Federated Fine-tuning of Billion-Sized Language Models across Mobile Devices

Mengwei Xu
Beijing University of Posts and Telecommunications

Yaozong Wu
Beijing University of Posts and Telecommunications

Dongqi Cai
Beijing University of Posts and Telecommunications

Xiang Li
Beijing University of Posts and Telecommunications

Shangguang Wang
Beijing University of Posts and Telecommunications

ABSTRACT

Large Language Models (LLMs) are transforming the landscape of mobile intelligence. Federated Learning (FL), a method to preserve user data privacy, is often employed in fine-tuning LLMs to downstream mobile tasks, an approach known as FedLLM. Though recent efforts have addressed the network issue induced by the vast model size, they have not practically mitigated vital challenges concerning integration with mobile devices, such as significant memory consumption and sluggish model convergence.

In response to these challenges, this work introduces `FwdLLM`, an innovative FL protocol designed to enhance the FedLLM efficiency. The key idea of `FwdLLM` is to employ backpropagation (BP)-free training methods, requiring devices only to execute "perturbed inferences". Consequently, `FwdLLM` delivers way better memory efficiency and time efficiency (expedited by mobile NPUs and an expanded array of participant devices). `FwdLLM` centers around three key designs: (1) it combines BP-free training with parameter-efficient training methods, an essential way to scale the approach to the LLM era; (2) it systematically and adaptively allocates computational loads across devices, striking a careful balance between convergence speed and accuracy; (3) it discriminatively samples perturbed predictions that are more valuable to model convergence. Comprehensive experiments with five LLMs and three NLP tasks illustrate `FwdLLM`'s significant advantages over conventional methods, including up to three orders of magnitude faster convergence and a 14.6× reduction in memory footprint. Uniquely, `FwdLLM` paves the way for federated learning of billion-parameter LLMs such as LLaMA on COTS mobile devices – a feat previously unattained.

1 INTRODUCTION

Large Language Models (LLMs)[1] such as GPTs and LLaMA have showcased an impressive ability to handle generic machine learning tasks [85]. As foundational models, pretrained LLMs can be fine-tuned for various downstream tasks and have been applied across a broad range of mobile applications, including but not limited to question answering, personal assistance, and image retrieval [23, 42, 82, 96, 100]. Early efforts have been invested to adapt LLMs to mobile devices while maintaining data privacy during the fine-tuning process. Often, these efforts employ federated learning, an approach known as FedLLM [18, 20, 21, 84, 90, 107, 109].

A salient feature of LLMs is their scalability: by incorporating more parameters, LLMs can continually evolve, achieving higher accuracy or even emergent abilities [33, 36, 79, 103]. Consequently, contemporary LLMs have grown enormously in size and are hard to be trained even on a GPU cluster [85], not to mention mobile devices. Recent research of FedLLM [20, 84, 107, 109] primarily address the network issue between devices and cloud aggregator, yet the convergence is still lengthy and being impractical for developers. Through pilot experiments (§2.2), we identify three key obstacles towards practical FedLLM.

• **Huge memory footprint.** The predominant on-device training algorithm [22, 91] necessitates extensive memory to store intermediate results such as activations and gradients. Although fine-tuning could omit most gradients with layers frozen, activations continue to demand considerable memory, often exceeding device capabilities. For example, 3.9 GB is required for RoBERTa-large. It results in extra I/O time to swap in/out data [49, 72] and makes the training task a highly likely victim of mobile OS's low memory killer [9]; in either way, the FedLLM convergence is significantly slowed down.

• **Incompatible with mobile accelerators.** Mobile SoCs are often furnished with powerful, fast-evolving DNN accelerators (NPUs), e.g., Google Edge TPU and Qualcomm Hexagon that are up to 30× faster than CPUs. Regrettably, on-device training is unsupported on nearly all mobile NPUs, since they are tailored for inference rather than training, and thus lack the requisite support for training-specific operations like SELECT_OPS [1] and dynamic gradient updating.

• **Limited device scalability.** In FL, only dozens of devices participate in training simtaneously, even when millions of IoT/smartphone devices are available . For instance, Google's deployed FL system samples merely around 1% of

[1]In this work, we mainly refer LLMs to transformer-based NLP models that exceed 100M parameters.

training-ready devices per round [8], because even a small number of devices can saturate learning performance, meaning additional devices do not further expedite convergence.

This work leverages a crucial observation: all above issues can be somehow traced to the use of backpropagation-based (termed **BP**) training algorithm [48, 94] on devices (details in §2.2). This prompts an essential question: *is it feasible to replace backpropagation with a more mobile-friendly training algorithm, thereby reinvigorating the FedLLM protocol?*

`FwdLLM`: training with "perturbed inferences". We thereby present `FwdLLM`, the first-of-its-kind system that enables practical and scalable FedLLM through BP-free training algorithm. Instead of calculating one exact gradient using BP, `FwdLLM` asks each device to perform perturbed inference: applying a few self-generated small perturbations to the model weights, and compare how their prediction output deviate from the ground truth labels with the unmodified model. Intuitively, if a perturbation makes the model more accurate (output closer to labels), the perturbation is likely to direct the model to global optima. In §3.2, we detail how such intuition leads to a mathmatical form to obtain a BP-free gradient that is an unbiased estimator of the true gradient. Such BP-free training algorithms [13, 31, 59, 75, 80] have been researched by the ML community for decades but seize very few attentions. Relying on only inferences, the computation on devices is much more memory efficient and NPU-compatible; more devices also scale to faster convergence to allow more perturbed inferences simltaneously.

While the idea is intriguing, `FwdLLM`'s design confronts three crucial challenges. (i) BP-free methods have only shown comparable performance with BP on tiny models like LeNet, as they demand proportionally increased perturbed inferences with its model size [13, 70, 75]. (ii) How many perturbed inferences are good enough before proceeding to the next round? It is a vital factor with strong impact on the training convergence. Using a fixed number perturbed inferences sees up to 2.99× longer training delay as compared to the final design of `FwdLLM` (§3.3). There is no silver-bullet setting that results in the fastest convergence under each condition. Rather, the optimal setting depends on the specific tasks and models; it also needs to be adapted on the fly even within the same training session, as the favorable setting drifts over time depending on the model's learning progress. (iii) The convergence speed of `FwdLLM` hinges on the perturbations generated. Contrary to most prior BP-free literature, which randomly samples perturbations from classic distributions [13, 30, 67], we discover that this method is often sub-optimal. Our empirical results in §3.4 shows that most randomly sampled perturbations are of low value (orthogonal to the true gradient) to the convergence.

`FwdLLM` addresses above challenges with three key designs.

First, `FwdLLM` integrates the perturbed inferences with parameter-efficient fine-tuning (PEFT) methods like LoRa [40] and Adapter [74]. It is based on a crucial observation that the training complexity of BP-free methods scales with its *trainable parameters* instead of total parameters, aligning well with PEFTs that necessitate minimal parameters for fine-tuning. In fact, the larger the LLM, the fewer PEFT trainable parameters required [34, 40, 58, 74]. While BP-free training and PEFT are priorly known and have been recently applied to FL in a few literature [20, 30, 70, 109], we are the first to identify their significance in FedLLM acceleration and investigate the system implications when they are orchestrated.

Second, `FwdLLM` employs an automatic and systematic strategy to manage the global perturbation inferences for developers. Unlike traditional FL protocols [8, 54, 68] that use a static, user-defined metric to control the computing loads on devices, `FwdLLM` augments the aggregator with an on-the-fly monitor that controls the timing to aggregate the gradients and proceed to the next round. Intuitively, as model approaches convergence, BP-free methods need more perturbations to accurately estimate the convergence direction. `FwdLLM` leverages a crucial heuristic that the variance across the BP-free gradients uploaded from different devices monotonically increases as model converges, which harmoniously paces with the number of perturbed inferences demanded for fast and stable convergence. Therefore, `FwdLLM` proposes a variance-controlled pacing mechanism that the perturbed inferences stops only when the variance observed on aggregator is smaller than a threshold. `FwdLLM` also judiciously prioritizes different means (by adjusting participant devices, training data size, and perturbation number) to adapt `global-PS` to maximize the efficiency.

Third, `FwdLLM` introduces a discriminative perturbation sampling method that generates perturbations more likely to contribute significantly to convergence. Concretely, `FwdLLM` asks devices to bypass the computing of low-value perturbations, i.e., those with nearly orthogonal convergence direction with the true gradients. To estimate the true gradients that are not known before aggregated on clouds, `FwdLLM` leverages the opportunity that the gradients direction changes smoothly during FL – an observation also exploited in prior FL literature [56, 92]. Thereby, the server always pre-computes the cosine similarities between the perturbations generated and the computed BP-free gradients of the previous round. The perturbations with small similarity will be filtered out and not computed by the devices.

We have implemented `FwdLLM` and evaluated it on 6 typical transformer-based models: ALBERT-base (0.01B), DistilBERT-base (0.07B), BERT-base (0.1B), RoBERTa-large (0.3B), and LLaMA (7B) and 3 classic NLP tasks. The on-device training is profiled on Google Pixel 7 Pro and Jetson TX2. The results demonstrate FwdLLM's impressive performance: compared to full-model fine-tuning, FwdLLM reduces the training time from 10.9–97.9 hours to 0.2–0.8 hours (up to 217.3× reduction); compared to more competitive baselines enhanced by different PEFT methods, FwdLLM still delivers 2.0×–93.4× speedup (10.6× on average). FwdLLM also reduces the memory footprint by up to 14.6× and 11.5× compared to them, respectively. Through orchestration with quantization (INT4), for the first time FwdLLM enables fine-tuning a billion-parameter model like LLaMA over COTS smartphones within only 10 minutes. The ablation study also shows the significance of FwdLLM's key designs in planning and manipulating perturbations.

Contributions In this work, we introduce a novel framework that facilitates practical and efficient federated LLM fine-tuning. This is achieved by using BP-free training algorithm. The framework, denoted as FwdLLM, innovatively integrates two new techniques that adaptively schedule the number of perturbations to be examined and selectively produce them to enhance their utility. Extensive experiments demonstrate that FwdLLM yields substantial improvements over existing baselines. Moreover, FwdLLM serves to unify the pathway of on-device inference and training. Rather than viewing these as two separate research domains that necessitate distinct methods and optimizations, FwdLLM enables researchers in mobile AI systems and hardware to concentrate on optimizing on-device inference. This focus, in turn, leads to more efficient federated learning processes.

2 BACKGROUND AND MOTIVATIONS

2.1 Federated Fine-tuning of LLMs

Large language model has been an revolutionary technique for its superior performance in serving generic, complex, and few-shot ML tasks. Training LLMs typically includes two crucial steps: (i) *pre-training* that endows the models with generic, rich knowledge of images/languages/etc, which requires large amounts of public training datasets and computing devices; (ii) *fine-tuning* that adapts the pre-trained models for various downstream tasks, which relies on domain-specific, privacy-sensitive data. Towards a privacy-friendly LLM training pipeline, there is a trend to combine LLM fine-tuning with federated learning, e.g., FedLLM [20, 24, 107].

We are at very early stage towards practical FedLLM. The gap between the tight resource constraint of edge devices and the extensive resource demand of on-device LLM training is huge, as demonstrated in both prior studies [17, 22, 91, 98, 101] and the following experiments. Recent attempts incorporate parameter-efficient fine-tuning techniques (LoRA [40], adapters [74], and prompt tuning [58]) into FedLLM and see significant improvements in saving the network traffic between devices and aggregator. Yet they do not fully address many other issues such as excessive memory footprint.

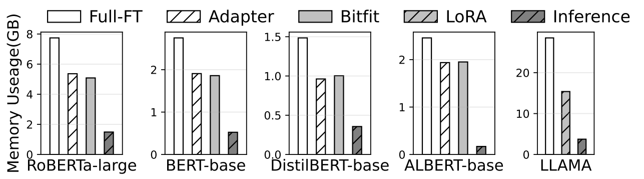


Figure 1: Peak memory footprint of different training methods and inference. Batch size: 8.

Algorithms	Trainable Parameters	Memory Footprint (GB)			
		Weights	Activations	Gradients	Total
FT-full	354.3M (100%)	1.3	5.1	1.3	7.7
FT-adapter	3.2M (9.0%)	1.3	3.9	0.02	5.2
FT-bitfit	0.3M (0.8%)	1.3	3.8	0.009	5.1
FT-lora	0.8M (2.2%)	1.3	3.8	0.01	5.1
Inference	/	1.3	0.2	0	1.5

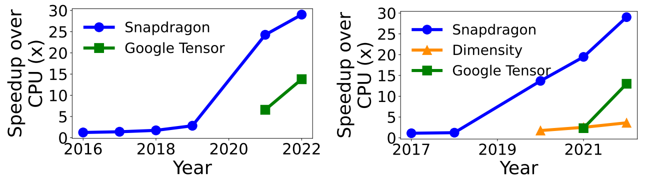
Table 1: The breakdown of memory footprint. Model: RoBERTa-large; batch size: 8. "FT": finetuning. "Activations" contain the activations for backward-gradient computation and optimizer states.

2.2 Preliminary Experiments of FedLLM

In this subsection, we reveal three crucial issues faced by FedLLM through pilot experiments.

• **FedLLM is hindered by the memory wall.** Figure 1 shows the peak memory usage in training various LLMs with a relatively small batch size (8). The observed memory expense is often unaffordable for edge devices, e.g., more than 7.7GBs for RoBERTa-large and 2.5GBs for ALBERT-base, while the typical mobile devices possess only 4GBs–12GBs DRAM. More severely, on smartphones, even only a small portion of that memory could be used for training tasks to not compromise user experience [17, 47, 53]. In contrast, inference consumes much less memory (e.g., less than 1GB) as it does not need to hold the intermediate computing results in memory as backpropagation does, which linearly scales up with batch size and sequence length.

PEFTs like adapter and bitfit methods cannot fundamentally reduce memory footprint as illustrated in Figure 1. They bring only 21.2%-35.2% memory savings across different models, which is inadequate to fit certain large models like ROBERTa-large or LLaMA into real mobile devices. We then break down the memory consumption of ROBERTa-large and summarize the results in Table 1. It explains why reducing trainable parameters cannot bring as significant memory saving: the activations generated during forward pass take

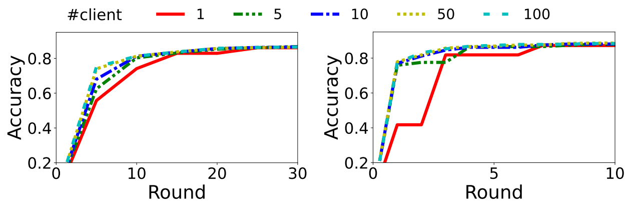


(a) ALBERT **(b) MobileBERT**

Figure 2: Performance evolvement of mobile NPU. Numbers are from AI Benchmark [2].

Model	Training Time (sec)			Inference Time (sec)		
	CPU	GPU	NPU	CPU	GPU	NPU
ALBERT	17.5	N/A	N/A	6.7	1.5	0.3
DistilBERT-base	6.9	N/A	N/A	3.4	0.8	0.2
BERT-base	14.0	N/A	N/A	6.9	0.8	0.3
RoBERTa-large	*28.1	N/A	N/A	11.7	2.9	0.8
LLaMA-7B (INT4)	N/A	N/A	N/A	22.1	N/A	N/A

Table 2: Per-batch (BS=8) training and inference time on Google Pixel 7 Pro. Library: llama.cpp for LLaMA and TFLite for others. "N/A": not supported. * is emulated in a infinite memory environment.



(a) Clients (w/ adapter) **(b) Clients (w/o adapter)**

Figure 3: Backpropagation-based FL has low device scalability.

Literature	Venue	Year	Total Devices	Devices per Round
Hermes [50]	MobiCom	2021	2,414	20 (0.8%)
PyramidFL [52]	MobiCom	2022	342,477	50 (0.01%)
FedAdapter [20]	MobiCom	2023	1,000	15 (0.15%)
FedBalancer [78]	MobiSys	2022	915	100 (10.9%)
Oort [45]	OSDI	2021	1,600,000	100 (0.006%)
FedNLP [104]	NAACL	2022	100	10 (10%)
C2A [43]	ACL	2023	100	25 (25%)
GradMA [65]	CVPR	2023	100	50 (50%)
FedScale [44]	ICML	2022	1,660,820	100 (0.006%)
FjORD [37]	NeurIPS	2021	3,400	10 (0.3%)

Table 3: Prior FL literature (mobile/system/AI) use a small ratio of devices in experiments. Maximal numbers are selected if many datasets are used.

up most of the memory usage, which cannot be eliminated even if the weights are not to be updated.

• **FedLLM's inability to leverage powerful mobile accelerators.** Modern mobile devices frequently come equipped with high-end NN accelerators. As Moore's Law approaches its limits, ASIC-based accelerators offer a promising pathway to sustain the growth in device capability in tandem with increasing model complexity. Figure 2 summarizes the speedup achieved by NPUs over CPUs for three popular mobile chip series: Qualcomm Snapdragon (Hexagon), Google Pixels (Tensor TPU), and MediaTek (Dimensity). The findings are compelling: the NPU speedup rate is steadily rising and reaches nearly 30× on the recent Snapdragon 8+ Gen 1 chip. This underscores the imperative to utilize NPUs for efficient on-device DNN execution.

Nevertheless, on-device training can hardly benefit from mobile NPUs. In Table 2, our measurements reveal the degree to which NPUs can accelerate DNN inference/training on the Google Pixel 7 Pro. While mobile NPUs significantly reduce inference latency compared to mobile CPUs and GPUs, they offer no support for DNN training. The reason behind this limitation is clear: these NPUs are tailored for inference tasks and thus lack the requisite support for backpropagation-specific operators such as `BroadcastGradient`, `ReluGrad`, `StridedSliceGrad`, and others [41]. Recent work [98] has enabled DNN training on Snapdragon Hexagon DSP, but this approach (i) compromises model accuracy due to lower data precision, and (ii) faces scalability challenges with other more proprietary NPUs, such as those found in Google Pixels and Huawei smartphones.

• **FedLLM has low device scalability.** Backpropagation-based FL struggles to enhance its convergence speed with more participant devices. As depicted in Figure 3, involving merely tens of devices per round leads to the saturation of convergence speed, regardless of the utilization of PEFT techniques. Escalating the device count to 100 results in only marginal improvements; for example, it yields just a 1.04× acceleration to reach an accuracy of 86% (with adapters).

In real-world scenarios, there could be easily more than millions devices, such as smartphones and IoTs, that are capable of contributing local data and computing resources. After filtering out devices in non-optimal conditions, such as those with low battery or high utilization, the remaining training-available device number still easily exceeds hundreds. For instance, in Google's deployed FL system [8], around 10,000 devices are simultaneously available for local training. The failure to scale to a larger number of idle devices seriously constrains the rate at which the model can converge. As indicated in Table 3, even when millions of devices are available, the existing literature on FL uniformly adopts the default experimental setting of using no more than 100 devices. The root cause of this scalability issue can be attributed to backpropagation-based optimizers [33, 36, 103] and can hardly be addressed at systems aspect.

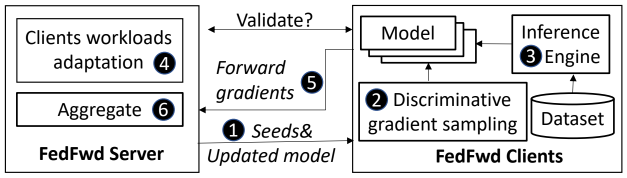


Figure 4: FwdLLM workflow.

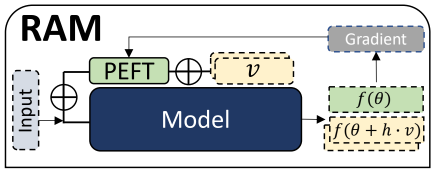


Figure 5: FwdLLM is memory-efficient. Dotted block will be released sequentially after computation.

3 FWDLLM DESIGN

3.1 Overview

FwdLLM is a cloud-device framework that aims to enable practical federated fine-tuning of LLMs across mobile devices. The key idea is to abandon backpropagation-based gradient descent, but uses "perturbed inference" that poses much less memory/compute pressure on devices and can be accelerated by ubiquitous mobile NN accelerators.

Simplified workflow As shown in Figure 4, FwdLLM employs a similar parameter-server architecture as traditional FedAvg protocol but mainly differs on the local computation. Per global round, the aggregator first sends the latest LLM updates (only for the trainable weights, denoted as $\mathcal{M}$) and random seeds to each available client ❶. Each client discriminatively samples N trainable weights perturbations based on the random seeds, and applies the perturbations to $\mathcal{M}$ to generate N perturbed LLMs ❷ (denoted as $\mathcal{M}_{i=1..N}$) (§3.4). A perturbation is essentially a vector with the same size as trainable parameter number that is sampled from a uniform distribution. The client then performs a forward pass on each $\mathcal{M}_i$ as well as $\mathcal{M}$ with local training data ❸, from which it gets a forward gradient by comparing their output difference (§3.2). Forward gradients are validated by the server (§3.3) to meet the variance-controlled pace ❹. Finally, the clients upload the validated forward gradients ❺ to the aggregator, where the gradients from different clients are aggregated and applied to $\mathcal{M}$ ❻. The above steps repeat till convergence.

FwdLLM's advantages by design. (1) FwdLLM is computationally efficient. For each client, calculating a forward gradient is equivalent to executing inferences twice (i.e., $f(\theta + h \cdot v)$ and $f(\theta)$), making its execution on NPU 19.5–27.5× faster than computing one backward gradient on a CPU, as previously demonstrated in Table 2. (2) FwdLLM is memory-efficient, as it does not require storage of the intermediate activations generated during the forward pass, which contribute to the majority of the memory footprint, as outlined in §2.2. The perturbation weights are parameter-efficient (details in §3.2), and the perturbed LLM can be generated sequentially, with each being immediately released once its inference is completed (Figure 5). Thus, the peak memory footprint of FwdLLM can be approximated as $size\_of(\mathcal{M}) + 2 \cdot trainable\_param$. (3) FwdLLM is highly scalable with respect to client numbers. As more clients compute forward gradients simultaneously, the aggregated gradient becomes closer to the real gradient, contributing to faster and more accurate convergence. Figure 6b illustrates that the continuous addition of perturbations enables the forward gradients to estimate the true gradient with greater precision and stability.

Implications on privacy From the cloud perspective, clients iteratively upload forward gradients, which are unbiased estimations of the true gradients, mirroring traditional FL methods based on backpropagation. Consequently, FwdLLM can be seamlessly integrated with common FL privacy enhancements, including differential privacy [12, 29], secure aggregation [30], and homomorphic encryption [106].

Unique challenges introduced by BP-free training First, BP-free training is not a panacea for all models. Existing studies [13, 70, 75] primarily validate its utility for diminutive models (a few million bytes), which are 1–3 orders of magnitude smaller than standard LLMs that we target. (2) The quantity of perturbations required to calculate a single forward gradient plays a pivotal role in determining convergence performance (elaborated in §3.3). However, this is not trivial to ascertain beforehand and has not been touched in prior literature [13, 30, 70, 75]. (3) The convergence speed of BP-free training is hampered by random perturbation generation [13, 75]. In following subsections, we will present three novel techniques to tackle those challenges, respectively.

3.2 Parameter-Efficient BP-Free FedLLM

Forward gradient The forward gradient method is selected as our BP-free algorithm because it is based solely on the directional derivative, which can be computed both precisely and efficiently via the forward pass [13]. Formally, to compute the directional derivatives of deep learning functions, denoted as f, with respect to a vector v at a point θ, the following equation can be used [48, 76]:

$$\nabla_v f(\theta) = \lim_{h \to 0} \frac{f(\theta + h \cdot v) - f(\theta)}{h}, \quad (1)$$

where $v \in N(0, 1)$ represents the weight perturbations, and $\nabla_v f(\theta)$ is the directional derivative of f at the model weight point θ in the direction v [25]. In simpler terms, $\nabla_v f(\theta)$ signifies the slope of f in the direction of v. The direction of steepest ascent can be identified using the gradient $\nabla f(\theta)$. Nevertheless, determining the gradient $\nabla f(\theta)$ can be computationally demanding, as it necessitates both a forward pass and a backward pass [13], which we have empirically validated in Section 2. As an alternative, we leverage the forward gradient g_v, which is more cost-effective to compute:

$$g_v(\theta) := \nabla_v f(\theta) v = (\nabla f(\theta) \cdot v) v, \tag{2}$$

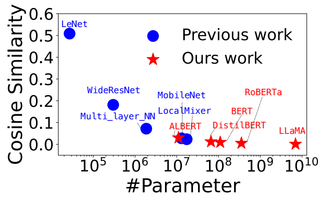
where $g_v(\theta)$ is established as an unbiased estimator of the gradient $\nabla f(\theta)$ [13].

Using Figure 9a as an illustrative example, we demonstrate the computation of gradients for the function $z = 2(x^2 + y^2)$ at the point $(0, 0.5)$. We can sample n direction vectors v from $N(0, 1)$ and compute $g_v(\theta)$ for each v. By taking the average of these forward gradients $g_v(\theta)$, we obtain an estimator for the true (backward) gradient $\nabla f(\theta)$.

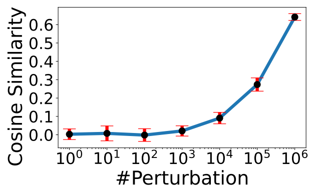
Forward gradient is not a panacea for all models. Although forward gradients have been proposed in previous literature [11, 13, 35, 66, 70, 81], their practical application has been limited, primarily due to the enormous demands on data and computation. Specifically, the requirements (i.e., perturbations per batch) grow exponentially with the parameter size. As shown in Figure 6, with the increase of parameter size, the generated forward gradients deviate significantly from the true gradients. Thus, to obtain reliable forward gradients, the required perturbations per batch increase exponentially with the parameter size. Prior forward gradient research has largely been restricted to evaluations on small-scale models such as LeNet and WideResNet [13, 70, 75].

Parameter-efficient BP-free learning To deal with this issue, FwdLLM exploits a key observation that the BP-free training complexity is primarily related to the *trainable* parameter size, rather than the total size. This observation is intuitively consistent with the mathematical foundation expressed as $f(\theta|\Theta, x)$, where the pre-trained model weight Θ and input x remain fixed, with only the PEFT weights θ being tunable. Fortunately, pre-trained LLMs have accumulated rich generic knowledge, thus requiring only a small number of new parameters to adapt to various downstream tasks. Therefore, for the first time, FwdLLM integrates BP-free training with PEFT methods.

In general, FwdLLM is compatible with various PEFT methods [34, 40, 55, 61, 64, 73, 83, 105] as demonstrated in our extensive offline experiments. We introduce an offline PEFT profiler designed to automatically identify the most suitable PEFT method for FwdLLM, using public dataset on clouds. A critical factor influencing FwdLLM performance is the number of trainable parameters. Consequently, we develope a similarity-aware profile, aiming to identify the optimal PEFT method that maximizes parameter savings while minimizing performance degradation. Specifically, we train the model



(a) Effect of model size.



(b) Effect of perturbation.

Figure 6: Effectiveness of forward gradients. (a) Increased model parameters make the generated forward gradient unreliable. (b) Adding perturbations could make the forward gradient computed more similar to gold gradients.

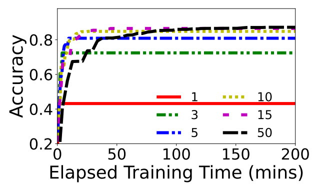


Figure 7: Optimal Global-PS varies across training.

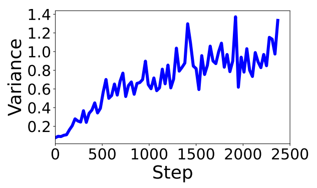


Figure 8: Gradient Variance throught training.

using the original parameters and various PEFT methods for a single iteration. Subsequently, we compute the similarity between the forward gradients and the BP gradients. This similarity is then utilized to gauge the efficacy of each PEFT method. In general, larger LLMs are conducive to more aggressive PEFT methods, like BitFit [105] and LoRa [40], which yield fewer trainable parameters.

3.3 Var.-controlled Perturbation Pacing

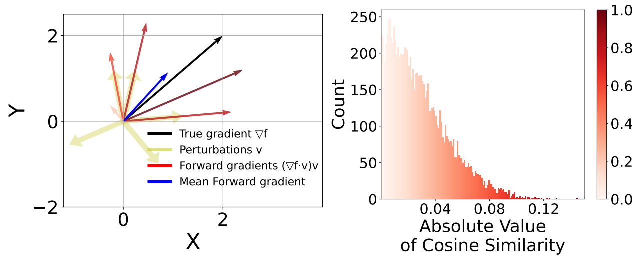
Key trade-off between accuracy and cost: In the design of FwdLLM, we identify a crucial trade-off between the rate of model convergence and the computational cost imposed on devices. Specifically, evaluating more perturbations leads to a more accurate forward gradient but also increases the inference cost. We introduce a new metric, global perturbation size (global-PS), defined as the total perturbations × local data size, aggregated across all clients per iteration. The success of FwdLLM greatly depends on choosing an appropriate global-PS, a topic that has not been previously explored in studies on forward gradient [13, 30, 75].

Adaptation of the global-PS on the fly: There is no universal global-PS setting that can optimize both accuracy and cost across varying scenarios. As illustrated in Figure 7, the optimal global-PS configuration changes throughout the training process, with FwdLLM favoring a monotonically increasing global-PS. For example, in the early training stages of DistilBERT on AGNEWS, the best `global-PS` is only 3 perturbations per client to reach 80% relative accuracy to convergence, while achieving 99% accuracy requires a global-PS of 50, leading to a 16.7× higher computation cost.

Systematic pacing strategy based on gradient variance: Requiring developers to manually control `global-PS` could be complex, as configurations vary across different models and datasets. Instead, `FwdLLM` offers an automatic and systematic strategy to manage the `global-PS` parameter, based on the observation that the numerical variance across the forward gradients uploaded by clients increases as the model approaches convergence. The variance is defined as $D(g) = \|\frac{1}{2}\left[(\bar{g_1}-\bar{g})^2 + (\bar{g_2}-\bar{g})^2\right]\|$, where $\bar{g_1}$ and $\bar{g_2}$ denote the means of the first and second halves of the forward gradients, respectively, and $\bar{g}$ denotes the mean of all the forward gradients. After 1500 training steps, we observed that this variance escalated from 0.078 to 1.182, marking a 15.2× increase. Such increased variance may necessitate more perturbations to accurately estimate the real gradient. `FwdLLM` simply uses a predefined variance threshold that must be met before aggregating gradients across devices, with the threshold being the only hyperparameter to tune. An empirically selected range of 0.5–1.5 has been observed to perform well across various models and datasets.

Prioritizing methods to adapt `global-PS`: The `global-PS` can be increased through three methods: (1) involving more devices; (2) having each device test more perturbations; (3) having each device test each perturbation on more local data. `FwdLLM` first prioritizes adding more devices, as concurrent computation facilitates fast convergence. Once device availability reaches its maximum, `FwdLLM` turns to the second method, asking clients to test more perturbations, for two reasons: (i) perturbations can be quickly and infinitely generated, and (ii) the result of the original LLM ($f(x)$) can be reused when calculating the directional derivatives on multiple perturbations ($f(x+v)$ and $f(x)$), thereby reducing the required forward propagations from $2*N$ to $N+1$.

Validating-computing pipeline: `FwdLLM` meticulously designs a pacing pipeline to validate incoming gradients. The primary principle is to ensure uninterrupted local forward gradient computations. After transmitting the forward gradients to the cloud aggregator, clients proceed to compute the forward gradients for the succeeding perturbation. Upon accumulating sufficient forward gradients to surpass the variance threshold, the aggregator instructs the client to halt forward gradient computations. It then aggregates the received forward gradients and dispatches the updated model to the client. This streamlined process guarantees timely reception of up-to-date pacing information by the server, eliminating futile waiting times for client feedback. Note that the validation step on cloud aggregator is lightweight, e.g., less than 10ms in our experiment setup.



(a) Samples **(b) Statistics**

Figure 9: Most of the gradients are nearly orthogonal to target gradients thus contributing little.

3.4 Discriminative Perturbation Sampling

The convergence speed of `FwdLLM` primarily hinges on the perturbations generated before training. Contrary to most prior forward-gradient literature, which randomly samples perturbations from classic distributions such as Gaussian functions [13, 30, 67], we discover that this method is often sub-optimal. `FwdLLM` instead employs a discriminative approach, selectively sampling perturbations that are *more likely to contribute larger gradients to the model's convergence.*

Minimal contribution from most forward gradients: Specifically, two randomly generated vectors in high dimensions could be orthogonal [28, 86, 89]. Recall that our forward gradient is calculated via $g_v(\theta) := \nabla_v f(\theta) v$, where $\nabla_v f(\theta) = \nabla f(\theta) \cdot v$ is the dot product of the gradient vector $\nabla f(\theta)$ and the perturbation vector v. Figure 9b shows our analysis of 10,000 perturbations generated while training DistilBERT on AGNEWS. We find that the majority have cosine similarity with the gradient vector $\nabla f(\theta)$ near zero; specifically, over 60% have a similarity less than 0.03, and they collectively contribute less than 29.6% of the final forward gradients. These perturbations contribute minimal forward gradient to model convergence.

Similarity-aware discriminative sampling: We propose a strategy to manipulate the perturbation generation process, making it more conducive to convergence. Figure 9a illustrates that when perturbation v is near orthogonal to gradient $\nabla f(\theta)$, the resulting forward gradients $g_v(\theta)$ (shallower red) contribute less. Based on this observation, we can select perturbations exhibiting high cosine similarity with the gradient vector $\nabla f(\theta)$. This technique eliminates the need to compute negligible contributions from certain perturbations. Our methodology leverages the opportunity that the gradients direction changes smoothly during FL – an observation also exploited in prior FL literature [56, 92].

Models	Arch.	Params.	PEFT	Infer. Libs
ALBERT-base [46]	Encoder-only	12M	BitFit	TFLite [5]
DistilBERT-base [77]	Encoder-only	66M	Adapter	TFLite [5]
BERT-base [27]	Encoder-only	110M	Bitfit	TFLite [5]
RoBERTa-large [63]	Encoder-only	340M	Bitfit	TFLite [5]
LLaMA [85]	Decoder-only	7B	LoRA	llama.cpp [6]

Table 4: Tested models. All models use PyTorch On TX2.

Therefore, it's sufficient to compute the cosine similarities only between the perturbations and the forward gradients from the preceding round. For practical implementation, perturbation manipulation computations are outsourced to the cloud. The cloud first generates a set of random seeds, filtering out perturbations with low cosine similarities to the prior round's forward gradients. The resultant seeds are then dispatched to clients for local perturbation generation.

4 IMPLEMENTATION AND SETUPS

FwdLLM prototype We have fully implemented the FwdLLM prototype atop FedNLP [104], which is one of the state-of-the-art frameworks to develop and evaluate FL methods on NLP tasks. There are two primary ways to implement forward gradients: numerical differentiation and analytical differentiation. We use the former as it is almost identical to forward inference. We therefore implement it with functorch library [3]. For various PEFT methods: adapter is implemented with AdapterHub [74], a library that facilitates the integration of different pre-trained adapters for downstream tasks; LoRa and BitFit are implemented by ourselves. The quantized LLaMA training is based on AutoGPTQ library [4].

Models We evaluate FwdLLM mainly on five popular LLMs, as shown in Table 4. Four of them are BERT-like models based on transformer encoders: (1) ALBERT-base (12M) [46]; (2) DistilBERT-base (66M) [77]; (3) BERT-base (110M) [27]; (4) RoBERT-large (340M) [63]. Those four models are extensively used in prior FedNLP research [19, 20, 104]. Apart from that, we also evaluate FwdLLM on the SoTA open-sourced generative language model (5) LLaMA-7B (INT4) [85]. To hold the whole LLaMA model in memory, we quantize it to INT4 format using GPTQ [32]. As far as we know, FwdLLM is the first attempt to apply federated learning to a billion-size models. FwdLLM selects different PEFT methods for different models with its offline profiler presented in §3.2: Adapter [74] for DistilBERT; LoRA [40] for LLaMA; and BitFit [105] for others. The pre-trained weights of above models are from Huggingface [95].

Datasets We experiment with three popular NLP datasets: (1) AGNEWS [108] is a news classification dataset with 4 classes. The number of training samples for each class is 30K and testing 1.9K. (2) YAHOO [108] is a topic classification dataset with 10 categories. Each category contains 140K training samples and 5,000 testing samples. (3) YELP-Polarity (YELP-P) [108] predicts a polarity label based on restaurant reviews. The number of each polarity is 280K/19K for training/testing. By default, we uniformly divide the datasets into 1,000 clients for AGNEWS and YELP-P, 10,000 clients for YAHOO. For non-iid settings, we follow prior literatures [20, 104] to divide datasets into skewed label distribution.

Hardware As prior FL literature [45, 50, 52, 104], our experiments are carried out in an semi-emulation manner on two GPU servers each with 8 × NVIDIA A100. The on-device training time is obtained on two popular edge devices: (1) Google Pixel 7 Pro (Pixel) is a popular mobile phones that is equipped with a Google Tensor G2 TPU, a Mali-G710 GPU and a Octa-core CPU. It runs Android 13 OS. We use TFLite [5] to train the four BERT-variant models and llamap.cpp [6] to run inference with LLaMA. The inference and training speed is previously illustrated in Table 2. (2) Jetson TX2 (TX2) [7] is a widely used edge board equipped with a 256-core NVIDIA Pascal GPU and a Dual-Core NVIDIA Denver 2 64-Bit CPU. We use PyTorch [71] for on-device inference and training. Note that Jetson TX2 has no NPU and is not the primary target platform of FwdLLM.

Metrics We mainly report time-to-accuracy metric and on-device runtime cost (memory, network, energy). The target convergence accuracy is 0.88 for AGNEWS, 0.65 for YAHOO and 0.82 for YELP-P, the same as prior work [20, 104, 108]

Baselines We compare FwdLLM to the following baselines: (1) Full-FT always fine-tunes the whole model [104]. (2) Adapter tuning (Adapter) introduces a small tunable module between transformer layers and freeze other parameters [39, 74]. (3) Bias tuning (BitFit) only tunes bias of each layer in the LLM [105]. (4) FedAdapter is the SoTA FedNLP fine-tuning framework that incorporates adapter tuning with layer freezing techniques [104], along with a progressive training paradigm to identify the optimal adapter configuration automatically [20]. All baselines use BP-based training, therefore can only use smartphone CPU as previously discussed in §2.2. Instead, FwdLLM can leverage mobile GPU and NPU as well. A recent work [98] ports training to mobile NPU, yet does not support transformer models.

FL settings Unless otherwise stated, FwdLLM selects 1000 clients per round, while baselines select 10 clients following prior work [20, 37, 50, 104]. This is because, as previously shown in §2.2, 10 clients already saturate the convergence performance of backpropgation-based FL methods; while FwdLLM is able to scale its convergence speed with more clients. Notably, not every device is training ready due to the runtime and battery status [8]; yet 1,000 clients are commonly available in a popular mobile app. We will also compare the performance of `FwdLLM` and baselines under different training-ready client numbers. `FwdLLM` and all baselines use the same set of hyper-parameters as prior work [20, 104]: local training epoch as 1; mini-batch size as 8; learning rate as 0.01; max sequence length as 64 for AGNEWS and 256 for others. The default FL aggregating algorithm is FedSGD [68] as later experiments will show that FedAVG [8] underperforms FedSGD due to the asymmetric compute/network cost. Network bandwidth is set to 10Mbps by default as prior literature [19, 20, 104].

5 EVALUATION

5.1 Convergence Performance on Sub-billion-sized Models

We first study the performance of `FwdLLM` on the four BERT-like models that have less than 1B parameters and are extensively used in prior FedNLP research [19–21, 104].

`FwdLLM` achieves significant improvements with mobile NPU. Table 5 summarizes the convergence time and Figure 10 illustrates the convergence process under the default setting. To reach the target accuracy, `FwdLLM` outperforms `Full-FT` by an significant factor of 217.3×. Compared with parameter-efficient fine-tuning baseline and the state-of-the-art federated fine-tuning system FedAdapter [20], `FwdLLM` can still beat them non-trivially (10.6× on average). The significant boost in performance partly stems from `FwdLLM`'s capacity to harness the powerful NPU, accelerating the training process. Unique to `FwdLLM` is its dependency solely on the forward pass, in contrast to other baselines that require the backward pass — a function not yet compatible with NPUs, as elaborated in Section 2.2.

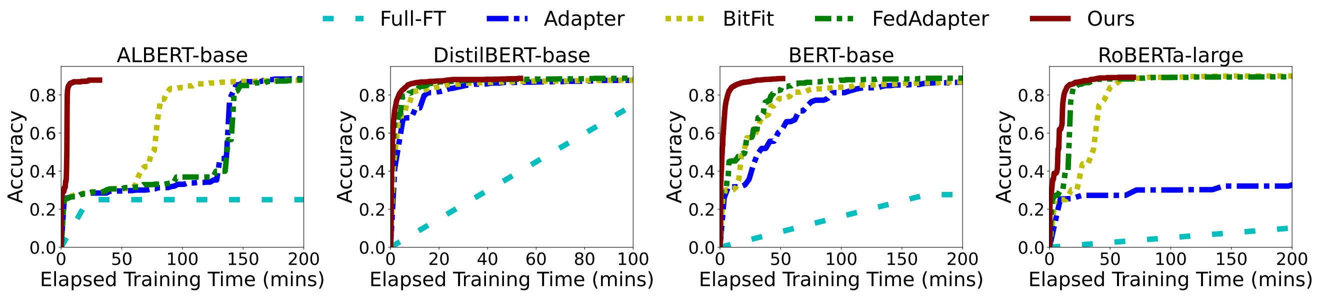
Intriguingly, our research reveals that, contrary to prior FL scenarios [20, 50, 52], FedLLM with PEFT methods favors FedSGD towards faster convergence compared to FedAVG. As detailed in Table 5, FedSGD using adapter leads to up to 8.04× faster convergence than FedAVG on the four tested models. This can be rationalized by understanding the inherent design of FedAVG. Primarily devised to address the communication overhead in federated learning, the benefits of FedAVG become muted, as PEFT methods have already significantly mitigated these overheads.

`FwdLLM` is versatile across different processors and hardware boards. Though designed primarily for NPUs, `FwdLLM` showcases commendable performance across multiple processors within the Google Pixel. On the GPU, it is up to 158.6× faster than the `Full-FT`, and it consistently outperforms several strong baselines, achieving speeds of 1.5× on average. Remarkably, only `FwdLLM` can exploit both the GPU and NPU in the Google Pixel, while other baselines are confined to the CPU, as illustrated in Table 2. Table 5 further showcases that, even on the Google Pixel's CPU, `FwdLLM` can still operate 70.4× faster than `Full-FT`. When compared to more advanced baselines, `FwdLLM` outstrips approximately 45.3% and matches the remainder. These results suggest that the design of `FwdLLM` is not solely dependent on NPUs; it's equally effective on other processors.

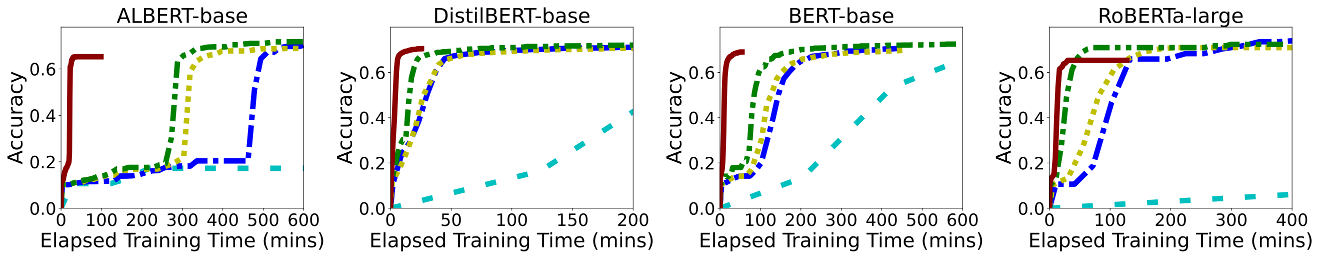
Consistently, `FwdLLM` surpasses other baselines regardless of the device or its underlying processor. As evidenced in Figure 11, when tested on a strong edge board like the Jetson TX2, equipped with a GPU and capable of running various baseline models, `FwdLLM` maintains its lead. Its convergence time showcases significant improvements, ranging from 12.2 to 3143.7× faster than `Full-FT`. Furthermore, it beats strong baselines in most of the cases, delivering speeds up to 39.1× faster. The advantage stems from the reality that a forward pass is inherently 3-5× quicker than a backward pass [13, 15]. The numerical gap between forward and backward pass are highly program dependent, which will be exaggerated in efficient inference engines [57, 62, 88]. Apart from that, our global cherry-picked forward gradients, refined through discriminative filtering, help to guide the training process to reach the target accuracy quickly.

`FwdLLM` exhibits enhanced scalability in convergence speed as the number of available training devices increases. Figure 12 demonstrates that `FwdLLM` adeptly leverages a growing number of clients to bolster convergence performance. With the client count increasing from 1 to 500, `FwdLLM`'s convergence duration notably reduces from 563.65 to 9.96 minutes. On the other hand, vanilla FedLLM methods, which rely on backward passes, fail to showcase a similar trend in scalability. For these approaches, the convergence duration plateaus once the client count exceeds 5. This behavior echoes the findings in §2.2, underscoring that `FwdLLM` outperforms backward pass-based FedLLM methods in scalability as more clients are added.

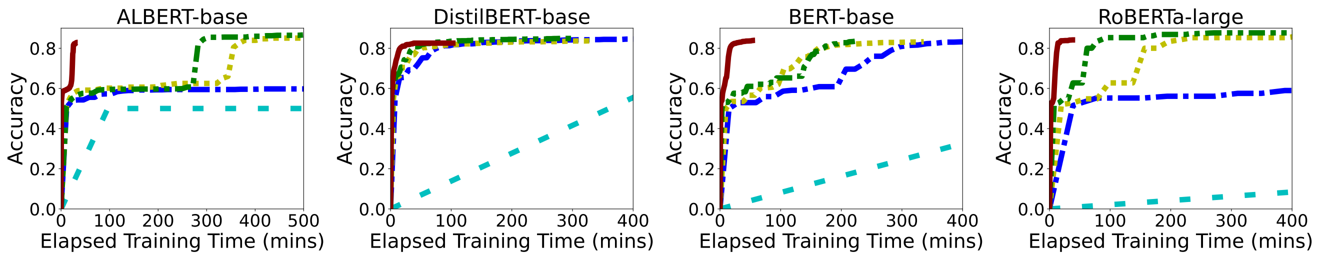
`FwdLLM` demonstrates resilience to non-iid data distributions. We evaluate the performance of `FwdLLM` under non-iid data distribution, as presented in Figure 13. Notably, while the performance of `FwdLLM` under non-iid slightly lags behind its IID counterpart — witnessing up to a 3.8% accuracy drop in AGNEWS dataset, it still achieves parity convergence accuracy with strong baseline methods under non-iid circumstances, with up to 337.5% faster. The resilience of `FwdLLM` to non-iid data distributions can be attributed to its ability to engage a vast number of clients, effectively harnessing their collective knowledge in a single round.



(a) AGNEWS



(b) YAHOO



(c) YELP-P

Figure 10: Overall Performance of FwdLLM and baselines. Processor: NPU for FwdLLM and CPU for others.

Convergence Time (mins)	ALBERT-base			DistilBERT-base			BERT-base			RoBERTa-large		
	AGNEWS	YAHOO	YELP-P	AGNEWS	YAHOO	YELP-P	AGNEWS	YAHOO	YELP-P	AGNEWS	YAHOO	YELP-P
Full-FT	4598.3	1076.0	5871.3	721.0	651.4	892.7	1535.2	1090.9	2217.4	3833.6	Err	Err
Adapter	168.3	509.9	948.3	84.7	115.3	119.6	250.1	311.8	370.8	860.0	132.7	1319.3
Adapter (FedAvg)	1325.6	2147.9	1119.6	136.9	485.7	141.2	595.2	1718.6	704.6	298.1	1067.0	410.4
Bitfit	174.8	350.5	367.0	76.4	134.8	116.7	272.8	366.3	307.2	58.9	131.4	196.3
FedAdapter	187.8	303.1	293.2	29.5	59.9	52.5	89.5	176.2	212.7	27.0	45.9	123.1
Ours (CPU)	227.1	315.9	271.6	61.5	110.5	92.2	200.7	462.7	242.8	194.3	277.3	95.3
Ours (GPU)	53.2	73.0	63.5	28.1	32.5	42.0	31.1	57.5	37.5	49.1	60.4	24.1
Ours (NPU)	22.7	30.4	27.0	21.9	18.1	32.7	27.6	49.0	33.2	28.9	30.1	14.1

Table 5: End-to-end performance. Device: Google Pixel 7P. Err: failed to reach target accuracy within 100 hrs.

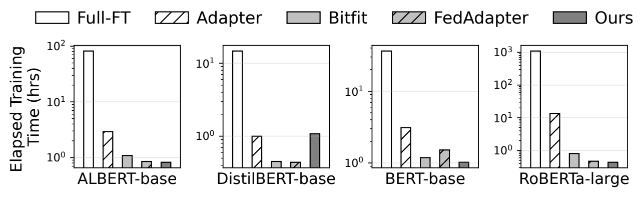


Figure 11: Convergence on Jetson TX2 and YELP-P.

Convergence time (mins)	DistilBERT-base		BERT-base	
	Uniform	Skewed	Uniform	Skewed
Adapter	23.4	37.8	92.3	158.2
Ours	9.1	11.2	28.4	51.2

Table 6: Summary of FwdFL performance under non-iid data distribution. Target accuracy is set as 0.84.

5.2 Convergence Performance on Billion-Sized Model (LLaMA)

We further assess FwdLLM using a billion-sized model, LLaMA-7B, on the AGNEWS dataset and the Pixel 7 Pro. To accommodate the model on these devices, we employed the

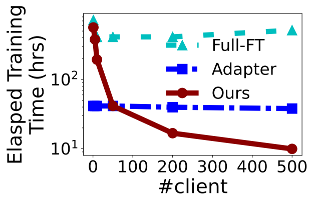


Figure 12: FwdLLM performance with different #clients.

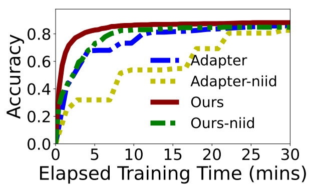
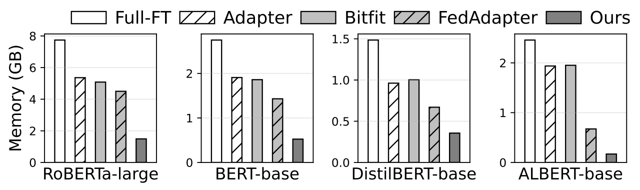
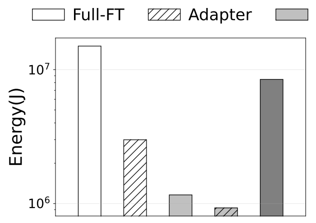
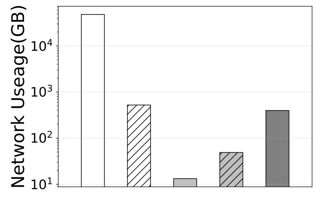


Figure 13: Performance under non-iid circumstance.



(a) Peak memory footprint





(b) Total energy cost **(c) Total network cost**

Figure 14: Resource cost of FwdLLM and baselines.

Methods	Mem. (GB)	Centralized Training (A100)			Federated Learning		
		Acc.	Round	Time	Acc.	Round	Time
BP, FP16	39.2	89.7	500	0.1 hrs	N/A due to memory inefficiency on Pixel 7 Pro (8GB)		
BP, INT8	32.4	88.6	500	0.06 hrs			
BP, INT4	28.5	87.8	500	0.04 hrs			
Ours, FP16	15.6	87.0	240	1.5 hrs			
Ours, INT8	7.9	86.9	260	0.8 hrs			
Ours (CPU), INT4	4.0	85.8	130	0.25 hrs	85.8	130	0.19 hrs
Ours (NPU*), INT4							0.07 hrs

Table 7: FwdLLM combined with INT4-based quantization is the only feasible approach in federated learning of LLaMA-7B. Dataset: AGNEWS; Acc.: accuracy (%). Centralized training and federated learning are conducted on NVIDIA A100 and Pixel 7 Pro, respectively. *: LLaMA currently is not supported by mobile NPU, therefore we emulate its speed based on the speedup of other BERT-like models.

quantization technique GPTQ [32], which is prevalently utilized with LLMs to minimize parameter redundancy. During the fine-tuning process, we adopted low-precision data formats like INT8/INT4 for the native LLaMa weights, while retaining FP32 for the trainable LoRa weights. Given that backpropagation-based training approaches are not feasible for such sizable models on mobile platforms, we conducted experiments using both centralized training (leveraging 1x NVIDIA A100) and federated learning (on smartphones).

With results shown in Table 7, we make following key observations. (1) For the first time, FwdLLM enables federated learning of billion-sized LLMs like LLaMA on COTS mobile devices. Combined with INT4 quantization, it takes only 0.19 hours for FwdLLM to achieve the target accuracy running on mobile CPU, which is comparable to a centralized training with one NVIDIA A100 GPU. This is mainly due to its ability to scale out its speed on a thousand devices simtaneously. If on-device training can be accelerated by NPU, FwdLLM is even much faster than the centralized training. (2) FwdLLM exhibits a harmonious orchestration with quantization strategies. Quantizing LLaMA weights to INT8 and INT4 effectively reduces the training-time memory footprint from 15.6GBs to 7.9GBs and 4.0GBs. Yet, the accuracy only degrades by less than 1.2%. This shows the great compatibility of FwdLLM with existing model compression algorithms.

5.3 System Cost

We analyze the resource cost during FedLLM, including the peak memory footprint, total energy and network (both uplink and downlink) expenditure on all participant devices. The experiments are performed with RoBERTa-large and YELP-P on Pixel 7 Pro. The results are depicted in Figure 14.

Memory footprint As shown in Figure 14a, FwdLLM achieves up to a 93% reduction in memory usage compared to FT and a 91.3% decrease relative to the robust PEFT benchmarks. Notably, only 169MB of memory is required for a single FwdLLM round, rendering it highly suitable for mobile devices. This efficiency stems from FwdLLM's design, which mandates only the execution of the forward pass, thus requiring storage primarily for the trainable model parameters and the perturbation weights; both are parameter-efficient as detailed in §3.1. Moreover, due to FwdLLM's sole reliance on inference functions, cutting-edge inference-centric memory optimization techniques [57, 62, 88] can be effortlessly incorporated to further reduce memory consumption.

Energy and network Compared to Full-FT, FwdLLM is able to save 43.6% energy consumption and 99.1% network cost on across devices. Yet, FwdLLM is not as energy/network-efficient as more competitive baselines with PEFT enhancements, e.g., 9.1× more energy consumption and 30.5× network traffic compared to Adapter approach. It is mainly ascribed to the involvement of 100× more devices per round, i.e., a price paid for scalability. Though, we deem such a price paid is bearable since FL participant devices are typically in a good condition, i.e., with battery being charged and non-metered nework connectivity like WiFi [8].

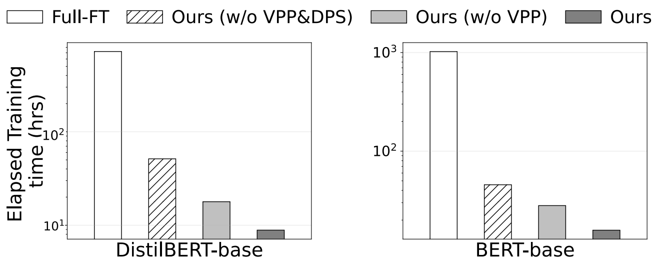


Figure 15: Model convergence delay with or without FwdLLM's key designs, showing their significance. VPP means Variance-controlled Perturbation Pacing, and DPS means Discriminative Perturbation Sampling.

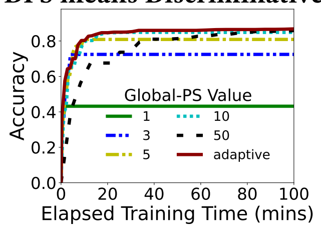


Figure 16: Adaptive vs Fixed. Model:DistilBERT. Dataset:AGNews

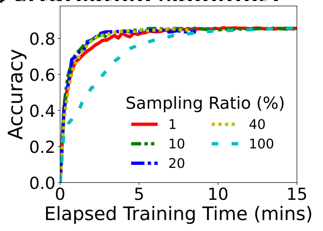


Figure 17: Different sampling ratio. Model:BERT. Dataset:AGNews

5.4 Significance of key designs

Dissecting system benefits Figure 15 reveals that each design element plays a pivotal role in enhancing the performance of FwdLLM. (1). Vanilla Forward-FL (FwdLLM without variance-controlled perturbation and discriminative perturbation sampling) dramatically reduces the convergence time, ranging from 14.13× to 22.45×. This improvement is ascribed to FwdLLM's capability to engage more clients in local PEFT training, thereby magnifying their contributions during the training phase. (2). Discriminative perturbation sampling can augment the convergence rate by 1.63× to 2.88×. This enhancement is realized by harnessing the full potential of each perturbation, enabling them to contribute more significantly to the gradient updates. (3). Variance-controlled perturbation pacing further bolsters convergence performance, with improvements ranging from 1.78× to 2.02×. This can be attributed to two primary factors: the early elimination of ineffective perturbations and the precise identification of optimal perturbations in subsequent phases.

Global-PS selection strategy Figure 16 demonstrates how our Global-PS planning strategy efficiently determines the best global-ps to achieve rapid convergence to a high accuracy. In comparison with a fixed, small global-ps value of 1, FwdLLM delivers an accuracy boost of 43.6%. And when pitted against a larger, fixed global-ps value of 50, FwdLLM achieves equivalent accuracy of 85% but 2.99× faster. This enhancement stems from that FwdLLM dynamically adapts the global-ps value as training progresses. It starts with a smaller global-ps to rapidly gain basic accuracy and transitions to a larger global-ps to hone in on higher accuracy in the later stages.

Sensitivity of discriminative sampling Figure 17 illustrates how varying sampling ratios influence the convergence performance. A 20% sampling ratio emerges as the most efficient, offering the best runtime improvement (2.33×) compared to not employing any sampling (equivalent to a 100% sampling ratio). Using a sampling ratio in the range of 10-40% yields comparable runtime benefits, suggesting that there's flexibility in choosing an effective hyperparameter, ensuring robustness of the sampling method. However, opting for extremely low sampling ratios is not always beneficial. For instance, a mere 1% sampling ratio hampers the convergence speed, taking 1.42× longer than the optimal 20% ratio. This is due to the limited gradient update directions, resulting in a decelerated convergence.

6 RELATED WORK

FedLLM With the recent rise of transformers and its variants [27, 38, 60, 77, 87], LLMs have achieved great success in various domians, including CV, NLP, etc. While its success is largely attributed to the pre-training paradigm, the privacy issue of LLMs has been a major concern, e.g., extraction attacks [10, 26, 111]. FedLLM is a promising direction to releave the privacy tension of LLMs [112]. To tackle the tight resource constraint on devices, most recent efforts resort to parameter-efficient fine-tuning (PEFT) techniques such as Adapter and LoRa [34, 40, 74, 105]. For instance, FedAdapter [20] addresses the adapter configuration problem for FedLLM. However, in §2 we show that PEFT-based FedLLM only mitigates the network bottleneck, but cannot significantly optimize the memory or speed of on-devce training. FwdLLM is built atop PEFT but significantly reduces the device-side overhead and accelerates convergence.

Backpropagation-free training Backpropagation is the most widely adopted but not the only way to train neural networks, such as zero-order optimization proposed in early 80s [11]. As models getting larger, reserachers are realizing that backpropagation becomes a burden to DNN training pipeline. Thus, a few new paradigms for backpropagation-free training are proposed [13, 35, 66, 81]. This work is built atop forward gradient for being mobile friendly. However, prior literature [13, 70, 75] of forward gradient still use toy models (MLP or small CNNs). BBTv2 [81] applys gradient-free methods to optimize the LLM prompts in a central cloud. BAFFLE, a concurrent work to FwdLLM [30], is the only one that combines backpropagation-free training with FL. However, it is not designed for LLM and lacks the two key techniques as presented in §3.3 and §3.4 that make backpropagation-free FL more systematic and efficient.

To our best knowledge, FwdLLM is the very first system that leverages backpropagation-free training algorithm to enable cross-device FL of billion-size LLM like LLaMA-7B.

FL Optimizations There have been tremendous efforts in making cross-device FL more efficient, including communication efficiency [16, 102], model compression/quantization [14, 97], client/data sampling [45, 50–52, 69, 93, 99, 110], and on-device training speedup [91, 98]. However, their benefits are modest compared to the gap between the LLMs like LLaMA and the resource constraint of mobile devices. FwdLLM innovates the FL protocol by abandoning backpropagation and therefore brings much more significant improvements.

7 CONCLUSIONS

In this study, we introduce FwdLLM, the pioneering federated learning framework for LLMs that operates without backpropagation. It borrows the wisdom from the forward gradients method and applies it to FedLLM training, so as to avoid memory-intensive backpropagation and enable scalable training of LLMs on mobile devices. To generate forward gradients more efficiently and precisely, we employ an adaptive perturbation generator that determines the number of perturbations on the fly. Additionally, we incorporate a discriminative sampler to selectively screen the generated perturbations. Comparative analyses reveal that FwdLLM outperforms contemporary FedLLM methods in both convergence time and scalability.

REFERENCES

[1] https://www.tensorflow.org/lite/guide/ops_select.
[2] https://ai-benchmark.com/ranking.html.
[3] https://pytorch.org/functorch/stable/.
[4] https://github.com/PanQiWei/AutoGPTQ.
[5] https://tensorflow.google.cn/lite.
[6] https://github.com/ggerganov/llama.cpp.
[7] https://developer.nvidia.com/embedded/jetson-tx2.
[8] Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.
[9] Android. Android: Low memory killer daemon. https://source.android.com/docs/core/perf/lmkd, 2022.
[10] 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.
[11] Andrew G Barto and Michael I Jordan. Gradient following without back-propagation in layered networks. In 1st Int. Conference Neural Nets, San Diego, volume 2, 1987.
[12] 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.
[13] Atılım Güneş Baydin, Barak A Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation. arXiv preprint arXiv:2202.08587, 2022.
[14] 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.
[15] Éric Blayo, Marc Bocquet, Emmanuel Cosme, and Leticia F Cugliandolo. Advanced data assimilation for geosciences: Lecture notes of the LES Houches School of Physics: Special issue, June 2012. OUP Oxford, 2014.
[16] 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.
[17] Dongqi Cai, Qipeng Wang, Yuanqiang Liu, Yunxin Liu, Shangguang Wang, and Mengwei Xu. Towards ubiquitous learning: A first measurement of on-device training performance. In Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning, pages 31–36, 2021.
[18] Dongqi Cai, Shangguang Wang, Yaozong Wu, Felix Xiaozhu Lin, and Mengwei Xu. Federated nlp in few-shot scenarios. arXiv preprint arXiv:2212.05974, 2022.
[19] Dongqi Cai, Shangguang Wang, Yaozong Wu, Felix Xiaozhu Lin, and Mengwei Xu. Federated nlp in few-shot scenarios. arXiv preprint arXiv:2212.05974, 2022.
[20] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Fedadapter: Efficient federated learning for modern nlp. arXiv preprint arXiv:2205.10162, 2022.
[21] 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.
[22] 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.
[23] Qingqing Cao, Noah Weber, Niranjan Balasubramanian, and Aruna Balasubramanian. Deqa: On-device question answering. In Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, pages 27–40, 2019.
[24] Chaochao Chen, Xiaohua Feng, Jun Zhou, Jianwei Yin, and Xiaolin Zheng. Federated large language model: A position paper. arXiv preprint arXiv:2307.08925, 2023.
[25] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. Mathematics for machine learning. Cambridge University Press, 2020.
[26] 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.
[27] 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.
[28] David Donoho and Jared Tanner. Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 367(1906):4273–4293, 2009.
[29] Cynthia Dwork. Differential privacy: A survey of results. In International conference on theory and applications of models of computation, pages 1–19. Springer, 2008.
[30] Haozhe Feng, Tianyu Pang, Chao Du, Wei Chen, Shuicheng Yan, and Min Lin. Does federated learning really need backpropagation? arXiv preprint arXiv:2301.12195, 2023.

[31] Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. arXiv preprint cs/0408007, 2004.

[32] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. arXiv preprint arXiv:2210.17323, 2022.

[33] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017.

[34] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In ICML, 2022.

[35] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. arXiv preprint arXiv:2212.13345, 2022.

[36] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. Advances in neural information processing systems, 30, 2017.

[37] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. Advances in Neural Information Processing Systems, 34:12876–12889, 2021.

[38] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. Advances in Neural Information Processing Systems, 33:9782–9793, 2020.

[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] Joo Seong Jeong, Jingyu Lee, Donghyun Kim, Changmin Jeon, Changjin Jeong, Youngki Lee, and Byung-Gon Chun. Band: coordinated multi-dnn inference on heterogeneous mobile processors. In Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services, pages 235–247, 2022.

[42] Liuyi Jin, Tian Liu, Amran Haroon, Radu Stoleru, Michael Middleton, Ziwei Zhu, and Theodora Chaspari. Emsassist: An end-to-end mobile voice assistant at the edge for emergency medical services. In Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services, pages 275–288, 2023.

[43] Yeachan Kim, Junho Kim, Wing-Lam Mok, Jun-Hyung Park, and SangKeun Lee. Client-customized adaptation for parameter-efficient federated learning. In Findings of the Association for Computational Linguistics: ACL 2023, pages 1159–1172, 2023.

[44] 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.

[45] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. Oort: Informed participant selection for scalable federated learning. arXiv preprint arXiv:2010.06081, 2020.

[46] 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.

[47] Niel Lebeck, Arvind Krishnamurthy, Henry M Levy, and Irene Zhang. End the senseless killing: Improving memory management for mobile operating systems. In 2020 USENIX Annual Technical Conference (USENIX ATC 20), pages 873–887, 2020.

[48] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. nature, 521(7553):436–444, 2015.

[49] Seulki Lee and Shahriar Nirjon. Fast and scalable in-memory deep multitask learning via neural weight virtualization. In Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services, pages 175–190, 2020.

[50] 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.

[51] 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.

[52] 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.

[53] Cong Li, Jia Bao, and Haitao Wang. Optimizing low memory killers for mobile devices using reinforcement learning. In 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pages 2169–2174. IEEE, 2017.

[54] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. Proceedings of Machine learning and systems, 2:429–450, 2020.

[55] 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.

[56] Xingyu Li, Zhe Qu, Bo Tang, and Zhuo Lu. Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients. arXiv preprint arXiv:2102.06329, 2021.

[57] Xiaobo Liang, Juntao Li, Lijun Wu, Ziqiang Cao, and Min Zhang. Dynamic and efficient inference for text generation via bert family. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2883–2897, 2023.

[58] 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.

[59] Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. IEEE Signal Processing Magazine, 37(5):43–54, 2020.

[60] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. arXiv preprint arXiv:2004.02178, 2020.

[61] 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.

[62] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14420–14430, 2023.
[63] 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.
[64] 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.
[65] Kangyang Luo, Xiang Li, Yunshi Lan, and Ming Gao. Gradma: A gradient-memory-based accelerated federated learning with alleviated catastrophic forgetting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3708–3717, 2023.
[66] Wan-Duo Kurt Ma, JP Lewis, and W Bastiaan Kleijn. The hsic bottleneck: Deep learning without back-propagation. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 5085–5092, 2020.
[67] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. arXiv preprint arXiv:2305.17333, 2023.
[68] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. arXiv preprint arXiv:1602.05629, 2:2, 2016.
[69] 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.
[70] Seonghwan Park, Dahun Shin, Jinseok Chung, and Namhoon Lee. Fedfwd: Federated learning without backpropagation. In Federated Learning and Analytics in Practice: Algorithms, Systems, Applications, and Opportunities, 2023.
[71] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703, 2019.
[72] 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.
[73] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In EACL, 2021.
[74] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. arXiv preprint arXiv:2007.07779, 2020.
[75] Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. arXiv preprint arXiv:2210.03310, 2022.
[76] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
[77] 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.
[78] 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.
[79] Nimit S Sohoni, Christopher R Aberger, Megan Leszczynski, Jian Zhang, and Christopher Ré. Low-memory neural network training: A technical report. arXiv preprint arXiv:1904.10631, 2019.
[80] Charles M Stein. Estimation of the mean of a multivariate normal distribution. The annals of Statistics, pages 1135–1151, 1981.
[81] Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuan-Jing Huang, and Xipeng Qiu. Bbtv2: towards a gradient-free future with large language models. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3916–3930, 2022.
[82] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. arXiv preprint arXiv:2004.02984, 2020.
[83] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning, 2022.
[84] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. Fedbert: When federated learning meets pre-training. ACM Trans. Intell. Syst. Technol., 13(4), aug 2022.
[85] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.
[86] Joel A Tropp. Improved analysis of the subsampled randomized hadamard transform. Advances in Adaptive Data Analysis, 3(01n02):115–126, 2011.
[87] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.
[88] Sairam Sri Vatsavai, Venkata Sai Praneeth Karempudi, and Ishan Thakkar. An optical xnor-bitcount based accelerator for efficient inference of binary neural networks. In 2023 24th International Symposium on Quality Electronic Design (ISQED), pages 1–8. IEEE, 2023.
[89] Roman Vershynin. High-dimensional probability: An introduction with applications in data science, volume 47. Cambridge university press, 2018.
[90] Boxin Wang, Yibo Jacky Zhang, Yuan Cao, Bo Li, H Brendan McMahan, Sewoong Oh, Zheng Xu, and Manzil Zaheer. Can public large language models help private cross-device federated learning? arXiv preprint arXiv:2305.12132, 2023.
[91] 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.
[92] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. IEEE journal on selected areas in communications, 37(6):1205–1221, 2019.
[93] 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.

[94] Paul J Werbos. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10):1550–1560, 1990.

[95] 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.

[96] Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong. Pecam: privacy-enhanced video streaming and analytics via securely-reversible transformation. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, pages 229–241, 2021.

[97] 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.

[98] 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.

[99] 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.

[100] Mengwei Xu, Feng Qian, Qiaozhu Mei, Kang Huang, and Xuanzhe Liu. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. IMWUT, 2(4):1–26, 2018.

[101] Mengwei Xu, Feng Qian, Qiaozhu Mei, Kang Huang, and Xuanzhe Liu. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2(4):1–26, 2018.

[102] 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.

[103] Yang You, Yuhui Wang, Huan Zhang, Zhao Zhang, James Demmel, and Cho-Jui Hsieh. The limit of the batch size. arXiv preprint arXiv:2006.08517, 2020.

[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] 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.

[107] Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Guoyin Wang, and Yiran Chen. Towards building the federated gpt: Federated instruction tuning. arXiv preprint arXiv:2305.05644, 2023.

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

[109] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5. IEEE, 2023.

[110] 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.

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

[112] Weiming Zhuang, Chen Chen, and Lingjuan Lyu. When foundation model meets federated learning: Motivations, challenges, and future directions. arXiv preprint arXiv:2306.15546, 2023.