Federated Few-Shot Learning for Mobile NLP

Dongqi Cai
Beiyou Shenzhen Institute

Shangguang Wang
Beiyou Shenzhen Institute

Yaozong Wu
Beiyou Shenzhen Institute

Felix Xiaozhu Lin
University of Virginia

Mengwei Xu
Beiyou Shenzhen Institute

ABSTRACT

Natural language processing (NLP) sees rich mobile applications. To support various language understanding tasks, a foundation NLP model is often fine-tuned in a federated, privacy-preserving setting (FL). This process currently relies on at least hundreds of thousands of labeled training samples from mobile clients; yet mobile users often lack willingness or knowledge to label their data. Such an inadequacy of data labels is known as a few-shot scenario; it becomes the key blocker for mobile NLP applications.

For the first time, this work investigates federated NLP in the few-shot scenario (FedFSL). By retrofitting algorithmic advances of pseudo labeling and prompt learning, we first establish a training pipeline that delivers competitive accuracy when only 0.05% (fewer than 100) of the training data is labeled and the remaining is unlabeled. To instantiate the workflow, we further present a system `FeS`[1] , addressing the high execution cost with novel designs: (1) Curriculum pacing, which injects pseudo labels to the training workflow at a rate commensurate to the learning progress; (2) Representational diversity, a mechanism for selecting the most learnable data, only for which pseudo labels will be generated; (3) Co-planning of a model's training depth and layer capacity. Together, these designs reduce the training delay, client energy, and network traffic by up to 46.0×, 41.2× and 3000.0×, respectively. Through algorithm/system co-design, `FeS` demonstrates that FL can apply to challenging settings where most training samples are unlabeled.

[1] `FeS` is available at https://github.com/UbiquitousLearning/FeS.

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3613277

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Federated Learning, Natural Language Processing, Few-shot Learning

ACM Reference Format:
Dongqi Cai, Shangguang Wang, Yaozong Wu, Felix Xiaozhu Lin, and Mengwei Xu. 2023. Federated Few-Shot Learning for Mobile NLP. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3570361.3613277

1 INTRODUCTION

Mobile NLP & training data Mobile NLP sees rich applications on mobile devices. Examples include auto completion, QA, and sentiment analysis [46, 75, 80, 83, 108]. NLP models are trained in two phases. (1) *Pre-training* initializes a foundation model (e.g., BERT [23]). This phase learns language representations. (2) *Fine-tuning* adjusts the model to specific NLP tasks and domains, such as to tag named identities (a task) in a user text message (a domain). Of NLP training, a core issue is training data. While pre-training is self-supervised and only needs unlabeled data, fine-tuning data is more difficult to obtain for the following reasons.

First, fine-tuning often requires mobile users' private data such as their text messages. Fortunately, such a privacy concern is mostly addressed by federated learning (FL) [13, 43, 62, 100, 104], in which clients cooperate to train a model without sharing their raw data.

A much bigger challenge is the need for data labels. While prior ML research tackled training with scarce labels (referred to as few-shot or zero-shot scenarios) [15, 21, 30, 31, 74], the mobile environment exacerbates such scarcity, as most mobile users lack motivations for labeling their data (e.g., to tag which words of a text message belong to named identities) [95] or the knowledge to do so (e.g., is a given sentence subjective or objective?) [71]. As a result, most clients are likely to have no labeled data (although they may have

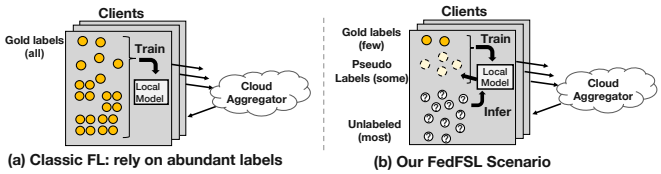


Figure 1: Classic and few-shot federated scenarios

abundant unlabeled data); across all clients, only a small fraction of the total data is labeled [73, 84, 95].

FedFSL We address mobile NLP training where data labeling challenges are at their extreme. (1) Highly scarce data labels, e.g. less than 0.1% of all, which are 1–2 orders of magnitude smaller than prior few-shot research [24, 26, 27, 42, 103, 111] (at least 4%). (2) Highly skewed label availability, as opposed to most prior work that rely on *uniformly* distributed labels [47, 51, 76, 104]. As we will show (§2.4), these two challenges render most prior work inadequate, resulting in significant loss in training accuracy. To this end, we design a runtime system for few-shot NLP learning in federated settings, which we refer to as FedFSL.

ML building blocks We identify two ML algorithms as key building blocks underpinning practical FedFSL. (1) *Pseudo labeling* [50] allows to train a model $\mathscr{M}$ with a small number of labeled samples (called gold labels) together with many unlabeled samples. To do so, $\mathscr{M}$ makes inference on unlabeled training data, from which high-confident results (i.e., pseudo labels) are selected to further train $\mathscr{M}$. Such positive feedback reinforces the model's capability in decision making. (2) *Prompt learning* [73] is a recent NLP advance that better adjusts pretrained models to downstream tasks. With it, even a weak model is able to generate pseudo labels with less errors. Integrating the two algorithms, we show that a model can be fine-tuned with only tens of gold labels, while still achieving 85.8%–97.0% of relative accuracy as compared to a fully supervised training with thousands of labels.

Challenges Despite of satisfactory accuracy, FedFSL can be far more expensive than standard FL, notably:

- **Planning for FSL**. Our FedFSL pipeline must run inference and training in an iterative fashion. In an inference round, each participating client generates pseudo labels. Using both its gold and pseudo labels, a client runs multiple training rounds before uploading its local model updates to the cloud server. Having aggregated updates from many clients, the server dispatches an updated model to clients for future pseudo labeling.
  This process crucially depends on a coherent plan for inference and training: what pseudo labels to be injected to the learning process and at what pace. For instance, generating too few pseudo labels per round slows down training; generating too many pseudo labels, especially when the model is still weak, results in excessive erroneous labels that mislead training. The decision must also be dynamic, catering to different datasets and different times in a training session.
- **Excessive on-device inference**. After receiving an aggregated model from the server, a mobile client may run inference on *all* its unlabeled data for pseudo labeling [17]. However, most of the inference is in vain, as only a small fraction (the most confident) of pseudo labels will be selected for subsequent training. The inference dominates the total energy cost, up to 87.4% per our measurement. The clients need an efficient mechanism for skipping generating pseudo labels for much of the data.
- **Training a large model on-device**. Language prompts rely on a large foundation model to work, as only a large model contains sufficiently rich knowledge amenable to extraction via prompts [12, 57]. Compared to models commonly used for FL, e.g., DistilBERT [72] and ALBERT [49], a large model such as RoBERTa-large [59] achieves up to 38.9% higher accuracy, at the cost of 7.6× more computation and 4.8× more memory. Training with a small batch size 4 takes 75.9 seconds, 910.8 joules, and over 10 GB of memory on mainstream mobile hardware.

Our solution We present FeS, the first framework that manages Federated Few-Shot learning for mobile NLP. Corresponding to the challenges above, FeS centers on three new designs.

Curriculum pacing (§3.1): To plan pseudo labeling and training, the key is to be commensurate with the learning progress. Intuitively, only as the model becomes confident via training, the client may pick increasingly more pseudo labels per round. Specifically, we characterize the *training pace* as a configuration $\pi = \langle f, n, k \rangle$, where f is the number of training rounds between re-generations of pseudo labels; n is the number of participating clients; k is the confidence threshold for selecting pseudo labels. Continuously weighting the model's recent learning progress against the training cost, FeS probes for configurations and switches to the most suitable ones, pacing through a dynamic *curriculum* of training.

On-device inference with representational diversity (§3.2): To reduce the inference effort, a client only selects its samples worth learning most and generates their pseudo labels accordingly. How to identify such samples? Motivated by representation learning [38, 61, 70], our rationale is to diversify their data representations. For instance, the selected samples could be sentences showing varying lengths and word frequencies. The rationale entails a lightweight implementation: a client estimates the *approximate* representations of its samples by running inference with a low-cost proxy model; the client only does the estimation once, ahead

of any training sessions; during a training session, the client selects samples for pseudo labeling by jointly considering representativeness and diversity. We are the first to apply representational filtering in the FedFSL scenario, i.e., a training-inference collaborative pipeline.

Co-planning of training depth and layer capacity (§3.3): A canonical approach to efficient fine-tuning is to only train the top layers[2] which encode task-specific knowledge while freezing the bottom layers which encode task-agnostic knowledge. For FedFSL however, we find that layer freezing alone is inadequate, resulting in inferior model accuracy. We attribute the observation to that much of the task-agnostic knowledge shall be adjusted as well in case of label scarcity. Therefore, FeS controls a model's training depth and its capacity jointly. Specifically, each client trains a model's top layers with full layer capacity, trains the middle layers with reduced capacity, and freezes the very bottom layers. The client reduces a layer's capacity by only tuning its bias while freezing its weights, a technique retrofitted from prior work [60, 78, 105]. FeS is the first to explore the synergy between layer freezing and model capacity, to our best knowledge.

It is worth noting that the above designs are compatible with existing FL optimizations such as client scheduling [53, 63] and training data sampling [52, 88]. FeS can be realized as an enhancement to existing FL frameworks, enabling them to operate with scarce labels.

Results We implement FeS and test it on two embedded devices: NVIDIA TX2 [1] and RaspberryPi 4B [2]. Large-scale training is emulated as prior FL literature does [48, 51, 53]. On a diverse set of NLP benchmarks, FeS reaches ~90% relative accuracy while requiring three orders of magnitude fewer samples. Compared to vanilla few-shot fine-tuning for NLP, FeS reduces the training delay from 2.1–9.1 hours to 0.1–0.4 hours (up to 46.0× reduction). Compared to strong baselines that use bias-only tuning [105], FeS still reduces the delay by 4.3×–14.6×. Our key designs contribute to the results significantly: curriculum pacing, representational filtering and depth/capacity co-planning reduce the delay by up to 3.5×/3.5×/62.3×, respectively. FeS for the first time fine-tunes a big language model (RoBERTa-large) on mobile/embedded devices with only 8GB of RAM; it reduces the network traffic for model aggregation by 1,841.7× and per-device energy consumption by 21.7× on average.

Contributions We present the first work that investigates NLP training with scarce data labels in a federated setting.

- **Algorithmic foundation** We identify the algorithm foundation as a combination of pseudo labels and language prompts. Compared to training with fully labeled data, we show it is possible to reduce the amount of labels by three orders of magnitude while still achieving 90% of the relative accuracy.
- **System designs** We tackle the high cost of FedFSL with novel designs: representational diversity (which optimizes inference), co-planning of learning depth and capacity (which optimizes training), and curriculum pacing (which orchestrates the two).
- **Experimental evaluation** Through experiments on real hardware, for the first time we show it is both desirable and practical for mobile devices to train NLP models – even with scarce labels.

[2]Top layers: layers closer to a model's output [39].

2 MOTIVATIONS

2.1 Mobile NLP and Its Obstacles

This work is concerned with NLP model *fine-tuning*: adapting a foundation model to various downstream tasks such as text classification and sequence tagging. While the foundation model is pre-trained by big companies *once*, the subsequent fine-tuning recurs for individual tasks and involves mobile clients. Therefore, fine-tuning has a strong impact on both the model accuracy and client efficiency.

Explored: fine-tuning is often privacy-sensitive. It relies on domain samples that are often generated by users, such as user reviews, messages, or emails. Collecting them to the cloud for training raises privacy concerns and is heavily regulated [65, 85]. In response, federated learning [62, 100] is the de facto approach that trains models with good accuracy without data sharing. Training NLP models in a federated setting is referred to as FedNLP [16, 104].

Unexplored: fine-tuning is often few-shot. While training samples on clients can be abundant, the *labeled* ones are often scarce. To exacerbate the problem, the numbers of labels could vary drastically across clients. Such skewed label distribution, combined with the non-IID data distribution nature in FL (e.g., skewed class distribution [55, 104]), could further degrade the fine-tuning accuracy. The causes for label scarcity are fundamental.

- *Users lack willingness.* Each sample is accessible to only one client user who can label it. Reports show that most users are reluctant to label their data [26, 99]. This is fundamentally different from traditional centralized or crowdsourced data labeling services that can recruit highly specialized data labelers [54].
- *Users lack expertise.* Data labeling for certain NLP tasks require domian-specific knowledge, which most users do not possess, e.g., cross-lingual transfer [68], Q&A [101], or biomedical text corpora understanding [114].
- *Diverse NLP tasks.* Downstream tasks are emerging over time, e.g., new domains, topics, or data distributions. Asking users to label a large amount of data for each task is tedious, inefficient, and impractical.

- *Mislabels are not uncommon.* Mislabels are common in real world, e.g., 6% in the well-established ImageNet or even more than 10% in other crowd-sourced datasets [64]. In FedFSL, since the labels from end users are merely impossible to be verified, we expect an even higher ratio of mislabels, which can significantly harm the model quality. Instead, trainers might only use the labels from very few, highly trustworthy people.

In essence, we argue that few-shot is a more realistic way to depict NLP training, a scenario we call as FedFSL. Unfortunately, FedFSL, in particular its system implications, are rarely investigated – in comparison, prior FL literature assumes abundant data labels (at least hundreds of thousands) uniformly distributed across clients.

2.2 Key Algorithm Blocks for FedFSL

In ML literature, there are two complementary approaches to address the few-shot issue. One is to exploit the abundant *unlabeled* data across clients. The other is to boost the model's ability of learning from few samples. For each approach, we identify a technique; together, they form the algorithmic foundation for FedFSL.

• **Pseudo labeling [50] allows training with few/zero labeled samples and many unlabeled samples**. As it trains a model, the trainer makes the current model infer on unlabeled data, and uses the inference results (i.e. pseudo labels) as if they are true labels for successive training. The efficacy of pseudo labeling has been established both empirically [19, 106] and theoretically [8, 50]. Intuitively, it works because training with pseudo labels encourages the model to learn a decision boundary that lies in a region where the example density is lower. Often, such a decision boundary yields good generalization performance (i.e. higher model accuracy), even though the true labels of individual samples remain unknown. In ML's lingo, such a training strategy roots in *entropy regularization*: the resultant model will make a prediction on unlabeled data with low class overlap (e.g. "great":0.9, "bad":0.1 rather than "great":0.6, "bad":0.4) and therefore low entropy. Pseudo labeling also tackles the challenge of skewed distribution of label classes. Since pseudo labeling involves more clients and therefore more diversified label classes for training, the fine-tuned model is likely to be more unbiased and accurate.

• **Prompt learning** [57] is a powerful NLP technique that boosts accuracy in model fine-tuning, which is commonly used in few-shot scenarios [31, 58, 60, 73]. For FedFSL, we find prompts crucial to the early stage of a training sessions, when the model is weak and can barely generate useful pseudo labels.

Given a task, standard NLP fine-tuning (without prompts) trains a new classification layer from scratch, which requires

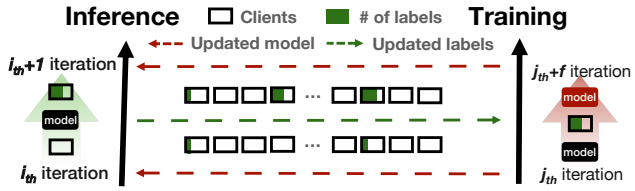


Figure 2: Workflow of FedFSL with pseudo labeling.

supervision from substantial labeled data. For example, suppose we fine-tune a foundation model to classify YELP reviews [110] and are only given two labeled samples [73]:

- T_1 (label=L1): "Most delicious pizza I've ever had."
- T_2 (label=L2): "You can get better sushi for half the price."

With such few samples, training a usable classification layer is impossible. Consider an unlabeled example:

- T_3 (label=?): Pizza was good. Not worth the price.

The model may predict T3's class probabilities closer to L1 if the task is to classify user satisfaction, or closer to L2 if the task is about whether price is mentioned. But without such a task description, the model can only randomly guess and generate an error-prone pseudo label.

To fix the problem, the insight of prompt learning is that the foundation model already encodes knowledge for performing various tasks; it just needs a prompt that describes what the task is.

In the example above, if ML developers augment *all* samples with a leading prompt, e.g. T3 becomes:

"**It was <MASK>**. Pizza was good..."

Then all input samples are reformulated as cloze-style phrases, which are exactly what the foundation model was pre-trained for – to predict missing words (masked out) in text. Next, the predicted masked word is mapped to a class, yielding a label. For instance, a predicted word "terrible" will map to label 1 and "great" will map to label 5. Compared to initializing a whole classification layer from scratch, the foundation model with prompts requires less finetuning before it can output labels with higher accuracy; in FedFSL, this means that the pseudo labels are less erroneous.

More formally, a cloze question is called a pattern and the mapping from words to classes is done by a verbalizer. Given a task, there exist multiple possible pattern/verbalizer pairs. The training loss is the cross-entropy between the correct answer and the distribution of probabilities among the tokens in the verbalizer. See recent surveys on prompt learning [12, 57] for more details.

2.3 Our System Model

System model: FedFSL workflow enhanced with above techniques We notice that pseudo labeling and prompt learning can well orchestrate and be complimentary to each

other: pseudo labeling heavily relies on the initial model accuracy to get enough, correct labels, for which prompt learning can help; in turn, prompt learning's ability is limited to the few number of data labels and especially their skewed distribution, for which pseudo labeling can help. Therefore, we construct an enhanced FedFSL workflow by orchestrating the two techniques atop FedNLP, shown in Fig. 2. This enhanced workflow is the algorithmic foundation of our future design, and is still dubbed as FedFSL for simplicity.

Our goal is to fine-tune a pre-trained language model $\mathscr{M}$ based on distributed clients' data. We assume that each client has a tiny training set with labels $\mathcal{T}$ (typically < 10) and a much larger set of unlabeled samples $\mathcal{D}$ (typically > 1000). In general, FeS consists of two loosely-coupled runtimes residing in a central server.

- **Inference runtime** that continously generates new pseudo labels on clients. Per f training rounds, it dispatches the global $\mathscr{M}$ to n clients, where the model exhaustively inferences on each local unlabeled data $\hat{x} \in D$ and generates a pseudo label $\hat{y}$. The data with the top k highest confidence (i.e., *logits*) are added as training samples. In subsequent training rounds, pseudo labels are treated equally as the gold labels. The pseudo labels that are generated in previous rounds will also be re-labeled to avoid forgetting events [82]. The above hyper-parameters $< f, n, k >$ indicate how inference runtime paces.
- **Training runtime** that follows a typical federated learning workflow to fine-tune $\mathscr{M}$. Per round, the runtime dispatches the global $\mathscr{M}$ to a random set of clients with at least one gold or pseudo label. The on-device training is assisted with prompts, provided by the trainers either in hand-crafted or automatic manner [29, 31, 56–58]. The updated models are then aggregated (default FedAvg protocol [62]) on the server as the new global $\mathscr{M}$. The process continues till $\mathscr{M}$ reaches a satisfactory accuracy. Notably, such a design is compatible with prior FL literature on client/data sampling [48, 51–53, 96], privacy enhancements [25, 107], and communication optimization [6, 90, 93].

2.4 Experimental Observations

Based on the FedFSL workflow presented above, we perform a set of early experiments on its performance. The results highlight the two sides of a coin: a satisfactory model accuracy yet huge resource cost on clients.

Observation-1: FedFSL achieves satisfactory accuracy with scarce data labels; for which both pseudo labeling and prompt learning are indispensable. Table 1 shows the convergence accuracy of RoBERTa-large [59] on 4 popular NLP datasets[3]. With only 64 data labels (0.005%–0.05% of the total dataset),

[3]You can find a detailed description of the datasets in §4.1.

Dataset	Full-set (oracle)	Vanilla-FedFSL	Prompt-Only	Pseudo-Only	Both (Ours)
AGNEWS (skewed)	93.0	64.8±3.1	68.4±2.4	67.5±1.3	**90.2**±0.5
MNLI (skewed)	85.0	37.7±5.6	42.4±5.8	42.7±6.3	**77.4**±1.2
YAHOO (skewed)	78.0	24.4±10.3	41.8±4.3	31.0±2.0	**66.9**±1.1
YELP-F (skewed)	70.0	38.3±8.8	51.2±1.8	45.7±4.4	**58.2**±2.4
YELP-F (uniform)	70.0	54.0±0.1	58.1±1.5	57.0±2.2	**61.9**±0.7

Table 1: Convergence accuracy with 64 gold labels. "Full-Set" assumes every data is labeled (an oracle case). "skewed" means the gold labels are located on few clients instead of uniformly distributed across clients.

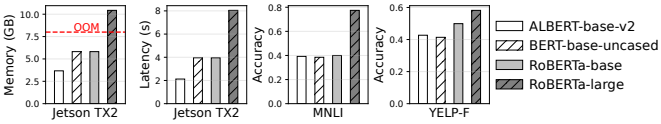


Figure 3: FedFSL convergence performance with different models and datasets. Batch size: 4.

FedFSL achieves 85.8%–97.0% relatively convergence accuracy to the full-set fine-tuning that assumes all data samples are labeled. The accuracy could be further boosted by involving more data labels. Neither pseudo labeling nor prompt learning alone is enough to exhibit a usable accuracy. With only one of them, the relative convergence accuracy is 40%–74%. Furthermore, the skewed label distribution challenges the task: on YELP-F, a vanilla FedNLP method results in much higher accuracy when the labels are uniformaly distributed; neverthelss, the challenge is mostly addressed by FedFSL that achieves satisfactory accuracy in both cases.

Observation-2: FedFSL incurs huge system cost. Our experiments highlight the excessive system cost (Figure 3), as against the commonsense that few-shot learning is usually fast and lightweight [21, 26, 32, 109]. For example, training RoBERTa-large on AGNEWS takes 3.3 hours to converge, 7.3 million Joules of energy, 68.4 GBs of network transmission, and 10.4 GB peak memory. The cost is about 1.4× higher than a full-set supervised FedNLP process on the same model and dataset. We then dive deeper into the implications behind and identify three challenges for a resource-efficient FedFSL system.

• **Orchestrating training and inference** FedFSL has two coupled components: a federated learning runtime that continuously updates a global model; an inference runtime that keeps generating pseudo labels. The two components must be paced harmoniously: the inference runtime generating too few pseudo labels could slow down the training; otherwise, generating too many pseudo labels could lead to resource waste or even excessive erroneous labels, especially when the global model is still weak. A mechanism to orchestrate the two components must be dynamic to fit the model learning progress.

• **Prompt learning needs large NLP model.** Compact NLP models have been proposed for mobile scenarios with acceptable accuracy degradation. However, our experiments demonstrate that a large, fully-fledged foundational language model is demanded in FedFSL. As shown in Figure 3, on `MNLI` and `YELP-F`, a large RoBERTa-large model (24 transformer blocks) can reach 91% and 88% relative accuracy to full-set fine-tuning, while Bert-base (12 transformer blocks) can only obtain 45% and 59%. The rationale is that a large language model also encodes rich knowledge for downstream tasks, from which hand-crafted prompts can better extract useful information within limited data labels.

• **Excessive on-device inference to obtain trustful pseudo labels.** Pseudo labeling typically requires performing inference on all unlabeled data, from which the most confident/valuable ones can be selected for future training. Note that the inference is not one-pass, as the model is continuously updated each round. According to our experiments, up to 87.4% of the total energy cost on clients is attributed to the inference. As will be shown in §3.2, randomly filtering out a large portion of samples to speed up pseudo labeling will degrade accuracy significantly.

3 FES DESIGN

Atop the FedFSL workflow presented in §2.2, `FeS` introduces three key techniques to make it systematically practical by addressing each of the challenges raised in §2.4.

3.1 Curriculum Pacing

`FeS` proposes to progressively speed up the pseudo labeling speed, i.e., adding more pseudo labels at a higher frequency. The rationales are two folds. First, at the beginning of FedFSL, the language model is relatively weak and the produced labels are prone to be erroneous, from which we better pick only the very confident ones; as the training progresses, the model becomes accurate enough to produce trustful pseudo labels in faster speed. Second, as the model gets more accurate, the utility of each data sample to further enhance the model diminishes. It demands recruiting more data to sustain an effective learning progress.

The configuration space. `FeS` further probes into more detailed pacing configurations, and distills three key parameters: f indicates frequency of updating pseudo labels, i.e., the number of training rounds before the next pseudo labeling; n indicates the number of clients selected to perform pseudo labeling; k indicates the curriculum ratio (linear increase) of data selected as pseudo labels for the subsequent training per selected client. As long as k is positive (e.g., 1%), the pseudo labeling speeds up (e.g., 1% selected at 1st round, 2% selected at 2nd round, etc). For example, <f,n,k>=<5,2,1> means inference line would provide 1% more pseudo labels compared to the prior updating from 2 clients every 5 rounds. The three parameters form a configuration (denoted as $< f, n, k >$) that can flexibly control the relative pacing between the training and inference. As shown in Figure 4a, various configuration leads to huge accuracy and cost tradeoff. Meanwhile, the best configuration varies across datasets and models, i.e., no silver-bullet configuration.

Algorithm 1: Our Pacing Configurator

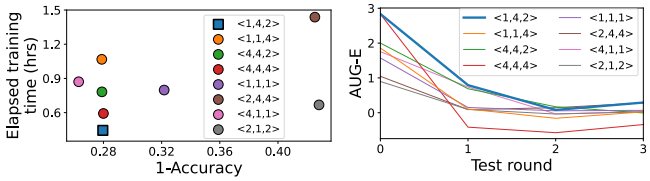
input : Target accuracy, acc;
Initial configuration list, l;
Number of candidate configurations, t;
AUG-E threshold, s;
Trial round, r.
output : Fine-tuned model, Θ_i (i=1,2...).

Function Cloud_controller():
  Iteration i=0;
  $l_{win}, l_{candidate}, i \leftarrow$ Switch(i, l); // initial pace configuration
  while Eval(Θ_i) < acc **do**
    Pacing(i, l_{win}); // training and labeling concurrently
    $E \leftarrow$ Compute AUG-E; // accuracy degradation detect
    if E<s **then**
      $l_{win}, l_{candidate}, i \leftarrow$ Switch($i, l_{candidate}$);
  Exit training.
Function Switch(i, $list$):
  $i_{tmp} \leftarrow i$;
  for l in $list$ **do**
    while $i < i_{tmp} + r$ **do**
      Pacing(i, l); i++;
    $E \leftarrow$ Compute AUG-E;
  $l_{win}, l_{candidate} \leftarrow$ Configurations with highest E and top-t E.
Function Pacing(i, l):
  $G_{train}, G_{label} \leftarrow$ Selects clients groups separately;
  Send model Θ_i and configuration l to G_{train} and G_{label};
  Parallel: $Client\_labeling(i), Client\_training(i)$;
  $\Theta_{i+1} \leftarrow$ Aggregate receive updated model from C_{train}.
Function Client_training(i):
  $\Theta_{i+1}(n) \leftarrow$ Update received model Θ_i on local labeled data;
  Send updated model $\Theta_{i+1}(n)$ to cloud.
Function Client_labeling(i):
  $labels \leftarrow$ Generate pseudo labels per pacing configuration.

Lightweight configuration searching To search for an effective configuration with low cost, we define a new metric *augment efficiency* (AUG-E) to measure the gradient of the time-to-accuracy curve.:

$$AUG\text{-}E(f, n, k) = \frac{\eta \Delta(acc)}{C_{infer}(f, n) + \theta \cdot C_{train}(k)} \quad (1)$$

where $C_{infer} = l_i \cdot n/f$ and $C_{train} = l_t \cdot k$. Here, l_i stands for inference latency and l_t stands for training latency per batch. AUG-E takes both (accuracy) gain and the cost for this gain into account. Higher the AUG-E, more accuracy benefit is brought from pseudo labeling cost.



(a) Impacts of configuration **(b) AUG-E across rounds**

Figure 4: AUG-E metric at early rounds (a) can help identify the pacing configurations that perform well in end-to-end training (b) regarding both accuracy and system cost (the higher AUG-E, the better). $< f, n, k >$ is the pacing configuration. Dataset: MNLI [91].

Algorithm 1 describes how FeS leverages AUG-E for configuration searching in details. At the beginning, FeS tries each configuration from an initial list we hand-picked through extensive offline experiments (default size 32). After only a few rounds, FeS evaluates those configurations using AUG-E: the best one will be selected for future pseduo labeling runtime (line 3, 10–16); the top-t (default 8) ones are packed into a candidate list for future update in case (see below).

To understand how AUG-E helps predict those efficient configurations ahead, we profile 8 random configurations $< f, n, k >$ and show their convergence performance and AUG-E metric on MNLI dataset [91]. Figure 4 shows that the most effective configurations with higher accuracy and smaller training time would obtain a relatively higher AUG-E score in the early stage of searching. For example, the configuration <1,4,2> with the highest AUG-E converges at 72.0% accuracy (2nd highest) within 0.4 hours (fastest). Therefore, we can use AUG-E as an indicator to the end-to-end performance of different pacing configurations.

Configuration switching In practice, we observe few cases that the picked configuration performs badly as training goes on. To mitigate the impacts of those corner cases, inspired by [16], FeS adopts a configuration switching mechanism at online. As described in Algorithm 1, once training alarms due to sharp accuracy degradation (line 6–8), i.e., AUG-E is below zero or extremely low, FeS seeks to switch the pacing configuration (usually speeds up the inference). More specifically, FeS repeats the configuration searching as it does at the beginning rounds (line 10–16), but only with the short top-t list of candidates that are proven to be relative effective as discussed above.

Cost analysis The cost of configuration searching is negligible as it spans only a few beginning rounds (typically 5); and online switching on top-t configurations rarely happens (typically < 2). Please note that the trials on different configurations could be amortized by leveraging the large amount of idle devices in federated learning [13, 98].

Wrongly labeled data If malicious clients are unluckily selected, FeS could encounter unexpected behavior such as low AUG-E score. Those wrongly labeled data will be flagged as 'unlabeled'. Subsequently, these data points undergo relabeling using our pseduo-labeling mechanism. Another advantage of FeS is that it requires only a small number of labeled training data, often in the tens. Consequently, it becomes easier for the cloud to identify trustworthy clients whose data labels are more likely to be accurate.

3.2 Representational Filtering

To circumvent the exhaustive labeling (model inference) of all local data, an intuitive approach is to early filter the data that is likely to contribute minimally to subsequent training. There are two key questions to be answered: (1) what metrics shall be used to quantify the value of a sample if it is important for training; (2) How can these metrics be efficiently extracted for each sample? Ideally, this process should be decoupled from the NLP model that is being continuously trained, allowing for an offline, one-pass operation.

Representativeness- and diversity-aware score The key idea of FeS is to jointly consider two data aspects: *representativeness* helps many text instances to find similar demonstrations, thus reducing duplicate labeling (inferring) cost for similar samples; *diversity* guarantees enough statistical utility, thus increasing the total convergence performance.

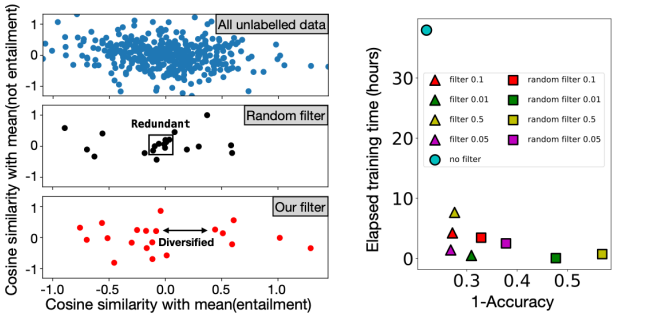
To do so, FeS first computes a vector representation for each unlabeled training instance (a sentence) $x \in \mathcal{X}$, by averaging the output vector for each of its word using the proxy model discussed below. For each sample x, we sort its k most similar samples in terms of the cosine similarity between the embedding vectors. Those example ids are denoted as $\mathcal{V}(x)$. We denoted representative vector as $\mathcal{E}(u) = \{x \mid u \in \mathcal{V}(x), x \in \mathcal{X}\}$. $|\mathcal{E}(u)|$ means the quantity of samples that sample u is similar to. Bigger the $|\mathcal{E}(u)|$, larger the representativeness.

Now let $\mathcal{L}$ and $\mathcal{U}$ denote the sets of already chosen samples and remaining samples, respectively. Initially, $\mathcal{L} = \emptyset$. Every remaining sample $u \in \mathcal{U}$ is scored as below:

$$\text{score}(u) = \sum_{x \in \mathcal{E}(u)} \rho^{-|\{\mathcal{V}(x) \cap \mathcal{L}\}|}, \quad \rho > 1 \tag{2}$$

where ρ discounts x that is close to the already selected instances, thereby encouraging diversity. A sample with higher score(u) would be preferred for labeling. Once it is selected, it will move from $\mathcal{U}$ to $\mathcal{L}$. Scores would be updated subsequently. According to Figure 5b, up to 95% online inferring cost could be saved without harming convergence performance apparently. This speeds up the end to end performance significantly.

Low-cost proxy model Existing importance-based filtering methods observe training loss [76], or weight norm [7, 44] to measure the importance of each training data. Those



(a) Representative diversity (b) End-to-end performance

Figure 5: FeS's representational filter helps find out a small portion of samples (20%) that are diversified in text semantic as compared to random filtering (a). In end-to-end experiments (b), it accelerates convergence by reducing pseudo labeling cost with negligible accuracy loss. Dataset: YAHOO [110].

methods are not feasible in FedFSL that lacks data labels. Inspired by sentence annotating task [38] and sentence-pair regression task [70], we propose to use a BERT-like model to obtain a feature representation for each data sample.

We use Sentence-BERT [70] as the proxy model, which is priorly used for sentence similarity comparison. It is a medium-sized NLP model that consists of 12 transformer blocks, which is notably smaller than the NLP model being trained in FedFSL as discussed in §2.2. Furthermore, the proxy model is trained offline and independent from the NLP model to be trained through FedFSL. Therefore, computing the vector representation of each data is one pass and incurs much less cost than performing inference each round. For example, when computing the vector representation of YELP-F [110], it incurs only an additional 8.4% time cost compared to the FedFSL training process. Moreover, this cost can be further amortized across multiple FedFSL tasks since they share the same representative vectors.

Micro experiments are conducted to show how this approach works better than a random strategy. We use 392 unlabeled samples from the same client on MNLI [91], and compare the filtered samples through our approach and a random one. In Figure 5a, we visualize the distance between those samples by the cosine similarity on the Sentence-BERT output. A key observation is that, our representativeness-aware filter can effectively select more diversified samples for pseudo labeling than a random filter. We further compare the end-to-end performance in Figure 5b with the same setting used in §4. Our approach filters 95% unlabeled samples and brings up to 7.2× convergence speedup on MNLI, with less than 1% accuracy loss. In comparison, randomly filtering data for pseudo labeling degrades the model accuracy significantly.

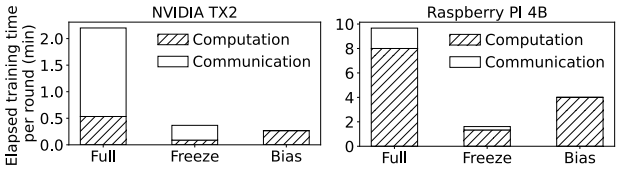


Figure 6: The per-round compute and communication time with different on-device training optimizations. Full: training everything; Freeze: freeze as many layers as possible with acceptable loss; Bias: training only bias for each layer.

3.3 Training Depth/Capacity Co-planning

In order to enhance training efficiency in terms of aspects such as energy consumption per batch and memory usage, a prevailing method is to limit the *depth* of the layers [33, 104]. This technique, also referred to as *freezing*, ensures that backward propagation is only performed across the top k layers nearest to the model output and encapsulate task-specific knowledge. Another underutilized strategy pertains to controlling the *capacity* of the layer, or determining how many weights to update within each layer. Recent research [18, 60, 105] indicates that adjusting only the bias of a layer, which typically represents a meager 0.1% of total weights, whilst maintaining the other weights as frozen, does not detrimentally affect model accuracy. This is attributed to that bias values are responsible for the predicted class or hidden-state. Altering the bias allows for an efficient modification of the output vocabulary, thereby mitigating the risks of catastrophic forgetting or overfitting [28, 36, 45, 102].

Observation: compute-network tradeoff by controlling the layer depth and capacity. In the context of FedFSL, however, we observe that controlling either the layer depth or capacity alone is inadequate towards resource-efficient training on client devices. Layer freezing can almost linearly scale down the resource cost with the freezed depth, e.g., up to 83% in AGNEWS (20 out of 24 layers) without accuracy degradation. However, the non-freezed layers still comprise of hundreds of MBs weights, in which case the network transmission bottlenecks the learning process. Meanwhile, controlling the layer capacity by only updating model bias brings much more significant reduction on the communication; yet the compute time bottlenecks as it still demands a complete forward-backward pass.

Naive use of bias-tuning brings a significant runtime reduction on high-end mobile devices. But on a much wimpy device, the speed-up drops. As shown in Figure 6, on NVIDIA TX2, bias-tuning reduces the elapsed training time per FL round by 8.2×. Because TX2 is with strong GPU capacity, the compute cost is not heavy, leaving the network as the bottleneck, which is alleviated by bias-tuning. While on Raspberry Pi 4B, bias-tuning's speed-up drops to 2.4×, because RPI4B computes slow with weak CPU capacity, which is the

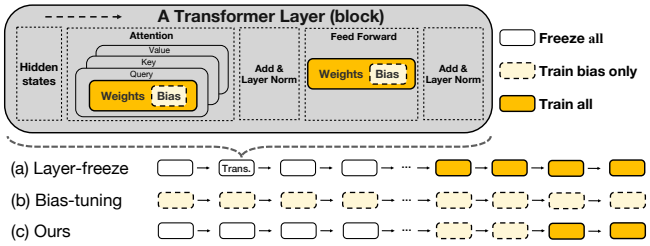


Figure 7: Comparing our approach of co-planning the tuned layer depth and capacity to traditional ones.

bottleneck that vanilla bias-tuning could not handle. In contrast, freezing the top layers can reduce the elapsed training time per round by 6.0× on RPI4B because it brings linear computation reduction.

Co-planning the tuned layer depth and capacity To balance the compute and network, FeS carefully controls both the layer depth and capacity. Simply applying both techniques, i.e., freezing some layers and updating the bias of the rest layers, does not fully exploit the potential of them. Instead, we propose a mixed depth/capacity tuning paradigm shown in Figure 7: trains very few top layers with full capacity; trains a few middle layers with reduced capacity (i.e., only bias updated); freezes the other bottom layers. The rationale of such a terraced design is that, the layer closer to the output encodes more downstream knowledge which can be extracted with limited data labels. The concrete tuning decision is done offline on cloud. Specifically, we use a FedFSL simulator that uses binary search to identify the optimal configuration. The simulator takes three types of input: pre-trained model; datasets, which could be a public one with similar classification difficulty to the private dataset distributed across clients; the estimated FedFSL runtime parameters including on-device training time and network bandwidth.

In comparison, a random scheme often results in low convergence accuracy or high resource costs due to either freezing too many or too few layers. For instance, when considering the AGNEWS dataset, compared to our cherry-picked plan (where we freeze 23 out of 24 layers, excluding only the bias of layers 16–23), freezing all 23 layers leads to a significant 4.1% accuracy loss, while freezing just 16 layers incurs a slowdown of 13.8× to achieve only a marginal accuracy gain.

Integration with other parameter-efficient fine-tuning methods Enormous parameter-efficient fine-tuning methods are off-the-shelf to fine-tune large language models, such as adapter [16, 67], LoRA [37, 40], etc. Our system already includes one popular technique: bitfit [105], that greatly saves tunable parameters while preserving the few-shot ability of large language models [60]. Other parameter-efficient fine-tuning methods could also interplay our co-planning schedule and reap benefits. For instance, adapters could be utilized to fine-tune the middle layers with reduced capacity, while fine-tuning the very top layers with full capacity, which will not only lead to high parameter-efficiency but also high computation-efficiency.

Dataset	AGNEWS [110]	MNLI [91]	YAHOO [110]	YELP-F [110]
# Training	120k	392.7k	1.4M	650k
# Test	7.6k	9.8k	60k	50k
# Clients	100	1000	1000	1000
# Labels	64	64	64	64
Distribution	Skewed	Uniform	Skewed	Skewed
Prompt	a ____ b	a ?____, b	Category: a ____ b	It was ____. a

Table 2: Evaluation datasets. Label quantity of each class follows prior work [104] where $\alpha = 1$. Please note that 64 is the total number of labels across clients, not per client.

Setup	Labeling		Training	
	Pacing	Optimization	Method	Optimization
FedCLS	/	/	Head-based	/
FedFSL	Static	/	Prompt-based	/
FedFSL-BIAS	Static	/	Prompt-based	Bias-only tuning
FeS (Ours)	Curriculum (§3.1)	Filtering (§3.2)	Prompt-based (§2.2)	Depth/Capacity Co-planning (§3.3)

Table 3: Summary of baselines used in experiments.

4 EVALUATION

We evaluate FeS to answer the following key questions: 1) How much performance improvement (in terms of time-to-accuracy and relative model accuracy) does FeS achieve? 2) How much performance improvement does FeS achieves across different number of gold labels? 3) How much performance improvement does each component of FeS contribute? 4) How much resource does FeS save?

4.1 Implementation and Setup

FeS prototype We have fully implemented the FeS prototype atop PET [73] and FedNLP [104]. PET is a popular prompt learning framework for NLP tasks. FedNLP is the state-of-the-art framework for evaluating NLP tasks under federated setting. As prior work [13], we adopt the parameter server (PS) architecture among the clients and central server. The on-device training and inference performance is tested with PyTorch 1.10, and then plugged into FedNLP framework. The models trained through prompt learning will be collected in the central server and aggregated through *FedAvg* [62] algorithm, which is also the default setting in prior FedNLP literature [104]. Both pseudo labeling and prompt learning randomly select clients for labeling and training per round.

Baselines We compare FeS to the following alternatives and the key differences are summarized in Table 3. (1) FedCLS is the vanilla federated fine-tuning method without optimizations [23, 72]. It trains only with the limited gold labels. (2) FedFSL implements pseudo labeling and prompt learning but

without our system optimizations. (3) FedFSL-BIAS runs the FedFSL pipeline; it however tunes only the layer bias while freezing the other weights. Both FedFSL and FedFSL-BIAS use static pacing, i.e., adding 100 pseudo labels per selected client per round. FeS and all baselines use the same set of hyper-parameters as prior literature [16, 73, 104]: mini-batch size as 4; local training epoch as 1; max sequence length as 256, per-round participant client number as 5.

Models We test two foundation NLP models: RoBERTa-large [59] (default) and its light version RoBERTa-base, composed of 24 and 12 transformer layers, respectively. They are downloaded directly from Huggingface [92]. We use RoBERTa-large for most of our experiments for its superior accuracy performance, as we discussed in §2.1. Generative tasks requiring mega large language models like GPT3 [15] are out of this work's scope. For most classification or seq2seq tasks, BERT-like models are adequate to achieve usable accuracy.

Dataset and few-shot setting We experiment with four popular NLP datasets and prompts[4], as shown in Table 2. (1) AGNEWS [110] is a news classification dataset. (2) MNLI [91] is a sentence understanding dataset. (3) YELP Review Full (YELP-F) [110] is a restaurant rating dataset. (4) YAHOO [110] is a question-answer pairing dataset. For each dataset, we follow prior work [16, 26] to randomly select gold labels. By default, the labels form a skewed distribution across clients to be more realistic to real-world situation as discussed in §2.1. For each dataset, we generate 3 different few-shot settings, on which we repeat the experiments and report the mean results.

Hardware As prior FL literature [48, 51, 53, 76, 104], our experiments are carried out in an emulation manner on a GPU server with 8x NVIDIA A40. The on-device training time is obtained on 2 development boards with similar hardware capacity to mainstream mobile devices, i.e., NVIDIA TX2 [1] and Raspberry Pi 4B [2] The default testbed device is NVIDIA TX2 without special statement. The numbers are then plugged into the emulation framework to calculate the elapsed time. We try various network bandwidths between clients and server, with default number as 1MB/s, a typical setting for mobile and IoT devices [4, 34].

4.2 End-to-end Performance

FeS significantly speeds up model convergence at high accuracy. As demonstrates in Figure 8a and Table 4, FeS achieves 86.8%-95.9% relative accuracy to a full-label fine-tuning on four datasets, using only 64 data labels. In comparison, FedCLS only converges at 27.9%–37.3% relative accuracy. The improved performance is attributed to the pseudo labeling and prompt learning techniques adopted by FeS. Accordingly, the other two baselines also achieve a competitive convergence accuracy. However, FeS is much faster to converge: FeS takes only 0.1–0.4 hours to reach the convergence accuracy of FedFSL on Jetson TX2, which is 8.2×–46.0× faster than FedFSL. Even when compared to the stronger baseline, FedFSL-BIAS, FeS still converges 4.3×–14.6× faster. On a wimpier device RPI 4B, FeS converges a few times slower due to the lengthened local training time. Nevertheless, it consistently outperforms FedFSL by 28.3× and FedFSL-BIAS by 10.8× on average.

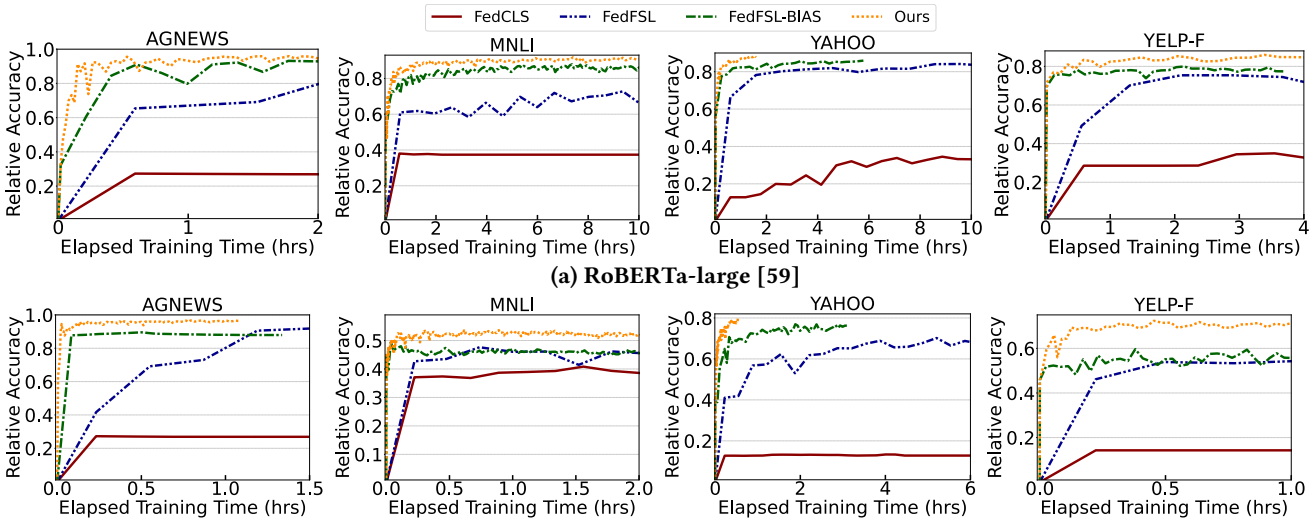
We also observe FeS achieves 3.4%–18.1% higher relative accuracy than FedFSL and FedFSL-BIAS, which are built atop the same algorithmic foundation. There are two potential reasons. First, by tuning fewer weights (bias), the training algorithm can better extract consolidated knowledge through fewer data labels. Second, FeS employs curriculum pacing that effectively orchestrates pseudo labeling with federated learning, whereas the two FedFSL baselines use a static pacing strategy that could lead to insufficient or excessive pseudo labels.

We then extend our experiments to RoBERTa-base and present the results in Figure 8b. FeS achieves 8.0%–66.9% higher relative accuracy than FedCLS. Compared to the more competitive baselines, i.e. FedFSL/FedFSL-BIAS, FeS maintains high convergence accuracy and is 24.1×/4.6× faster on average, respectively. In most cases, except for AGNEWS, which is a relatively easy task, using small models with weaker knowledge results in lower accuracy, as discussed in §2. For instance, RoBERTa-base sees a decrease of 38.5%, 8.2% and 13.9% in convergence accuracy on MNLI, YAHOO and YELP-F, respectively, when compared to RoBERTa-large. Fortunately, FeS significantly reduces the large model fine-tuning cost while maintaining its high performance.

FeS outperforms baselines in various network environments. Figure 9 reports the performance of FeS and baselines under various network environments from 0.1MB/s to 10MB/s, which cover the typical WiFi and cellular bandwidth. Our key observation is that FeS consistently outperforms other baselines under different network settings, with improvement more significant at lower bandwidths. For instance, with a bandwidth of 10MB/s, FeS is 19.0× and 33.6× faster than FedFSL on AGNEWS and MNLI, respectively. When the bandwidth drops to 0.1MB/s, the improvement reaches 224.3× and 661.7×, respectively. This is due to FeS significantly reducing network transmission by tuning very few parameters (~0.1% of all.)

Our results show that FeS also outperforms FedFSL-BIAS by 5.6×–7.4×. Those benefits arise from FeS's greater efficiency in both computation and communication, which is also verified by the end-to-end performance.

[4]We attempt 6, 2, 6, 4 different prompts that are widely used in prior work for each dataset, and use the one with highest accuracy. The verbalizers are the same as the previous literature [73].



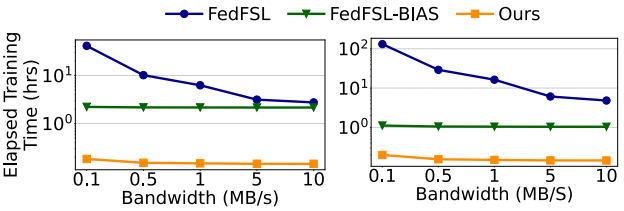
(a) RoBERTa-large [59]

(b) RoBERTa-base [59]

Figure 8: Overall Performance of FeS and baselines. Device: Jetson TX2.

Dataset	AGNEWS					MNLI					YAHOO					YELP-F				
Perf.	Conv. Acc.	Time-to-acc (hr)				Conv. Acc.	Time-to-acc (hr)				Conv. Acc.	Time-to-acc (hr)				Conv. Acc.	Time-to-acc (hr)			
		TX2		RPI			TX2		RPI			TX2		RPI			TX2		RPI	
		acc1	acc2	acc1	acc2		acc1	acc2	acc1	acc2		acc1	acc2	acc1	acc2		acc1	acc2	acc1	acc2
FedCSL	27.9%	X	X	X	X	37.3%	X	X	X	X	34.6%	X	X	X	X	35.7%	X	X	X	X
FedFSL	92.5%	3.3	3.3	50.0	50.0	74.1%	9.2	X	137.5	X	84.3%	8.3	X	125.0	X	75.3%	2.1	X	31.3	X
FedFSL-BIAS	92.5%	1.7	1.7	25.0	25.0	88.1%	0.5	11.7	7.5	175.0	85.9%	3.3	5.3	50.0	80.0	79.4%	0.2	2.1	2.5	10.4
Ours	95.9%	0.4	0.4	5.5	5.5	92.2%	0.2	0.8	2.5	12.5	88.5%	0.3	0.7	5.0	10.0	86.8%	0.1	0.5	1.3	7.5

Table 4: The final convergence accuracy ("Conv. Acc.") and the elapsed training time ("Time-to-acc") to reach different relative accuracy. NLP model: RoBERT-large. "acc1"/"acc2" are the final convergence accuracy of FedFSL/FedFSL-BIAS, respectively. "X" means the accuracy cannot be achieved.



(a) AGNEWS (b) MNLI

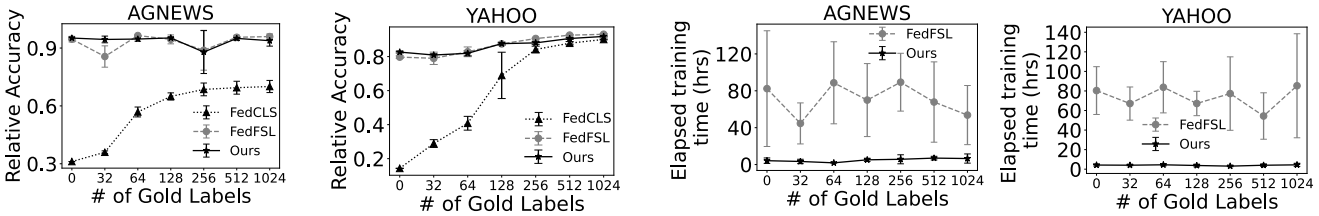
Figure 9: FeS outperforms baselines under all network bandwidths to reach the convergence accuracy of FedFSL.

4.3 Impacts of Initial Gold Labels

We vary the initial data labels and compare the performance of FeS to baselines on two datasets: AGNEWS and YAHOO. As shown in Figure 10a, FeS performs on par with or slightly higher than FedFSL in terms of relative accuracy from 0–1024 initial data labels, which is up to 64.1% higher than FedCLS. In some cases, FeS achieves satisfactory zero-shot performance, e.g., 95.2% relative accuracy on AGNEWS while FedCSL only reaches 31.1%. This observation paves the way for future research on zero-shot learning in mobile NLP. Furthermore, FeS also significantly reduces the end-to-end convergence time under various initial data labels. For a fair comparison, we only compare FedFSL and FeS that perform alike. As shown in Figure 10b, to reach the same accuracy, FeS reduces the elapsed training time by up to 18.3× and 17.1× on AGNEWS and YAHOO, respectively.

4.4 Significance of Key Designs

We perform an ablation study to understand the contribution of each key technique of FeS presented in §3. As shown in Figure 11, we find each of them significantly contributes to the results: (1) The co-planning of training depth and capacity reduces the convergence time by 8.0×–62.3× on different datasets. The significant improvement comes from that most of bottom layers (up to 66.7% on AGNEWS) are skipped, which reduces the training latency linearly. Some middle layers (up to 33.3%) are tuned with reduced capacity, and thus reduces the network traffic. (2) With the model optimized for training, we observe the pseudo labeling accounts for more than 70% of the total computation cost. The representative diversity mechanism filters out up to 95% of the data, further reducing the training time by 1.2×–3.5×. (3) Curriculum pacing further reduces the training time by 1.6×–3.5× by selecting a (sub-)optimal pacing configuration.



(a) Convergence accuracy (b) Training time

Figure 10: The training accuracy (a) and training time (b) with different number of gold labels.

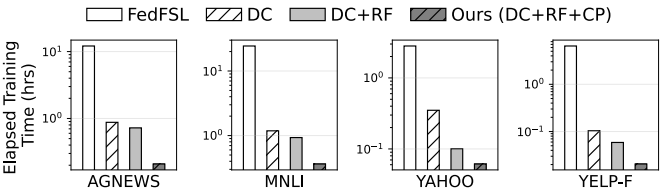


Figure 11: Model convergence delays with and without FeS's key designs, showing their significance. DC: training depth/capacity co-planning; RF: representative filtering; CP: curriculum pacing.

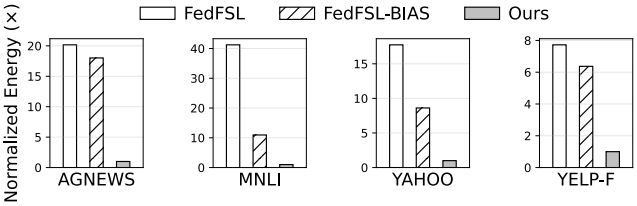


Figure 12: The total energy consumption of all clients, normalized to that of FedCLS.

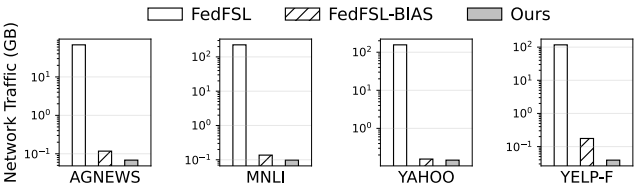


Figure 13: The total network traffic of all clients.

4.5 Client Resource Cost

Energy consumption Figure 12 illustrates the average energy consumed during mobile NLP training tasks on each device. It shows that FeS saves the energy consumption remarkably, e.g., 7.7×–41.2× reduction compared to FedFSL and 6.4×–18.0× reduction compared to FedFSL-BIAS. This improvement comes from the reduced network transmission time, the on-device training/labeling computations, and the cherry-picked orchestrating pace.

Network traffic Figure 13 reports the total network traffic incurred during fine-tuning to reach the convergence accuracy of FedFSL. It shows that FeS saves 1841.7× on average and up to 3000.0× (reducing from 224.6 GB to 0.04 GB) network traffic compared to FedFSL on four datasets. Please note that reducing the network traffic not only speeds up the convergence, but also mitigates the overhead on clients and the monetary cost to FL developers. The cost is billed by the amount of data transmitted on public cloud platforms such as AWS [5], which charges $0.01/GB.

Memory footprint As shown in Figure 14, our training depth and capacity co-planning mechanism can reduce the memory footprint by 4.3–4.5 times, which is crucial for practical deployment on mobile devices. For example, FedFSL requires 10.4 GB memory[5] to train RoBERTa-large, which is 2.4× higher than training RoBERTa-base. This excessive memory requirement would lead to out-of-memory and training failure on mobile devices which typically have only 8GB RAM. FedFSL-BIAS reduces the memory usage of training RoBERTa-large to 5.8 GB, which is still too large for mobile devices. Because it only bypasses the memory bottleneck of the weight update, but not the intermediate activations which is the main memory bottleneck [18]. In comparison, FeS only requires 2.3 GB memory due to the shallow training depth and greatly saved intermediate activations.

Remark Training can be done when no user interactions are present, e.g. when phone is idle/charged overnight which is nearly a "clean" environment without other co-running applications to share memory. Moreover, memory inefficiency can be compensated with acceptable training overhead through advanced memory optimizations such as batch splitting and model weight caching [87]. During the end-to-end convergence, which typically takes between 0.1 to 0.8 hours, each device typically engages in a few rounds of training, with each round lasting only a few tens of seconds. As a result, FeS shall not compromise user experience.

5 RELATED WORK

Few-shot learning (FSL) and FedFSL FSL has been one of the hottest topics in machine learning research, as it is considered more akin to how human intelligence works [30, 69, 77, 79, 89]. FeS identifies two complementary algorithmic blocks, i.e., pseudo labeling [8, 19, 50, 106] and prompt learning [57], and demonstrates satisfactory accuracy under federated context. Prior work [41] introduced iterative

[5]Tested on a central server.

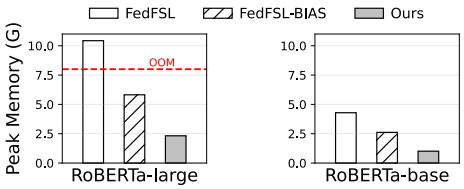


Figure 14: Memory footprint of on-device training.

pseudo labeling into prompt learning for centralized training of vision-language model, but it employed a fixed iterative pace that could result in poor federated performance, as we have demonstrated. Our harsh but practical assumptions of highly scare and skewed data labels invalidates existing FedFSL methods [24, 26, 27, 42, 103, 111], which assume at least 4% of the data to be uniformly labeled across clients. As far as we know, FeS achieves the state-of-the-art performance in the FedFSL scenario. At system aspect, FSL is generally considered lightweight due to fewer rounds of training [21, 26, 32, 109]. However, we show that the cost could be substantial in FL due to the use of the pseudo labeling (requiring on-device inference) and prompt learning (requiring large NLP models). We are the first to tackle these system challenges and enable practical FedFSL.

Large language model (LLMs) Few-shot prompt tuning on LLMs such as GPT-3 [15] can rival the performance of fully-supervised medium-size models like RoBERTa [59]. However, these models are too large (e.g., 175B+ parameters for GPT-3) to be deployed on devices after training, and deploying them on the cloud leads to privacy concerns [3, 81] and network delays [35, 97]. We pioneer practical federated prompt tuning, allowing resource-constrained devices to achieve comparable few-shot performance while preserving privacy and supporting offline inference.

FedNLP aims to achieve both high accuracy and privacy preservation in NLP model fine-tuning. Recently, there are a few literature investigating its implications, but mostly at the algorithm aspect. [104] builds a benchmark for popular FedNLP tasks and datasets in a standard FL workflow. [10] enhances the privacy of FedNLP by orchestrating with differential privacy. SEFL [86] eliminates the need for the trusted entities and is resilient to client dropouts in FedNLP tasks. Those work are orthogonal to FeS. [16] is the only work that we are aware of that tackles with huge system cost of FedNLP. It proposes a FedNLP framework based on lightweight, automatically configured adapters at runtime. However, the adapter cannot be applied in few-shot NLP scenarios according to our experiments.

FL system optimizations The huge resource cost of cross-device FL has been well recognized by the research community. In respond, lots of efforts have been invested, including communication efficiency optimizations [13, 98], model compression/quantization [11, 93], client/data sampling [48, 51–53, 63, 88, 96, 112], and on-device training speedup [87, 94]. Instead, FeS addresses unique challenges raised by the few-shot scenarios: pacing between pseudo labeling and training; filtering redundant unlabeled data for pseudo labeling. The design of FeS is mostly compatible with most optimizations above as its FL training is loosely coupled with the pseudo labeling.

Attacks in FL It is well known that FL cannot fully guarantee privacy preservation, e.g., extraction attacks [9, 22, 113]. However, dropout, a common training technique used in ML, is proven to be very effective to defend against those attacks [22]. Moreover, [113] demonstrates that most attacks suffer a significant decrease in success ratio when training batch sizes are set greater than 1. Apart from that, most data extraction attacks tend to be extremely resource-intensive [9, 22, 113]. Though larger models leak more information than the smaller ones, it incur larger inversion cost either (e.g., about 1672.52s for reconstructing one sentence [113]). FeS avoids revealing training data and raises the barrier for attackers (i.e. it requires much higher attack capability and much longer time). Furthermore, integrating various privacy-preserving techniques, such as differential privacy [10] and secure aggregation [14], can further enhance the security of FL. FeS is parameter-efficient and thereby shall be easy to integrate with them.

6 CONCLUSIONS

FeS is a FedFSL framework that enables practical few-shot NLP fine-tuning on federated mobile devices. At algorithm aspect, it incorporates pseudo labeling and prompt learning to achieve usable accuracy with only tens of data labels. At system aspect, it proposes three novel techniques, i.e., by pacing training and labeling, early filtering unlabeled data, and reducing the tuning depth/capacity, to address the unique challenge of huge resource cost raised by its algorithmic foundation. On extensive experiments, FeS shows superior system performance over existing approaches.

ACKNOWLEDGMENTS

This research was supported by National Key Research and Development Program of China #2020YFB1805500, NSFC #62032003, #61921003, #62102045, Beijing Nova Program #Z211100002121118, Young Elite Scientists Sponsorship Program by CAST #2021QNRC001, and CCF-Alibaba Innovative Research (AIR). Dongqi Cai was supported by BUPT Excellent Ph.D. Students Foundation #CX2023124. The authors thank the anonymous reviewers and the shepherd for their insightful feedbacks.

REFERENCES

[1] https://developer.nvidia.com/embedded/jetson-tx2.

[2] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[3] https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal.

[4] The state of wifi vs mobile network experience as 5g arrives. https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2018-11/state_of_wifi_vs_mobile_opensignal_201811.pdf, 2018.

[5] Amazon ec2 on-demand pricing. https://aws.amazon.com/ec2/pricing/on-demand/, 2022.

[6] Ahmed M Abdelmoniem and Marco Canini. Towards mitigating device heterogeneity in federated learning via adaptive model quantization. In Proceedings of the 1st Workshop on Machine Learning and Systems, pages 96–103, 2021.

[7] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. arXiv preprint arXiv:1511.06481, 2015.

[8] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In 2020 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2020.

[9] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. Lamp: Extracting text from gradients with language model priors. Advances in Neural Information Processing Systems, 35:7641–7654, 2022.

[10] Priyam Basu, Tiasa Singha Roy, Rakshit Naidu, Zumrut Muftuoglu, Sahib Singh, and Fatemehsadat Mireshghallah. Benchmarking differential privacy and federated learning for bert models. arXiv preprint arXiv:2106.13973, 2021.

[11] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In International Conference on Machine Learning, pages 560–569. PMLR, 2018.

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

[13] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. Proceedings of Machine Learning and Systems, 1:374–388, 2019.

[14] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1175–1191, 2017.

[15] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[16] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Autofednlp: An efficient fednlp framework. arXiv preprint arXiv:2205.10162, 2022.

[17] Dongqi Cai, Yaozong Wu, Haitao Yuan, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Towards practical few-shot federated nlp. In Proceedings of the 3rd Workshop on Machine Learning and Systems, pages 42–48, 2023.

[18] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce activations, not trainable parameters for efficient on-device learning. arXiv preprint arXiv:2007.11622, 2020.

[19] Paola Cascante-Bonilla, Fuwen Tan, Yanjun Qi, and Vicente Ordonez. Curriculum labeling: Revisiting pseudo-labeling for semi-supervised learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 6912–6920, 2021.

[20] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. arXiv preprint arXiv:1604.06174, 2016.

[21] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. arXiv preprint arXiv:1904.04232, 2019.

[22] Jieren Deng, Yijue Wang, Ji Li, Chenghong Wang, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. Tag: Gradient attack on transformer-based language models. In The 2021 Conference on Empirical Methods in Natural Language Processing, 2021.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[24] Enmao Diao, Jie Ding, and Vahid Tarokh. Semifl: Semi-supervised federated learning for unlabeled clients with alternate training. Advances in Neural Information Processing Systems, 35:17871–17884, 2022.

[25] Cynthia Dwork. Differential privacy: A survey of results. In International conference on theory and applications of models of computation, pages 1–19. Springer, 2008.

[26] Chenyou Fan and Jianwei Huang. Federated few-shot learning with adversarial learning. In 2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt), pages 1–8. IEEE, 2021.

[27] Tiantian Feng and Shrikanth Narayanan. Semi-fedser: Semi-supervised learning for speech emotion recognition on federated learning using multiview pseudo-labeling. arXiv preprint arXiv:2203.08810, 2022.

[28] Robert M French. Catastrophic forgetting in connectionist networks. Trends in cognitive sciences, 3(4):128–135, 1999.

[29] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2021, pages 3816–3830. Association for Computational Linguistics (ACL), 2021.

[30] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In 6th International Conference on Learning Representations, ICLR 2018, 2018.

[31] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. Ppt: Pre-trained prompt tuning for few-shot learning. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, pages 8410–8423, 2022.

[32] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. arXiv preprint arXiv:1902.11175, 2019.

[33] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4805–4814, 2019.

[34] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. Mp-dash: Adaptive video streaming over preference-aware multipath. In Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies, pages 129–143, 2016.

[35] Osama Haq, Mamoon Raja, and Fahad R Dogar. Measuring and improving the reliability of wide-area cloud paths. In Proceedings of the 26th International Conference on World Wide Web, pages 253–262, 2017.

[36] Douglas M Hawkins. The problem of overfitting. Journal of chemical information and computer sciences, 44(1):1–12, 2004.

[37] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In International Conference on Learning Representations, 2021.

[38] SU Hongjin, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. Selective annotation makes language models better few-shot learners. In The Eleventh International Conference on Learning Representations, 2022.

[39] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In International Conference on Machine Learning, pages 2790–2799. PMLR, 2019.

[40] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2021.

[41] Tony Huang, Jack Chu, and Fangyun Wei. Unsupervised prompt learning for vision-language models. arXiv preprint arXiv:2204.03649, 2022.

[42] Wonyong Jeong, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Federated semi-supervised learning with inter-client consistency & disjoint learning. In International Conference on Learning Representations, 2020.

[43] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. Foundations and Trends® in Machine Learning, 14(1–2):1–210, 2021.

[44] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In International conference on machine learning, pages 2525–2534. PMLR, 2018.

[45] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

[46] Seohyun Kim, Jinman Zhao, Yuchi Tian, and Satish Chandra. Code prediction by feeding trees to transformers. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pages 150–162. IEEE, 2021.

[47] Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. Fedscale: Benchmarking model and system performance of federated learning at scale. In International Conference on Machine Learning, pages 11814–11827. PMLR, 2022.

[48] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In 15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21), pages 19–35, 2021.

[49] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. International Conference on Learning Representations, 2020.

[50] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Workshop on challenges in representation learning, ICML, volume 3, page 896, 2013.

[51] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, pages 420–437, 2021.

[52] Anran Li, Lan Zhang, Juntao Tan, Yaxuan Qin, Junhao Wang, and Xiang-Yang Li. Sample-level data selection for federated learning. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications, pages 1–10. IEEE, 2021.

[53] Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. Pyramidfl: A fine-grained client selection framework for efficient federated learning. In Proceedings of the 28th Annual International Conference on Mobile Computing And Networking, pages 158–171, 2022.

[54] Guoliang Li, Yudian Zheng, Ju Fan, Jiannan Wang, and Reynold Cheng. Crowdsourced data management: Overview and challenges. In Proceedings of the 2017 ACM International Conference on Management of Data, pages 1711–1716, 2017.

[55] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In 2022 IEEE 38th International Conference on Data Engineering (ICDE), pages 965–978. IEEE, 2022.

[56] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 4582–4597, 2021.

[57] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 55(9):1–35, 2023.

[58] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. arXiv preprint arXiv:2110.07602, 2021.

[59] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[60] Robert Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. In Findings of the Association for Computational Linguistics: ACL 2022, pages 2824–2835, 2022.

[61] Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. Active learning by acquiring contrastive examples. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 650–663, 2021.

[62] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics, pages 1273–1282. PMLR, 2017.

[63] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In ICC 2019-2019 IEEE international conference on communications (ICC), pages 1–7. IEEE, 2019.

[64] CG Northcutt, Anish Athalye, and J Lin. Pervasive label errors in ml benchmark test sets, consequences, and benefits. In NeurIPS 2020 Workshop on Security and Data Curation Workshop, 2020.

[65] Stuart L Pardau. The california consumer privacy act: Towards a european-style privacy regime in the united states. J. Tech. L. & Pol'y, 23:68, 2018.

[66] Xuan Peng, Xuanhua Shi, Hulin Dai, Hai Jin, Weiliang Ma, Qian Xiong, Fan Yang, and Xuehai Qian. Capuchin: Tensor-based gpu memory management for deep learning. In Proceedings of the

Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, pages 891–905, 2020.

[67] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 46–54, 2020.

[68] Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. Xcopa: A multilingual dataset for causal commonsense reasoning. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2362–2376, 2020.

[69] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In International conference on learning representations, 2016.

[70] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992, 2019.

[71] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. International Journal of Computer Vision, 126(2):144–157, 2018.

[72] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.

[73] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 255–269, 2021.

[74] Timo Schick and Hinrich Schütze. True few-shot learning with prompts—a real-world perspective. Transactions of the Association for Computational Linguistics, 10:716–731, 2022.

[75] Taihua Shao, Yupu Guo, Honghui Chen, and Zepeng Hao. Transformer-based neural network for answer selection in question answering. IEEE Access, 7:26146–26156, 2019.

[76] Jaemin Shin, Yuanchun Li, Yunxin Liu, and Sung-Ju Lee. Fedbalancer: data and pace control for efficient federated learning on heterogeneous clients. In Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services, pages 436–449, 2022.

[77] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. Advances in neural information processing systems, 30, 2017.

[78] Weiqi Sun, Haidar Khan, Nicolas Guenon des Mesnards, Melanie Rubino, and Konstantine Arkoudas. Unfreeze with care: Space-efficient fine-tuning of semantic parsing models. In Proceedings of the ACM Web Conference 2022, pages 999–1007, 2022.

[79] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1199–1208, 2018.

[80] Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. Intellicode compose: Code generation using transformer. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pages 1433–1443, 2020.

[81] Latanya Sweeney. Simple demographics often identify people uniquely. Health (San Francisco), 671(2000):1–34, 2000.

[82] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. International Conference on Learning Representations, 2019.

[83] Betty Van Aken, Benjamin Winter, Alexander Löser, and Felix A Gers. How does bert answer questions? a layer-wise analysis of transformer representations. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pages 1823–1832, 2019.

[84] Bram van Berlo, Aaqib Saeed, and Tanir Ozcelebi. Towards federated unsupervised representation learning. In Proceedings of the third ACM international workshop on edge systems, analytics and networking, pages 31–36, 2020.

[85] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). A Practical Guide, 1st Ed., Cham: Springer International Publishing, 10(3152676):10–5555, 2017.

[86] Chenghong Wang, Jieren Deng, Xianrui Meng, Yijue Wang, Ji Li, Sheng Lin, Shuo Han, Fei Miao, Sanguthevar Rajasekaran, and Caiwen Ding. A secure and efficient federated learning framework for nlp. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7676–7682, 2021.

[87] Qipeng Wang, Mengwei Xu, Chao Jin, Xinran Dong, Jinliang Yuan, Xin Jin, Gang Huang, Yunxin Liu, and Xuanzhe Liu. Melon: Breaking the memory wall for resource-efficient on-device machine learning. In Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services, pages 450–463, 2022.

[88] Su Wang, Mengyuan Lee, Seyyedali Hosseinalipour, Roberto Morabito, Mung Chiang, and Christopher G Brinton. Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications, pages 1–10. IEEE, 2021.

[89] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. ACM computing surveys (csur), 53(3):1–34, 2020.

[90] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. Advances in Neural Information Processing Systems, 31, 2018.

[91] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Proceedings of NAACL-HLT, pages 1112–1122, 2018.

[92] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771, 2019.

[93] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized sgd and its applications to large-scale distributed optimization. In International Conference on Machine Learning, pages 5325–5333. PMLR, 2018.

[94] Daliang Xu, Mengwei Xu, Qipeng Wang, Shangguang Wang, Yun Ma, Kang Huang, Gang Huang, Xin Jin, and Xuanzhe Liu. Mandheling: mixed-precision on-device dnn training with dsp offloading. In Proceedings of the 28th Annual International Conference on Mobile Computing And Networking, pages 214–227, 2022.

[95] Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. Limubert: Unleashing the potential of unlabeled data for imu sensing applications. In Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, pages 220–233, 2021.

[96] Jie Xu and Heqiang Wang. Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective. IEEE Transactions on Wireless Communications, 20(2):1188–1200, 2020.

[97] Mengwei Xu, Zhe Fu, Xiao Ma, Li Zhang, Yanan Li, Feng Qian, Shangguang Wang, Ke Li, Jingyu Yang, and Xuanzhe Liu. From cloud to

edge: a first look at public edge platforms. In Proceedings of the 21st ACM Internet Measurement Conference, pages 37–53, 2021.

[98] Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu. Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data. In Proceedings of the Web Conference 2021, pages 935–946, 2021.

[99] Peijun Yang, Haibin Cai, and Zhiming Zheng. Improving the quality of crowdsourcing labels by combination of gold data and incentive. In 2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), pages 10–15. IEEE, 2018.

[100] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 13(3):1–207, 2019.

[101] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 72–77, 2019.

[102] Xue Ying. An overview of overfitting and its solutions. In Journal of physics: Conference series, volume 1168, page 022022. IOP Publishing, 2019.

[103] Hongzheng Yu, Zekai Chen, Xiao Zhang, Xu Chen, Fuzhen Zhuang, Hui Xiong, and Xiuzhen Cheng. Fedhar: Semi-supervised online learning for personalized federated human activity recognition. IEEE Transactions on Mobile Computing, 2021.

[104] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Christophe Dupuy, Rahul Gupta, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: Benchmarking federated learning methods for natural language processing tasks. Findings of NAACL, 2022.

[105] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. arXiv preprint arXiv:2106.10199, 2021.

[106] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. Advances in Neural Information Processing Systems, 34:18408–18419, 2021.

[107] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In 2020 USENIX annual technical conference (USENIX ATC 20), pages 493–506, 2020.

[108] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4):e1253, 2018.

[109] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. Revisiting few-sample bert fine-tuning. International Conference on Learning Representations, 2021.

[110] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. Advances in neural information processing systems, 28, 2015.

[111] Yuchen Zhao, Hanyang Liu, Honglin Li, Payam Barnaghi, and Hamed Haddadi. Semi-supervised federated learning for activity recognition. arXiv preprint arXiv:2011.00851, 2020.

[112] Yuxi Zhao and Xiaowen Gong. Quality-aware distributed computation and user selection for cost-effective federated learning. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 1–6. IEEE, 2021.

[113] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. Advances in neural information processing systems, 32, 2019.

[114] Qile Zhu, Xiaolin Li, Ana Conesa, and Cécile Pereira. Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text. Bioinformatics, 34(9):1547–1554, 2018.