# **PROMPTFL: Let Federated Participants Cooperatively Learn Prompts Instead** of Models — Federated Learning in Age of Foundation Model

Tao Guo, Song Guo, Junxiao Wang, Wenchao Xu

Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China

#### Abstract

Quick global aggregation of effective distributed parameters is crucial to federated learning (FL), which requires adequate bandwidth for parameters communication and sufficient user data for local training. Otherwise, FL may cost excessive training time for convergence and produce inaccurate models. In this paper, we propose a brand-new FL framework, PROMPTFL, that replaces the federated model training with the federated prompt training, i.e., let federated participants train prompts instead of a shared model, to simultaneously achieve the efficient global aggregation and local training on insufficient data by exploiting the power of foundation models (FM) in a distributed way. PROMPTFL ships an off-theshelf FM, i.e., CLIP, to distributed clients who would cooperatively train shared soft prompts based on very few local data. Since PROMPTFL only needs to update the prompts instead of the whole model, both the local training and the global aggregation can be significantly accelerated. And FM trained over large scale data can provide strong adaptation capability to distributed users tasks with the trained soft prompts. We empirically analyze the PROMPTFL via extensive experiments, and show its superiority in terms of system feasibility, user privacy, and performance.

### Introduction

The ever-growing edge devices, e.g., smart phones, autonomous vehicles, etc., are generating various types and rapidly growing big data. Artificial intelligence (AI) has shown its success to mine the big edge data and produce accurate models that can replace human decisions timely and properly. Traditional AI paradigms require to gather all raw data to a cloud center for centralized training, which can incur significant communication overhead and potential privacy leakage, and thus are not desirable for edge users.

Federated learning (FL) has emerged to conduct distributed machine learning that allows multiple edge users to jointly train a shared model without sharing their raw data, which has been demonstrated great success in many edge applications, e.g., input word prediction, voice assistant, etc. (Hard et al. 2018; Liang et al. 2020), that can mine massive distributed data without exposing users' privacy, and thus are widely applied in various edge scenarios. The FL training process comprises of two iterative phases, i.e., local training and global aggregation. Thus the learning performance is determined by both the effectiveness of the parameters from local training and smooth aggregation of them. However, these two requirements are not easy to satisfy in edge environment, i.e., edge users often have limited bandwidth and insufficient data, which can cause inefficient parameters aggregation, excessive training time and reduced model accuracy.

Existing research efforts have focused on improving the FL optimization process (Li et al. 2020; Zhao et al. 2018) or refining model architectures (Qu et al. 2022), but this does not change that FL inherently entails a large number of communication rounds and a large amount of labeled data for training, which are often unavailable for edge users. Such challenges are particularly salient under the combined effect of a long training process and unfavorable factors such as non-IID and unbalanced data, limited communication bandwidth, and unreliable and limited device availability.

We revisits the question of how FL mines the distributed data in iterative training rounds, and exploit the emerging foundation model (FM) to optimize the FL training. FM refers to large neural model that trained on large scale data and has strong adaptation capability for various downstream tasks. We let federated participants cooperatively learn prompts instead of models to unleash the power of FM in a distributed way, whereby both the local training and global aggregation can be significantly accelerated.

We investigate the behavior of the nascent model in a standard FL setting using popular off-the-shelf FMs, *e.g.*, CLIP, and methods for FM adaptation. We propose PROMPTFL, a framework that replaces existing federated model training with prompt training, i.e., FL clients train prompts instead of a model, which can simultaneously exploit the insufficient local data and reduce the aggregation overhead. PROMPTFL ships an off-the-shelf public CLIP to users and apply continuous prompts (*a.k.a.* soft prompts) for FM adaptation, which requires very few data samples from edge users. The framework is technically very simple but effective. The focus of our investigation is whether it meets the key principles:

- Feasibility. What are the system costs? We examine the feasibility of PROMPTFL on modern hardware, focusing conservatively on personal cell phones. We demonstrate the feasibility of the system in terms of overhead in communication, training, and inference dimensions.
- **Performance.** Are PROMPTFL competitive with FL? FL does not baseline against any such approach, so we im-

plement a proof-of-concept in the framework, spanning a range of popular image classification tasks. We observe PROMPTFL competitive with strong FL baselines.

• **Privacy.** Is PROMPTFL privacy-preserving? We show that PROMPTFL keeps data on each device private, aiming to learn global prompts updated only by communicating gradients rather than the data itself, and thus not less private than FL.

## **Preliminaries**

#### Foundation Model

AI is going through a paradigm shift with the rise of models (*e.g.*, BERT, GPT-3, CLIP, DALL-E·2) trained on broad data using self-supervision at scale that can be adapted to a wide range of downstream tasks. Researchers call these models foundation models (FMs) to emphasize their key core. From a technical standpoint, FMs are not new. However, the sheer size and scope of FMs over the past few years has expanded our imagination of what is possible. FMs are scientifically interesting for their impressive performance and capabilities, but what makes them critical to research is that they are rapidly being integrated into real-world deployments of AI systems, with profound implications for users.

**CLIP** Contrastive Language-Image Pre-Training (CLIP) is a neural network trained on hundreds of millions of (image, caption) pairs (Radford et al. 2021). CLIP encodes images and captions separately as vectors, enabling users with visual modality samples to retrieve, score, or classify samples from textual modalities. Models are often very fragile and only know very specific things you trained them to do. CLIP extends the knowledge of classification models to a wider range of things by leveraging semantic information in text. Standard classification models completely discard the semantic meaning of class labels and simply enumerate numeric classes behind the scenes; CLIP works by understanding the meaning of the classes. ALIGN is another CLIP-like vision-language pre-training (Jia et al. 2021).

**Image Classification with CLIP** CLIP pre-trains an image encoder and a text encoder to predict which images are paired with which texts. We can use this behavior to convert the CLIP to an image classifier. We may convert all [class] to captions such as "picture of [class]" and predict the caption class CLIP estimates the best pairing with the given image. In many previous works, this has involved prompt template engineering, in which human engineers or algorithms search for the best template for each class (Fürst et al. 2021; Li et al. 2021; Singh et al. 2022; Yuan et al. 2021).

## **Federated Learning**

Recent neural models require large amounts of training data (Dodge et al. 2020), and users typically hold limited-scale labeled data. To address the challenge of lack of sufficient data for individual users, federated learning of data across multiple privacy spheres (*i.e.*, users) has become a popular framework.

The term *federated learning* was introduced by (McMahan et al. 2017). In a centralized setting, the federated server

initially sends global model parameters to each client. After training with local data, the participants are only required to share gradients for model updates. Then the server aggregates the gradients and transmits the updated model back to each client. More specifically, federated learning is a machine learning setting where a set of n clients (*e.g.*, mobile devices) collaboratively train a model under the orchestration of a federated server (*e.g.*, service provider), while the training data of clients is stored locally and not exchanged (Kairouz et al. 2021). The federated server orchestrates the collaborative training process, by repeating the following steps until training is converged:

**Client Selection** Given the unstable client availability, for the round t of federated learning, the federated server samples a small subset of m clients meeting eligibility requirements out of all n clients to participate in the learning.

**Local Training** Upon notification of being selected at the round t, each selected client downloads the current parameters  $\theta$  of global model and a training program from the federated server. Each selected client locally computes an update to the global model on its local training data by executing the training program. More specifically, the gradients updated at one client (denoted as G), are computed by  $\frac{\partial \ell(X,y,\theta)}{\partial \theta}$ , where X, y denote the batches of training data and corresponding labels, and  $\ell(\cdot)$  refers to the loss function.

The gradients G in typical federated learning settings are the minimum that must be shared to the server, corresponding to FedSGD method. In FedAVG (McMahan et al. 2017), models are consecutively updated on more batches of local data, which can be several epochs of training, and then shared. We note that a common way is to share the updated model  $\theta + G$ , but this practically amounts to sharing G since all participants know  $\theta$ .

**Global Aggregation** Upon having received local updates from m clients, the federated server aggregates these updates and update its global model, and initiates next round learning. In addition to the federated learning framework that relies on the centralized server node, there are also some federated learning implementations based on the decentralized framework (Roy et al. 2019; Lalitha et al. 2018; Hu, Jiang, and Wang 2019). This means that the aggregation of gradients does not necessarily occur in a fixed federation server, but may also occur in some clients.

## **Prompt-Based Federated Learning**

We hypothesize that an off-the-shelf public CLIP-like model is shipped to the user device. The CLIP-like model is a powerful image classifier that utilizes linguistic knowledge to classify images. In other words, CLIP already knows a lot about the content of images. But to unleash the power of CLIP in FL, we need to take advantage of something called prompt engineering that was mentioned in the preliminaries.

## **Prompt Engineering**

The prompting function  $f_{\text{prompt}}(\cdot)$  is applied to modify the class label **y** into a prompt  $\mathbf{y}' = f_{\text{prompt}}(\mathbf{y})$ . The most natural form of implementing a prompting function is to manually



Figure 1: Framework and workflow of PROMPTFL. Each client includes a prompt learner (with only a small amount of trainable parameters) and an out-of-the-box CLIP (with backbone frozen). The federated server aggregates only the parameter updates of prompt learners over multiple users, and transmit the updated parameters back to each user.

create an intuitive template based on human introspection. For example, as referred in (Brown et al. 2020) we may manually craft prefix prompts to handle an image classification task by using templates like "picture of [class]" or "a photo of a [class]". Based on that, many approaches have been proposed to automate the template design process.

Specifically, the automated prompting can be further separated into discrete prompts (*a.k.a.* hard prompts), where the prompt is an actual text string, and continuous prompts (*a.k.a.* soft prompts), where the prompt is performed directly in the embedding space of the model (Liu et al. 2021). Discrete prompts constraint that the embeddings of template words be the embeddings of natural language words (Shin et al. 2020; Gao, Fisch, and Chen 2021). Thus, discrete prompting is a clear way to visualize what "word" are learned for the vectors (Deng et al. 2022).

Our paper adopts continuous prompts instead of discrete prompts in FL for the reason that (1) Our purpose of prompt construction is to find a way to enable FL to efficiently perform the image classification tasks, not for human interpretation, there is no need to limit prompts to human-interpretable natural language. (2) The templates have their own parameters that can be tuned based on training data from the user, which is a natural compatibility connecting FL and prompting. More related topics of continuous prompts can refer to (Li and Liang 2021; Lester, Al-Rfou, and Constant 2021; Tsimpoukelli et al. 2021; Hambardzumyan, Khachatrian, and May 2021; Zhou et al. 2021).

#### Framework to Learn Prompts in FL

The framework of PROMPTFL is presented in Figure 1. Each FL client consists of a prompt learner and an out-of-the-box CLIP model. PROMPTFL introduces only a small amount of trainable parameters in the prompt learner while keeping the CLIP backbone frozen. In other words, during local training, only the parameters of the prompt learner are updated while the whole CLIP model turns off gradients in both the image and the text encoder. The federated server is designed to aggregate only the parameter updates of prompt learners over multiple users, and transmit the updated parameters back to each user. Thus, PROMPTFL evolves the goal of FL from

model training to prompt learner training.

The CLIP backbone comprises two encoders, one for images and the other for texts. The image encoder will map high-dimensional images into a low-dimensional embedding space. The network of the image encoder can take the form of a CNN such as ResNet50 (He et al. 2016) or Vision Transformer (Dosovitskiy et al. 2021). The text encoder will generate text representations from input. The network of the text encoder is a Transformer (Vaswani et al. 2017).

**Prompt Learner** Given a pre-trained CLIP backbone, the input to the text encoder is designed in the form of [prompt vectors][class]. Inspired by the simple and straightforward prompt design in (Zhou et al. 2021), we introduce a set of p continuous embeddings of dimension d in the [prompt vectors]. d is same as the dimension of word embeddings in the text encoder, thus 512 by default. p is a hyperparameter specifying the number of embeddings. In a word, [prompt vectors] are p learnable d-dimensional vectors.

Given a batch of image-text pairs, CLIP will maximize the cosine similarity for matched pairs while minimize the cosine similarity for all other unmatched pairs. Since CLIP is pre-trained to predict whether an image matches a textual description, it can compute the classification loss and logits by aligning the two embedding spaces for images and texts (*i.e.*, [prompt vectors][class]) respectively. Formally, let  $q(\cdot)$ and  $h(\cdot)$  be the feature extraction function of the image and text encoder. Let  $w_i = h(\mathbf{P}, \mathbf{K}_i)$  be the weight vector generated by the text encoder, where  $i \in [1, k]$ . k denotes the number of classes and each  $(\mathbf{P}, \mathbf{K}_i)$  is derived from the prompt in the form of [prompt vectors][class]<sub>*i*</sub>, where [class]<sub>*i*</sub> is replaced by the word embedding vector of specific class label name. Let  $\cos[\cdot|\cdot]$  denote the cosine similarity used by CLIP. By forwarding a  $(\mathbf{P}, \mathbf{K}_i)$  and an image **x**, the classification prediction probability and logits are computed as

$$\mathbf{p}(\mathbf{y} = i | \mathbf{x}) = \frac{\exp\left(\cos[g(\mathbf{x})|h(\mathbf{P}, \mathbf{K}_i)]\right)}{\sum_{j=1}^{k} \exp\left(\cos[g(\mathbf{x})|h(\mathbf{P}, \mathbf{K}_j)]\right)}, \quad (1)$$

where **P** is the only part that is updated in local back propagation and aggregated in the federated server.

Frameworks	PromptFL	Federated Learning	Modern Mobile Phone Hardware			
Dimensions	(150M parameter model)	(100M parameter model)	(E. Freedman 2021)			
Communication	600 MB File Download	40 GB File Download +	54 Mbps Downstream RateLimit			
	1 4 Minutes	40 GB File Upload	12 Mbps Upstream RateLimit			
	1.4 Minutes	Totally 9 Hours	(O'Dea 2021)			
Training	Almost None	4 TFLOPs	1.5 TFLOPs, 8 GB RAM			
Inference	60 GFLOPs	40 GFLOPs	1.5 TFLOPs, 8 GB RAM			
Storage	600 MB on Disk	400 MB on Disk	1 TB on Disk			

Table 1: System cost comparison. Assumes 32 local training batch size, 1 local training epoch, 100 total communication rounds for FL. Assumes 196 input sequence length, full precision for PROMPTFL and FL.

Prompting are particularly useful in the FL case, as using prompts to push the model in the correct direction is particularly effective. This feature enables prompting to converge quickly in FL, requires less data per user, and is less affected by adverse factors in the process, *e.g.*, non-IID and unbalanced data, limited communication bandwidth, and unreliable and limited device availability. In this paper, the prompt learner employed in PROMPTFL though simple and straightforward as a bridge to our core idea is easy to follow. We also envision that more complex and effective bridges would be there to replace the role and should be a valuable direction.

## System Feasibility

We examine the feasibility of PROMPTFL on modern hardware, focusing conservatively on personal cell phones. We notice that users can access GPUs from their mobile phones. Enterprise users have more abundant resources. Without loss of generality, we take a 100M parameter model for FL and 150M parameter CLIP backbone for image similarity-search of PROMPTFL. The prompt learner introduces only a small number of parameters, that can be ignored. We assume that the FL configures 32 local training batch size, 1 local training epoch, and 100 total communication rounds, which suggested in (Qu et al. 2022). We also assume that both FL and PROMPTFL configure 196 input sequence length and the full precision. The system cost comparison is summarized in Table 1 along the following dimensions:

**Communication** The average download speed within the globe for mobile internet was 54 Mbps, and the average upload speed for mobile internet was 12 Mbps that reported by 2021 (O'Dea 2021). PROMPTFL requires locally downloading while FL requires communicating the model repeatedly between users and the federated server. Thus, the communication cost in terms of file transfer volume is that it takes only 1.4 minutes to transfer 600MB for PROMPTFL, and 9 hours for FL to transfer 40GB.

**Training and Inference** FL requires FLOPs computed by  $(2 \times 3 \times \text{model parameters} \times \text{local training epoch} \times \text{local training batch size} \times \text{input sequence length})$  for training, while the training FLOPs of PROMPTFL is much smaller and negligible compared to FL. For both PROMPTFL and FL, inference requires FLOPs computed by  $(2 \times \text{model parameters} \times \text{input sequence length})$ , in the setting where the key and value vectors for attention computation are cached. Compared to the acceptable computational and storage costs, the RAM on the

modern cell phones is a key bottleneck. We believe that this bottleneck will no longer be a problem in the near future as the techniques evolve: (1) Out-of-the-box offloading inference (Rajbhandari et al. 2021). (2) Trends for more RAM (Patterson 2022) and tiny CLIPs (Sisodia 2021). (3) Inference with quantization methods (Gholami et al. 2021).

**Compatibility** Apart from image classification, many different vision tasks are compatible with PROMPTFL, such as object detection (Gu et al. 2021), video understanding (Xu et al. 2021) and visual question answering (Shen et al. 2021). This means that the system cost of PROMPTFL is shared by many tasks. The prompt learner incurs these costs per personal task specific user subset requires. PROMPTFL is thus competitive in terms of economics.

## **Privacy Concerns**

As we have outlined in the framework, PROMPTFL achieves to train prompts in concert with the federated server. Each participant user only needs to upload its local parameter update of the prompt learner rather than the raw data of images. Such a method avoids leakage of raw images, thereby better adapting to the privacy-preserving settings of the FL. On the other hand, the parameters of prompt learner only describes the correlation between classes and textual prompts, and do not directly contain any visual feature embeddings. Also, the parameters of prompt learner are static (*i.e.*, inputagnostic) across the training data. This is useful when faced with a server that wants to recover the raw data from an update (Zhu, Liu, and Han 2019).

Inference APIs While pre-trained CLIPs are available for download at the time of writing this paper, high-performance models in these domains are often costly to train. For example, the CLIP model trained on 400 million labeled images. The training process took 30 days across 592 V100 GPUs (Radford et al. 2021). This would have cost million dollars to train on AWS on-demand instances. The value of these models and their exposure over publicly-accessible APIs make us rethink the framework of PROMPTFL. As illustrated in Figure 1, we hypothesize that the model APIs typically return low-dimensional outputs like confidence scores or logits, so information leakage is significantly reduced (Dziedzic et al. 2022). In such a case, the prompt learner can still be trained normally, because the CLIP backbone is kept frozen during the training process. The difference is that users need to make queries to the model APIs with their private images.

		IID			Extreme Non-IID				LEARNABLE		
BENCHMARK	Method	Accuracy $\uparrow$		<i>F-1</i> ↑		Accuracy $\uparrow$		$F$ -1 $\uparrow$		PARAMETERS	
		Rn50	Vit	Rn50	Vit	Rn50	Vit	Rn50	Vit	Rn50	Vit
Caltech101	PromptFL	90.18	94.65	86.09	91.76	88.72	94.12	83.98	90.48	0.1%	0.01%
	Finetuning FL	90.02	93.1	84.72	89.07	29.78	29.89	12.2	12.2	100%	100%
	FL from scratch	32.41	32.49	10.51	12.89	-	-	-	-	100%	100%
Flowers102	PromptFL	88.14	90.5	87.62	90.14	66.3	74.75	60.14	69.13	0.1%	0.01%
	Finetuning FL	92.6	91.9	91.56	<b>90.7</b>	24.4	24.5	10.68	11.18	100%	100%
	FL from scratch	33.17	38	25.7	32.5	-	-	-	-	100%	100%
OxfordPets	PROMPTFL	88.5	92.89	88.44	92.8	87.03	89.51	86.85	88.45	0.1%	0.01%
	Finetuning FL	90.38	92.1	90.06	91.92	24.83	25.27	11.3	11.93	100%	100%
	FL from scratch	10.25	8.722	7.624	8.318	-	-	-	-	100%	100%
Food101	PROMPTFL	78.0	85.75	77.9	85.66	78.1	85.88	78.03	85.8	0.1%	0.01%
	Finetuning FL	69.28	76.68	69.08	76.85	22.92	23.8	10.19	10.73	100%	100%
	FL from scratch	21.11	21.03	19.75	19.92	-	-	-	-	100%	100%

Table 2: **Performance of PROMPTFL against existing FL framework on the four datasets**. The table report the accuracy and F-1 score according to the corresponding backbone and method. The best score of each group appears in bold. Compared with finetuning and training from the scratch, PROMPTFL only update  $0.01\% \sim 0.1\%$  parameters, however, still outperforms other methods across datasets. Given the poor result in training from the scratch even with iid mode, we assume that the performance from the non-iid setting can be even wore, so we omit the result in this row.

Some lightweight secure inference techniques like (Liu et al. 2020) can be used in the framework to protect privacy.

## **Experiments**

Our experiments aim to answer the following research questions that are important for the practical deployment of FL methods, while also contributing to our understanding of the PROMPTFL paradigm.

- Is PROMPTFL able to train a competitive performance in FL as compared to which have been the de-facto method on image classification tasks?
- Is PROMPTFL capable of handling heterogeneous data distributions (*a.k.a.* non-IID settings) across clients?
- Is PROMPTFL competitive with the de-facto method in terms of computational communication overhead?
- What is the difference between PROMPTFL and the finetuning of visual pre-trained models in FL?
- What practical tips help the service provider and participants deploy PROMPTFL in FL?

### **Experimental Setup**

**Datasets** We select a representative collection of recognition datasets used in CLIP as our benchmarks. **General Objects:** Caltech101 (Fei-Fei, Fergus, and Perona 2004) for general object detection. **Fine-grained Categories:** Flowers102 (Nilsback and Zisserman 2008), OxfordPets (Parkhi et al. 2012) and Food101 (Bossard, Guillaumin, and Gool 2014) for fine-grained classification from diversified categories.

**Baselines** As compared to our proposed PROMPTFL, we choose current representative framework in FL, FedAVG,

by updating and averaging the model weights collaboratively among server and clients. We compare both training from the scratch and fine-tuning with pretrained models as our baseline method. We select the most prevailing models, Vit\_b16 and Retnet50, as our backbone in both our image encoder of PROMPTFL and the corresponding backbone in the baseline method.

Fine-tuning vs. Prompting How does the prompting differ from the existing adaptation method in FL? Currently in vision, the standard adaptation method is fine-tuning. Therefore we consider fine-tuning as the de-facto way of adapting visual pre-trained models in FL. Fine-tuning is highly flexible in its usage: it can adapt the pre-trained models to new input domains or new tasks with different output semantics. Yet it also requires some level of access to the pre-trained models: often entire parameters. Unlike fine-tuning, prompting adapts the inputs to a pre-trained model by modifying the model's inputs. This opens up unique applications: the inputspace adaptation puts control in the hands of the FL user; FL users only need to find the prompts, they don't need to control the pre-trained model itself while training and testing. In this way, FL users can provide adapted images and prompts to an online API that can only operate on their inputs. On the other hand, fine-tuning is typically conditioned on inputs. Its update also directly contains some embeddings of visual feature information. In contrast, the prompts we explore in this paper are input-agnostic across the training data. So the prompting can prevent leaking of user's private information from FL update to a certain extent.

**CLIP PROMPTFL** For CLIP, an image-language model, PROMPTFL organizes users to collaboratively learn prompts as the CLIP's output transformation function. Given a frozen pre-trained CLIP  $\mathcal{F}$  and a task dataset  $\mathbf{D}\{(\mathbf{x}_m, \mathbf{y}_m)\}$  across



Figure 2: **Performance of PROMPTFL with different class distribution.** Bars represent accuracy and lines indicate F-1 score. We range the class distributed on each client from entirely disparate to 10%, 20% and 50% number of classes repeated on more than one client. Compared with the collapse of existing framework in 2, the performance of PROMPTFL remains stable and competitive. Further more, 50% overlapping of classes shows slightly improvement across majority of datasets and backbone.



Figure 3: **Performance of PROMPTFL with different clients and shots.** The overall performance enhanced as the number of shots increasing. However, as the classes on each client become sufficient, the performance of clients with different clients reach similar optimal results, which on the other hand reveals that clients number do not affect the performance of PROMPTFL.

clients, the target of PROMPTFL is to learn a single, static, task-specific prompting  $f_{prompt}$  on class space parameterized by [prompt vectors]. Image classes are represented by labels (*e.g.*, 'panda') which are then prompted (*i.e.*, '[prompt vectors][panda]') to specify the context of the user's task. We follow CLIP's protocol and compute the cosine similarity of the embeddings for each class, normalized to a probability distribution via softmax. The class with the highest probability is selected as the model output. The prompting is added to the class space to form a prompted output  $\mathbf{y} + v_f$ . During training, PROMPTFL will maximize the likelihood of the correct label y,

$$\max_{f_{\text{recovert}}} \mathbf{p}_{\mathcal{F}; f_{\text{prompt}}}(\mathbf{y} + v_f | \mathbf{x}), \tag{2}$$

while the gradient updates are applied only to the [prompt vectors]  $v_f$  and the CLIP parameters  $\mathcal{F}$  remain frozen. During validation, the optimized prompt is added to all test-time classes,  $\mathbf{D}_{\text{test}}\{(\mathbf{x}_m, \mathbf{y}_m + v_f)\}$ , which will be then processed through the frozen  $\mathcal{F}$ .

Training Details To validate the effectiveness of our method, we compare the performance of PROMPTFL with existing framework by 1)training the collaborative model from the scratch and 2)fine-tuning the full model with pretrained weights. We evaluate the performance across four representative dataset used in CLIP for both general objects and fine-grained classification. We report the performance with two representative and influential backbone, Resnet50(38.3M parameters) and Vit b16(86.6M parameters). For the evaluation metrics, we select three aspects to assess the performance of each method, 1) representative Top-1 accuracy on the test set, 2)F1 score to measure the weighted and unified average of precision and recall, which is more useful especially on unbalanced class distribution, 3) as well as the computational and communication cost reported in Fig. 4. We presuming that higher result on accuracy and F-1 score as well as lower result on computation latency will lead to better a framework, detailed comparison results show the superior if PROMPTFL in Tab. 2.

All experiments are conducted with Pytorch on GeForce RTX 3090 GPU. Training is performed with SGD with 0.001 learning rate. Tab. 2 measures the overall performance of PROMPTFL against existing framework from the perspective of two data distribution settings. For the iid setting, each client shares the same classes, while for the extreme non-iid setting, each client owns the independent and non-overlapping classes. We can see that from Tab. 2, PROMPTFL obtains superior results with similar or better accuracy and F1 value, but with only  $0.01\% \sim 0.1\%$  learnable parameters with the iid setting. Further more, with



Figure 4: **Comparison of computation and communication cost of PROMPTFL and Finetuning FL.** We measure the communication cost by the size of uploaded data per round, and observe that finetuning FL takes up to 110 times of cost more than PROMPTFL. Furthermore, finetuning and training from scratch take 2 to 3 times of round more than PROMPTFL for training, which exacerbate the communication expenses. We also utilize GPU memory usage, training GPU time and training data usage to evaluate the computational cost. Training GPU time is calculated by the time of training 50 epoch and training data usage is reported by training food101, which we can observe that finetuning require 250× more than PROMPTFL. We can see that PROMPTFL surpasses the existing framework in the entire aspects of communication and computation efficiency.

the non-iid setting PROMPTFL achieves competitive performance on both accuracy and efficiency against existing framework. Superior outcome on both settings manifest the advantage of our proposed PROMPTFL.

Data Distribution Analysis After obtaining the decent performance in both extreme iid and non-iid setting, we hope to further testify the stability of PROMPTFL and figure out the impact of different data distribution on clients to the performance of PROMPTFL. To observe the intermediate status, we select p% overlapped ratio of classes from 0%to 10%, 20% and 50%, which means that p% of classes will appear on more than one client and the remaining 1 - p%classes only shows on single client. Fig. 2 reports the accuracy and F1 with corresponding distribution. From the result, we observe that given the circumstance that the class on each client is sufficient, the distribution of class has not much impact on the performance of PROMPTFL, only a tiny improvement when the overlapping of classes reaches 50%. On the contrary, existing framework shows miserable stability when encountering shifted class distribution other than unified mode by observing the Tab. 2

**Impact of number of shots** Following the few-shot evaluation setting adopted in CLIP, we use 2, 4, 8, 16 shots in training PROMPTFL and validate the performance with corresponding test sets. Unlike the circumstance in the centralized mode where data only from a single entrance, PROMPTFL involves several participants. Thus we redeclare that the number of shots for the FL mode implies the overall shot containing the entire participants, which is more practical and accord with the class unbalance scenarios in the realworld. From the result in 3, we observe that as the number of training examples per class increases, the performance of PROMPTFL enhanced. Furthermore, for the setting of adequate clients with sufficient class number on each client, the accuracy for each setting reveals rather steady.

**Comparison with different clients** Next, to eliminate the possible impact caused by different clients, we further study the performance of PROMPTFL with different clients from 16 to 32 to 64, with the iid mode that each client owns ran-

dom set of classes. Also, to avoid that the collapse of performance due to deficiency of class on each client, especially for the case with large clients number, we also range the number of shots from 2 to 4 to 8 to 16. We observe that for different number of clients, performance will reach similar optimum for as the classes on each client is sufficient. For example, in caltech101, all settings achieve around 89% with 16 shots.

Computation and Communication Cost Analysis We also analyse the efficiency of PROMPTFL with regard to the computation and communication cost during training. We measure the communication cost by the size of uploaded data per round, and the total round to be transmitted. For the computation cost, we calculate the GPU memory utilization and training GPU time for given steps. Fig. 4 shows the comparison between existing finetuning framework and our proposed PROMPTFL. We observe that PROMPTFL can save at most 110 times communication cost per round compared to existing prevailing method, let alone that PROMPTFL takes half of rounds to reach convergence, which makes a wider disparity in communication cost between them. As for the computation cost, we report the comparison of GPU time as in the same given steps, where PROMPTFL remains outperform existing framework around 3 times. Further more, there is huge advantage that PROMPTFL consumes far less GPU memory during training, which can alleviate the system burden in practical.

### Conclusion

Overall, there are many unknowns about PROMPTFL and this paper sets out to investigate its feasibility. In summary: (1) We demonstrate the system feasibility of PROMPTFL on modern hardware, in terms of overhead in communication, training, and inference. (2) We show that PROMPTFL keeps data on each device private, aiming to learn global prompts updated only by communicating gradients rather than the data itself, and thus not less private than FL. (3) We implement a proof-of-concept in the framework, spanning a range of popular image classification tasks. We find PROMPTFL to be competitive with strong FL baselines.

## References

Bossard, L.; Guillaumin, M.; and Gool, L. V. 2014. Food-101–mining discriminative components with random forests. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems* (*NeurIPS*), 33: 1877–1901.

Deng, M.; Wang, J.; Hsieh, C.-P.; Wang, Y.; Guo, H.; Shu, T.; Song, M.; Xing, E. P.; and Hu, Z. 2022. RLPrompt: Optimizing Discrete Text Prompts With Reinforcement Learning. *arXiv preprint arXiv:2205.12548*.

Dodge, J.; Ilharco, G.; Schwartz, R.; Farhadi, A.; Hajishirzi, H.; and Smith, N. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dziedzic, A.; Dhawan, N.; Kaleem, M. A.; Guan, J.; and Papernot, N. 2022. On the Difficulty of Defending Self-Supervised Learning against Model Extraction. In *Proceedings of the International Conference on Machine Learning (ICML).* 

E. Freedman, A. 2021. Apple a15 bionic powers iphone 13 and ipad mini.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Fürst, A.; Rumetshofer, E.; Tran, V.; Ramsauer, H.; Tang, F.; Lehner, J.; Kreil, D.; Kopp, M.; Klambauer, G.; Bitto-Nemling, A.; et al. 2021. Cloob: Modern hopfield networks with infoloob outperform clip. *arXiv preprint arXiv:2110.11316*.

Gao, T.; Fisch, A.; and Chen, D. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *Proceedings* of the Annual Meeting of the Association for Computational Linguistics (ACL).

Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M. W.; and Keutzer, K. 2021. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*.

Gu, X.; Lin, T.-Y.; Kuo, W.; and Cui, Y. 2021. Openvocabulary Object Detection via Vision and Language Knowledge Distillation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hambardzumyan, K.; Khachatrian, H.; and May, J. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*  Hard, A.; Rao, K.; Mathews, R.; Ramaswamy, S.; Beaufays, F.; Augenstein, S.; Eichner, H.; Kiddon, C.; and Ramage, D. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 

Hu, C.; Jiang, J.; and Wang, Z. 2019. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*.

Jia, C.; Yang, Y.; Xia, Y.; Chen, Y.-T.; Parekh, Z.; Pham, H.; Le, Q.; Sung, Y.-H.; Li, Z.; and Duerig, T. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and Trends*® *in Machine Learning*, 14(1–2): 1–210.

Lalitha, A.; Shekhar, S.; Javidi, T.; and Koushanfar, F. 2018. Fully decentralized federated learning. In *Proceedings of the NeurIPS Workshop on Bayesian Deep Learning*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).* 

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

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).* 

Li, Y.; Liang, F.; Zhao, L.; Cui, Y.; Ouyang, W.; Shao, J.; Yu, F.; and Yan, J. 2021. Supervision Exists Everywhere: A Data Efficient Contrastive Language-Image Pre-training Paradigm. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv* preprint arXiv:2107.13586.

Liu, Z.; Wu, Z.; Gan, C.; Zhu, L.; and Han, S. 2020. Datamix: Efficient privacy-preserving edge-cloud inference. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of* 

the International Conference on Artificial Intelligence and Statistics (AISTATS).

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *Proceedings* of the Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP).

O'Dea, S. 2021. Average global mobile and fixed broadband download & upload speed worldwide.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. 2012. Cats and dogs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*).

Patterson, B. 2022. Blake's ios device specifications grid.

Qu, L.; Zhou, Y.; Liang, P. P.; Xia, Y.; Wang, F.; Adeli, E.; Fei-Fei, L.; and Rubin, D. 2022. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Rajbhandari, S.; Ruwase, O.; Rasley, J.; Smith, S.; and He, Y. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (HPCA).* 

Roy, A. G.; Siddiqui, S.; Pölsterl, S.; Navab, N.; and Wachinger, C. 2019. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*.

Shen, S.; Li, L. H.; Tan, H.; Bansal, M.; Rohrbach, A.; Chang, K.-W.; Yao, Z.; and Keutzer, K. 2021. How Much Can CLIP Benefit Vision-and-Language Tasks? In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Shin, T.; Razeghi, Y.; Logan IV, R. L.; Wallace, E.; and Singh, S. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Singh, A.; Hu, R.; Goswami, V.; Couairon, G.; Galuba, W.; Rohrbach, M.; and Kiela, D. 2022. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 

Sisodia, V. 2021. Distillation of clip model and other experiments.

Tsimpoukelli, M.; Menick, J. L.; Cabi, S.; Eslami, S.; Vinyals, O.; and Hill, F. 2021. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 200–212.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

Xu, H.; Ghosh, G.; Huang, P.-Y.; Okhonko, D.; Aghajanyan, A.; Metze, F.; Zettlemoyer, L.; and Feichtenhofer, C. 2021. VideoCLIP: Contrastive Pre-training for Zero-shot Video-Text Understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (*EMNLP*).

Yuan, L.; Chen, D.; Chen, Y.-L.; Codella, N.; Dai, X.; Gao, J.; Hu, H.; Huang, X.; Li, B.; Li, C.; et al. 2021. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*.

Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; and Chandra, V. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

Zhou, K.; Yang, J.; Loy, C. C.; and Liu, Z. 2021. Learning to prompt for vision-language models. *arXiv preprint arXiv:2109.01134*.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems* (*NeurIPS*), 32.