FedBPT: Efficient Federated Black-box Prompt Tuning for Large Language Models

Jingwei Sun [1] **Ziyue Xu** [2] **Hongxu Yin** [2] **Dong Yang** [2] **Daguang Xu** [2] **Yudong Liu** [1] **Zhixu Du** [1] **Yiran Chen** [1] **Holger R. Roth** [2]

Abstract

Pre-trained language models (PLM) have revolutionized the NLP landscape, achieving stellar performances across diverse tasks. These models, while benefiting from vast training data, often require fine-tuning on specific data to cater to distinct downstream tasks. However, this data adaptation process has inherent security and privacy concerns, primarily when leveraging user-generated, device-residing data. Federated learning (FL) provides a solution, allowing collaborative model fine-tuning without centralized data collection. However, applying FL to finetune PLMs is hampered by challenges, including restricted model parameter access due to the high encapsulation, high computational requirements, and communication overheads. This paper introduces **Fed**erated **B**lack-box **P**rompt **T**uning (FedBPT), a framework designed to address these challenges. FedBPT allows the clients to treat the model as a black-box inference API. By focusing on training optimal prompts and utilizing gradient-free optimization methods, FedBPT reduces the number of exchanged variables, boosts communication efficiency, and minimizes computational cost and memory consumption. Experiments highlight the framework's ability to drastically cut communication and memory costs while maintaining competitive performance. Ultimately, FedBPT presents a promising solution for efficient, privacy-preserving fine-tuning of PLM in the age of large language models. Our code is available in NVIDIA FLARE.

[1]Department of Electrical Computer Engineering, Duke University, Durham, USA [2]NVIDIA, Santa Clara, USA. Correspondence to: J. S. <jingwei.sun@duke.edu>, H. R. <hroth@nvidia.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Large language models (LLM) have shown increasing power on various NLP tasks (Devlin et al., 2018; Raffel et al., 2020; Brown et al., 2020; Fedus et al., 2022; Zhang et al., 2021; Zeng et al., 2021; Sun et al., 2021; Qiu et al., 2020). Typically, these models are trained on a diverse range of text from books, articles, and websites to gain a broad understanding of human language and are known as the pre-trained language models (PLMs). However, task-specific data is often required to adapt PLMs to perform specific tasks or be more accurate in real-world scenarios. This fine-tuning process relies heavily on user-generated data on devices, providing a wealth of contextual insights and nuanced use cases that reflect actual human interaction and needs. In practice, it is challenging to use these devices and data securely. Data needs to be collected and stored for training, but exchanging and storing sensitive data carries security risks and privacy concerns. To overcome the issue of data isolation, federated learning (FL) can be applied to enable numerous devices to collaboratively finetune PLMs over decentralized data while preserving data privacy (McMahan et al., 2017; Sun et al., 2020).

Although fine-tuning PLMs through FL presents promising opportunities, three challenges constrain their real-world application on edge devices. Especially for LLMs, these challenges include (1) limited access to the model parameters due to the high encapsulation, (2) computational and memory costs for local clients, and (3) communication overhead in the FL system. In the real world, mobile frameworks like TensorFlow Lite, PyTorch Mobile, and Apple CoreML, deep learning models are usually integrated or 'encapsulated' within a mobile application. This encapsulation involves converting the model into a binary format that is optimized for efficient mobile inference, and the direct manipulation of model parameters after conversion is typically restricted. Such a high encapsulation is also preferred by the model developers to ensure that the model is used as intended and protects its intellectual property, and it also means that end-users or third-party developers have limited ability to alter or inspect the model. Additionally, even if the clients could access the model parameters, it is impractical for devices with limited resources (Wen et al., 2013; Alyamkin et al., 2019) to conduct local PLM

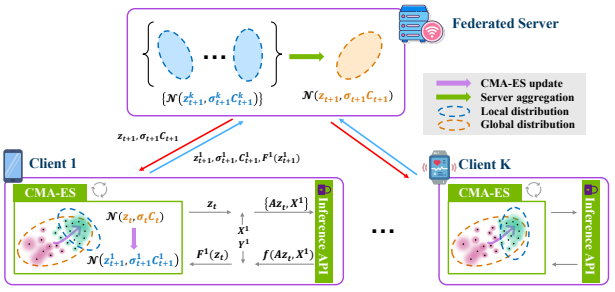


Figure 1. Overview of FedBPT. The clients in FedBPT adopt a gradient-free optimization (CMA-ES) to search for optimal distributions of the prompt based on local data. The clients are not required to access the PLM parameters, and only inference of the PLM is conducted during the search. The server aggregates the uploaded local distributions to derive the globally optimal distribution of the prompt. The global distribution will be sent back to the clients for the next round of search.

fine-tuning, which is extremely memory-intensive and brings high computational overhead. Moreover, fine-tuning PLMs through FL requires the clients and server to frequently exchange model parameters or gradients, usually on a scale of millions or even billions of parameters. Such intensive communication cost is unfeasible for commercial edge devices with limited communication bandwidth. To this end, existing works (Sun et al., 2022a; Chen et al., 2022b; Zhao et al., 2023; Xu et al., 2023) apply parameter-efficient fine-tuning (PEFT) methods of PLMs to FL to reduce resource costs. Effective PEFT methods include adapter tuning (Houlsby et al., 2019), prefix tuning (Li & Liang, 2021), LoRA (Hu et al., 2021) and BitFit (Zaken et al., 2021). These techniques primarily freeze most parameters of PLMs and update only a few additional parameters, which can reduce communication costs significantly. However, these PEFT methods still require the clients to access model parameters and gradients for local training. Even if the computational cost could be reduced, these gradient-based PEFT methods requiring back-propagation are still unfeasible for most edge devices with limited resources, such as mobile phones and AR headsets.

To solve these challenges simultaneously, we propose a new framework called **Fed**erated **B**lack-box **P**rompt **T**uning (**FedBPT**) as shown in Figure 1. The goal of FedBPT is to train an optimal prompt to improve the performance of the frozen PLMs. The clients and the server exchange prompts rather than model parameters, which reduces the communicated variables from millions or billions to only hundreds, improving communication efficiency significantly. The clients in FedBPT adopt a gradient-free optimization method rather than gradient-based methods to conduct local training, which frees the clients from being required to access the model parameters. In addition, only forward-propagation without back-propagation is needed for local training, which can reduce computational cost and memory consumption.

We conducted experiments on multiple datasets using SOTA PLMs. The results show that FedBPT reduces the communication cost by more than 500k$\times$ while achieving comparable results with the baselines that require model parameter access and back-propagation for optimization. FedBPT can also reduce the memory footprint by more than 3$\times$ without applying any additional efficient inference technique. By proposing FedBPT, we offer a solution to break down data silos in the era of LLMs without the limiting factors of requiring full model parameter access, large communication bandwidth, and device compute capacity.

We summarize our contributions as follows:

- We present three challenges in applying FL to adapt PLMs in the real world, including the requirement of model access, communication cost, and on-device compute capacity.
- We propose a federated black-box prompt tuning framework (FedBPT) that enables the devices to adapt PLMs in the real world collaboratively by solving the above-mentioned challenges simultaneously.
- We evaluate FedBPT on multiple datasets with SOTA PLMs. FedBPT achieves comparable accuracy with the gradient-based methods that require clients to access model parameters while reducing communication and memory costs significantly.

2. Related Works

2.1. Federated Learning

Federated learning (FL) (Konečnỳ et al., 2016; McMahan et al., 2017; Sun et al., 2022b) is a prominent distributed learning strategy, particularly beneficial for tasks that prioritize privacy. However, its application faces challenges due to the non-IID nature of distributed datasets. The heterogeneous data distribution across devices compromises accuracy relative to traditional centralized training. Numerous research efforts (Kairouz et al., 2021; Zhao et al., 2018; Chai et al., 2020; Li et al., 2018) have sought to mitigate this performance degradation. Recent works (Chen et al., 2022a; Nguyen et al., 2022) demonstrate that fine-tuning the pre-trained models through FL suffers less from the non-IID issue. Empirical research by Weller et al. (2022) suggests that Pretrained Language Models (PLMs) can diminish the effects of non-IID data and bridge the accuracy discrepancy with centralized training. Their results show that when applying PLMs, even the vanilla FedAvg can achieve comparable model performance with centralized training. These works indicate that FL presents a promising avenue for fine-tuning PLMs by leveraging user data while upholding privacy

standards. However, PLMs, especially large-scale ones, introduce considerable communication overheads in FL scenarios, making federated training cumbersome and often unsuitable for practical applications. Additionally, the training of PLMs typically demands ample labeled data to ensure satisfactory accuracy – a condition that may be unattainable for individual device users. Notably, many local devices are constrained by limited computational capacity and memory, making the local training of PLMs a challenging endeavor. Diverging from these studies, our work delves into adapting PLMs within FL, especially under tight resource constraints.

2.2. Prompt-based Learning

Prompt-based learning has gained significant attention in the realm of LLMs. Its essence is rooted in leveraging minimal examples or specific cues to guide a PLM toward the desired output. This contrasts with traditional supervised learning, where a model is trained explicitly using extensive labeled data. OpenAI's GPT-3 (Brown et al., 2020) marked a pivotal turn in the exploration of prompt-based learning. The sheer scale of GPT-3 made it possible to produce relevant outputs with carefully crafted prompts (Brown et al., 2020; Lester et al., 2021) without the need for task-specific model fine-tuning. However, manually designed prompts still suffer a performance gap compared with a fine-tuned model (Brown et al., 2020; Schick & Schütze, 2020; Gao et al., 2020; Sun et al., 2022d). Recent works demonstrate that the prompt does not have to represent natural language. It can also be optimized efficiently in continuous space with gradient descent (Li & Liang, 2021; Hambardzumyan et al., 2021; Qin & Eisner, 2021; Liu et al., 2023; Zhong et al., 2021; Liu et al., 2021). In the case of only tuning the continuous prompt while keeping the parameters of large PLMs untouched, one can retain the efficient training benefits while matching the performance of full model tuning. Prompt tuning (Lester et al., 2021; Li & Liang, 2021) was proposed to fine-tune a continuous vector concatenated to the input embeddings. Unlike manual prompt design conducted at the vocabulary level, prompt tuning optimizes the prompt in the embedding space. Based on this idea, p-tuning (Liu et al., 2021; 2022; 2023) was proposed to improve the performance further. Similar to prompt tuning, p-tuning also learns concrete prompts in the embedding space. However, in p-tuning, an additional LSTM model is required to predict token embeddings.

3. Preliminary: Black-box Prompt Tuning

Common language understanding tasks can be formulated as a classification task to predict for a batch of input texts X the labels Y. Prompt tuning is to train a continuous prompt vector $\boldsymbol{p} \in \mathbb{R}^D$ such that the prediction performance can be improved when the model is fed the optimal prompt vector $\boldsymbol{p}^*$ together with the input X. The objective of prompt tuning can be formulated as

$$\boldsymbol{p}^* = \underset{\boldsymbol{p} \in \mathcal{P}}{\operatorname{argmin}} \mathcal{L}(f(\boldsymbol{p}; X), Y), \tag{1}$$

where $f(\cdot)$ is the PLM inference API, $\mathcal{L}(\cdot)$ is the loss function and $\mathcal{P}$ is some search space of interest. To optimize $\boldsymbol{p}$, gradient-based methods (e.g., SGD) can be applied by conducting back-propagation of the model f. Recently, a gradient-free optimization, **B**lack-**B**ox **T**uning (BBT) (Sun et al., 2022c), was also proposed to optimize the prompt $\boldsymbol{p}$ without back-propagation. Based on the observation that large-scale PLMs have a low intrinsic dimensionality (Aghajanyan et al., 2020; Qin et al., 2021), BBT optimizes $\boldsymbol{z} \in \mathbb{R}^d$ in a much smaller subspace ($d \ll D$) and uses a random projection matrix $\boldsymbol{A} \in \mathbb{R}^{D \times d}$ to project $\boldsymbol{z}$ on the original prompt space $\mathcal{P}$. The objective can be formulated as

$$\boldsymbol{z}^* = \underset{\boldsymbol{z} \in \mathcal{Z}}{\operatorname{argmin}} \mathcal{L}(f(\boldsymbol{A}\boldsymbol{z}; X), Y). \tag{2}$$

To optimize z, BBT adopts a gradient-free optimizer CMA-ES (Covariance Matrix Adaptation Evolution Strategy) (Hansen, 2016), a widely used evolutionary algorithm for non-convex black-box optimization in the continuous domain. CMA-ES maintains a parameterized search distribution, i.e., a multivariate normal distribution. In each iteration, CMA-ES samples a population of new query solutions from the multivariate normal distribution as

$$\boldsymbol{z}_{t+1,i} \sim \boldsymbol{m}_t + \sigma_t \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_t), \tag{3}$$

where $i = 1, ..., \lambda$ and λ is the population size. $\boldsymbol{m}_t \in \mathbb{R}^d$ and $\boldsymbol{C}_t \in \mathbb{R}^{d \times d}$ are the mean vector and covariance matrix of the search distribution at iteration step t, respectively. σ_t is the standard deviation that controls the step length. $\boldsymbol{m}_t, \boldsymbol{C}_t$ and σ_t are updated by maximizing the likelihood of successful steps, which are the steps with lower loss values (cf. (Hansen, 2016) for more details).

4. Method

To solve the challenges of model access, communication cost, and computational cost simultaneously, we propose **Fed**erated **B**lack-box **P**rompt **T**uning method (FedBPT) to train an optimal prompt in a federated fashion by adapting BBT to federated learning. Unlike FL methods communicating model parameters, the clients in FedBPT train and communicate with the server prompts rather than the model parameters, which is communication efficient. To optimize prompts, the clients only need to conduct inference rather than back-propagation, significantly reducing the computational cost and memory usage. The FL server aggregates the local prompts uploaded by the client and is completely agnostic to the employed LLM architecture. During training, the clients can treat the model as a black box: neither the clients nor the server requires access to the PLM parameters.

4.1. Problem Formulation

Suppose there are K clients in FL, and each client hosts a private dataset $D^k = (X^k, Y^k)$ consisting of n^k samples $\{x_i^k, y_i^k\}_{i\in[n^k]}$. Given a global projected matrix $\boldsymbol{A}$ in Equation (2), the clients collaboratively train an optimal z with the objective to solve:

$$\boldsymbol{z}^* = \underset{\boldsymbol{z}}{\operatorname{argmin}} \sum_{k\in[K]} \frac{n^k}{\sum_{k\in[K]} n^k} F^k(\boldsymbol{z}), \tag{4}$$

where $F^k(\boldsymbol{z})$ is the loss of client k:

$$F^k(\boldsymbol{z}) = \mathcal{L}\big(f(\boldsymbol{A}\boldsymbol{z}; X^k), Y^k\big) = \sum_{i\in[n^k]} \mathcal{L}\big(f(\boldsymbol{A}\boldsymbol{z}; x_i^k), y_i^k\big). \tag{5}$$

4.2. Overview of FedBPT

In FedBPT, the clients optimize local objectives based on BBT. Thus, unlike previous FL works, FedBPT aggregates the CMA-ES parameters applied by the clients to conduct BBT rather than the deep learning models. At the start of the training, the server initializes and distributes the projection matrix $\boldsymbol{A}$ to the clients. Then, the server and clients will freeze and apply $\boldsymbol{A}$ to calculate the prompt with the received $\boldsymbol{z}$. In each communication round (e.g., the t-th round), the server first sends the up-to-date global CMA-ES parameters, including the mean vector $\boldsymbol{z}_t$, covariance matrix $\boldsymbol{C}_t$ and the search step σ_t to clients. Then, the clients (e.g., the k-th client) conduct BBT to optimize the received CMA-ES parameters by minimizing their local loss, i.e., Equation (5). After local optimization, the clients upload their locally optimal parameters and the local loss value $F^k(\boldsymbol{z}_{t+1}^k)$ to the server. After the server receives all CMA-ES parameters, it aggregates the local parameters and updates the global CMA-ES parameters for the next communication round. After the training is completed (e.g., T communication rounds), the mean vector of the global CMA-ES $\boldsymbol{z}_T$ will be adopted to compute the optimal prompt $\boldsymbol{p}_T = \boldsymbol{A}\boldsymbol{z}_T$.

The primary distinction between FedBPT and earlier FL algorithms lies in the use of BBT for optimization. Yet, integrating BBT into FL algorithms, such as FedAvg, is not straightforward. Simply combining BBT and FedAvg cannot achieve decent performance. The first challenge is how to aggregate CMA-ES parameters on the server effectively. Unlike aggregating deep learning models, directly averaging CMA-ES parameters, mostly consisting of distribution statistics, is not feasible. The second challenge is the prompt overfitting problem caused by data distribution shifts across clients, which is common under non-IID settings. We will introduce these challenges in detail and our solutions in the following sections.

4.3. Server-level CMA-ES Algorithm

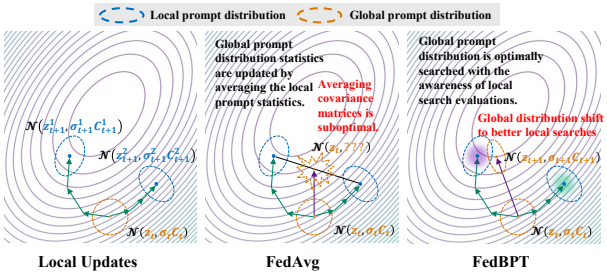


Figure 2. Comparison of aggregation between directly using FedAvg and FedBPT. FedAvg derives the global distribution by directly averaging the local distribution statistics. Mathematically, the arithmetic mean of the covariance matrices is not equivalent to the covariance matrix of our targeted optimal global distribution. In FedBPT, the server applies CMA-ES to derive the global prompt distributions with the awareness of the evaluation results of the uploaded local distributions.

After receiving local CMA-ES parameters, the server conducts aggregation on the server to derive a global search distribution that can guide the clients' search in the next communication round. Directly averaging the models uploaded by the clients following FedAvg is not effective for FedBPT. In FedBPT, the clients locally optimize the CMA-ES parameters parameterized by multivariate normal distribution statistics. Mathematically, the arithmetic mean of the covariance matrices is not equivalent to the covariance matrix of our targeted optimal global distribution, as is empirically shown in Sec. 5.2. In addition, CMA-ES is a random search algorithm that cannot guarantee to achieve a local optimum as with gradient-based optimization algorithms. Directly averaging optimal and inferior local search results makes it difficult to achieve a global optimum. To derive an optimal global search distribution on the server, we adopt a server-level CMA-ES algorithm to update the search distribution statistics based on the local search results. The comparison of aggregations by directly applying FedAvg and FedBPT is shown in Figure 2.

The intuition of the server-level CMA-ES is to consider the local search results as a set of solutions sampled by the server. The server then evaluates these sampled solutions and updates the search distributions for the next communication round. Suppose a set of clients $\mathbb{S}_t$ participate in training in the t-th communication round. The server-level CMA-ES takes the received mean vectors $\{\boldsymbol{z}_{t+1}^k\}_{k\in\mathbb{S}_t}$ as the sampled solutions and the local loss values $\{F^k(\boldsymbol{z}_{t+1}^k)\}_{k\in\mathbb{S}_t}$ as the corresponding search step loss. To update the CMA-ES parameters, the **search step length** is required. **However, the server-level "sampling" is conducted by multiple local search steps, and the server-level search step length σ_t is intractable.** Directly applying a local search step length causes the model to diverge. We provide a theoretical explanation for

this divergence later. To solve this problem, we theoretically derive a corrected search step σ_t' on the server formulated as

$$\sigma_t' = 2\sqrt{\sum_{k\in\mathbb{S}_t'}\sum_{j=1}^{I}\left(\sigma_{t,j}^k\right)^2/(|\mathbb{S}_t|\cdot\lambda_k)}, \tag{6}$$

where $\mathbb{S}_t'$ is the set of $\frac{|\mathbb{S}_t|}{2}$ clients that upload z_{t+1}^k with the lowest local loss values $F^k(\boldsymbol{z}_{t+1}^k)$. $\sigma_{t,j}^k$ is the step length of client k's j-th local search iteration in communication round t. I is the number of local search iterations, and λ_k is the local search population of client k. Then, the overall algorithm of FedBPT is shown in Algorithm 1.

Algorithm 1 Training Algorithm of FedBPT.

Server executes:

initialize the projection matrix $\boldsymbol{A}$ and distribute it to the clients
initialize the global CMA-ES parameters $\{\boldsymbol{z}_0,\sigma_0,\boldsymbol{C}_0\}$
for each round $t=0,1,...$ **do**
  for each client $k\in S_t$ **in parallel do**
    $\{\boldsymbol{z}_{t+1}^k, \{\sigma_{t,j}^k\}_{j\in[I]}, F^k(\boldsymbol{z}_{t+1}^k)\} \leftarrow$ ClientUpdate$(\boldsymbol{z}_t,\sigma_t,\boldsymbol{C}_t)$
  end for
  $\sigma_t' = 2\sqrt{\sum_{k\in\mathbb{S}_t'}\sum_{j=1}^{I}\left(\sigma_{t,j}^k\right)^2/(|\mathbb{S}_t|\cdot\lambda_k)}$
  $\{\boldsymbol{z}_{t+1},\sigma_{t+1},\boldsymbol{C}_{t+1}\} \leftarrow$ CMA-ES$\left(\left\{\boldsymbol{z}_{t+1}^k, F^k\left(\boldsymbol{z}_{t+1}^k\right)\right\}_{k\in S_t};\boldsymbol{z}_t,\sigma_t',\boldsymbol{C}_t\right)$
end for

ClientUpdate$(\boldsymbol{z}_t,\sigma_t,\boldsymbol{C}_t)$**:**

$\boldsymbol{z}_{t,1}^k,\sigma_{t,1}^k,\boldsymbol{C}_{t,1}^k \leftarrow \boldsymbol{z}_t,\sigma_t,\boldsymbol{C}_t$
for each local iteration j from 1 to $I-1$ **do**
  Randomly sample a set of binary masks M_j^k with the same shape of X^k with a rate r_p of elements that are zeros
  Randomly sample a set of tokens $\hat{X}^k$ with the same shape of X^k
  for $i\in\lambda_k$ **do**
    $\boldsymbol{z}_{t,j,i}^k \sim \mathcal{N}(\boldsymbol{z}_{t,j},\sigma_{t,j}\boldsymbol{C}_{t,j})$
    $\hat{F}^k(\boldsymbol{z}_{t,j,i}^k) = \frac{\mathcal{L}\left(f(\boldsymbol{A}\boldsymbol{z}_{t,j,i}^k;X^k),Y^k\right)}{\mathcal{L}\left(f(\boldsymbol{A}\boldsymbol{z}_{t,j,i}^k;X^k\odot M_j^k+\hat{X}^k\odot\left(1-M_j^k\right)),Y^k\right)}$
  end for
  $\left\{\boldsymbol{z}_{t,j+1}^k,\sigma_{t,j+1}^k,\boldsymbol{C}_{t,j+1}^k\right\} \leftarrow$ CMA-ES$\left(\left\{\boldsymbol{z}_{t,j,i}^k,\hat{F}^k\left(\boldsymbol{z}_{t,j,i}^k\right)\right\}_{i\in[\lambda_k]};\boldsymbol{z}_{t,j}^k,\sigma_{t,j}^k,\boldsymbol{C}_{t,j}^k\right)$
end for
$\boldsymbol{z}_{t+1}^k \leftarrow \boldsymbol{z}_{t,I}^k$
$F^k(\boldsymbol{z}_{t+1}^k) = \mathcal{L}\left(f(\boldsymbol{A}\boldsymbol{z}_{t+1}^k;X^k),Y^k\right)$
return $\{\boldsymbol{z}_{t+1}^k,\{\sigma_{t,j}^k\}_{j\in[I]},F^k(\boldsymbol{z}_{t+1}^k)\}$ to server

We then introduce how we derive the corrected search step σ_t'. After the server receives the locally updated prompt vectors $\{z_{t+1}^k\}_{k\in\mathbb{S}_t}$ and the corresponding loss values $\{F^k(\boldsymbol{z}_{t+1}^k)\}_{k\in\mathbb{S}_t}$, the server updates the CMA-ES parameters as shown in Algorithm 2.

Algorithm 2 CMA-ES update.

CMA-ES$(\{z_i,f_i\}_{i\in[\lambda]},z,\sigma,C,p_\sigma,p_c)$**:**

$z_{1...\lambda} \leftarrow z_{s(1)...s(\lambda)}$ with $s(i)=\text{argsort}(f_{1...\lambda},i)$
$z' \leftarrow \frac{1}{\mu}\sum_{k=1}^{\mu} z_i$
$p_\sigma \leftarrow$ Update-$p_\sigma(z',z,C,p_\sigma)$
$p_c \leftarrow$ Update-$p_c(z',z,C,p_\sigma,p_c)$
$\sigma' \leftarrow \sigma\times\exp(\frac{|p_\sigma|}{E|N(0,I)|}-1)$
$C' \leftarrow (1-c_1-c_\mu+c_s)C + c_1 p_c p_c^T + \frac{1}{\mu c_\mu}\sum_{k=1}^{\mu}\frac{z_i-z}{\sigma}(\frac{z_i-z}{\sigma})^T$
Return $z',\sigma',C',p_\sigma,p_c$

Without the loss of generality, we assume that $\left[z_{t+1}^1,...,z_{t+1}^{|\mathbb{S}_t|}\right]$ is ordered satisfying that $F^1(\boldsymbol{z}_{t+1}^1) < ... < F^{|\mathbb{S}_t|}(\boldsymbol{z}_{t+1}^{|\mathbb{S}_t|})$ as stated in line 1 of Algorithm 2. The server updates z_{t+1} following

$$z_{t+1} = \sum_{k=1}^{\mu}\frac{1}{\mu}z_{t+1}^k = z_t + \sum_{k=1}^{\mu}\frac{1}{\mu}(z_{t+1}^k - z_t), \tag{7}$$

where we apply *Rank-μ-Update* (Hansen, 2016) and the μ best z_{t+1}^k with lowest z_{t+1}^k are averaged to update z_{t+1}. There is no issue to update z_{t+1} on the server (line 2 in Algorithm 2), but to update σ_{t+1} and $\boldsymbol{C}_{t+1}$, the intermediate coefficients *evolution path* values $p_{\sigma,t+1}$ and $p_{c,t+1}$ for this round should be derived first (line 3&4 in Algorithm 2). For simplicity, we derive σ_t' from the computation of $p_{\sigma,t+1}$, which is also applicable to $p_{c,t+1}$. CMA-ES derives the evolution path $p_{\sigma,t+1}$ following

$$p_{\sigma,t+1} \leftarrow (1-c_\sigma)p_{\sigma,t} + \sqrt{1-(1-c_\sigma)^2}\sqrt{\mu}C_t^{-1/2}\frac{z_{t+1}-z_t}{\sigma_t}, \tag{8}$$

where c_σ is an artificial hyper-parameter satisfying $c_\sigma \leq 1$ (Hansen, 2016). The σ_t is intractable in FedBPT, and we need to derive an estimated σ_t' to conduct CMA-ES on the server correctly. The key to estimating the global search step on the server is to guarantee that the term $\sqrt{\mu}C_t^{-1/2}\frac{z_{t+1}-z_t}{\sigma_t}$ follows a standard normal distribution

$$\sqrt{\mu}C_t^{-1/2}\frac{z_{t+1}-z_t}{\sigma_t} \sim \mathcal{N}(0,I) \tag{9}$$

under neutral selection, which means that the server randomly selects z_{t+1}^k to update the CMA-ES parameters. Based on this rule of estimation, we first derive the distribution of $z_{t+1}-z_t$. From Equation (7), we have

$$z_{t+1} - z_t = \sum_{k=1}^{\mu}\frac{1}{\mu}(z_{t+1}^k - z_t), \tag{10}$$

where z_{t+1}^k is formulated as

$$z_{t+1}^k = z_t + \sum_{j=1}^{J}\sigma_{t,j}^k \times \sum_{i=1}^{\mu_l}\frac{1}{\mu_l}z_{t,j,i}^k \sim \mathcal{N}\left(0,C_{t,j}^k\right), \tag{11}$$

where J is the number of local iterations in one round of training, and μ_l is the rank of local *Rank-μ-Update*. $\sigma^k_{t,j}$ is the search step length of client k's j-th iteration in round t. $z^k_{t,j,i}$ is the i-th sampled point in client k's j-th iteration of search in round t. When the clients conduct limited iterations of local training in one round, we make an assumption that the local covariance matrix $C^k_{t,j}$ in one round will not change significantly, then we have

$$z^k_{t+1} \approx z_t + \sum_{j=1}^{J} \sigma^k_{t,j} \times \sum_{i=1}^{\mu_l} \frac{1}{\mu_l} z^k_{t,j,i} \sim \mathcal{N}\left(0, C^k_t\right). \tag{12}$$

Therefore, we have

$$\begin{aligned} &\frac{C_t^{-1/2}(z_{t+1}-z_t)}{\sqrt{\sum_{k=1}^{\mu}\left(\frac{1}{\mu}\right)^2 \sum_{j=1}^{J}\left(\sigma^k_{t,j}\right)^2 \sum_{i=1}^{\mu_l}\left(\frac{1}{\mu_l}\right)^2}} \\ =&\sqrt{\mu}\frac{C_t^{-1/2}(z_{t+1}-z_t)}{\sqrt{\sum_{k=1}^{\mu}\sum_{j=1}^{J}\left(\sigma^k_{t,j}\right)^2/(\mu\cdot\mu_l)}} \sim \mathcal{N}(0,I). \end{aligned} \tag{13}$$

Compared with Equation (9), we derive an estimated global search step length as

$$\sigma'_t = \sqrt{\sum_{k=1}^{\mu}\sum_{j=1}^{J}\left(\sigma^k_{t,j}\right)^2/(\mu\cdot\mu_l)}. \tag{14}$$

In this paper, we set $\lambda_1 = ... = \lambda_K$ and $\mu_l = \frac{\lambda_k}{2}$, where λ_k is the local population size of the k-th client, and we set $\mu = \frac{|\mathbb{S}_t|}{2}$. Then without the restriction on the order of $\left[z^1_{t+1},...,z^{|\mathbb{S}_t|}_{t+1}\right]$, we have

$$\sigma'_t = 2\sqrt{\sum_{k\in\mathbb{S}'_t}\sum_{j=1}^{J}\left(\sigma^k_{t,j}\right)^2/(|\mathbb{S}_t|\cdot\lambda_k)}, \tag{15}$$

where $\mathbb{S}'_t$ is the set of $\frac{|\mathbb{S}_t|}{2}$ clients that upload z^k_{t+1} with the lowest local loss values $F^k(\boldsymbol{z}^k_{t+1})$.

4.4. Local Black-box Prompt Tuning against Overfitting

In real life, client data are non-IID distributed, which causes label-skew across clients (Li et al., 2018). The server-level CMA-ES evaluates the clients' search results based on the uploaded local loss values. Such label-skew makes local searches overfitted to local data distributions by achieving low local loss values and makes it difficult for the server to evaluate their performance on the global data distribution. This overfitting issue is more serious when adopting BBT for local training. Gradient-based optimizations (e.g., SGD) incorporate both data and label information into the gradient for updating. In contrast, when using Equation (2) as the local training objective, BBT modifies the CMA-ES parameters based primarily on how close predictions are to the labels while using the data only indirectly. It is a practical label-skew case in which most of a client's data is distributed in one class (Tang et al., 2022). In this case, a local CMA-ES might learn a prompt that triggers the frozen PLM to generate predictions corresponding to the dominant class, regardless of the input. We conduct experiments to demonstrate such an issue, which can be found in Appendix A

To mitigate this overfitting issue, we propose a perturbation method to regularize the local training objective and avoid CMA-ES selecting overfitting prompts. For a sample $\{x^k_i, y^k_i\}$ of client k, we randomly generate a binary mask m^k_i with an artificial rate r_p of elements that are zeros. We then randomly sample a sentence $\hat{x}^k_i$ from the vocabulary with the same length of x^k_i as shown in Figure 3, and the local training objective for the k-th client is formulated as

$$\boldsymbol{z}^* = \underset{\boldsymbol{z}\in\mathcal{Z}}{\operatorname{argmin}} \sum_{i\in[n^k]} \frac{\mathcal{L}\left(f(\boldsymbol{A}\boldsymbol{z}; x^k_i), y^k_i\right)}{\mathcal{L}\left(f\left(\boldsymbol{A}\boldsymbol{z}; x^k_i \odot m^k_i + \hat{x}^k_i \odot \left(1-m^k_i\right)\right), y^k_i\right)}. \tag{16}$$

The intuition is that given a perturbed input, the PLM should not be confident of generating a correct prediction even when fed an optimal prompt.

A detailed algorithm of applying server-level CMA-ES and the local perturbation method can be found in Algorithm 1.

5. Experiments

5.1. Experimental Setup

Datasets and Models We conduct experiments on three language understanding datasets: (1) The SST-2 (Socher et al., 2013) is a popular sentiment analysis dataset. The SST-2 dataset consists of sentences taken from movie reviews along with their corresponding sentiment labels. Each sentence is annotated as either "positive" or "negative" based on the sentiment conveyed. (2) The Yelp polarity (yelp) is another sentiment analysis dataset, which consists of reviews on Yelp along with their corresponding sentiment labels of "positive" or "negative". (3) The AG's News dataset (OpenAI) is a large-scale topic classification dataset for the task of categorizing news articles into one of four predefined topic classes. The dataset is based on the AG's corpus, a collection of news articles from various sources. We evaluate our FedBPT on two PLMs: (1) RoBERTa (Liu et al., 2019) is a variation of the BERT model. It is pre-trained using a variant of the masked language modeling (MLM) objective, whose objective is to predict masked tokens in a given text sequence. In this paper, we apply the version of 356 million parameters. (2) Llama 2 (Touvron et al., 2023) is a SOTA PLM released by Meta, which is a collection of foundation language models ranging from 7 billion to 70 billion parameters. Llama 2 models are trained on 2 trillion tokens and have double the context length than Llama 1. In this paper, we

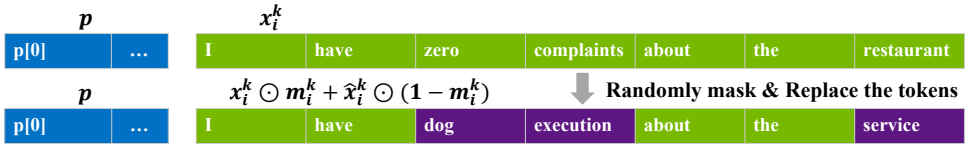


Figure 3. We randomly mask and replace the tokens to perturb a sentence. The PLM should be confused about the perturbed sentence even given an optimal prompt.

evaluate FedBPT on the model with 7 billion parameters.

Baselines We compare our black-box tuning FL framework with several gradient-based and gradient-free methods. For gradient-based methods, we compare with four baselines: (1) **FedAvg** (McMahan et al., 2017) is the most widely-used algorithm for FL. In FedAvg, the clients fine-tune the whole model and transmit the updated model parameters. (2) **FedPrompt** (Zhao et al., 2023) is the SOTA work of applying FL to adapt the PLM with high communication efficiency. The clients in FedPrompt learn and transmit prompts, which reduces the communication cost significantly. (3) **FedP-tuning** is built on FedPrompt by replacing the local training from prompt tuning to p-tuning (Liu et al., 2022), which is more advanced and proven to achieve higher performance on downstream tasks. (4) **FedLoRA** (Yi et al., 2023) applies LoRA (Hu et al., 2021) for local training and transmits the low-rank adapter parameters, which can reduce communication cost. For gradient-free methods, we consider three baselines: (1) **Manual Prompt** is adapted following the templates and label words in Appendix C to conduct zero-shot evaluation. (2) **In-context Learning** Following (Brown et al., 2020), we randomly select up to 5 training samples and concatenate them with the input texts. (3) **FedAvg-BBT** is a baseline by simply combining BBT (Sun et al., 2022c) and FedAvg. We build this baseline for comparison to show the effectiveness of our designed server-level prompt searching.

FL setup & Hyperparameters We follow FedPrompt (Zhao et al., 2023) to design our FL setup. The system has ten clients, and all of the clients participate in training in each round. Considering the real world, where many users possess only a limited amount of labeled data, we conduct experiments under few-shot settings. We randomly select 40 samples for each class to construct a training set D_{train}. We conduct experiments in both IID and non-IID settings. More detailed hyperparameters can be found in Appendix B.

5.2. Experimental Results

Results of RoBERTa. The results when adopting RoBERTa as the PLM are shown in Tab. 1. Compared with the gradient-based methods, FedBPT achieves comparable or even higher accuracy with drastically reduced trainable parameters. Specifically, FedBPT achieves an accuracy of 0.92% higher than FedPrompt and only 0.69% lower than the best gradient-based baseline FedP-tuning for SST-2 under the non-IID setting. Meanwhile, FedBPT reduces the trainable parameters by more than 100× and 30,000× compared with FedPrompt and FedP-tuning, respectively. The trainable parameters are required to be transmitted in each communication round, which means that FedBPT reduces the communication cost of one device in one round from 120MB to only 4KB compared with FedP-tuning. For AG's News and Yelp, FedBPT can also achieve comparable accuracy under IID and non-IID settings. Notably, FedAvg and FedLoRA cannot improve the accuracy under both IID and non-IID settings. This demonstrates that directly fine-tuning the parameter space of LLMs is not feasible in realistic FL settings, even if adapting LoRA when the clients hold limited labeled samples. We document the memory usage by one client of different methods on SST-2 in Tab. 2. It is shown that FedBPT can reduce memory costs by more than 3× compared with gradient-based methods.

Method	Mem.
FedPrompt	5.8 GB
FedP-tuning	6.1 GB
FedAvg	7.2 GB
FedLoRA	2.0 GB
In-context Learning	2.1 GB
FedBPT	1.8 GB

Table 2. Memory footprint on SST-2 by applying RoBERTa.

Compared with gradient-free baselines, FedBPT achieves higher accuracies under IID and non-IID settings for all the datasets. FedBPT achieves accuracies of 2.3%, 4.57%, and 1.08% higher than FedAvg-BBT under non-IID settings for SST-2, AG's News, and Yelp, respectively. It is shown that FedAvg-BBT achieves limited accuracy improvement compared with manual prompts for all the datasets, which demonstrates that **simply combining FedAvg and BBT cannot achieve decent performance**.

Results of Llama 2. The number of trainable parameters when applying Llama 2 as the PLM is shown in Tab. 3. The trade-off between the communication cost of one device in one round and model accuracy is shown in Figure 4. We have two important observations: (1) For Llama 2, FedBPT can improve the accuracy significantly compared with the gradient-free baselines and achieve comparable accuracies with the gradient-based methods in most settings. Specifically, FedBPT improves accuracy by more than 12%, 11%, and 13% for SST-2, AG's News, and Yelp compared with the manual prompts under non-IID settings, respectively.

Method	Trainable Params.	SST-2 Acc.(%) IID	SST-2 Acc.(%) non-IID	AG's NEWS Acc.(%) IID	AG's NEWS Acc.(%) non-IID	Yelp Acc.(%) IID	Yelp Acc.(%) non-IID
Gradient-based methods							
FedPrompt	51K	90.25	85.55	87.72	85.62	91.44	91.47
FedP-tuning	15M	**90.6**	**87.16**	**88.17**	**86.11**	**93.61**	**91.63**
FedAvg	355M	84.7	82.4	77.43	76.54	88.25	88.03
FedLoRA	786K	84.6	84.53	77.85	75.9	88.52	88.2
Gradient-free methods							
Manual prompt	0	83.6		75.75		88.37	
In-Context Learning	0	79.7		76.96		89.65	
FedAvg-BBT	500	84.45	84.17	76.54	76.46	89.64	89.72
FedBPT	*500*	***87.16***	***86.47***	***82.36***	***81.03***	***91.12***	***90.8***

Table 1. Results under both IID and non-IID settings with RoBERTa as the backbone model.

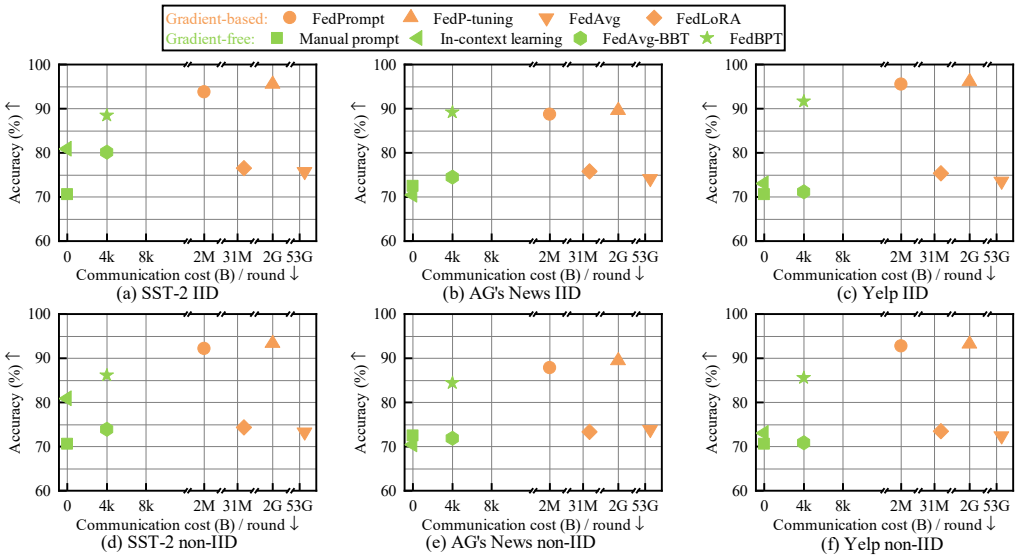


Figure 4. The results under IID and non-IID settings with Llama 2 as the backbone model.

Method	FedPrompt	FedP-tuning	FedAvg	FedLoRA	Manual	FedAvg-BBT	FedBPT
Trainable Params.	205K	235M	7B	4M	0	500	500

Table 3. Number of trainable parameters when adopting Llama 2 as the backbone model.

FedBPT can achieve slightly higher accuracy than FedPrompt under the AG's News IID setting, while the gradient-free baselines experience declines in accuracy of over 15%. (2) FedBPT reduces the number of trainable parameters compared with gradient-based methods even more significantly than adopting RoBERTa. Specifically, compared with FedP-tuning, FedBPT reduces the trainable parameters from 235M to only 500, which makes FedBPT reduce the communication cost of one device in one round from nearly 2GB to 4KB.

In summary, FedBPT can achieve much higher accuracy than gradient-free baselines and comparable accuracy as gradient-based methods for both RoBERTa and Llama 2 models. In addition, the number of trainable parameters does not increase when the model scale is larger. The reason is that FedBPT adopts a projection matrix to project the embedding space to a low-dimension space, which enables the clients to conduct CMA-ES learning to train a low-dimensional vector. This scalability is essential considering the rapid growth of the PLM parameter scale, which allows the clients in FedBPT not to pay more communication costs when the FL system adopts larger PLMs. Some gradient-based methods outperform FedBPT in accuracy for some settings, which is expected. However, we should realize that gradient-based methods require conducting back-propagation, which are not always realistic for most cases of FL on edge devices, and only the gradient-free methods are feasible in many cases.

5.3. Ablation studies

Local population size (λ_k). In each iteration of local search, the clients (e.g., the k-th client) sample λ_k candidates for evaluation. We study the effect of local population λ_k on the model accuracy. We set λ_k from 5 to 20, and conduct experiments on SST-2 and AG's News for RoBERTa. The results are shown in Figure 5. It is shown that the model accuracy of FedBPT is not sensitive to λ_k. Thus, in real applications, λ_k can be set relatively small to reduce computational cost.

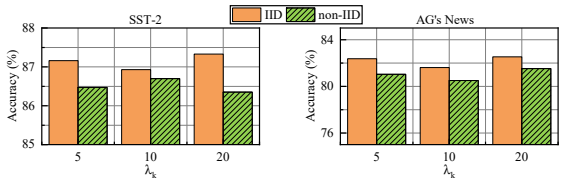


Figure 5. Results of FedBPT adopting RoBERTa with different λ_k.

Local binary mask rate (r_p). We study the effect of the rate of zeros in the binary masks m_i^k that local devices apply to perturb input and avoid overfitting. We conduct experiments on SST-2 and AG's News under the non-IID setting for RoBERTa. As introduced in Sec. 4.4, a larger r_p means that more tokens in a sentence will be randomly replaced. We set r_p from 0% to 80%, and the results are shown in Tab. 4. It is shown that applying the random placement can improve the global accuracy compared with simply adopting vanilla BBT for local training (i.e., $r_p = 0$). This illustrates the effectiveness of our designed random placement in mitigating the local overfitting challenge.

Dataset	SST-2				
r_p	0	0.2	0.4	0.6	0.8
Acc. (%)	84.86	85.21	86.03	**86.47**	86.12
Dataset	AG's News				
r_p	0	0.2	0.4	0.6	0.8
Acc. (%)	78.28	80.92	**81.03**	80.75	80.83

Table 4. Results of FedBPT adopting RoBERTa with different r_p under non-IID settings.

6. Conclusion

We introduced an FL framework, FedBPT, allowing clients to adapt black-box PLMs efficiently using gradient-free optimization. This approach eliminates the need for clients to access and finetune model parameters and only requires forward propagation for local training, thus lowering computational and memory demands for devices. Evaluations of several datasets with SOTA PLMs revealed that FedBPT matches the accuracy of gradient-based methods but with markedly less communication and memory overhead.

Acknowledgements

This material is based upon work supported by the U.S. National Science Foundation under award No.2112562. This work is also supported by ARO W911NF-23-2-0224. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. National Science Foundation, ARO, and their contractors.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning and Federated Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aghajanyan, A., Zettlemoyer, L., and Gupta, S. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

Alyamkin, S., Ardi, M., Berg, A. C., Brighton, A., Chen, B., Chen, Y., Cheng, H.-P., Fan, Z., Feng, C., Fu, B., et al. Low-power computer vision: Status, challenges, and opportunities. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):411–421, 2019.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Chai, Z., Chen, Y., Zhao, L., Cheng, Y., and Rangwala, H. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. *ArXivorg*, 2020.

Chen, H.-Y., Tu, C.-H., Li, Z., Shen, H.-W., and Chao, W.-L. On pre-training for federated learning. *arXiv preprint arXiv:2206.11488*, 2022a.

Chen, J., Xu, W., Guo, S., Wang, J., Zhang, J., and Wang, H. Fedtune: A deep dive into efficient federated fine-tuning with pre-trained transformers. *arXiv preprint arXiv:2211.08025*, 2022b.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Hambardzumyan, K., Khachatrian, H., and May, J. Warp: Word-level adversarial reprogramming. *arXiv preprint arXiv:2101.00121*, 2021.

Hansen, N. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

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

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

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

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Liu, X., Ji, K., Fu, Y., Tam, W. L., Du, Z., Yang, Z., and Tang, J. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., Yang, Z., and Tang, J. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, 2022.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. Gpt understands, too. *AI Open*, 2023.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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

Nguyen, J., Wang, J., Malik, K., Sanjabi, M., and Rabbat, M. Where to begin? on the impact of pre-training and initialization in federated learning. *arXiv preprint arXiv:2210.08090*, 2022.

OpenAI. Ag's news dataset.

Qin, G. and Eisner, J. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*, 2021.

Qin, Y., Wang, X., Su, Y., Lin, Y., Ding, N., Liu, Z., Li, J., Hou, L., Li, P., Sun, M., et al. Exploring lowdimensional intrinsic task subspace via prompt tuning. *arXiv preprint arXiv:2110.07867*, 2021.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10): 1872–1897, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Schick, T. and Schütze, H. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*, 2020.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sun, G., Mendieta, M., Yang, T., and Chen, C. Exploring parameter-efficient fine-tuning for improving communication efficiency in federated learning. *arXiv preprint arXiv:2210.01708*, 2022a.

Sun, J., Li, A., Wang, B., Yang, H., Li, H., and Chen, Y. Provable defense against privacy leakage in federated learning from representation perspective. *arXiv preprint arXiv:2012.06043*, 2020.

Sun, J., Li, A., Duan, L., Alam, S., Deng, X., Guo, X., Wang, H., Gorlatova, M., Zhang, M., Li, H., et al. Fedsea: A semi-asynchronous federated learning framework for extremely heterogeneous devices. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, pp. 106–119, 2022b.

Sun, T., Shao, Y., Qian, H., Huang, X., and Qiu, X. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855. PMLR, 2022c.

Sun, T.-X., Liu, X.-Y., Qiu, X.-P., and Huang, X.-J. Paradigm shift in natural language processing. *Machine Intelligence Research*, 19(3):169–183, 2022d.

Sun, Y., Wang, S., Feng, S., Ding, S., Pang, C., Shang, J., Liu, J., Chen, X., Zhao, Y., Lu, Y., et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*, 2021.

Tang, M., Ning, X., Wang, Y., Sun, J., Wang, Y., Li, H., and Chen, Y. Fedcor: Correlation-based active client selection strategy for heterogeneous federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10102–10111, 2022.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Weller, O., Marone, M., Braverman, V., Lawrie, D., and Van Durme, B. Pretrained models for multilingual federated learning. *arXiv preprint arXiv:2206.02291*, 2022.

Wen, S., Zeng, Z., Huang, T., and Chen, Y. Passivity analysis of memristor-based recurrent neural networks with time-varying delays. *Journal of the Franklin Institute*, 350(8): 2354–2370, 2013.

Xu, M., Song, C., Tian, Y., Agrawal, N., Granqvist, F., van Dalen, R., Zhang, X., Argueta, A., Han, S., Deng, Y., et al. Training large-vocabulary neural language models by private federated learning for resource-constrained devices. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

yelp. Yelp dataset challenge. URL https://www.yelp.com/dataset/challenge.

Yi, L., Yu, H., Wang, G., and Liu, X. Fedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*, 2023.

Zaken, E. B., Ravfogel, S., and Goldberg, Y. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

Zeng, W., Ren, X., Su, T., Wang, H., Liao, Y., Wang, Z., Jiang, X., Yang, Z., Wang, K., Zhang, X., et al. Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation. *arXiv preprint arXiv:2104.12369*, 2021.

Zhang, Z., Han, X., Zhou, H., Ke, P., Gu, Y., Ye, D., Qin, Y., Su, Y., Ji, H., Guan, J., et al. Cpm: A large-scale generative

chinese pre-trained language model. *AI Open*, 2:93–99, 2021.

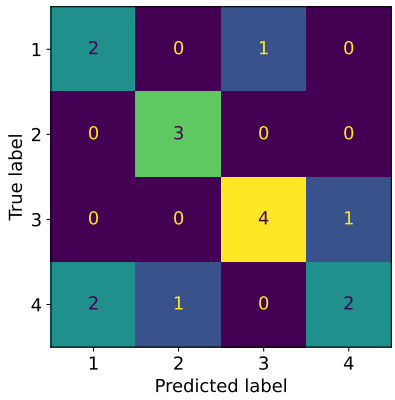
Zhao, H., Du, W., Li, F., Li, P., and Liu, G. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

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

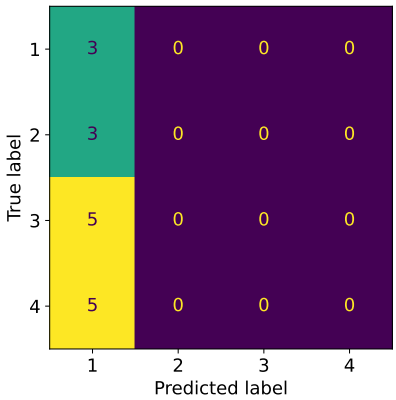
Zhong, Z., Friedman, D., and Chen, D. Factual probing is [mask]: Learning vs. learning to recall. *arXiv preprint arXiv:2104.05240*, 2021.

A. Overfitting Problem of Local Training via BBT

To demonstrate the overfitting problem of local training by applying BBT, we conduct experiments on AG's NEWS (OpenAI), a topic classification dataset with four data classes. We simulate an FL client to train prompts for a pre-trained RoBERTa (Liu et al., 2019) model using BBT. The simulated client holds data following the Dirichlet distribution, commonly applied in previous FL papers (Hsu et al., 2019; Tang et al., 2022) for non-iid setting, and more than 90% of its data are in class one. The confusion matrix evaluated with the prompt trained by this client is shown in Figure 6. It is shown that all of the data will be classified as class one after applying the prompt trained by this client, which demonstrates the problem of overfitting caused by local BBT.



(a) Results without prompt.



(b) Results of locally trained prompts.

Figure 6. Confusion matrix of a client holding data that more than 90% is in class one.

B. Experimental Hyperparameters Setup

For IID settings, we split the training dataset D_{train} evenly. For non-IID settings, we follow previous works to split the data following the Dirichlet distribution parameterized by α. We maintain a default setting of α = 1.0 throughout our experiments. The initial search step length σ_1 is 1. We set local iteration I to 8 and the local population λ_k to be 5 for all clients.

C. Manual Prompt Templates

Model	Dataset	Template
RoBERTa	SST-2	$\langle S \rangle$. It was $[MASK]$.
	AG's News	$[MASK]$ News: $\langle S \rangle$
	Yelp	$\langle S \rangle$. It was $[MASK]$.
Llama 2	SST-2	What is the sentiment of this sentence: $\langle S \rangle$? $[OUTPUT]$
	AG's News	$\langle S \rangle$. The news is about $[OUTPUT]$
	Yelp	$\langle S \rangle$. It is $[OUTPUT]$

Table 5. Manual prompt templates. $\langle S \rangle$: sentence in the dataset. $\langle MASK \rangle$: mask token adopted by RoBERTa. $[OUTPUT]$: output placeholder of Llama 2.

We set manual prompts for different datasets as shown in Tab. 5. We set different prompts for RoBERTa and Llama 2 according to their pre-training strategy. For RoBERTa pre-trained on masked language modeling (MLM) tasks, we apply manual prompts to organize the datasets as MLM datasets. For Llama 2, which is pre-trained on causal language modeling (CLM) tasks, we organize the samples to prompt the model to generate labels.