for each training round, with the objective of maximizing accuracy while minimizing the number of communication rounds. Yang et al. (2020) selects the subset of clients with minimal class imbalance.

Model-based PFL approaches aim to enable FL models to adapt to the diverse data distributions among clients. Jiang et al. (2019) and Fallah et al. (2020) use meta-learning to find a global model that is optimized using the client local data. Achituve et al. (2021) proposes learning a single kernel function shared by all clients, but use a personalized Gaussian processes classifier for each client. Bui et al. (2019) and Liang et al. (2020b) achieve personalization by using a global model, and a local model on top of it. The local model does not share parameters with the server. Huang et al. (2021) regularized stronger collaboration amongst clients with similar data distributions. In (Shamsian et al., 2021) for each client the server saves an embedding vector, and a hypernetwork is used to produce a personalized model from the embedding. Both the embedding vectors and the hypernetwork train using the clients labeled data. Shamsian et al. (2021) also suggest creating a personalized model for a novel labeled client, by freezing the hypernetwork, and fine-tune an embedding vector using the client labeled data. This is different from our setup, where the new client do not have labels.

Adapting a model to a new distribution at inference time was also considered as a variant of domain adaptation (DA) (Wang et al., 2020a; Kim et al., 2021; Liang et al., 2020a).
Our new client setup can be seen as an adaptation of this setup to the federated setup.

**Differential privacy (DP).** The goal of DP is to share information about a dataset while withholding information about individuals in the dataset. Although privacy is a key motivations of PFL, it has been shown that some private information is exposed in the process (Mothukuri et al., 2021). (Melis et al., 2019) shows that an adversarial participant can infer the presence of exact data points in others training data. (Hitaj et al., 2017) shows that an adversarial participant can generate other clients private data. Several works utilizes DP to protect the client privacy (Xie et al., 2018; Augenstein et al., 2019; Zhu & Philip, 2019; Hao et al., 2019; Truex et al., 2019). However, adding noise to the embedding of the clients may harm the model performance. Unlike these papers, we focus on the privacy of the novel client, so we do not harm the training process, and each novel client can choose its privacy-accuracy trade off in real-time.

### 3. The Learning Setup

We now formally define the learning setup of ITU-PFL.

Following the notation in (Baxter, 2000) we define $X$ to be an input space and $Y$ an output space. $P$ is a probability distribution over the data $X \times Y$. Let $l$ be a loss function $l : Y \times Y \rightarrow R$. $H$ is a set of hypotheses with $h : X \rightarrow Y$. The error of a hypothesis $h$ over a distribution $P$ is $err_P(h) = \int_{X \times Y} l(h(x), y) dP(x, y)$.

In ITU-PFL, we are given a federation of $N$ clients $c_1, \ldots, c_N$ for training, and other, novel, clients added at inference time. For simplicity of notation, we consider a single novel client $c_{new}$. Let $\{P_i\}_{i=1}^N$ be the data distributions of training clients, and $P_{new}$ the data distribution for $c_{new}$. Each training client has access to $m_i$ IID samples from its distribution $P_i$, $S_i = \{(x_{ij}^i, y_{ij}^i)\}_{j=1}^{m_i}$.

The goal of ITU-PFL is to use data from training clients $\{S_i\}_{i=1}^N$ to learn a mechanism that can assign a hypothesis $h_{new} \in H$ when given unlabeled data from a novel client $S_{new} = \{x_{j}^{new}\}_{j=1}^m$. That hypothesis should minimize the error on the novel client. For any distribution of its data $P_{new}$, that error is simply $err_{P_{new}}(h_{new}) = \int_{X \times Y} l(h_{new}(x), y) dP_{new}(x, y)$.

### 4. Our approach

In ITU-PFL, we are evaluated by the quality of inference on a unlabeled novel client, one that was never seen during training, and has its own data distribution. Fundamentally, this problem is not about producing a single trained model, but about producing a ”meta mechanism” that can provide a model ”on-demand”, given (a descriptor of) a novel client. A natural candidate for such a meta-mechanism, are hypernetworks (HNs). HNs are neural networks that output the weights of another network, conditioned on some input, and can therefore produce ”on-demand” models. Since the weights of the generated model are a (differentiable) function of the HN parameters, training the HN is achieved simply by propagating gradients from the generated (client) model. HNs have already been shown to be effective for PFL by (Shamsian et al., 2021).

To generate a model for a novel client, the HN should be fed with a proper descriptor that summarizes the client dataset. We propose to learn a *client encoder* that takes as input the data of the novel client and produces a dense descriptor.

**Client encoding.** The HN expects to receive as input a descriptor of a client. The client encoder module takes unlabeled data and produces a dense descriptor. Consider a representation space $\mathcal{E}$ of clients, we aim to obtain a representation $e_i \in \mathcal{E}$ for client $i$.

The architecture of the encoder should fit the data it handles. We consider three different architectures for the encoder. First, note that the client data is a set without any particular order. This property of invariant to permutations can be handled by applying a DeepSet architecture (Zaheer et al., 2017). In this architecture, each data point is fed to the same model and produces a feature vector. Then, an invariant pooling operation (usually mean) is applied to all outputs. Finally, another model is applied to that output to produce the final descriptor.

Although DeepSet is invariant to input permutations, it does not take into account the symmetries of each element itself. For example, for images, the encoder should be invariant to translation. To handle both set-level and element-level symmetries we used Deep Sets for Symmetric elements layers (DSS) (Maron et al., 2020).
Figure 2. Training and inference workflow. (a) In training, for each client \(i\): (1) The server sends the client encoder \(g_y\) to the client. (2) The client computes locally its embedding vector \(e_i\) and sends it to the server. (3) The server feeds the embedding vector to the hypernetwork \(f_\theta\), and sends the client the personalized model weights \(f_\theta(e_i)\). (4) The client uses its labeled data to train locally the personalized model. Then, the client sends the weights difference \(\Delta \theta\) back to the server. The server updates the hypernetwork using the validation set. (b) For inference on novel client: (1) The server sends the encoder \(g_y\) to the client. (2) The client computes locally its descriptor \(e_{\text{new}}\) and sends it to the server. (3) The server uses the hypernetwork \(f_\theta\) to produce a personalized model from the descriptor, and sends the model to the client.

Both DeepSet and DSS use uniform pooling to aggregate the information from all elements in the set. In practice, some elements (samples) may be more important when determining a client descriptor. In addition, in some cases one may want to consider several elements together when computing the descriptor. For example, the descriptor could take into account the distribution of samples and treat differently rare and typical samples. To model those interactions we used set transformer (ST) (Lee et al., 2019). ST uses attention to aggregate all representations of the elements to one description.

We treated the architecture as a hyper parameter and selected it using the validation set.

**Hypernetworks.** Assume first that there already exists a representation space \(\mathcal{E}\) of clients, and a client \(i\) has a representation \(e_i \in \mathcal{E}\). This representation is fed as input to the HN, which then produces a personalized model for the client. We explain below how the representation is learned.

A hypernetwork \(f_\theta\) parametrized by \(\theta\) embodies a mapping from client-embedding space to hypotheses (model) space \(f_\theta : \mathcal{E} \rightarrow \mathcal{H}\). Any client with an embedding vector \(e_i\) is mapped by the hypernetwork to a personalized model \(h_i = f_\theta(e_i)\).

**Training.** The hypernetwork learns to produce a personalized model from a client descriptor by optimizing

\[
L_\mathcal{H}(\theta, e_1, ..., e_N) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} l(f_\theta(e_i)(x_j^i), y_j^i),
\]

using training (labeled) clients \(c_1, \ldots, c_N\). When training the client encoder, batches from the same client yield different representations, and we found that this variability hurts training. To alleviate this issue, we first learned an embedding of each training client, namely, a mapping from a client identity \(i\) to a dense descriptor \(e_i\). This embedding layer was trained in a standard way jointly with the hypernetwork. Then, we trained the client encoder, as described above, to minimize \(L_2\) distance between \(e_i\) and the predicted descriptor using batches from \(c_i\). More details in App. B.

**Flow.** Figure 2 illustrates the workflow of ITUFL-HN. During training (2a), the hypernetwork \(f_\theta\) optimizes \(L_\mathcal{H}\) and the encoder \(g_y\) optimizes \(L_{\text{encoder}} = \sum_{i=1}^{n} L_2(g_y((x_j^i)_{j=1}^m_i), e_i)\). Specifically, in each communication step, we repeat those 4 steps: (1) The server selects a random client and sends it to the current encoder \(g_y\). (2) The client locally predicts its embedding \(e_i\) and sends it back to the server. (3) Using its embedding, the server uses hypernetwork to generate a customized network \(h_i\) and communicates it to the client. (4) The client then locally trains that network on its data and communicates the delta between the weights before and after the training back to the server. Using the chain rule, the server can train the hypernetwork and the encoder.

At inference time (2b), the server sends the encoder to the novel client. The client uses the encoder to calculate an embedding \(e_{\text{new}}\) and sends it to the server. The server uses the hypernetwork to predict the client personalized model \(h_{\text{new}}\) from the embedding and sends the result to the client. The client then applies its personalized model locally without revealing its data.

5. **Generalization bound**

The data of the new client and the clients in the federation may come from different distributions. In the most general case, there is no guarantee that learning a model for the labeled clients leads to a good model for the novel client. We now show that previous bounds developed for multi-task learning (MTL) and for domain adaptation (DA), can be applied to the ITU-PFL setup, and bound the generalization error for the novel client. Intuitively, the bound has two terms, one captures the domain shift, and a second the usual generalization error.

**Theorem 5.1.** Let \(\mathcal{H}\) be hypotheses space, \(P_{\text{new}}\) be the data distribution of a novel client, and \(Q\) be a distribution over client data distributions, namely \(P_i\) is drawn from \(Q\).

The generalization error of the novel client
is bound by $err_{P_{\text{new}}}(H) \leq \epsilon + \frac{1}{2} \int_{P} \inf_{h \in H} d_{H}(H(P), P_{\text{new}}) dQ(P)$. Here, $\epsilon$ is the approximation error of a client in the federation from Theorem 2 in (Baxter, 2000) and $d_{H}(H)$ is the distance measure between probability distributions defined in (Ben-David et al., 2010). The error $err(H)$ and the empirical error $\hat{err}(H)$ are defined in detail in appendix A.

Proof: See appendix A for the detailed proof.

6. Experiments

We evaluate ITUFL-HN using five benchmark datasets.

6.1. Experiment setup and evaluation protocol.

We evaluate ITUFL-HN on the proposed ITU-PFL setup where novel clients are presented to the server at inference time.

Client split: To quantify performances on novel clients, we first randomly partition clients to $N_{\text{train}}$ training clients and $N_{\text{novel}}$ novel clients. We used $N_{\text{novel}} = N/10$. Unless stated otherwise, we report the average accuracy over novel clients: $\frac{1}{N_{\text{novel}}} \sum_{i=1}^{N_{\text{novel}}} \frac{1}{m_i} \sum_{j} m_i \cdot \text{Acc}(\psi_i(x_j^i), y_j^i)$. Where $\psi_i$ is the model chosen by the server to evaluate novel client $i$ with $m_i$ samples. To conduct a fair comparison, training is limited to 500 steps for all evaluated methods. In each step, the server communicates with a 0.1 fraction of training clients following the protocol of each method. Specific model architectures are described for each experiment separately.

Sample split and HP tuning: We split the samples of each training client into a training set and validation set. Validation samples were used for hyperparameter tuning for all methods and datasets. Specifically, we tuned training batch size, learning rate of each method, weight-decay values and the number of inner training epochs, and an early stopping point. Results are reported using the best hyperparameters for each experiment. See appendix C for more details.

6.2. Baselines

We evaluate and compare the following FL methods: (1) **FedAVG** (McMahan et al., 2017a), perhaps the most widely used FL algorithm. Parameters of local models are averaged element-wise with weights proportional to sizes of the client datasets. (2) **FedProx** (Sahu et al., 2018a) adds a proximal term to the client cost functions, thereby limiting the impact of local updates by keeping them close to the global model. (3) **FedMA** (Wang et al., 2020c) constructs the shared global model in a layer-wise manner by matching and averaging hidden elements with similar feature extraction signatures. Since in FL there is one global model, all novel clients are evaluated using this single global model. For PFL methods the comparison is less straightforward. PFL methods produce a separate models per training client, but does not have a way to generalize to a novel unlabeled client. For unlabeled client we can only approximate the novel client model with one of the training client models. We devised three ways to use the training models for inference with a novel client, and tested them using (4) **pFedHN** (Shamsian et al., 2021).

(4a) **PFL-sampled**: Draw a trained client model uniformly at random. We evaluate this baseline by computing the mean accuracy of all personalized models on each novel client.

(4b) **PFL-Ensemble**: A stronger baseline is achieved by taking into account all personalized models when predicting for a single novel client. This is achieved by averaging the logits of all models for each prediction. In practice, this method requires sending multiple models to each novel client. Therefore, it is expensive in both communication and computation costs.

(4c) **PFL-nearest**: We measured the distance between a novel client and all trained clients, and use the model of the closest training client. We measure the distance using A-distance (Ben-David et al., 2007), which measures how hard it is to separate data points from two different clients using a linear model. Since that model can get gradients from each client separately, it can be used in the FL setup.

6.3. Results on CIFAR data

We first evaluate ITUFL-HN using CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009). We follow two existing protocols to split the data across clients.

(1) **Pathological split**: As proposed by (McMahan et al., 2017a), we sort the training samples by their labels and partition them into $N \cdot K$ shards. Each client is then randomly assigned $K$ of the shards. This results in $N$ clients with the same number of training samples and a different distribution over labels. In our experiments, we use $N = 100$ clients, $K = 2$ for CIFAR10 and $K = 5$ for CIFAR100.

(2) **Dirichlet allocation**: We follow the procedure by (Hsu et al., 2019a) to control the magnitude of distribution shift between clients. For each client $i$, samples are drawn independently with class labels following a categorical distribution over classes with a parameter $q_i \sim Dir(\alpha)$. Here, $Dir$ is the symmetric Dirichlet distribution and $\alpha$ is the concentration parameter. We conduct three experiments for each of the two datasets with $\alpha \in \{0.1, 1, 10\}$. Smaller values of alpha imply larger distribution shifts between clients.

Implementation details: There are three different mod-
We evaluate ITUFL-HN using two of the suggested geometries. A fully connected neural network, with three hidden layers followed by Global-Max and Global-Mean operations over batch samples after the first layer. The output dimension is the size of the embedding dimension rather than the number of labels. Hypernetwork: The hypernetwork is a fully connected neural network, with three hidden layers and multiple linear heads per target weight tensor.

Results: Table 1 compares ITUFL-HN to FL and PFL baselines on CIFAR-10 and CIFAR-100. ITUFL-HN performs better than all baselines in all evaluated datasets and split, except for CIFAR-10 $\alpha = 10$ split.

### 6.4. iNaturalist

iNaturalist is a dataset for Natural Species Classification based on the iNaturalist 2017 Challenge (Horn et al., 2018). The dataset has 1,203 classes. Following Hsu et al. (2020), we evaluate ITUFL-HN using two of the suggested geographical splits of iNaturalist: iNaturalist-Geo-1k with 368 clients, and iNaturalist-Geo-300 with 1,208 clients.

Implementation details: We use a LeNet-style network (LeCun et al., 1998) with two convolutions and two max-pooling operations over batch samples after the first layer. The client encoder has three fully connected layers with Global-Max and Global-Mean operations over batch samples after the first layer. The target model is a simple fully connected network with two max-pooling operations over batch samples after the first layer. The output dimension is the size of the embedding dimension.

### 6.5. Landmarks

Landmarks is a dataset for landmark recognition based on the 2019 Landmark-Recognition Challenge (Weyand et al., 2020). The dataset has 2,028 classes. Following Hsu et al. (2020), we split the dataset into clients by authorship. The resulting Landmarks-User-160k split contains 1,262 clients. Implementation is as in Sec. 6.4.

### 6.6. Yahoo Answers

Yahoo Answers is a question-classification dataset created by Zhang et al. (2015). The dataset contains 1.4 million train samples from 10 classes. We split the data to 1000 clients using Dirichlet allocation procedure with $\alpha = 10$.

Implementation details: We used BERT (Devlin et al., 2019) to extract a 768 dimensions feature vector for each sample. All other details are as in Sec. 6.4.

### 6.7. The effect of distribution shift on generalization

When ITUFL-HN generalizes to a new client, we expect the quality of generalization to depend on how similar the novel client is to the training clients. Clients that differ significantly from the training clients may perform poorly compared to clients that are similar to training clients.

To quantify this effect, we generated clients at varying similarity levels by creating new splits of CIFAR-100 using Dirichlet allocation as in Sec. 6.3 while varying $\alpha \in \{0.1, 0.25, 0.5, 1, 10\}$. To measure the similarity between a novel client and all training clients, we first compute the empirical label distributions of each client. Then, we computed their KL-divergence, and took $D_{KL}$ between the novel client and the nearest trained client.

---

Table 1. Accuracy on novel unlabeled clients, CIFAR10 and CIFAR100: Values are averages and standard error across clients.

| split       | CIFAR-10 | CIFAR-100 |
|-------------|----------|-----------|
|             | pathological | $\alpha = 0.1$ | $\alpha = 1$ | $\alpha = 10$ | pathological | $\alpha = 0.1$ | $\alpha = 1$ | $\alpha = 10$ |
| FedAvg      | 50.3 ± 2.9 | 58.7 ± 3.6 | 48.4 ± 2.9 | 66.2 ± 0.5 | 16.2 ± 1.3 | 17.9 ± 1.0 | 13.5 ± 1.7 | 30.2 ± 0.4 |
| FedProx     | 54.2 ± 2.0 | 53.9 ± 2.2 | 54.2 ± 1.0 | 52.8 ± 0.6 | 5.5 ± 1.2 | 15.9 ± 0.7 | 20.6 ± 0.6 | 12.4 ± 0.6 |
| FedMA       | 42.9 ± 1.8 | 49.3 ± 3.4 | 54.5 ± 0.9 | 53.8 ± 0.6 | 11.2 ± 0.7 | 12.6 ± 1.0 | 6.5 ± 0.4 | 7.3 ± 0.3 |
| PFL-sampled | 24.8 ± 1.0 | 61.0 ± 3.7 | 49.4 ± 3.0 | 68.5 ± 0.7 | 3.9 ± 0.4 | 13.3 ± 0.5 | 3.4 ± 1.4 | 32.4 ± 0.1 |
| PFL-ensemble| 47.6 ± 3.2 | 62.2 ± 3.7 | 49.4 ± 3.0 | 68.5 ± 0.7 | 7.8 ± 1.8 | 20.4 ± 1.2 | 3.4 ± 1.4 | 32.7 ± 0.2 |
| PFL-nearest | 24.4 ± 6.2 | 63.1 ± 3.5 | 49.4 ± 0.9 | 68.5 ± 0.7 | 6.5 ± 2.7 | 14.3 ± 0.6 | 3.4 ± 0.4 | 32.1 ± 0.2 |
| ITUFL-HN (ours) | 59.5 ± 3.5 | 66.0 ± 3.0 | 62.9 ± 1.0 | 68.1 ± 0.5 | 19.5 ± 2.1 | 26.4 ± 0.1 | 32.9 ± 0.9 | 33.6 ± 0.1 |
Table 2. Accuracy on novel unlabeled clients for iNaturalist, Landmarks and Yahoo Answers. Values are averages and SEMs across all novel clients.

| Split          | iNaturalist | Landmarks | Yahoo Answers |
|----------------|-------------|-----------|---------------|
|                | Geo-300     | Geo-1k    | User-160k     | User-1K       |
| FedAvg         | 36.1 ± 1.6  | 36.9 ± 1.1| 34.8 ± 1.3    | 27.6 ± 0.1   |
| FedProx        | 17.4 ± 0.8  | 26.5 ± 1.7| 13.8 ± 1.0    | 13.9 ± 0.1   |
| FedMA          | 13.4 ± 0.7  | 17.5 ± 0.8| 3.8 ± 0.4     | 25.0 ± 0.1   |
| PFL - sampled  | 25.6 ± 1.4  | 27.2 ± 0.9| 37.4 ± 1.3    | 33.2 ± 0.1   |
| PFL - ensemble | 31.5 ± 1.6  | 36.6 ± 1.2| 39.1 ± 1.4    | 33.2 ± 0.1   |
| PFL - nearest  | 24.2 ± 3.7  | 26.9 ± 4.6| 33.1 ± 3.6    | 13.2 ± 0.1   |

ITUFL-HN (ours) 37.5 ± 1.7 41.6 ± 1.2 41.1 ± 1.4 35.8 ± 0.2

Figure 3. Accuracy of novel clients vs. distribution shift between the novel client and training clients. Shown results for CIFAR-100 across multiple splits to $N_{train} = 90$ train clients and $N_{novel} = 10$ novel clients using symmetric Dirichlet distributions with varying parameter $\alpha$ (see Sec. 6.7). Accuracies of novel clients are reported against the KL-divergence (over label distribution) from the nearest train client for each method.

6.8. Robustness to covariate shift

The domain shift of a novel client is often caused by using different sensors or different environments. We evaluated the robustness of ITUFL-HN to five common data corruptions, blur, rotation, brightness, contrast, and saturation, using CIFAR-10 data. Data from training clients were kept without corrupted.

For blur, we used a $7 \times 7$ Gaussian filter, with $\sigma$ chosen uniformly at random in $(0.1, 2.0)$. For brightness, contrast, and saturation, we use a factor that is chosen uniformly at random in $(0.5, 1.5)$. The rotation transformation used an angle chosen uniformly at random in $(0, 1.5)$.

Table 3 summarizes the results. ITUFL-HN outperforms the evaluated FL and PFL baselines when evaluated on corrupted novel clients.

7. Differential Privacy

A key aspect of federated learning is data privacy, restricting clients from sharing their data directly with the hub. Unfortunately, since data is used to train the federated model, and the federated model produces models for other clients, some private information may be exposed (see a recent survey by Mothukuri et al., 2021). In this section, we analyze theoretically and experimentally the privacy of a novel client and characterize how it can protect its privacy by applying differential privacy (DP) (Dwork et al., 2006b). We further study the trade-off between the privacy of a novel client and the accuracy of its personalized model.

We first define key concepts and our notation. We use $(\epsilon, \delta)$-differential privacy as defined by Dwork et al. (2006a). We say that two datasets $D, D'$ are adjacent if they differ in a single instance.

Definition 7.1. A randomization mechanism $M : D \rightarrow R$ satisfies a $(\epsilon, \delta)$-differential privacy if for any two adjacent inputs $d, d' \in D$ and for any subset of outputs $S \subseteq R$ it
We now evaluate empirically the effect of adding Gaussian additive noise to the embedding of a novel client. To meet the client can reach better performance. For a given \((\mid \Delta \mid)\) sensitivity of the encoder is bounded by \(B\). Hence bound the noise magnitude necessary to achieve privacy. The following lemma shows that when using a deep-set encoder, we can bound the sensitivity of the encoder, hence bound the noise magnitude necessary to achieve privacy. See proof in appendix D.

**Lemma 7.2.** Let \(g\) be a deep-set encoder, written as: \(g(D) = \psi(\sum_{x \in D} \phi(x))\). If \(\psi\) is a linear function with Lipschitz constant \(L_\phi\), and \(\phi\) is bounded by \(B_\phi\), then the sensitivity of the encoder is bounded by \(\Delta g \leq \frac{1}{|D|} L_\phi B_\phi\).

The lemma shows that the sensitivity of the encoder decreases linearly with the size \(|D|\) of the novel client dataset. For a given \((\epsilon, \delta)\), lower sensitivity let us use lower Gaussian noise to achieve the desired privacy. This in turn means the client can reach better performance.

We now evaluate empirically the effect of adding Gaussian additive noise to the embedding of a novel client. To meet the conditions in lemma 7.2, we normalized the output of \(\phi\) to be on a unit sphere, so \(B_\phi = 1\). In addition, we average the output of \(\phi\), so \(L_\phi = 1\). We used \(\delta = 0.01\) and compare different values of \(\epsilon\) and dataset size.

Figure 4 shows that with sufficient data, a novel client can protect its privacy without compromising the performance of the personalized model. For example, assume the desired privacy is \(\epsilon = 0.3\). If the client feeds the DP-encoder with 3000 samples (or more), the HN creates a personalized model that is as accurate as a non-DP model.

**8. Conclusion**

This paper describes ITU-PFL, a new real-world federated setup, where a model trained in a federated learning manner, transfers to novel clients that were not available during training, and do not even have labeled data.

We describe ITUFL-HN, a novel approach to ITU-PFL, based on learning an encoder that learns a space of clients, and a hypernetwork that maps a client to its corresponding model in an "on-demand" way. We evaluated ITUFL-HN on five benchmark datasets, showing that it usually generalizes better than current FL and modified PFL methods. We also analyze and bound the generalization error for the novel client and analyze applying differential privacy for the novel client. We hope this paper will encourage the research community to consider generalization to novel clients when designing FL methods.

**References**

Achituve, I., Shamsian, A., Navon, A., Chechik, G., and Fetaya, E. Personalized federated learning with gaussian processes. 2021.
Agarwal, N., Suresh, A. T., Yu, F., Kumar, S., and Memahan, H. B. cpsgd: Communication-efficient and differentially-private distributed sgd. arXiv preprint arXiv:1805.10559, 2018.

Augenstein, S., McMahan, H. B., Ramage, D., Ramaswamy, S., Kairouz, P., Chen, M., Mathews, R., et al. Generative models for effective ml on private, decentralized datasets. arXiv preprint arXiv:1911.06679, 2019.

Baxter, J. A model of inductive bias learning. Journal of artificial intelligence research, 12:149–198, 2000.

Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al. Analysis of representations for domain adaptation. Advances in neural information processing systems, 19:137, 2007.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. Machine learning, 79(1-2):151–175, 2010.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. Learnability and the vapnik-chervonenkis dimension. Journal of the ACM (JACM), 36(4):929–965, 1989.

Bui, D., Malik, K., Goetz, J., Liu, H., Moon, S., Kumar, A., and Shin, K. G. Federated user representation learning. arXiv preprint arXiv:1909.12535, 2019.

Dai, X., Zhou, K., Yang, H., Ng, K. K., Cheng, J., and Fan, Y. Hyper-sphere quantization: Communication-efficient sgd for federated learning. arXiv preprint arXiv:1911.04655, 2019.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Duan, M., Liu, D., Chen, X., Liu, R., Tan, Y., and Liang, L. Self-balancing federated learning with global imbalanced data in mobile systems. IEEE Transactions on Parallel and Distributed Systems, 32(1):59–71, 2021.

Duchi, J. C., Jordan, M. I., and Wainwright, M. J. Privacy aware learning. Journal of the ACM (JACM), 61(6):1–57, 2014.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 486–503. Springer, 2006a.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In Theory of cryptography conference, pp. 265–284. Springer, 2006b.

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

Fan, C. and Huang, J. Federated few-shot learning with adversarial learning. arXiv preprint arXiv:2104.00365, 2021.

Haddadpour, F. and Mahdavi, M. On the convergence of local descent methods in federated learning. arXiv preprint arXiv:1910.14425, 2019.

Hanzely, F. and Richtárik, P. Federated learning of a mixture of global and local models. arXiv preprint arXiv:2002.05516, 2020.

Hao, M., Li, H., Xu, G., Liu, S., and Yang, H. Towards efficient and privacy-preserving federated deep learning. In ICC 2019-2019 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE, 2019.

Hitaj, B., Ateniese, G., and Perez-Cruz, F. Deep models under the gan: information leakage from collaborative deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 603–618, 2017.

Horn, G. V., Aodha, O. M., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. J. The inaturalist species classification and detection dataset. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8769–8778, 2018.

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

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

Hsu, T.-M. H., Qi, H., and Brown, M. Federated visual classification with real-world data distribution. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16, pp. 76–92. Springer, 2020.
Huang, Y., Chu, L., Zhou, Z., Wang, L., Liu, J., Pei, J., and Zhang, Y. Personalized cross-silo federated learning on non-iid data. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pp. 7865–7873, 2021.

Jiang, Y., Konečný, J., Rush, K., and Kannan, S. Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv: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 International Conference on Machine Learning, pp. 5132–5143. PMLR, 2020.

Kim, Y., Cho, D., Han, K., Panda, P., and Hong, S. Domain adaptation without source data. IEEE Transactions on Artificial Intelligence, 2021.

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

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In International Conference on Machine Learning, pp. 3744–3753. PMLR, 2019.

Liang, J., Hu, D., and Feng, J. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In International Conference on Machine Learning, pp. 6028–6039. PMLR, 2020a.

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

Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don’t use large mini-batches, use local sgd. arXiv preprint arXiv:1808.07217, 2018.

Maron, H., Litany, O., Chechik, G., and Fetaya, E. On learning sets of symmetric elements. In International Conference on Machine Learning, pp. 6734–6744. PMLR, 2020.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics, pp. 1273–1282. PMLR, 2017a.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. arXiv preprint arXiv:1710.06963, 2017b.

Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In 2019 IEEE Symposium on Security and Privacy (SP), pp. 691–706. IEEE, 2019.

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G. A survey on security and privacy of federated learning. Future Generation Computer Systems, 115:619–640, 2021.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In International Conference on Artificial Intelligence and Statistics, pp. 2021–2031. PMLR, 2020.

Sahu, A. K., Li, T., Sanjabi, M., Zaheer, M., Talwalkar, A., and Smith, V. On the convergence of federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127, 3:3, 2018a.

Sahu, A. K., Li, T., Sanjabi, M., Zaheer, M., Talwalkar, A., and Smith, V. On the convergence of federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127, 3, 2018b.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510–4520, 2018.

Shamsian, A., Navon, A., Fetaya, E., and Chechik, G. Personalized federated learning using hypernetworks. arXiv preprint arXiv:2103.04628, 2021.

Stich, S. U. Local sgd converges fast and communicates little. arXiv preprint arXiv:1805.09767, 2018.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. Towards personalized federated learning. arXiv preprint arXiv:2103.00710, 2021.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. A hybrid approach to privacy-preserving federated learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, pp. 1–11, 2019.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. arXiv preprint arXiv:2006.10726, 2020a.

Wang, H., Kaplan, Z., Niu, D., and Li, B. Optimizing federated learning on non-iid data with reinforcement learning. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 1698–1707. IEEE, 2020b.
A. Generalization Bound

**Proof.** The main idea of the proof is to first bound the error of all (labeled) training clients, using results from multi-task learning. Then, treat the novel client as a target domain in a domain adaptation problem, and bound its shift from training clients.

The error of the novel client for a given hypothesis space $H$ is $\inf_{h \in H} err_{P_{\text{new}}}(h)$. Since $P_{\text{new}}$ is independent of $Q$, we can integrate over all $P \sim Q$ and obtain

$$err_{P_{\text{new}}}(H) := \inf_{h \in H} err_{P_{\text{new}}}(h)$$

$$= \int_{P \sim Q} \inf_{h \in H} err_{P_{\text{new}}}(h) dQ(P). \quad (2)$$

Using Theorem 2 from (Ben-David et al., 2010) with $P_{\text{new}}$ treated as the target domain and $P$ as the source domain, gives that $\forall h, \forall P : err_{P_{\text{new}}}(h) \leq err_{P}(h) + \frac{1}{2} \hat{d}_{H}(P, P_{\text{new}})$. Plugging into Eq. (2) gives

$$err_{P_{\text{new}}}(H) \leq \int_{P \sim Q} \inf_{h \in H} \left[ err_{P}(h) + \frac{1}{2} \hat{d}_{H}(P, P_{\text{new}}) \right] dQ(P) \quad (3)$$

$$= err_{Q}(H) + \frac{1}{2} \int_{P \sim Q} \inf_{h \in H} \hat{d}_{H}(P, P_{\text{new}}) dQ(P).$$

Since $err_{Q}(H)$ is unknown, we use Theorem 2 from (Baxter, 2000) to bound the error of the novel client. That yields the bound in the theorem.

We now explain that in details. If we have only one task and one domain, the most common solution is to find $h \in \mathcal{H}$ that minimizes the loss function on a training set sampled from probability distribution $P$. In general, $\mathcal{H}$ is a hyperparameter, defined by the network architecture. This (Blumer et al., 1989) shows that in this simple case, the generalization error is bounded. The bound depends upon the “richness” of $H$. Choosing a “rich” $H$ (with large VC-dimension), let the generalization error be larger.

In ITU-PFL, each client may use a different hypothesis. Instead of bounding the error of one chosen hypothesis, we bound the error of the chosen hypotheses space that each client chooses from. This way, we can bound the error of a novel client, without assuming anything about the way it chose from the hypothesis space.

First, we find $H$ that contains hypotheses that can fit all data of the clients. Second, for each client, we select the best hypothesis $h \in H$ according to the client data. We define $Q$ as a distribution over $P$, so, each client sample from $Q$ the distribution $P_{t}$. We further define $\mathbb{H}$ as a hypothesis space family, where each $H \in \mathbb{H}$ is a set of functions $h : X \rightarrow Y$.

The first goal is to find a hypothesis space $H \in \mathbb{H}$ that minimizes the weighted error of all clients, assuming each
client uses the best hypothesis $h \in H$. We define this error using the following loss:

$$err_Q(H) := \int_P \inf_{h \in H} err_P(h)dQ(P)$$  

while $err_P(h) := \int_{X \times Y} l(h(x), y)dP(x, y)$. In practice $Q$ in unknown, so we can only estimate $err_Q(H)$ using the sampled clients and their data.

For each client $i = 1..n$, we sample the client training data from $X \times Y \sim P_i$. We denote the sampled training set with $z_i := (x_1, y_1), ..., (x_m, y_m)$, and $z = z_1, ..., z_n$. The empirical error of a specific hypothesis is defined by $\hat{err}_z(h) := \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i)$. Now the empirical loss to minimize is

$$\hat{err}_z(H) := \frac{1}{n} \sum_{i=1}^{n} \inf_{h \in H} \hat{err}_{z_i}(h)$$

(Baxter, 2000) shows that if the number of clients $n$ satisfies $n \geq \max(2^{32} \rho^2 \log(\frac{8C(H^r)}{\delta}), \frac{4}{\rho^2})$, and the number of samples per client $m$ satisfies $m \geq \max(2^{32} \rho^2 \log(\frac{8C(H^r)}{\delta}), \frac{4}{\rho^2})$, then with probability $1 - \delta$ all $H \in \mathcal{H}$ satisfies

$$err_Q(H) \leq \hat{err}_z(H) + \epsilon$$  

were $C(H^r)$ and $C(H^r)$ are the covering numbers defined in (Baxter, 2000), and can be referred as a way to measure the complexity of $H$. Note that a very “rich” $H$ makes $\hat{err}_z(H)$ small, but it increases the covering number, so for the same amount of data, $\epsilon$ increases.

For the PFL setup, this is enough, since we can ensure that for a client that sampled from $Q$ and was a part in the federation, the chosen hypothesis $h \in H$ has an error close to the empirical one $\hat{err}_z(H)$. For a novel client, this may not be the case. The novel client may sample from a different distribution over $P$. In the general case, the novel client may even have a different distribution over $X \times Y$. In the most general case, the error on the novel client can not be bound. In DA, a common distribution shift is a covariate shift, where $P(x)$ may change, but $P(y|x)$ remains constant. This assumption lets us bound the error of the novel client.

Ben-David et al. (2010) proofed that for a given $H \in \mathcal{H}$, if $S$ and $T$ are two datasets with $m$ samples, then with probability $1 - \delta$, for every hypothesis $h \in H$:

$$err_T(h) \leq err_S(h) + \frac{1}{2} d_{H\Delta H}(S, T)$$

$$+ 4 \sqrt{\frac{2d \log(2m) + \log(\frac{2}{\delta})}{m}} + \lambda$$

where $err_D(h) = E_{(x, y) \sim D}[|h(x) - y|]$ is the error of the hypothesis on probability distribution of the domain $D$. $d_{H\Delta H}(S, T)$ is a distance measure between the domains $S$ and $T$, and $\lambda = \arg \max_{h \in H} err_T(h) + err_S(h)$. Note that for over-parametrized models like deep neural networks $\lambda$ should be very small. To keep the analysis shorter we assume this is the case. We further assumed $m$ is big enough to neglect $4 \sqrt{\frac{2d \log(2m) + \log(\frac{2}{\delta})}{m}}$. Those assumptions are not mandatory, and the following analysis can be done without it.

**B. More details about training**

In ITUFL-HN two main components are trained using a federation of labeled clients: A hypernetwork and a client encoder. The hypernetwork optimizes the $L_{HN}$ loss defined in 8 by updating both its own weights $\theta$ and the clients representations $\{e_i\}_{i=1}^N$.

$$L_{HN}(\theta, e_1, ..., e_N) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} l(f_0(e_i)(x_j^i), y_j^i)$$  

The client encoder trains to predict the representations learned by the hypernetwork from the client raw data by minimizing $L_{encoder}$ defined in 9. At inference time, a novel client feeds its data to the client encoder and gets an embedding vector. Then, feeding the embedding vector to the hypernetwork produces a custom model for the client.

$$L_{encoder} = \sum_{i=1}^{n} L_2(g_\theta(\{x_j^i\}_{j=1}^{m_i}), e_i)$$

In detail, in each communication step, we repeat those 4 steps: (1) The server selects a random client and sends it to the current encoder. (2) The client locally predicts its embedding and sends it back to the server. (3) Using its embedding, the server uses hypernetwork to generate a customized network and communicates it to the client. (4) The client then locally trains that network on its data and communicates the delta between the weights before and after the training back to the server. Using the chain rule, the server can train the hypernetwork and the encoder (see Figure 2).

When training the client encoder, batches from the same client yield different representations, and we found that this variability hurts training. We design 2 approaches to alleviate this issue: (1) We calculate the embedding using all the client data, without breaking it into mini-batches. (2) We first learned an embedding of each training client, namely, a mapping from a client identity $i$ to a dense descriptor $e_i$. This embedding layer was trained in a standard way jointly with the hypernetwork. Then, we trained the client encoder. We tested both methods, and choose between them using
cross validation. Practically, for the iNaturalist experiments, the second method was better. For all other datasets, the first method was better. We now explain the second approach in detail.

In each communication step, the server selects a random client. Using its current embedding, the hypernetwork generates a customized network and communicates it to the client. The client then locally trains that network on its data for a predefined number of local epochs. As in (Shamsian et al., 2021), the client communicates the delta between the weights before and after the training back to the server. Using the chain rule, the server can train the hypernetwork and the embeddings to optimize $L_{HN}$ (see Figure 2).

In addition to the custom target model, the server sends the client the current encoder, and the current embedding of the client. Similar to the previous step, the client trains the encoder locally to predict the embedding from the client data by optimizing $L_{encoder}$, then, the updates of the encoder are sent back to the server for aggregation.

Up to this point, the client encoder trains in parallel to the hypernetwork, and has no influence on the hypernetwork weights or the embeddings of the labeled clients. We found that freezing the encoder and fine-tuning the hypernetwork using the trained encoder predictions improve the results of our method. This is done by optimizing the hypernetwork parameters $\theta$ using $L_{Fine-tune}$.

$$L_{Fine-tune}(\theta) = \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{m_i} l(f_{\theta}(g_{\gamma}(\{x_j^i\}_{j=m_i}^m))(x_j^i), y_j^i)$$

(10)

However, this fine-tune step reduces the performance of the labeled clients. Since our goal is to generalize well to novel clients, we ignored this effect. Note that in a real-world application, the server may save a version of the hypernetwork before fine-tuning it, and used it when generating models for the original federation.

C. Experimental Details

For all experiments presented in the main text, we use a fully-connected hypernetwork with 3 hidden layers of 100 hidden units each. The size of the embedding dim is $N_{tags}$. Experiments are limited to 500 communication steps. In each step, communication is done with $0.1 : N_{train}$.

Hyperparameter Tuning We divide the training samples of each train client into 85% / 15% train / validation sets. The validation sets are used for hyperparameter tuning and early stopping of all baselines and datasets. The searched hyperparameters and corresponding values by method: **FedAVG**: The local momentum $\mu_{local}$ is set to 0.5. We search over local learning-rate $\eta_{local} \in \{1e-1, 1.5e-2, 1e-2, 5e-3, 1e-3\}$, number of local epochs $K \in \{1, 2, 5, 10\}$ and batch size $\{16, 32, 64\}$. **FedProx and FedMA**: We used the hyperparameters used by Wang et al. (2020c) in the official code that provided by the authors. **pFedHN**: We set $\mu_{local} = 0.9$. We search over learning-rates of the hypernetwork, embedding layer and local training: $\eta_{hn}, \eta_{embedding}, \eta_{local} \in \{1e-1, 1.5e-2, 1e-2, 5e-3, 1e-3\}$, weight decays $wd_{hn}, wd_{embedding}, wd_{local} \in \{1e-3, 1e-4, 1e-5\}$, number of local epochs $K \in \{1, 2, 5, 10\}$ and batch size $\{32, 64\}$. **ITUFL-HN**: We perform the optimization using the same parameters and values as in pFedHN. In addition, we search over the learning-rate of the client encoder $\eta_{encoder} \in \{1e-1, 1.5e-2, 1e-2, 5e-3, 1e-3\}$.

CIFAR(Section 6.3) We use a LeNet-based target network with two convolution layers with 16 and 32 filters of size 5 respectively. Following these layers are two fully connected layers of sizes 120 and 84 that output logits vector. The client encoder follows the same architecture with an additional fully connected layer of size 200 followed by Mean-global-pooling for the first 100 units and Max-global-pooling for the other 100 units. Global-pooling is done over the samples of a batch.

INaturalist, Landmarks and Yahoo Answers Data(Sections 6.4-6.6) We use a simple fully-connected network with two Dense layers of size 500 each, followed by a Dropout layer with a dropout probability of 0.2. The client encoder is a fully-connected network with three Dense layers of size 500. The first layer is followed by Mean-global-pooling for the first 250 units and Max-global-pooling for the other 250 units.

D. Differential Privacy

Proof.

$$\Delta g := \max_{D,D'} \|g(D) - g(D')\|$$

$$= \max_{D,D'} \|\psi\left(\frac{1}{|D|} \sum_{x \in D} \phi(x)\right) - \psi\left(\frac{1}{|D'|} \sum_{x \in D'} \phi(x)\right)\|.$$  

(11)

Denote $d \in D$ and $d' \in D'$ as the only nonidentical instance between $D$ and $D'$, so $D/d = D'/d'$. Then

$$\Delta g = \max_{D,D'} \|\psi\left(\frac{1}{|D|} \sum_{x \in D/d} \phi(x) + \phi(d)\right)$$

$$- \psi\left(\frac{1}{|D'|} \sum_{x \in D'/d'} \phi(x) + \phi(d')\right)\|$$

$$= \max_{d,d'} \|\psi(\phi(d) - \phi(d'))\|.$$  

(12)
Figure 5. Test accuracy (±SEM) for CIFAR-10 novel client while applying DP. As $\epsilon$ decreases, we need more data to preserve the same accuracy of the model.

Assume that $\phi$ is bounded with $B_{\phi}$, so $|\phi(d) - \phi(d')| < 2B_{\phi}$. Then from the linearity of $\psi$:

$$\Delta g \leq \frac{1}{|D|} L_{\psi} |\phi(d) - \phi(d')| \leq \frac{2}{|D|} L_{\psi} B_{\phi}$$  \hspace{1cm} (13)

We also compare different privacy levels, using lower values of $\epsilon$. Figure 5 shows that more privacy (lower $\epsilon$), requires a bigger dataset to maintain the same accuracy of the model.

E. Embedding Space Visualization

To get more intuition for the way the client encoder captures similarities between clients, we visualized the client embedding space. Figure 6 shows the TSNE plot of the embedding space using a set transformer encoder. It is hard to visualize which client should be close to which one. In order to help capture some of those similarities, we sign each class with a different marker. We plot each client using the marker of its higher class. Figure 6 shows that some areas are related with some labels. For example, the area marked with an ellipse contains mostly clients who have 8 and 9 as their higher label. Note that those client have another class, so it is not expected to contain all clients with the label 8 or 9.

Figure 6. TSNE plot of CIFAR10 pathological split embedding space. We sign each class with a different marker, and plot each client using the marker of its higher class.