Federated Learning from Pre-Trained Models: A Contrastive Learning Approach

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated Learning (FL) is a machine learning paradigm that allows decentralized clients to learn collaboratively without sharing their private data. However, excessive computation and communication demands pose challenges to current FL frameworks, especially when training large-scale models. To prevent these issues from hindering the deployment of FL systems, we propose a lightweight framework where clients jointly learn to fuse the representations generated by multiple fixed pre-trained models rather than training a large-scale model from scratch. This leads us to a more practical FL problem by considering how to capture more client-specific and class-relevant information from the pre-trained models and jointly improve each client's ability to exploit those off-the-shelf models. In this work, we design a **Fed**erated **P**rototype-wise **C**ontrastive **L**earning (FedPCL) approach which shares knowledge across clients through their class prototypes and builds client-specific representations in a prototype-wise contrastive manner. Sharing prototypes rather than learnable model parameters allows each client to fuse the representations in a personalized way while keeping the shared knowledge in a compact form for efficient communication. We perform a thorough evaluation of the proposed FedPCL in the lightweight framework, measuring and visualizing its ability to fuse various pre-trained models on popular FL datasets.

1 Introduction

Federated Learning (FL) is a new promising field of machine learning that allows multiple clients to train together without sharing their private data [1]. Vanilla FL aims to train a single global model over all participating clients by periodically synchronizing their model parameters. However, the learned model usually does not perform well on all clients due to the statistical heterogeneity among local datasets [2]. Personalized Federated Learning (PFL) is proposed to solve this problem by training a personalized model for each client. Recent studies on PFL leverage various techniques to enable more common underlying information shared across different clients [3, 4, 5, 6]. So far, FL and PFL have been widely used in real-world tasks, including computer vision [7], natural language processing [8], and graph data mining [9].

However, the models in real-world applications are usually large-scale neural networks which incur *high computation costs* and require *high communication bandwidth* when trained from scratch. This can make it infeasible to train such models in some practical FL scenarios, e.g., low-resource device-based federated learning. To alleviate the above issues, we propose a lightweight FL framework that uses *multiple fixed pre-trained backbones* as the encoder, followed by learnable layers to fuse the representations generated by the backbones for each client. The proposed framework is capable of fusing the representations generated by pre-trained models with various architectures or obtained from various source data, expanding the scope of federated learning by integrating off-the-shelf foundation mod-

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

els. Also, it makes it possible to utilize large-scale pre-trained models, e.g., VisionTransformer [10] and Swin Transformer [11], in a computation resource-constrained case to enhance the overall performance. Using the pre-trained foundation models as the fixed encoder can efficiently reduce costs because neither complicated backward propagation computation nor large-scale neural network transmission between the server and clients is needed during the training stage. As shown in Table 1, compared to training a ResNet18 [12] from scratch using FedAvg [1], a much smaller number of parameters are communicated per round when learning to fuse the representations generated by fixed pre-trained models. Besides, higher performance can be achieved when using pre-trained models after the same communication rounds. Compared with using a single pre-trained model, using multiple models pre-trained on diverse datasets can provide a more comprehensive view for an input sample [13].

Table 1: Compared with training a single ResNet18 from scratch, less training time per round and fewer learnable parameters are needed when using multiple fixed ResNet18 pre-trained on different datasets [14]. Experiments are implemented with FedAvg [1] on Digit-5 dataset [15] under feature shift non-IID setting [16]. The number of communication rounds is 50.

Mode	Param.↓	Time↓	Acc↑
Training from scratch	11M	0.95s	31.75(3.07)
Using 1 pre-trained model	0.13M	0.31s	38.65(2.65)
Using 3 pre-trained models	0.40M	0.64s	40.47(3.05)

To enable a better personalized representation ability for each client under this lightweight FL framework, we need to select an appropriate information carrier to share common underlying knowledge across clients. Motivated by [17, 18, 19], class-wise prototypes, defined as "a representative embedding" for a specific class, can be an effective information carrier for communication between the server and clients. Sharing prototypes allows for better knowledge sharing across various learning domains, which has been proved in transfer learning [20] and multi-task learning [21] scenarios. To efficiently extract the useful shared information learned from the pre-trained models via prototypes, we design an algorithm called **Fed**erated **P**rototype-wise **C**ontrastive **L**earning (FedPCL) where both local prototypes and global prototypes are used for knowledge sharing in a supervised contrastive manner. By maximizing the agreement between the fused representation and its corresponding prototypes with contrastive learning, class-relevant information and semantically meaningful knowledge are captured by each client. Concretely, global prototypes force the fused representation to be closer to the global class center, while local prototypes force clients to share more higher-level feature information in a pairwise way. Using prototypes to realize inter-client communication allows for knowledge sharing in a latent space and brings additional benefits. Compared with directly transmitting the representation of a sample, prototypes can eliminate bias from a single sample and protect clients' privacy. Compared with model parameters, prototypes are much smaller in size, which significantly reduces communication costs.

We quantitatively evaluate the performance of FedPCL and state-of-the-art FL algorithms under the lightweight framework based on benchmark foundation models and datasets. We also perform extensive experiments to validate the effectiveness of FedPCL in fusing representations output by different backbones and its capability to integrate knowledge from backbones with different model architectures. Our main contributions are summarized as follows:

- We take the first step towards integrating pre-trained models into federated learning to form a lightweight FL framework, which substantially reduces computation and communication costs of current FL frameworks.
- We further propose FedPCL, a novel algorithm that uses class prototypes as the information carrier and conducts contrastive learning during local updates, which allows clients to share more class-relevant knowledge from both local and global prototypes.
- Experiments are conducted on a variety of benchmark foundation models and datasets to measure FedPCL's ability to fuse various pre-trained models for each client. The results indicate that FedPCL outperforms baselines with a better personalization and knowledge integration ability.

2 Related Work

Personalized Federated Learning. Personalized Federated Learning (PFL) aims to train a personalized model for each client in the FL framework so that the model can achieve better performance on the local dataset. Existing approaches for PFL are based on various techniques. [22, 23, 24] add an additional term to the original loss function of each client to produce better personalized models

according to the private data. [16, 25] share part of the model and keep personalized layer private to achieve personalization. [26] proposes to use a central hypernetwork to generate personalized models for clients. [27] enables a more flexible personalization by adaptively weighted aggregation. [4, 28] study PFL from a meta-learning perspective where a meta-model is learned to generate the initialized local model for each client.

Contrastive Learning. Contrastive learning has been widely applied to self-supervised learning scenarios in recent years, achieving state-of-the-art performance in the unsupervised training of deep image models [29, 30]. A number of works are focused on learning an encoder where the embeddings of the same sample are pulled closer and those of different samples are pushed apart [31, 32, 33]. [34] extends contrastive learning from self-supervised settings to fully supervised settings, enabling us to better exploit label information with contrastive learning. There are also some works incorporate contrastive learning into federated learning to assist local training to achieve higher model performance [35, 36, 37].

Prototype Learning. Prototypes have been widely used in a variety of tasks in transfer learning [20, 38], multi-task learning [21, 39], and meta-learning [17, 40]. It is usually defined as the mean feature vectors of samples within the same class [41, 42]. The authors in [43] represent task-agnostic information by prototypes for distributed machine learning systems and propose a multi-task model fusion method that integrates prototypes for a new task. Since the prototype has the ability to generalize semantic knowledge from similar samples, it is used to assist the local training in federated learning by several studies [44, 19, 36, 35].

Pre-Trained Foundation Model. Due to the huge number of parameters and the broad data available for training, pre-trained foundation models can better capture knowledge when being applied to downstream tasks [45]. Recently, pre-trained models (e.g., ViT [10], DETR [46], BERT [47]) have been widely investigated in both vision and natural language processing (NLP) tasks [48]. Extracting task-specific knowledge from the pre-trained models has the potential to achieve the state-of-the-art performance due to their generality and adaptability to different tasks [49, 50, 51]. However, how to integrate the pre-trained models into a federated learning paradigm and keep the whole framework lightweight still remains an open problem.

3 Problem Formulation

In this section, we formulate our proposed lightweight FL framework which integrates off-the-shelf pre-trained models as fixed backbones and learns to fuse them adaptively. We start from the general framework of FL, and then explicitly define the problem and explain the proposed global objective.

General FL Framework. Formally, the global objective of general FL across m clients is

$$\min_{(w_1, w_2, \cdots, w_m)} \frac{1}{m} \sum_{i=1}^{m} \frac{|D_i|}{N} L_i\left(w_i; D_i\right) \tag{1}$$

where L_i and w_i are the local loss function and model parameters for client i, respectively. D_i is the private dataset of the i-th client. N is the total number of samples among all clients.

The objective of general FL, such as vanilla FedAvg and its variants, is to learn an optimal global model w across m clients, where $w = w_1 = w_2 = \cdots = w_m$ [1]. This can be achieved by periodically synchronizing the model parameters of all clients at the server. However, parameter synchronization sometimes deteriorates the performance on local datasets in the presence of data heterogeneity [25, 52]. Some recent studies [5, 3] also explore personalized FL by applying various constraints and regularization terms, which allows clients to keep different models $w_i, i \in [1, m]$ to achieve higher performance on their local datasets.

The Proposed Lightweight FL Framework. Similar to most FL frameworks, there are m clients and a central server involved in the multiple pre-trained backbone-based framework. Each client $i \in [1, m]$ owns K shared and fixed backbones and a private dataset D_i that cannot be shared with each other. Each learning model can be seen as a combination of at least two parts: (i) *Feature Encoder* $r(\cdot; \Phi^*) : \mathbb{R}^d \to \mathbb{R}^{K \times d_e}$, comprising K fixed pre-trained backbones, each of which maps the raw sample $\mathbf{x}$ of size d to a representation vector of size d_e. K representation vectors are concatenated together as the output $r(\mathbf{x}; \Phi^*)$, denoted as $r_{\mathbf{x}}$ for short. (ii) *Projection Network* $h\left(\cdot; \theta_i\right) : \mathbb{R}^{K \times d_e} \to \mathbb{R}^{d_h}$, which fuses the K representation vectors and maps $r_{\mathbf{x}}$ from one latent space to another for further representation learning. The formal definition is provided as follows.

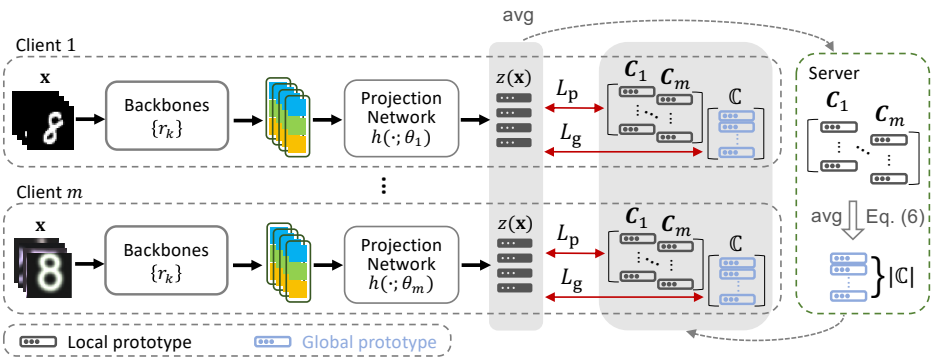


Figure 1: An overview of the proposed lightweight federated learning framework. This example assumes that for each client, there are three pre-trained backbones, with the block in different colors illustrating their backbone-specific representation.

Definition 3.1. Let ϕ_k^*, where $k \in [1, K]$, denote the optimal parameter of the k-th backbone pre-trained on a specific dataset, r_k be the embedding function of the k-th backbone, and $\mathbf{x}$ denote an instance sampled from a local dataset. We define the *concatenated representation* output by the feature encoder as

$$r(\mathbf{x}; \Phi^*) := \text{concat}\left(r_1(\mathbf{x}; \phi_1^*), \ldots, r_K(\mathbf{x}; \phi_K^*)\right). \tag{2}$$

For client i, the projection network h, parameterized by θ_i, aims to fuse the representations output by multiple backbones into another abstract space. The output of the projection network is computed as

$$z(\mathbf{x}) = h\left(r_{\mathbf{x}}; \theta_i\right). \tag{3}$$

Based on the above definition, we formulate the global objective of the lightweight FL framework as

$$\begin{aligned} &\min_{\{\theta_1, \theta_2, \ldots, \theta_m\}} \sum_{i=1}^{m} \frac{|D_i|}{N} \mathbb{E}_{(\mathbf{x},y)\in D_i}\left[L_i\left(\theta_i; z\left(\mathbf{x}\right), y\right)\right] \\ &\text{s.t. } z(\mathbf{x}) = h\left(r(\mathbf{x}; \Phi^*); \theta_i\right) \text{ where } \Phi^* = \{\phi_1^*, \phi_2^*, \cdots, \phi_K^*\}. \end{aligned} \tag{4}$$

The target of the framework is to learn the personalized projection network $\{\theta_i\}_{i=1}^m$ for each client. Given an input sample $\mathbf{x}$, the representations output by pre-trained backbones are concatenated together as $r(\mathbf{x}; \Phi^*)$. Then, the projection network $h(\theta_i)$ optimized for each client converts the concatenated representation to $z(\mathbf{x})$.

The proposed multi-backbone setting is friendly to most FL methods. The learnable parameter θ in the projection network can be fully or partly shared across clients for different purposes. For example, FedAvg [1] and its variants [53, 54, 55, 56] can synchronize θ every communication round, and personalized FL methods can also adapt to this framework without much modification [22, 25].

4 Federated Prototype-wise Contrastive Learning (FedPCL)

In this section, we elaborate our proposed algorithm **Fed**erated **P**rototype-wise **C**ontrastive **L**earning (FedPCL), which is illustrated in Figure 1. In each client, $z(\mathbf{x})$ is generated by the projection network which fuses the representations from multiple backbones. Then, to share the common underlying knowledge across clients, we employ a prototype-based communication scheme to transmit and aggregate prototype sets between the server and clients. With the prototypes returned from the server, we perform local optimization via well-crafted prototype-wise contrastive loss function, which extracts class-relevant information while sharing more inter-client knowledge in the latent space. We provide detailed illustration for each procedure in the rest of this section.

Prototype as the Information Carrier. To capture more class-relevant information and semantically meaningful knowledge, we propose to transmit prototypes between the server and clients. Compared with transmitting the learnable model parameters, there are several advantages brought by transmitting class-wise prototypes. Firstly, prototype is more compact in form, which significantly decreases communication costs required during the training process. Secondly, non-parametric communication

allows each client to learn a more customized local model without synchronizing parameters with others. Thirdly, prototypes are high-level statistic information rather than raw features, which raise no additional privacy concerns to the system and are robust to gradient-based attacks.

To extract useful class-relevant information from the local side, we construct a local prototype as the information carrier for knowledge uploading. Specifically, it is defined in the latent space of projection network's output by the mean of the fused representations within the same class j,

$$C_i^{(j)} := \frac{1}{|D_{i,j}|} \sum_{(\mathbf{x},y)\in D_{i,j}} z\left(\mathbf{x}\right), \tag{5}$$

where $D_{i,j}$ refers to the subset of D_i composed of all instances belonging to class j, and $\boldsymbol{C}_i$ denotes the local prototype set of the i-th client. After the above computation, the local prototype set of each client is sent to the central server for knowledge aggregation, which shares the local class-relevant information extracted on each specific client based on its local dataset.

Server Aggregation. After receiving local prototype sets $\{\boldsymbol{C}_i\}_{i=1}^m$ from all participating clients, the server first computes the global prototype as

$$\bar{C}^{(j)} := \frac{1}{|\mathcal{N}_j|} \sum_{i\in\mathcal{N}_j} \frac{|D_{i,j}|}{N_j} C_i^{(j)}, \tag{6}$$

where $\mathcal{N}_j$ denotes the set of clients that own instances of class j, and N_j denotes the number of instances belonging to class j over all clients. The global prototype set is denoted as $\mathbb{C} = \{\bar{C}^{(1)}, \bar{C}^{(2)}, \dots\}$. With such an aggregation mechanism, the global prototype set summarize coarse-grained class-relevant knowledge shared by all clients, which provides a high-level perspective for representation learning.

After aggregation, the server sends both the global prototype set and the full local prototype sets collected from all clients back to every clients. For the situation where only a few classes in each client in some non-IID cases, we introduce a prototype padding procedure in the server to ensure that each local prototype set contains prototypes corresponding to all classes:

$$C_i^{(j)} = \begin{cases} C_i^{(j)}, & i \in \mathcal{N}_j \\ \bar{C}^{(j)}, & i \notin \mathcal{N}_j \end{cases}. \tag{7}$$

Prototype-based communication and aggregation allow each client to own a unique projection network that is able to fuse the general representations in a customized way. The returned local prototype sets can encourage a mutual learning from a client-relevant perspective while the global prototype set, where each element indicates a class center in the overall data, provides a chance to learn from a highly summarized client-irrelevant perspective.

Local Training. After receiving the prototype sets from the server, the main target of local training is to efficiently extract useful knowledge from the local and the global prototypes, respectively, so as to maximally benefit local representation learning. To achieve that, we propose a prototype-wise supervised contrastive loss that consists of two terms, i.e., global term and local term.

To force the fused representation $z(\mathbf{x})$ generated by the local projection network to be closer to its corresponding global class center so as to extract more class-relevant but client-irrelevant information, we define the global prototype-based loss term as

$$L_{\mathrm{g}} = \sum_{(\mathbf{x},y)\in D_i} -\log \frac{\exp\left(z_{\mathbf{x}} \cdot \bar{C}^{(y)}/\tau\right)}{\sum_{y_a\in A(y)} \exp\left(z_{\mathbf{x}} \cdot \bar{C}^{(y_a)}/\tau\right)}. \tag{8}$$

where $z_{\mathbf{x}}$ represents $z(\mathbf{x})$ for short, $A(y) := \{y_a \in [1, |\mathbb{C}|] : y_a \neq y\}$ is the set of labels distinct from y, τ is the temperature that can adjust the tolerance for feature difference [57, 33, 34]. For a specific instance $\mathbf{x}$ sampled from D_i, we use an inner dot product to measure the similarity between the fused representation $z_{\mathbf{x}}$ and prototypes.

Apart from the global term, to align $z(\mathbf{x})$ with each client's local prototypes by alternate client-wise contrastive learning in the latent space and enable more inter-client knowledge sharing, we define the local prototype-based loss term as

$$L_{\mathrm{p}} = \sum_{(\mathbf{x},y)\in D_i} -\frac{1}{m}\sum_{p=1}^{m} \log \frac{\exp\left(z_{\mathbf{x}} \cdot C_p^{(y)}/\tau\right)}{\sum_{y_a\in A(y)} \exp\left(z_{\mathbf{x}} \cdot C_p^{(y_a)}/\tau\right)}. \tag{9}$$

Given the ground-truth label y of $\mathbf{x}$, a prototype set can be divided as one positive prototype $C^{(y)}$ and a set of negative prototypes $C^{(y_a)}$. Both L_{g} and L_{p} force the representation of a sample to be closer to those positive prototypes and apart from those negative prototypes. $\bar{C}^{(y)}$ in L_{g} and $C^{(y)}$ in L_{p} summarize abstract class-relevant information to different granularity, which provides guidance for optimizing the local projection network from different perspectives. For the i-th client, the local objective function in Eq. (4) is defined as a combination of L_{g} and L_{p} in the following form,

$$L\left(\theta_i; z\left(\mathbf{x}\right), y, \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m}\right) = L_{\mathrm{g}}\left(\theta_i; z\left(\mathbf{x}\right), y, \mathbb{C}\right) + L_{\mathrm{p}}(\theta_i; z\left(\mathbf{x}\right), y, \{\boldsymbol{C}_p\}_{p=1}^{m}). \tag{10}$$

At the end of the local training in each round, clients upload their local prototype set $\boldsymbol{C}_i$ to the server. We present the FedPCL algorithm in detail in Algorithm 1.

Prototype-based Inference. After adequate optimization, in each client, the local prototypes generated by projection network not only contain compact local class-wise information but also absorb general knowledge from all participating clients. Considering their powerful representation capability, we leverage the local prototypes to predict unknown labels in the inference stage. Concretely, for a test sample, we first calculate the similarity scores between the output of projection network and every local prototypes. Then, the predicted result is the class with the maximum similarity score.

Generalization Bound. We provide insights into the performance analysis of FedPCL. A detailed description and derivations can be found in Appendix C.

Algorithm 1 FedPCL

Input: $D_i, \theta_i, i = 1, \cdots, m$, and K pre-trained backbones with parameters $\phi_1^*, \phi_2^*, \cdots, \phi_K^*$, respectively.
Server executes:
Initialize prototype sets $\{\boldsymbol{C}_p\}_{p=1}^{m}$.
for each round $T = 1, 2, ...$ **do**
  for each client i **in parallel do**
    $\boldsymbol{C}_i \leftarrow$ LocalUpdate$(i, \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m})$
  end for
  Update global prototype by Eq. (6).
end for
LocalUpdate$(i, \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m})$:
for each local epoch **do**
  for each batch in D_i **do**
    Compute L_{g} by Eq. (8) with global prototypes.
    Compute L_{p} by Eq. (9) with local prototypes.
    Update θ_i by Eq. (10).
  end for
end for
Compute local prototypes by Eq. (5).
return $\boldsymbol{C}_i$

Theorem 4.1. (Generalization Bound of FedPCL.) *Consider an FL system with m clients. Let $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_m$ be the true data distribution and $\widehat{\mathcal{D}}_1, \widehat{\mathcal{D}}_2, \cdots, \widehat{\mathcal{D}}_m$ be the empirical data distribution. Denote the projection network h as the hypothesis from $\mathcal{H}$ and d be the VC-dimension of $\mathcal{H}$. The total number of samples over all clients is N. Then, with probability at least $1-\delta$:*

$$\begin{aligned} &\max_{(\theta_1,\theta_2,\ldots,\theta_m)} \left| \sum_{i=1}^{m} \frac{|D_i|}{N} L_{\mathcal{D}_i}\left(\theta_i; \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m}\right) - \sum_{i=1}^{m} \frac{|D_i|}{N} L_{\widehat{\mathcal{D}}_i}\left(\theta_i; \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m}\right) \right| \\ &\leq \sqrt{\frac{N}{2} \log \frac{(m+1)|\mathbb{C}|}{\delta}} + \sqrt{\frac{d}{N} \log \frac{eN}{d}}. \end{aligned} \tag{11}$$

Theorem 4.1 indicates that compared with the model trained on an ideal data distribution, the performance of the empirically trained model is associated with the VC-dimension of $\mathcal{H}$ and the size of the prototype set $\mathbb{C}$. An expected performance gap can be achieved by using appropriate projection network and number of classes. The generalization bound is also important to ensure a satisfying performance of FedPCL, especially when applying it to some safety-critical scenarios [3, 58, 59].

5 Experiments

5.1 Experimental Setup

Datasets and Non-IID Settings. We evaluate our proposed framework on the following three benchmark datasets: Digit-5 [15], Office-10 [60], and DomainNet dataset [61]. **Digit-5** [15] is a benchmark dataset for digit recognition, including five benchmark datasets, namely SVHN, USPS, SynthDigits, MNIST-M, and MNIST. **Office-10** [60] is a standard benchmark dataset consisting of four datasets, namely Amazon, Caltech, DSLR and WebCam. **DomainNet** [62] is a large-scale dataset consisting of six datasets, namely Clipart, Info, Painting, Quickdraw, Real, and Sketch.

Table 2: Test accuracy under the feature shift non-IID setting. In the column labeled BB (short for backbone), *s* is for a single pre-trained backbone and *m* is for multiple pre-trained backbones. # of Comm Params refers to the average number of parameters sent from a client to the server per round.

BB	**Method**	MNIST	SVHN	USPS	Synth	MNIST-M	Avg	# of Comm Params
s	FedAvg	70.65(1.15)	17.10(0.20)	70.24(1.62)	32.90(0.75)	29.33(1.18)	44.04(0.98)	133,632
	pFedMe	71.13(3.63)	13.18(1.78)	69.20(0.30)	36.25(3.35)	25.25(2.25)	43.00(2.26)	133,632
	PerFedAvg	52.68(7.03)	16.28(1.23)	53.66(6.58)	29.05(3.45)	24.38(2.38)	35.21(4.13)	133,632
	FedRep	64.00(2.20)	17.88(1.08)	70.44(1.27)	36.50(1.55)	31.90(0.05)	44.14(2.03)	131,072
	FedProto	80.40(2.75)	17.03(0.38)	88.47(0.91)	40.90(1.10)	32.85(0.75)	51.93(1.18)	2,560
	Solo	60.40(2.25)	15.60(0.20)	75.28(4.48)	34.65(0.05)	28.48(0.53)	42.88(1.50)	-
	Ours	**82.75**(0.40)	**18.12**(0.42)	**88.82**(0.15)	**41.40**(0.60)	**33.05**(0.95)	**52.83**(0.21)	2,560
m	FedAvg	71.68(2.93)	18.45(0.45)	72.95(0.86)	37.35(1.35)	33.70(2.55)	46.83(1.63)	395,776
	pFedMe	67.45(2.70)	15.43(0.38)	65.66(7.20)	33.55(4.60)	31.80(0.20)	42.78(3.01)	395,776
	PerFedAvg	56.03(2.73)	17.03(0.63)	57.55(0.27)	34.90(2.80)	30.98(1.53)	39.30(1.59)	395,776
	FedRep	77.25(1.75)	16.40(0.50)	80.25(0.32)	37.63(2.18)	36.53(0.28)	49.61(1.05)	393,216
	FedProto	83.78(0.83)	17.90(0.10)	**91.74**(0.00)	43.70(2.45)	36.43(1.58)	54.71(0.99)	2,560
	Solo	70.43(4.63)	15.00(0.40)	84.90(0.24)	37.18(2.73)	34.35(2.20)	48.37(2.04)	-
	Ours	**84.65**(0.15)	**19.38**(0.63)	90.74(0.53)	**44.73**(0.37)	**37.25**(0.28)	**55.34**(0.34)	2,560

To mimic non-IID scenarios in a more general way, we investigate three different non-IID settings: (i) *Feature shift* non-IID: The datasets owned by clients have the same label distribution but different feature distributions. (ii) *Label shift* non-IID: The datasets owned by clients have the same feature distribution but different label distributions which is simulated by Dirichlet distribution with parameter α [58, 55]. (iii) *Feature & Label shift* non-IID: The datasets owned by clients are different in both label distribution and feature distribution, which is more common but challenging in real-world scenarios. Details about the non-IID settings can be found in Appendix A.1.

Baselines and Implementation. We compare our proposed method with popular FL algorithms including FedAvg [1], pFedMe [22], PerFedAvg [4], FedRep [25], FedProto [19], and Solo, i.e., training independently within each client. In feature shift and label shift non-IID setting, the number of clients is 5. In feature & label shift setting, the number of clients is 5, 4, 6 for Digit-5, Office-10, and DomainNet, respectively.

Similar to [13, 14], we use the ResNet18 architecture [12] pre-trained on Quick Draw, Aircraft, and CU-Birds [14] as the backbones. For all baseline methods, the backbone module is followed by two fully connected layers, corresponding to projection network and classifier, respectively, whereas for our method, the backbone module is followed by only one fully connected layer as the projection network. The output dimension of each backbone and the projection network are 512 and 256, respectively. Note that for a fair comparison, all baselines use the same network architecture on top of the frozen backbones as FedPCL except their essential classifier.

We use a batch size of 32, and an Adam [63] optimizer with weight decay 1e-4 and learning rate 0.001. The default setting for local update epochs is $E = 1$ and the temperature τ is 0.07. We implement all the methods using PyTorch and conduct all experiments on one NVIDIA Tesla V100 GPU. Details about the model and each dataset can be found in Appendix A.1.

5.2 Performance Comparison

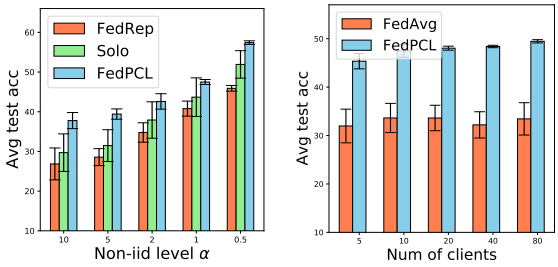
Table 2 and Table 3 report the results of our method and baselines in mean (std) format over clients with three independent runs. Table 2 shows the results of two cases: (1) *s*: a single backbone is available; (2) *m*: multiple backbones are available.

Table 3: Test accuracy under the feature & label shift non-IID setting for Office-10, under the label shift non-IID setting for DomainNet.

Method	Office-10	Domainnet
FedAvg	33.84(4.59)	28.09(2.91)
pFedMe	30.00(1.41)	32.65(0.72)
Per-FedAvg	26.04(1.46)	34.64(0.54)
FedRep	37.24(1.54)	48.82(0.55)
FedProto	34.54(2.65)	44.48(0.58)
Solo	36.38(0.54)	46.70(0.75)
Ours	**41.40**(1.19)	**52.92**(3.47)

The results suggest that: (1) compared with single backbone cases, multiple pre-trained backbones lead to higher test accuracy in most cases and about a $1\% - 4\%$ test accuracy improvement for our method; (2) apart from locally training each client (Solo), our method achieves relatively smaller deviation across different runs compared with most baselines, which demonstrates that FedPCL is able to fuse the representations in a more stable way; (3) the number of communicated parameters in prototype-based method is much lower than that of the model parameter-based methods.

Robustness to varying levels of heterogeneity. In the label shift non-IID setting, we use Dirichlet distribution with parameter $\alpha > 0$ to simulate the heterogeneity level over 10 clients. As α becomes smaller, the label distribution becomes more heterogeneous. We evaluate our proposed FedPCL with FedRep and Solo by varying $\alpha \in \{0.5, 1, 2, 5, 10\}$ on 10 clients. Figure 2(a) suggests that FedPCL achieves the best performance for all values of α. Moreover, FedPCL has the smallest deviation across several runs, which implies the stable convergence and robustness of FedPCL.



(a) (b)

Figure 2: Test accuracy on Digit-5 under varying non-IID levels (*left*) and varying numbers of clients (*right*). Both are conducted in the label shift non-IID setting. In (a), non-IID level is controlled by α, ranging from 0.5 to 10. In (b), the number of clients (m) ranges from 5 to 80.

Effect of varying numbers of participating clients. We also compare FedPCL with the vanilla FedAvg by varying the number of clients (m) from 5 to 80 as shown in Figure 2(b). The non-IID level remains the same when more clients participate in the training procedure. The results show that the average test accuracy of FedPCL is higher than that of FedAvg by about 15%. Furthermore, the deviation of FedPCL across three runs is much lower than that of FedAvg, especially when m becomes larger. This indicates the scalability of the lightweight framework and the potential and ability of FedPCL when adapting to a large-scale FL system.

5.3 Integrating Backbones with Various Architectures

The above experiments use backbones with the same architecture but pre-trained on different source data, showing the capability of FedPCL on integrating knowledge from different source domains. In Table 4, we also show that our proposed lightweight framework is capable of (i) using pre-trained models with different architectures, i.e., a two-layer CNN, AlexNet [64], and VGGNet [65]; (ii) integrating large-scale pre-trained models, i.e., ViT [10, 66], to enhance the performance without the need of huge computation resources to train or fine-tune them. Pre-trained models with different representation abilities should be adaptively leveraged for different tasks to achieve expected performance. They can be selected based on a small set of data before large-scale training.

Table 4: Integrating backbones with various architectures into the proposed framework. Experiments are implemented with FedPCL under feature & label shift non-IID setting.

Backbone-Domain	Digit-5	Office-10
[ResNet18-QuickDraw, ResNet18-Aircraft, ResNet18-Birds]	42.87(1.47)	41.40(1.19)
[MLP-ImageNet, AlexNet-ImageNet, VGG11-ImageNet]	**55.80**(2.09)	70.11(1.78)
[tiny-ViT-ImageNet, small-ViT-ImageNet, base-ViT-ImageNet]	40.96(2.87)	**84.63**(2.57)

5.4 Ablation Study

In this section, we present the results for various ablation experiments to test the effect of global/local prototypes and the contrastive loss. More results can be found in Appendix A.2.3.

Effect of the contrastive loss. We test the performance of the local loss function in different forms: (1) Cross Entropy loss; (2) Cross Entropy loss + ProtoDist term, which takes the distance between prototype and fused representation as an additional term to regularize the cross entropy loss; (3) Supervised Contrastive loss that only uses local embedding for contrastive learning; (4) Our prototype-wise contrastive loss that uses both global and local prototypes for local contrastive learning. As shown in Table 5, our prototype-wise supervised contrastive loss outperforms others.

Table 5: Comparison between the cases when different local losses are used for local training. Experiments are conducted on Digit-5 dataset under the feature & label shift non-IID setting where the Dirichlet parameter α is 1, the number of clients is 5, and the number of pre-trained backbones is 3.

Loss Type	**Acc**
Cross Entropy	32.98(3.44)
Cross Entropy + ProtoDist [19]	43.94(3.02)
Supervised Contrastive [34]	42.18(3.25)
Ours	**45.22(3.65)**

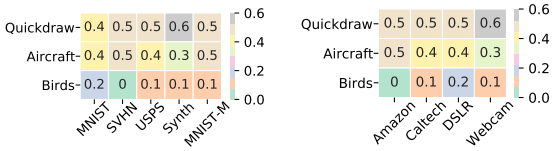
Effect of prototypes. To verify the effectiveness of different kinds of prototypes used for local training in our proposed FedPCL, we compare the following three cases: (i) Only global prototypes computed at the server are used for local training; (ii) Only local prototypes aggregated at the server are used for local training; (iii) Both global and local prototypes are used for local training. All these three cases are implemented under three non-IID settings. As shown in Table 6, without global or local prototypes being used for local supervised contrastive learning, the performance drops 0.3%-2%, indicating that the knowledge conveyed by global prototypes and local prototypes can benefit the local learning framework from different perspectives.

Table 6: Comparison between the cases when only global prototypes are used for local training, only local prototypes from all clients are used for local training, and both of them are used for local training. Experiments are conducted on Digit-5 dataset under three non-IID settings where the Dirichlet parameter α is 1, the number of clients is 5, and the number of pre-trained backbones is 3.

Prototypes	Feature shift non-IID	Label shift non-IID	Feature & Label shift non-IID
Global only	54.69(0.14)	44.75(1.77)	43.79(4.09)
Local only	55.01(0.10)	44.52(1.73)	43.08(3.87)
Global and local	**55.34**(0.34)	**45.35**(1.58)	**45.22**(3.65)

5.5 Visualizing the Fusing Results of FedPCL

To better understand how FedPCL fuses the representation output by backbones, we visualize the normalized similarity scores under the feature shift non-IID setting. The heatmap in Figure 3 summarizes the values of the normalized similarity scores, computed by the inner product between local prototypes in the single backbone case and the multi-backbone case. For example, in Figure 3(a), the element 0.5 on row 1 and column 2 represents the similarity score between the local prototype computed when only one backbone pre-trained in Quickdraw is available and the local prototype computed when three backbones pre-trained in Quickdraw, Aircraft, and Birds are available.



(a) Digit-5 (b) Office-10

Figure 3: Similarity scores generated on two datasets: (a) Digit-5 and (b) Office-10. Rows correspond to the pre-trained backbones $r_k(\mathbf{x})$ and columns correspond to different local datasets with feature shift.

We find that the backbone pre-trained on Quickdraw contributes more to the fused prototype, while the backbone pre-trained on Birds contributes the least. Clients who own a specific dataset show different levels of utilization when fusing the representations from various backbones.

5.6 Incorporating with Privacy-Preserving Techniques

To further eliminate concerns about privacy leakage, we also incorporate FedPCL with privacy-preserving techniques to observe the variation in performance. Concretely, we add various random noise to the prototypes to be communicated and the original images, respectively. The results show that the performance of FedPCL remains high after noise injection. Details are given in Appendix A.2.4.

6 Conclusion

To address the problems on excessive computation and communication demands in current federated learning frameworks, we propose a lightweight framework that leverages multiple neural networks as fixed pre-trained backbones to replace the learnable feature extractor. To customize the general representations generated by these backbones for each client, class-wise prototypes are shared across the clients and the server. To efficiently extract the shared knowledge from the prototypes, we develop FedPCL algorithm that uses contrastive learning at the client-side during the local update. Extensive experiments are conducted to show the superiority of FedPCL under the proposed lightweight framework.

This research has great potential to bring existing FL frameworks to a new paradigm and provide more options on local neural network architectures. However, we mainly focus on the vision task in this work. Methods for language tasks can be further explored in future studies. Considering the evident motivation and effectiveness of the lightweight framework and FedPCL, we believe this work is intriguing for future studies.

References

[1] H Brendan McMahan, Eider Moore, Daniel Ramage, et al. Communication-efficient learning of deep networks from decentralized data. *AISTATS*, 2017.

[2] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, et al. Advances and open problems in federated learning. *arXiv:1912.04977*, 2019.

[3] Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, and Feijie Wu. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[4] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. In *Advances in Neural Information Processing Systems*, 2020.

[5] Canh T Dinh, Nguyen Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems*, 2020.

[6] Jaehoon Oh, SangMook Kim, and Se-Young Yun. FedBABU: Toward enhanced representation for federated image classification. In *International Conference on Learning Representations*, 2022.

[7] Sangjoon Park, Gwanghyun Kim, Jeongsol Kim, Boah Kim, and Jong Chul Ye. Federated split task-agnostic vision transformer for covid-19 cxr diagnosis. *Advances in Neural Information Processing Systems*, 34, 2021.

[8] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: A research platform for federated learning in natural language processing. *arXiv preprint arXiv:2104.08815*, 2021.

[9] Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

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

[13] Lu Liu, William L Hamilton, Guodong Long, Jing Jiang, and Hugo Larochelle. A universal representation transformer layer for few-shot image classification. In *International Conference on Learning Representations*, 2020.

[14] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

[15] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *European Conference on Computer Vision*, pages 561–578. Springer, 2020.

[16] Xiaoxiao Li, Meirui JIANG, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *International Conference on Learning Representations*, 2020.

[17] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4077–4087, 2017.

[18] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. In *NeurIPS*, 2021.

[19] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. FedProto: Federated prototype learning over heterogeneous devices. In *AAAI*, 2022.

[20] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer learning for image classification with sparse prototype representations. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[21] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011.

[22] Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *NeurIPS*, 2020.

[23] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. *Advances in Neural Information Processing Systems*, 33:2304–2315, 2020.

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

[25] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, 2021.

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

[27] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*, 2020.

[28] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[29] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[30] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8392–8401, 2021.

[31] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2018.

[32] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019.

[33] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[34] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[35] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021.

[36] Xutong Mu, Yulong Shen, Ke Cheng, Xueli Geng, Jiaxuan Fu, Tao Zhang, and Zhiwei Zhang. Fedproc: Prototypical contrastive federated learning on non-iid data. *arXiv preprint arXiv:2109.12273*, 2021.

[37] Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *arXiv preprint arXiv:2010.08982*, 2020.

[38] Zhaohong Deng, Yizhang Jiang, Fu-Lai Chung, Hisao Ishibuchi, Kup-Sze Choi, and Shitong Wang. Transfer prototype-based fuzzy clustering. *IEEE transactions on fuzzy systems*, 24(5):1210–1232, 2015.

[39] Yongxin Yang and Timothy Hospedales. A unified perspective on multi-domain and multi-task learning. In *3rd International Conference on Learning Representations*, 2015.

[40] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning to propagate for graph meta-learning. *Advances in Neural Information Processing Systems*, 32:1039–1050, 2019.

[41] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. In *International Conference on Learning Representations*, 2015.

[42] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, pages 1269–1277, 2015.

[43] Nghia Hoang, Thanh Lam, Bryan Kian Hsiang Low, and Patrick Jaillet. Learning task-agnostic embedding of multiple black-box experts for multi-task model fusion. In *International Conference on Machine Learning*, pages 4282–4292. PMLR, 2020.

[44] Umberto Michieli and Mete Ozay. Prototype guided federated learning of visual feature representations. *arXiv preprint arXiv:2105.08982*, 2021.

[45] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, et al. Pre-trained models: Past, present and future. *AI Open*, 2021.

[46] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[47] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, 2019.

[48] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[49] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[50] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[51] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[52] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 2020.

[53] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *MLSys*, 2020.

[54] Ming Xie, Guodong Long, Tao Shen, Xianzhi Wang, Zhou Tianyi, and Jing Jiang. Multi-center federated learning. *arXiv:2005.01026*, 2020.

[55] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020.

[56] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *NeurIPS*, 2020.

[57] Oliver Zhang, Mike Wu, Jasmine Bayrooti, and Noah Goodman. Temperature as uncertainty in contrastive learning. *arXiv preprint arXiv:2110.04403*, 2021.

[58] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *NeurIPS*, 2020.

[59] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv:2002.10619*, 2020.

[60] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2066–2073. IEEE, 2012.

[61] Xingchao Peng and Kate Saenko. Synthetic to real adaptation with generative correlation alignment networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1982–1991. IEEE, 2018.

[62] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[65] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[66] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Li Fei-Fei, Ehsan Adeli, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[67] Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. The quick, draw!-AI experiment. *Mount View, CA, accessed Feb*, 17(2018):4, 2016.

[68] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[69] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2019.

[70] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021.

[71] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*, pages 12133–12143. PMLR, 2021.

[72] Gary Cheng, Karan Chadha, and John Duchi. Federated asymptotics: a model to compare federated learning algorithms. *arXiv preprint arXiv:2108.07313*, 2021.

[73] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10174–10183, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.6 and Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix C.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 5.1 and source code in supplemental material.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.1 and Appendix A.1.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5.2.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.1.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.1.

(b) Did you mention the license of the assets? [Yes] See Section 5.1.

(c) Did you include any new assets either in the supplemental material or as a URL? [No] We do not use any new datasets in this paper.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] The datasets used in this paper are publicly available.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experimental Details and Additional Results

A.1 Experimental Details

A.1.1 Non-IID Settings.

(a) Feature shift non-IID (b) Label shift non-IID (c) Feature & label shift non-IID

Figure 4: Illustration of three non-IID settings on Digit-5 dataset. Each dot represents a set of samples within a certain class allocated to a client. The feature shift non-IID is indicated by dot colors while the label shift non-IID ($\alpha = 1$) is indicated by dot sizes.

To mimic non-IID scenarios in a more general way, we investigate three different non-IID settings as follows and visualize them in Figure 4.

- *Feature shift* non-IID: The datasets owned by clients have the same label distribution but different feature distributions. The number of classes and the number of samples per class are the same across clients.
- *Label shift* non-IID: The datasets owned by clients have the same feature distribution but different label distributions. Similar to existing works [58, 55], we use Dirichlet distribution with parameter α to allocate examples for this kind of non-IID setting.
- *Feature & Label shift* non-IID: The datasets owned by clients are different in both label distribution and feature distribution, which is more common but challenging in real-world scenarios.

(a) Digit-5 (b) Office-10 (c) DomainNet

Figure 5: Examples of raw instances from three datasets: Digit-5 (*left*), Office-10 (*middle*), and DomainNet (*right*). We present five classes for each dataset to show the feature shift across their sub-datasets.

A.1.2 Visualizatin of Raw Samples.

Some examples of raw instances can be found in Figure 5.

A.1.3 Model Architecture.

For the single backbone cases, we use ResNet18 pre-trained on Quickdraw as the backbone. For the multiple backbone cases, we use three pre-trained ResNet18 as the backbones. They are pre-trained on Quick Draw [67], Aircraft [68], and CU-Birds [13] public dataset, respectively. The model architecture following the backbone module is shown in Table 7.

Table 7: The model architecture with learnable parameters for each client. FC refers to fully connected layer and BN refers to the BatchNormalization layer. K refers to the number of available pre-trained backbones, which is 3 in our experiments.

Layer	Details
1	FC(512$\times K$, 256), ReLU, BN(256)
2	FC(256, 10)

A.1.4 Training Details.

We provide the detailed settings for the experiments conducted in Section 5.2.

For data splitting, we use a portion of data as training samples (10%) and the rest as test samples. We first take out a 20% subset of the training set for validation and return the validation set back to the training set and retrain the model after selecting the optimal hyperparameters.

Table 8, 9, 10 show the data partitioning details in feature shift non-IID, label shift non-IID, and feature & label shift non-IID, respectively. We run each algorithm till the convergence of its loss.

Table 8: Detailed statistics for three benchmark datasets in feature shift non-IID, label IID setting (Table 2).

Datasets	MNIST	SVHN	USPS	SynthDigits	MNIST-M
# of clients	1	1	1	1	1
# of classes per client	10	10	10	10	10
# of samples per class	10	10	10	10	10

Table 9: Detailed statistics for three benchmark datasets in label shift non-IID, feature IID setting (Table 11).

Datasets	MNIST-M	Caltech	Real
# of clients	5	5	5
# of samples per client	[152,92,112,72,70]	[76,79,111,70,112]	[153,95,111,55,84]

Table 10: Detailed statistics for three benchmark datasets in feature & label shift non-IID setting (Table 11).

Datasets	Digit-5	Office-10	DomainNet
# of clients	5	4	6
# of samples per client	[100,75,112,120,65]	[89,108,58,120]	[137,230,270,204,175,152]

For hyperparameter tuning, we use grid search to find the optimal hyperparameters including learning rate, weight decay, and the output dimension of each backbone. Concretely, the grid search is carried out in the following search space:

- learning rate: $\{0.1, 0.01, 0.001, 0.0001\}$
- weight decay: {1e-3, 5e-4, 1e-4, 5e-5, 1e-5}
- output dimension of each backbone: $\{128, 256, 512, 1024\}$

A.2 Additional Experiments

A.2.1 Performance under Three Non-IID Settings.

Table 11 shows the performance of FedPCL and baseline methods under label shift non-IID scenarios.

A.2.2 Fairness across Clients.

Following the metrics in [69], we verify the advantage of FedPCL in fairness and report the results over 40 and 80 clients in Table 12 and Table 13, respectively. We list the average, the worst 10%,

Table 11: Test accuracy under the (1) label shift non-IID setting, (2) feature & label shift non-IID setting. For the former (feature IID, label non-IID), we use MNIST-M, Caltech, and Real as the datasets of all clients, respectively. For the latter (feature non-IID, label non-IID), we use Digit-5, Office-10, and DomainNet as the datasets, respectively.

Feature	**Method**	**Single backbone**			**Multi-backbone**		
		Digit-5	Office-10	Domainnet	Digit-5	Office-10	DomainNet
IID	FedAvg	30.42(5.34)	28.84(1.01)	25.78(1.48)	31.97(3.48)	23.60(1.72)	28.09(2.91)
	pFedMe	28.14(1.26)	22.53(5.28)	29.43(1.30)	32.82(1.74)	25.73(3.26)	32.65(0.72)
	Per-FedAvg	27.22(7.11)	36.74(3.96)	31.37(3.35)	30.95(2.94)	25.10(0.55)	34.64(0.54)
	FedRep	34.14(7.37)	36.35(0.53)	**41.95**(3.35)	39.27(1.35)	37.95(0.91)	48.82(0.55)
	FedProto	38.02(3.89)	39.04(0.13)	34.41(0.74)	42.17(3.23)	40.78(0.93)	44.48(0.58)
	Solo	36.40(3.71)	37.82(1.79)	39.06(3.27)	41.33(2.22)	39.59(2.49)	46.70(0.75)
	Ours	**40.88**(2.09)	**40.76**(0.21)	39.63(3.31)	**45.35**(1.58)	**42.13**(0.77)	**52.92**(3.47)
Non-IID	FedAvg	31.03(4.29)	25.67(1.79)	16.40(1.25)	32.98(3.44)	33.84(4.59)	19.25(2.03)
	pFedMe	25.13(8.31)	22.65(1.10)	16.56(2.37)	28.10(8.80)	30.00(1.41)	18.46(2.04)
	Per-FedAvg	28.94(0.60)	25.87(0.19)	18.68(1.47)	31.95(1.81)	26.04(1.46)	20.96(1.83)
	FedRep	34.16(3.40)	32.50(2.86)	19.64(1.53)	39.28(2.44)	37.24(1.54)	30.28(1.01)
	FedProto	41.49(2.75)	37.47(1.27)	19.37(0.14)	43.94(3.02)	34.54(2.65)	29.45(0.39)
	Solo	35.56(3.16)	35.54(1.05)	17.84(0.87)	38.35(2.55)	36.38(0.54)	27.15(0.52)
	Ours	**42.87**(1.47)	**37.64**(1.99)	**24.90**(1.61)	**45.22**(3.65)	**41.40**(1.19)	**35.23**(0.29)

the worst 20%, the worst 40%, the best 10% of test accuracy over all clients across three runs in Digit-5 dataset. We also report the variance of the accuracy distribution across clients (the smaller, the fairer) [69]. The results demonstrate that in such a multi-backbone scenario, it is easier to obtain a fair solution using prototype-based communication rather than model parameter/gradient-based communication, because the information conveyed by prototypes is more local data distribution-independent while the information conveyed by model parameters tends to be affected by local data distribution. Detailed experimental results on fairness can be found in Table 12 and 13.

Table 12: The average, the worst, the best, and the variance of the test accuracy of 40 clients on Digit-5.

Method	Average	Worst 10%	Worst 20%	Worst 40%	Best 10%	Variance
FedAvg	32.20(2.17)	16.73(1.73)	20.31(2.21)	25.01(0.71)	54.60(3.33)	11.16(0.35)
Ours	48.39(0.25)	35.35(2.63)	37.58(3.07)	40.82(3.16)	62.72(1.33)	8.45(0.06)

Table 13: The average, the worst, the best, and the variance of the test accuracy of 80 clients on Digit-5.

Method	Average	Worst 10%	Worst 20%	Worst 40%	Best 10%	Variance
FedAvg	33.44(3.34)	12.34(2.67)	15.53(2.90)	19.53(2.40)	56.87(2.74)	13.20(1.35)
Ours	49.46(0.34)	30.39(4.57)	34.37(3.40)	39.52(2.66)	66.46(2.38)	10.60(0.25)

A.2.3 Effect of the Number of Backbones.

The number of pre-trained backbones can be adjusted for a specific task correspondingly. To study its effect, we compare the performance and parameter size when different numbers of fixed backbones are used. The results in Table 14 indicate that more pre-trained backbones can lead to better performance but consume more computing resources and memory space.

A.2.4 Privacy Protection.

We incorporate FedPCL with privacy-preserving techniques to observe its variation in performance. Concretely, we add random noise of various distributions into the communicated prototypes and the original images, respectively. Table 15 shows that the performance of FedPCL remains high after injecting noise to the prototypes. Figure 6 visualizes an original image of a bike from Office-10 dataset and what it looks like after the noise injection operation. It is hard to tell the objects in the right column of Figure 6(c), but the accuracy only drops about 2%, compared to vanilla FedPCL.

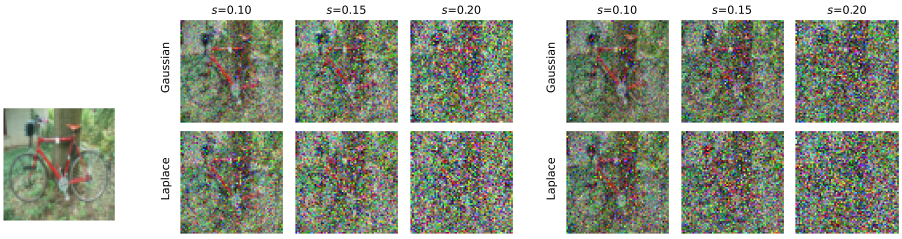
Table 14: Effect of the number of pre-trained backbones. Experiments are conducted on Digit-5 dataset under the feature & label shift non-IID setting where the Dirichlet parameter α is 1, and the number of clients is 5.

# of Backbones	# of Training Params.	# of Fixed Params.	Acc
1	133,632	11M	42.87(1.47)
2	264,704	22M	44.49(1.75)
3	395,776	33M	45.22(3.65)

In conclusion, FedPCL has the potential to combine with privacy-preserving techniques without an obvious decrease in performance.

Table 15: The performance of FedPCL on Office-10 dataset after incorporating privacy-preserving techniques. Here, we consider using multiple backbones under the label shift non-IID setting with $\alpha = 1$ and $m = 5$. s represents the scale parameter for the noise distribution generation and $p \in (0, 1)$ represents the perturbation coefficient of the noise.

Methods	Add Noise to	Noise Type	Acc(Std)
FedRep	/	/	37.95(0.91)
FedPCL	/	/	42.13(0.77)
FedPCL	Prototype	*Laplace* ($s = 0.05, p = 0.1$)	39.93(3.48)
		Gaussian ($s = 0.05, p = 0.1$)	40.04(2.09)
		Laplace ($s = 0.05, p = 0.2$)	40.24(3.01)
		Gaussian ($s = 0.05, p = 0.2$)	41.83(3.57)
	Image	*Laplace* ($s = 0.2, p = 0.1$)	38.29(2.87)
		Gaussian ($s = 0.2, p = 0.1$)	41.10(0.57)
		Laplace ($s = 0.2, p = 0.2$)	36.93(2.33)
		Gaussian ($s = 0.2, p = 0.2$)	40.14(0.78)



(a) Original (b) $p = 0.1$ (c) $p = 0.2$

Figure 6: Visualization of (a) an original image of a bike from Office-10 dataset; (b-c) applying noise injection privacy-preserving techniques to the original image. Specifically, given an original image x and a perturbation coefficient $p \in (0, 1)$, $\widetilde{x} = (1 - p)x + e$ is the image used for training. Here, we provide the results of two kinds of random noise, Gaussian (upper row) and Laplace noise (lower row). The value of p in Figure 6(b) and Figure 6(c) is 0.1 and 0.2, respectively.

B Discussion

B.1 Applying FedPCL to a Wider Range of Scenarios

Since there are many off-the-shelf foundation models available nowadays and there is a trend to use them in CV and NLP communities, it is a promising research direction to integrate those pre-trained foundation models into an FL framework. Compared with training from scratch, using pre-trained models can save more computation and communication resources.

Our proposed framework is limited to the cases where pre-trained models are available. This is mainly due to the fact that most types of data, e.g., image, text, graph, have corresponding pre-trained models nowadays. For the cases without pre-trained models, using multiple fixed encoders can be an alternative solution, which is another interesting problem and can be explored in the future.

We do think incorporating pre-trained models into existing learning frameworks is a promising trend in deep learning, especially when models are becoming larger and larger in scale and hard to train from scratch. So far, the idea to utilize pre-trained foundation models has been proposed for CV and NLP tasks and achieved certain improvement [45, 70, 71].

B.2 Comparison to Related Works

This paper has provided novel contributions in terms of (1) integrating pre-trained models into federated learning, (2) proposing a novel algorithm FedPCL which allows clients to share knowledge via prototype-based local contrastive learning.

Although prototypical learning and contrastive learning exist in prior work, they are still based on the learnable parameter aggregation scheme [35, 36, 44] or just use prototypes/contrastive learning to regularize the original local training [19]. Instead of synchronizing learnable parameters, our method allows each client to keep their own local parameters while extracting shared knowledge only by contrastive learning. Prototype is used as the information carrier to achieve that.

Some state-of-the-art PFL methods do not work well under our proposed lightweight framework. Their performance is mainly due to the following two reasons.

- Most of these baselines are designed based on the training-from-scratch framework where there are a large number of learnable parameters. The proposed new setting where most parameters are fixed is not friendly to some PFL methods.
- When there are feature/label shift non-IID across clients, the local performance might be further deteriorated after parameter aggregation. The deterioration can be more significant when only a small number of parameters are locally trained. Since most PFL methods are still based on the parameter aggregation scheme, their performances are inevitably affected.

Performance might be enhanced by decreasing the number of shared parameters, optimizing local training schemes, and introducing fine-tuning procedures during local training, etc.

There is also a branch of FL studying fine-tuning techniques for FL [72, 73]. These works mainly focus on how to adjust the local/global model to improve its representation ability on biased/generic data distribution, while our work utilizes fixed pre-trained foundation models and focuses on improving the fusing ability.

C Proof of Generalization Bound

Theorem C.1. (Generalization Bound of Alg. 1). *Consider an FL system with m clients. Let $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_m$ be the true data distribution and $\widehat{\mathcal{D}}_1, \widehat{\mathcal{D}}_2, \cdots, \widehat{\mathcal{D}}_m$ be the empirical data distribution. Denote the projector head h as the hypothesis from $\mathcal{H}$ and d be the VC-dimension of $\mathcal{H}$. The total number of samples over all clients is N. Then, with probability at least $1-\delta$:*

$$\begin{aligned}&\max_{(\theta_1,\theta_2,\ldots,\theta_m)} \left| \sum_{i=1}^{m} \frac{|D_i|}{N} L_{\mathcal{D}_i}\left(\theta_i; \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m}\right) - \sum_{i=1}^{m} \frac{|D_i|}{N} L_{\widehat{\mathcal{D}}_i}\left(\theta_i; \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m}\right) \right| \\ &\leq \sqrt{\frac{N}{2} \log \frac{(m+1)|\mathbb{C}|}{\delta}} + \sqrt{\frac{d}{N} \log \frac{eN}{d}}.\end{aligned} \tag{12}$$

Proof. We start from the McDiarmid's inequality that

$$\mathbb{P}\left[g\left(X_1, \ldots, X_n\right) - \mathbb{E}\left[g\left(X_1, \ldots, X_n\right)\right] \geq \epsilon\right] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n} c_i^2}\right) \tag{13}$$

when

$$\sup_{x_1,\ldots,x_n} \left|g\left(x_1, x_2, \ldots, x_n\right) - g\left(x_1, x_2, \ldots, x_n\right)\right| \leq c_i. \tag{14}$$

Eq. 13 equals to

$$\mathbb{P}\left[g\left(\cdot\right)-\mathbb{E}\left[g\left(\cdot\right)\right]\leq\epsilon\right]\geq 1-\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n}c_i^2}\right) \tag{15}$$

which means that with probability at least $1-\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n}c_i^2}\right)$,

$$g\left(\cdot\right)-\mathbb{E}\left[g\left(\cdot\right)\right]\leq\epsilon. \tag{16}$$

Let $\delta=\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{n}c_i^2}\right)$, the above can be rewritten as with probability at least $1-\delta$,

$$g\left(\cdot\right)-\mathbb{E}\left[g\left(\cdot\right)\right]\leq\sqrt{\frac{\sum_{i=1}^{n}c_i^2}{2}\log\frac{1}{\delta}}. \tag{17}$$

For prototype, by substituting $g\left(\cdot\right)$ with

$$\max_{(\theta_1,\theta_2,\ldots,\theta_m)}\left(\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\mathcal{D}_i}\left(\theta_i;\cdot\right)-\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\widehat{\mathcal{D}}_i}\left(\theta_i;\cdot\right)\right), \tag{18}$$

we can obtain that with probability at least $1-\delta$, the following holds for a specific prototype,

$$\begin{aligned}&\max_{(\theta_1,\theta_2,\ldots,\theta_m)}\left(\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\mathcal{D}_i}\left(\theta_i;\cdot\right)-\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\widehat{\mathcal{D}}_i}\left(\theta_i;\cdot\right)\right)\\&-\mathbb{E}\left[\max_{(\theta_1,\theta_2,\ldots,\theta_m)}\left(\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\mathcal{D}_i}\left(\theta_i;\cdot\right)-\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\widehat{\mathcal{D}}_i}\left(\theta_i;\cdot\right)\right)\right]\leq\sqrt{\frac{N}{2}\log\frac{1}{\delta}},\end{aligned} \tag{19}$$

Considering there are $(m+1)|\mathbb{C}|$ prototypes in total, by using Boole's inequality, with probability at least $1-\delta$, the following holds,

$$\begin{aligned}&\max_{(\theta_1,\theta_2,\ldots,\theta_m)}\left(\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\mathcal{D}_i}\left(\theta_i;\mathbb{C},\{\boldsymbol{C}_p\}_{p=1}^{m}\right)-\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\widehat{\mathcal{D}}_i}\left(\theta_i;\mathbb{C},\{\boldsymbol{C}_p\}_{p=1}^{m}\right)\right)\\&\leq\mathbb{E}\left[\max_{(\theta_1,\theta_2,\ldots,\theta_m)}\left(\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\mathcal{D}_i}\left(\theta_i;\mathbb{C},\{\boldsymbol{C}_p\}_{p=1}^{m}\right)-\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\widehat{\mathcal{D}}_i}\left(\theta_i;\mathbb{C},\{\boldsymbol{C}_p\}_{p=1}^{m}\right)\right)\right]\\&+\sqrt{\frac{N}{2}\log\frac{(m+1)|\mathbb{C}|}{\delta}},\end{aligned} \tag{20}$$

where N is the total number of samples over all clients.

$$\begin{aligned}&\mathbb{E}\left[\max_{(\theta_1,\theta_2,\ldots,\theta_m)}\left(\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\mathcal{D}_i}\left(\theta_i;\mathbb{C},\{\boldsymbol{C}_p\}_{p=1}^{m}\right)-\sum_{i=1}^{m}\frac{|D_i|}{N}L_{\widehat{\mathcal{D}}_i}\left(\theta_i;\mathbb{C},\{\boldsymbol{C}_p\}_{p=1}^{m}\right)\right)\right]\\&\leq\mathbb{E}\left[\sum_{i=1}^{m}\frac{|D_i|}{N}\max_{\theta_i}\left(L_{\mathcal{D}_i}\left(\theta_i;\mathbb{C},\{\boldsymbol{c}_p\}_{p=1}^{m}\right)-L_{\widehat{\mathcal{D}}_i}\left(\theta_i;\mathbb{C},\{\boldsymbol{c}_p\}_{p=1}^{m}\right)\right)\right]\\&\overset{(a)}{\leq}\sum_{i=1}^{m}\frac{|D_i|}{N}\Re_i(\mathcal{H})\\&\leq\sum_{i=1}^{m}\frac{|D_i|}{N}\sqrt{\frac{d}{|D_i|}\log\frac{e|D_i|}{d}}\\&\leq\sum_{i=1}^{m}\frac{|D_i|}{N}\sqrt{\frac{d}{|D_i|}\log\frac{eN}{d}}\\&\overset{(b)}{\leq}\sqrt{\frac{d}{N}\log\frac{eN}{d}}\end{aligned} \tag{21}$$

where $\mathcal{H}$ is the hypothesis set of projector head h, d is the VC-dimension of $\mathcal{H}$, N is the total number of samples over all clients. (a) follows from the definition of Rademacher complexity

$$\widehat{\Re}_S(\mathcal{G})=\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{g\in\mathcal{G}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i g\left(z_i\right)\right] \tag{22}$$

where m is the number of samples, σ_i refers to independent uniform random variable taking value in $\{-1, +1\}$, and (b) follows from Jensen's inequality.

So,

$$\begin{aligned}
&\max_{(\theta_1,\theta_2,\ldots,\theta_m)} \left| \sum_{i=1}^{m} \frac{|D_i|}{N} L_{\mathcal{D}_i}\left(\theta_i; \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m}\right) - \sum_{i=1}^{m} \frac{|D_i|}{N} L_{\widehat{\mathcal{D}}_i}\left(\theta_i; \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^{m}\right) \right| \\
&\leq \sqrt{\frac{N}{2} \log \frac{(m+1)|\mathbb{C}|}{\delta}} + \sqrt{\frac{d}{N} \log \frac{eN}{d}}.
\end{aligned} \tag{23}$$

□