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

Anonymous Authors[1]

Abstract

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. 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, 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. We perform a thorough evaluation of the proposed FedPCL in the lightweight framework, measuring 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 (McMahan et al., 2017). 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 (Kairouz et al., 2019). 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 (Zhang et al., 2021a; Fallah et al., 2020; T Dinh et al., 2020; Oh et al., 2022). So far, FL and PFL have been widely used in real-world tasks, including computer vision (Park et al., 2021), natural language processing (Lin et al., 2021), and graph data mining (Xie et al., 2021).

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 models. Also, it makes it possible to utilize large-scale pre-trained models, e.g., VisionTransformer (Dosovitskiy et al., 2020) and Swin Transformer (Liu et al., 2021), 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.

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 (Snell et al., 2017; Luo et al., 2021; Tan et al., 2022), 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 (Quattoni et al., 2008) and multi-task learning (Kang et al., 2011) scenarios. To efficiently extract the useful shared information learned from the pre-trained models

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022. Do not distribute.

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:

2. Problem Formulation

We formulate our proposed lightweight FL framework which integrates off-the-shelf pre-trained models as fixed backbones and learns to fuse them adaptively.

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* $h(\cdot; \theta_i) : \mathbb{R}^{K \times d_e} \to \mathbb{R}^{d_h}$, 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.

Definition 2.1. 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

$$r(\mathbf{x}; \Phi^*) := \text{concat}\left(r_1(\mathbf{x}; \phi_1^*), \ldots, r_K(\mathbf{x}; \phi_K^*)\right). \quad (1)$$

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

$$z(\mathbf{x}) = h\left(r_{\mathbf{x}}; \theta_i\right). \quad (2)$$

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

$$\min_{\{\theta_1, \theta_2, \ldots, \theta_m\}} \sum_{i=1}^{m} \frac{|D_i|}{N} \mathbb{E}_{(\mathbf{x},y) \in D_i}\left[L_i\left(\theta_i; z\left(\mathbf{x}\right), y\right)\right]$$
$$\text{s.t. } z(\mathbf{x}) = h\left(r(\mathbf{x}; \Phi^*); \theta_i\right) \text{ where } \Phi^* = \{\phi_1^*, \phi_2^*, \cdots, \phi_K^*\}. \quad (3)$$

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 $z(\mathbf{x})$.

3. FedPCL

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.

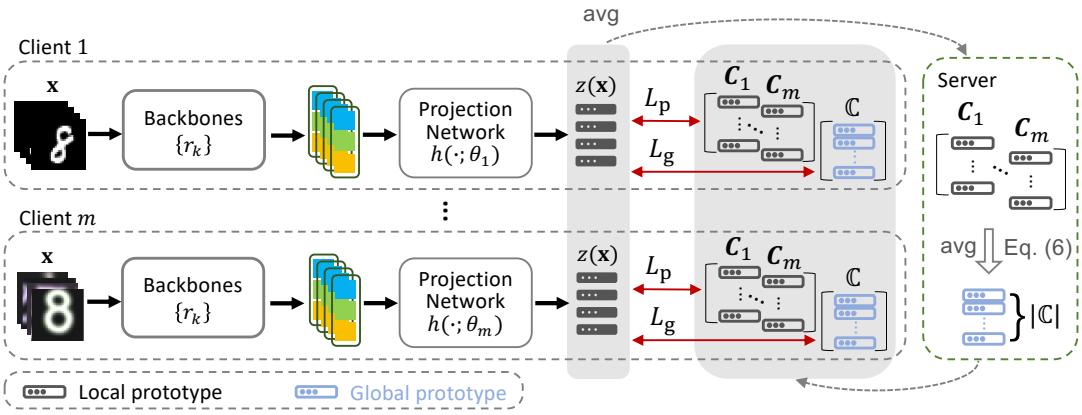


Figure 1. An overview of the proposed lightweight federated learning framework. This example assumes that for each client, there are three pre-trained backbones, with the block in different colors illustrating their backbone-specific representation.

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,

$$C_i^{(j)} := \frac{1}{|D_{i,j}|} \sum_{(\mathbf{x},y)\in D_{i,j}} z\left(\mathbf{x}\right), \tag{4}$$

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

$$\bar{C}^{(j)} := \frac{1}{|\mathcal{N}_j|} \sum_{i\in\mathcal{N}_j} \frac{|D_{i,j}|}{N_j} C_i^{(j)}, \tag{5}$$

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.

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

$$L_{\mathrm{g}} = \sum_{(\mathbf{x},y)\in D_i} -\log \frac{\exp\left(z_{\mathbf{x}} \cdot \bar{C}^{(y)}/\tau\right)}{\sum_{y_a\in A(y)} \exp\left(z_{\mathbf{x}} \cdot \bar{C}^{(y_a)}/\tau\right)}. \tag{6}$$

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 (Zhang et al., 2021b; Chen et al., 2020; Khosla et al., 2020). 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

$$L_{\mathrm{p}} = \sum_{(\mathbf{x},y)\in D_i} -\frac{1}{m}\sum_{p=1}^{m} \log \frac{\exp\left(z_{\mathbf{x}} \cdot C_p^{(y)}/\tau\right)}{\sum_{y_a\in A(y)} \exp\left(z_{\mathbf{x}} \cdot C_p^{(y_a)}/\tau\right)}. \tag{7}$$

For the i-th client, the local objective function in Eq. (3) is defined as a combination of L_{g} and L_{p} in the following form,

$$\begin{aligned} L\left(\theta_i; z\left(\mathbf{x}\right), y, \mathbb{C}, \{\boldsymbol{C}_p\}_{p=1}^m\right) &= L_{\mathrm{g}}\left(\theta_i; z\left(\mathbf{x}\right), y, \mathbb{C}\right) \\ &+ L_{\mathrm{p}}(\theta_i; z\left(\mathbf{x}\right), y, \{\boldsymbol{C}_p\}_{p=1}^m). \end{aligned} \tag{8}$$

4. Experiments

4.1. Experimental Setup

Datasets and Non-IID Settings. We evaluate our proposed framework on the following three benchmark datasets:

Table 1. 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

Digit-5 (Zhou et al., 2020), Office-10 (Gong et al., 2012), and DomainNet dataset (Peng & Saenko, 2018). To mimic non-IID scenarios in a more general way, we investigate three different non-IID settings. 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 (McMahan et al., 2017), pFedMe (Dinh et al., 2020), PerFedAvg (Fallah et al., 2020), FedRep (Collins et al., 2021), FedProto (Tan et al., 2022), and Solo, i.e., training independently within each client. Details about the model and each dataset can be found in Appendix A.1.

4.2. Performance Comparison

Table 2. 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)

Table 1 and Table 2 report the results of our method and baselines in mean (std) format over clients with three independent runs. 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.

Table 3. 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)

4.3. Integrating Backbones with Various Architectures

In Table 3, we also show that our proposed lightweight framework is capable of (i) fusing representations generated by backbones with different architectures, i.e., a two-layer CNN, AlexNet (Krizhevsky et al., 2012), and VGGNet (Simonyan & Zisserman, 2014). (ii) integrating large-scale pre-trained models, i.e., ViT (Dosovitskiy et al., 2020), to enhance the performance without huge computation resources to train or fine-tune them.

5. Conclusion

To address the problems on excessive computation and communication demands in current FL 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.

References

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Collins, L., Hassani, H., Mokhtari, A., and Shakkottai, S. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning*, 2021.

Dinh, C. T., Tran, N. H., and Nguyen, T. D. Personalized federated learning with moreau envelopes. In *NeurIPS*, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning: A meta-learning approach. In *Advances in Neural Information Processing Systems*, 2020.

Gong, B., Shi, Y., Sha, F., and Grauman, K. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 2066–2073. IEEE, 2012.

Jongejan, J., Rowley, H., Kawashima, T., Kim, J., and Fox-Gieg, N. The quick, draw!-AI experiment. *Mount View, CA, accessed Feb*, 17(2018):4, 2016.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., et al. Advances and open problems in federated learning. *arXiv:1912.04977*, 2019.

Kang, Z., Grauman, K., and Sha, F. Learning with whom to share in multi-task feature learning. In *ICML*, 2011.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2019.

Lin, B. Y., He, C., Zeng, Z., Wang, H., Huang, Y., Soltanolkotabi, M., Ren, X., and Avestimehr, S. Fednlp: A research platform for federated learning in natural language processing. *arXiv preprint arXiv:2104.08815*, 2021.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. In *NeurIPS*, 2020.

Liu, L., Hamilton, W. L., Long, G., Jiang, J., and Larochelle, H. A universal representation transformer layer for few-shot image classification. In *International Conference on Learning Representations*, 2020.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

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

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

McMahan, H. B., Moore, E., Ramage, D., et al. Communication-efficient learning of deep networks from decentralized data. *AISTATS*, 2017.

Oh, J., Kim, S., and Yun, S.-Y. FedBABU: Toward enhanced representation for federated image classification. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=HuaYQfggn5u.

Park, S., Kim, G., Kim, J., Kim, B., and Ye, J. C. Federated split task-agnostic vision transformer for covid-19 cxr diagnosis. *Advances in Neural Information Processing Systems*, 34, 2021.

Peng, X. and Saenko, K. Synthetic to real adaptation with generative correlation alignment networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1982–1991. IEEE, 2018.

Quattoni, A., Collins, M., and Darrell, T. Transfer learning for image classification with sparse prototype representations. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2008.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *NeurIPS*, pp. 4077–4087, 2017.

T Dinh, C., Tran, N., and Nguyen, T. D. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems*, 2020.

Tan, Y., Long, G., Liu, L., Zhou, T., Lu, Q., Jiang, J., and Zhang, C. FedProto: Federated prototype learning over heterogeneous devices. In *AAAI*, 2022.

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

Xie, H., Ma, J., Xiong, L., and Yang, C. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

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

Zhang, O., Wu, M., Bayrooti, J., and Goodman, N. Temperature as uncertainty in contrastive learning. *arXiv preprint arXiv:2110.04403*, 2021b.

Zhou, K., Yang, Y., Hospedales, T., and Xiang, T. Learning to generate novel domains for domain generalization. In *European Conference on Computer Vision*, pp. 561–578. Springer, 2020.

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 2. 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 2.

- *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 (Lin et al., 2020; Wang et al., 2020), 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 3. 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 3.

A.1.3. Model Architecture.

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 (Kingma & Ba, 2014) 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.

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 (Jongejan et al., 2016), Aircraft (Maji et al., 2013), and CU-Birds (Liu et al., 2020) public dataset, respectively. The model architecture following the backbone module is shown in Table 4.

Table 4. 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 4.2. Table 5, 6, 7 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 5. Detailed statistics for three benchmark datasets in feature shift non-IID, label IID setting (Table 1).

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

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 7. Detailed statistics for three benchmark datasets in feature & label shift non-IID setting (Table 8).

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]

A.2. Additional Experiments

A.2.1. Performance under Three Non-IID Settings.

Table 8 shows the performance of FedPCL and baseline methods under label shift non-IID scenarios.

Table 8. 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)

A.2.2. Fairness across Clients.

Following the metrics in (Li et al., 2019), we verify the advantage of FedPCL in fairness and report the results over 40 and 80 clients in Table 9 and Table 10, respectively. We list the average, the worst 10%, 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) (Li et al., 2019). 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 9 and 10.

Table 9. 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 10. 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. 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 11, our prototype-wise supervised contrastive loss outperforms others.

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;

Table 11. 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 (Tan et al., 2022)	43.94(3.02)
Supervised Contrastive (Khosla et al., 2020)	42.18(3.25)
Ours	**45.22(3.65)**

(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 12, 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 12. 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.

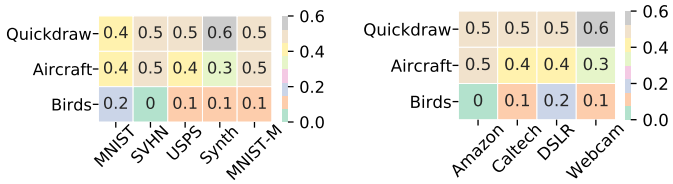
Prototypes	Feature shift non-IID	Label shift non-IID	Feature & Label shift non-IID
Global only	54.69(0.14)	44.75(1.77)	43.79(4.09)
Local only	55.01(0.10)	44.52(1.73)	43.08(3.87)
Global and local	**55.34**(0.34)	**45.35**(1.58)	**45.22**(3.65)

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 13 indicate that more pre-trained backbones can lead to better performance but consume more computing resources and memory space.

Table 13. 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)

A.2.4. Visualizing the Fusing Results of FedPCL



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

Figure 4. 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.

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 4 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 4(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.

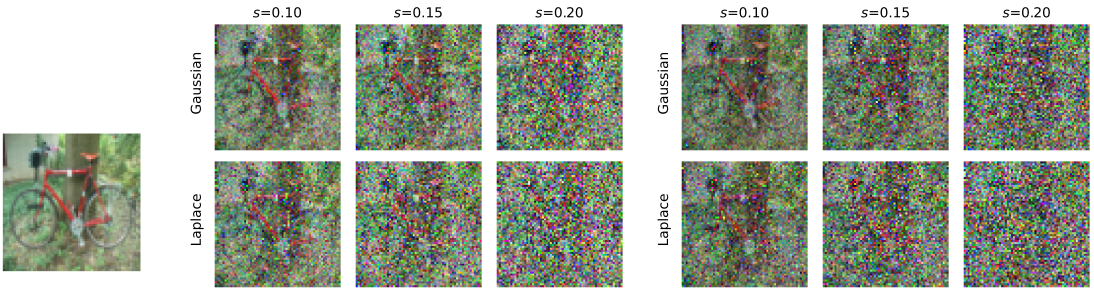
We find that the backbone pre-trained on Quickdraw contributes more to the fused prototype, while the backbone pre-trained on Birds contributes the least. Clients who own a specific dataset show different levels of utilization when fusing the representations from various backbones.

A.2.5. 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 14 shows that the performance of FedPCL remains high after injecting noise to the prototypes. Figure 5 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 5(c), but the accuracy only drops about 2%, compared to vanilla FedPCL. In conclusion, FedPCL has the potential to combine with privacy-preserving techniques without an obvious decrease in performance.

Table 14. 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 5. 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 5(b) and Figure 5(c) is 0.1 and 0.2, respectively.