Efficient Collaborative LLM Fine-Tuning over Heterogeneous Mobile Devices via Many Backbones to One Side-Network Tuning

Xingke Yang[1,*], Liang Li[2,*], Sicong Li[1], Liwei Guan[3], Hao Wang[5], Xiaoqi Qin[3], Jiang Liu[4]
Xin Fu[1], Miao Pan[1]

[1]University of Houston, USA [2]Pengcheng Laboratory, China
[3]Beijing University of Posts and Telecommunications, China
[4]Waseda University, Japan [5]Stevens Institute of Technology, USA
xyang56@cougarnet.uh.edu mpan2@central.uh.edu

Abstract

Collaboratively fine-tuning (FT) large language models (LLMs) over heterogeneous mobile devices fosters immense potential applications of personalized intelligence. However, such a vision faces critical system challenges. Existing federated LLM fine-tuning approaches remain limited by prohibitive on-device resource overhead, straggler-prone synchronous aggregation, and the assumption of a unified pretrained backbone across heterogeneous devices. In this paper, we propose EC-MobiLLM, a novel design for efficient collaborative LLM fine-tuning across heterogeneous mobile devices and heterogeneous backbones. EC-MobiLLM adopts a server-assisted side-tuning paradigm to minimize on-device overhead and pioneers non-blocking asynchronous collaboration to accelerate training. Moreover, EC-MobiLLM introduces adaptive feature alignment to support collaboration across heterogeneous backbones, aligning with practical needs in real-world deployments. Extensive experimental results demonstrate that EC-MobiLLM can maintain robust fine-tuning performance while achieving extremely low on-device memory, with at least 95.2% reduction in computation overhead, 93.2% reduction in communication costs and $5.1\times$ faster convergence compared to existing methods, validating its efficacy for practical LLM adaptation over heterogeneous mobile devices.

1 Introduction

Fine-tuning (FT) large language models (LLMs) for domain-specific tasks unlocks significant potential for novel applications, driving growing demand for personalized intelligence [21, 36, 3]. In practice, the limited scale and diversity of data on an individual mobile device is often insufficient to support robust fine-tuning, motivating collaborative LLM fine-tuning across devices to improve model performance. Early works [39, 27] explored collaborative LLM FT by combining parameter-efficient fine-tuning (PEFT) methods (e.g., Adapters [13] and LoRA [14]) with the traditional federated learning (FL) [19] paradigm. These approaches follow the standard FL loop (local training $\rightarrow$ upload $\rightarrow$ server aggregation $\rightarrow$ model distribution) and enable cross-device collaboration by exchanging only lightweight trainable modules.

However, local training in these methods faces critical bottlenecks due to the limited computational power and memory capacity of mobile devices. For example, tuning a model with 1.3B-parameters with LoRA typically requires more than 14.5 GB of GPU memory, which exceeds the 4–12 GB

*Equal Contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

available on most mobile devices [17]. To further alleviate on-device burden, split federated learning (SFL) approaches further reduce device load by offloading deeper layers to the server [18, 5]. However, its frequent exchange of activations and gradients further introduces substantial communication overhead. Moreover, the inherently synchronous aggregation protocol of FL and SFL requires strict sequential scheduling, leading to accumulated latency and dramatically prolonged convergence time—an issue amplified by the sheer size of modern LLMs.

Furthermore, in real-world deployments, participating mobile devices are often highly heterogeneous in computational power, memory capacity, and network bandwidth. To alleviate the resulting straggler effect, some studies [1, 7] explore heterogeneous assignment of tuning modules, where devices are equipped with PEFT modules of different sizes (e.g., client-specific LoRA ranks) to adjust on-device computation, memory, and communication costs. However, these approaches enforce a unified pretrained backbone for all devices. Under this constraint, only adjusting LoRA ranks provides limited differentiation in on-device cost and fine-tuning performance. As a result, constraining high-resource devices to the same base model as low-resource ones wastes their capabilities. In practice, such devices should be able to deploy larger backbones to achieve stronger representations and a higher performance ceiling.

To tackle the above issues, we propose EC-MobiLLM, a novel and efficient framework for collaborative LLM fine-tuning over heterogeneous mobile devices, built on the server-assisted side-tuning principle. Specifically, we decouple resource-intensive gradient computation from mobile devices by hosting all trainable parameters within a shared side-network on the server, while each mobile device retains only its own resource-appropriate frozen backbone LLM. During collaborative fine-tuning, each mobile device executes forward propagation computations on local data using its frozen backbone and uploads only selected intermediate activations to the server. The server performs asynchronous backpropagation using these activations, computing gradients and updating the shared side-network and lightweight projection matrices - without requiring global synchronization. In this way, EC-MobiLLM allows mobile devices to bypass costly on-device backpropagation and optimizer steps, drastically reducing client memory and computational load. By enabling server-side side-network training to proceed immediately upon receiving any mobile device's activations, EC-MobiLLM eliminates the fundamental straggler bottleneck inherent in synchronous FL aggregation protocols. Crucially, EC-MobiLLM supports heterogeneous backbone sizes across devices while preserving cross-device collaborative LLM FT, enabling each device to benefit from the shared data diversity and achieve better fine-tuning performance.

Our salient contributions are summarized as follows:

(i) We propose EC-MobiLLM, a novel framework that pioneers mobile-friendly, asynchronous server-assisted side-tuning for LLM adaptation across distributed mobile data. Our design decouples computation by having devices perform only forward passes and upload activations, where the server asynchronously updates a unified side-network per-client activation arrival. This eliminates synchronization bottlenecks and removes all gradient computation from devices.

(ii) We design adaptive mechanisms that enable EC-MobiLLM to support heterogeneous mobile devices through cross-backbone alignment techniques. This ensures heterogeneous devices with different size models can collaboratively train a unified side-network that consolidates knowledge from all devices.

(iii) We implement and evaluate EC-MobiLLM across diverse mobile platforms (NVIDIA Jetson TX2, Xavier NX, and AGX Xavier) and model scales (sub-billion to billion parameters). Experiments across multiple tasks and system settings demonstrate that EC-MobiLLM achieves extremely low on-device memory usage, with at least 95.2% reduction in computation overhead, 93.2% reduction in communication costs and $5.1\times$ faster convergence compared to existing methods. It also delivers state-of-the-art and highly robust LLM fine-tuning performance.

2 Background and Motivations

Inefficiencies of Federated LLM FT. The limited scale and diversity of data on individual mobile devices necessitate collaborative LLM fine-tuning across devices to enhance model performance. FL has emerged as a dominant paradigm for this purpose, where devices perform local training and upload parameter updates to a central server for aggregation. However, even with PEFT techniques

such as Adapters [13], LoRA [14], and BitFit [37], locally fine-tuning LLMs under the FL paradigm remains challenging due to the substantial computation and memory demands of modern LLMs. To further alleviate on-device memory burden, split federated learning (SFL) approaches further reduce device load by offloading deeper layers to the server [18, 5, 12] but introduces communication and coordination bottlenecks. Specifically, devices sequentially exchange activations/gradients with the server during forward/backward passes, while the device-side sub-models require periodic weight aggregations across devices. Alternatively, forward-only methods like FwdLLM [34] eliminate backpropagation by estimating gradients through parameter perturbations. While reducing activation memory to inference levels, this approach requires multiple forward passes per update - increasing device computational overhead. In addition, sequential execution of clients and servers in FL also hinders fast convergence in collaborative LLM fine-tuning. Collectively, existing federated LLM fine-tuning approaches exhibit critical gaps and cannot simultaneously optimize on-device memory usage, computational overhead, communication cost, and fine-tuning performance.

Heterogeneity Challenges for collaborative LLM FT. The significant variation in computational power, memory capacity, and network bandwidth across mobile devices introduces fundamental limitations to traditional synchronous federated learning protocols, which require the central server to wait for model updates from all participating devices before every-round global aggregation. This synchronization barrier creates unavoidable delays caused by slow devices (stragglers), forcing faster devices to remain idle during waiting periods [24]. Moreover, memory heterogeneity also leads to significant resource waste. Existing heterogeneous collaborative LLM FT approaches such as FlexLoRA [1], FLoRA [30], and HetLoRA [7] largely follow the standard FL+LoRA framework. They assume the same pretrained backbone across clients, configure client-specific LoRA ranks to match individual computing resource budgets, and design aggregation schemes to effectively combine LoRA updates with heterogeneous ranks. However, as shown in Table 1, the difference in training memory usage across various LoRA ranks is not significant. In practice, capable devices naturally prefer to load larger, more powerful models for achieving better performance. In summary, existing FL-based collaborative LLM fine-tuning approaches still face a significant applicability gap in real-world deployments. This calls for a breakthrough collaborative LLM fine-tuning framework that better accommodates heterogeneous devices and supports flexible asynchronous collaboration.

Table 1: Fine-tuning accuracy and memory footprint under different LoRA ranks and backbones (batch size: 16; sequence length: 256; Task: MNLI).

Pre-trained Model	OPT-350M			OPT-1.3B
LoRA Rank (r)	r = 16	r = 64	r = 128	r = 64
Accuracy	76.1	76.5	76.8	81.1
Memory(GB)	6.65	6.70	6.79	14.49

Efficient On-device LLM Fine-Tuning. Enabling LLM fine-tuning directly on mobile devices requires innovative architectures that minimize memory and computational load. Server-assisted side-tuning, pioneered by MobiLLM [17], addresses this by decoupling trainable side-networks from frozen backbones and offloading all the gradient computation to the server. Different from split learning's strictly sequential pipeline, MobiLLM allows the device and the server to compute in parallel. Advancing this approach, PAE MobiLLM [35] (as illustrated in Fig. 1) introduces key optimizations: Mobile devices perform only a single forward pass through frozen backbones, compute the output deviation $\Delta y = \text{Label}_y - y_{\text{pre}}$ as the training target for the server-side side network, and upload token-selected intermediate activations $(\mathbf{A}_1, \ldots, \mathbf{A}_L)$ together with Δy. The server then trains the side-network exclusively using these sparse activation-deviation pairs $(\mathbf{A}_1, \cdots, \mathbf{A}_L, \Delta\mathbf{y})$. A server-cached replay mechanism further reduces device overhead by limiting local data processing to the first epoch, with subsequent iterations handled server-side. This achieves an efficient balance between device resource consumption, communication costs, and training speed for on-device LLM fine-tuning. While highly effective in single-device settings, scaling to multi-device collaboration introduces fundamental challenges, including heterogeneity in both device resources and backbone models. We bridge this gap by extending the side-tuning paradigm to multi-device collaborative LLM fine-tuning with novel architectural innovations.

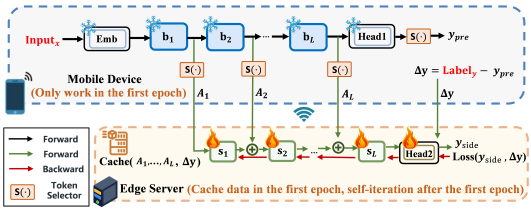
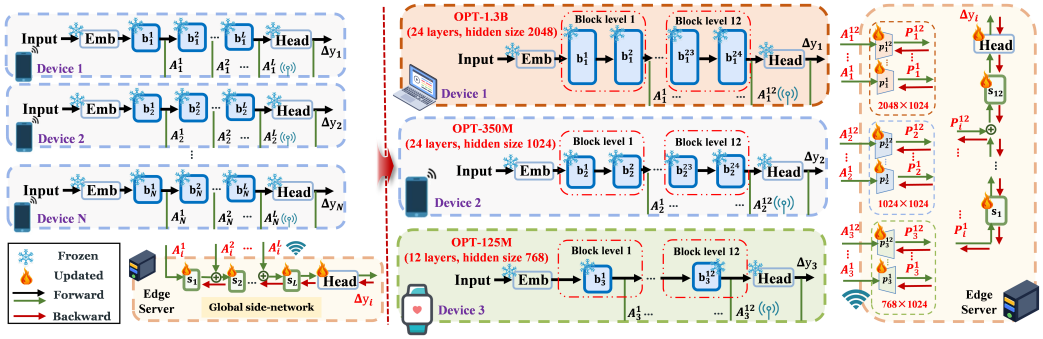


Figure 1: An overview of the PAE MobiLLM.



(a) EC-MobiLLM in the Homogeneous Model Scenario (b) EC-MobiLLM in the Heterogeneous Model Scenario

Figure 2: An overview of the EC-MobiLLM.

3 EC-MobiLLM Design

EC-MobiLLM is a collaborative LLM fine-tuning framework designed for heterogeneous mobile devices, supporting collaborative training with heterogeneous backbone models. Fig. 2 presents an overview of the EC-MobiLLM system. EC-MobiLLM provides a practical and efficient collaboration framework through inference-level on-device overhead, non-blocking asynchronous collaboration, and device-adaptive backbone selection, making it fundamentally different from conventional FL-based collaborative LLM fine-tuning methods. Specifically, each device retains only its own resource-appropriate frozen backbone LLM and performs a single forward pass using local data, without requiring on-device backpropagation. The device can then upload selected sparse intermediate activations and training target to the server at any time, without requiring synchronization between devices. The server trains a unified side network model using activation values uploaded from multiple devices, without requiring special parameter aggregation. Furthermore, the server can cache and reuse received activation values to independently and rapidly complete iterative training of the side network, without requiring continuous device-server data exchange.

Note that the concerns about data privacy arising from the transmission of intermediate activations in EC-MobiLLM are aligned with the definitions adopted in split learning related architectures [27] and Google's federated learning (FL) work [19]. Similar to these approaches, EC-MobiLLM is naturally compatible with existing advanced privacy-preserving techniques, such as differential privacy (DP) mechanisms [10] (e.g., adding DP noise to activations, gradients, outputs, or objective functions) and secure multiparty computation [9]. Further discussion is provided in Appendix A.3.

3.1 Non-blocking asynchronous collaboration

Before collaborative training starts, devices communicate their backbone model specifications to the server, enabling it to initialize the side-network parameter structure and identify which intermediate activation layers each device should upload (Sec 3.2). During collaborative fine-tuning, devices and the server run in parallel, and no synchronization is required across devices:

Mobile Devices. Each mobile device performs forward propagation through its frozen backbone using mini-batches sampled from its local dataset. For each mini-batch, device i transmits selected intermediate activations $(\mathbf{A}_i^1, \ldots, \mathbf{A}_i^L)$ from the backbone layers, along with the prediction residual Δy_i (defined as the difference between ground-truth label and backbone output) to the server as the side network training signal, consistent with PAE MobiLLM's design. Here, the selected activations mean that only the activations at token positions involved in loss computation are uploaded. For instance, for a classification task, this reduces the per-batch communication from 33.4 MB (full activations) to 0.78 MB [35]. The mobile device continuously samples mini-batches from its local training set for a full epoch, performing local computation while simultaneously uploading activations. Since the backbone model on the mobile device remains frozen and is always kept in evaluation mode, stochastic layers such as dropout are disabled. Therefore, the intermediate activations produced for the same input samples stay unchanged across training iterations. As a result, after the device finishes one full pass over its local dataset, it does not need to perform redundant computations. Instead, the server can reuse the cached data for iterative computations.

Server Training. Upon receiving these activation-deviation pairs $(\mathbf{A}_i^1, \cdots, \mathbf{A}_i^L, \Delta\mathbf{y}_i)$ from any device, the server trains the side-network immediately, i.e., processing each device's contribution upon arrival without global synchronization. For each incoming sample, the server first checks the hidden size of the uploaded activations, applies the corresponding projection modules p in the forward pass, and then runs the shared side-network forward pass. It computes the loss between the side-network output and the deviation Δy, and backpropagates along the same path. In other words, activations from different hidden-size backbones jointly train the shared side-network while separately updating their size-specific projection modules. After each update, the server caches received activations to construct an activation repository for iterative side-network training. During idle periods, the server trains continuously on cached samples to maximize computational efficiency. This design ensures uninterrupted training despite slow devices—eliminating straggler bottlenecks while maintaining full utilization of server resources. After all devices complete uploading, the server performs efficient standalone tuning using the comprehensive cached dataset.

After collaborative fine-tuning, devices download their specific projection modules alongside the shared side-network for local inference. These downloaded modules are lightweight, typically amounting to only about 1%–2% of the pretrained backbone parameters [17, 35], and introduces only minor additional inference overhead on mobile devices.

3.2 Heterogeneity-aware Cross-model Alignment

Beyond computational heterogeneity through non-blocking device-server collaboration, EC-MobiLLM fundamentally resolves memory-driven model capacity divergence across mobile devices where deployable backbone sizes are dictated by each device's memory constraints. Specifically, EC-MobiLLM allows each device to load a pre-trained backbone model scaled to its hardware capacity. As illustrated in Fig. 2 (b), devices may employ different-sized models (e.g., OPT-125M, OPT-350M, OPT-1.3B [38]), resulting in backbone architectures with varying numbers of transformer layers and hidden dimensions. This architectural diversity introduces two key challenges: (1) server-side adapters must accommodate diverse backbone structures, and (2) uploaded activations exhibit inconsistent shapes across devices. To resolve these issues, we introduce two structural alignment mechanisms—layer-wise activation sampling and hidden dimension scaling—that unify activation patterns across heterogeneous devices for collaborative training of the shared side-network on the server.

Layer-Wise Activation Sampling. Our layer-wise activation sampling mechanism addresses mismatches in transformer layer counts across backbone models. Inspired by empirical evidence in LST [26] demonstrating that not all layers require dedicated adaptation modules, we propose selectively extracting activations from strategic layer positions.

As shown in Fig. 2(b), when devices use backbones with differing layer counts (e.g., 12 vs. 24 layers), we partition all models into a fixed number of blocks (e.g., 12 blocks). Activations are then sampled exclusively from the final layer of each block. The server-side shared side-network is configured with an equal number of adapter modules, each processing one aligned activation block. Through comparative experiments of various strategies (Sec. 5.2), we established optimal configuration guidelines: the block count is set to the minimum layer depth among participating models (e.g., 12 blocks for 12/24-layer models), while deeper models are partitioned at uniform intervals (e.g., sampling every other layer in a 24-layer backbone). This approach ensures layer-wise structural consistency while preserving representational capacity.

Hidden Size Scaling. Pre-trained models of different scales exhibit varying hidden sizes, preventing direct integration of their activations into a unified side-network adapter module. To resolve this, we introduce dedicated trainable lightweight bottleneck projection modules for each backbone model type, mapping activations to a consistent hidden size for shared side-network processing. The trainable bottleneck projection module can be formulated as $P = L_1 L_2$, where $P \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, $L_1 \in \mathbb{R}^{d_{\text{in}} \times d_r}$ and $L_2 \in \mathbb{R}^{d_r \times d_{\text{out}}}$.

As shown in Fig. 2 (b), three distinct backbone models (hidden sizes: 2048, 1024, 768) require dimension standardization. We configure the shared side-network adapter modules with a target hidden size of 1024 and deploy projection modules $(p_i^1, \ldots, p_i^{12})$ on the server for each model variant. Consider device 1 using OPT-1.3B (hidden size 2048): its activations $(\mathbf{A}_1^1, \ldots, \mathbf{A}_1^{12})$ pass through corresponding projection modules $(p_1^1, \ldots, p_1^{12}) \in \mathbb{R}^{2048*1024}$, producing transformed activations

($\mathbf{P}_1^1, \ldots, \mathbf{P}_1^{12}$) with uniform 1024-dimensional features. These standardized activations then feed into the shared adapter modules ($s_i^1, \ldots, s_i^{12}$), which enable collaborative training across heterogeneous models. Our empirical study (Sec. 5.2) suggests selecting mid-range dimensions (e.g., 1024 for 768/1024/2048 scenarios) can optimize efficiency-accuracy balance in multi-device scenarios, while prioritizing larger dimensions can preserve representation capacity in scenarios involving only two model sizes.

4 Experimental Setup

Implementation: The experimental testbed consists of a server with the NVIDIA A100 GPU and three types of heterogeneous client devices representing increasing computational capabilities for LLM processing: (1) NVIDIA Jetson TX2 (8GB RAM, 1.3 TFLOPS), (2) NVIDIA Jetson Xavier NX (8GB RAM, 6 TFLOPS peak), and (3) NVIDIA Jetson AGX Xavier (16GB RAM, up to 10 TFLOPS peak).

Models: To systematically evaluate EC-MobiLLM's performance, we employ two representative pre-trained LLM architectures: i) decoder-based OPT series (OPT-1.3B, OPT-350M, OPT-125M), and ii) encoder-based RoBERTa series (RoBERTa-large(350M) and RoBERTa-base(125M)). This selection ensures architectural diversity while maintaining mobile compatibility. All models are initialized via HuggingFace Transformers [33].

Datasets: We take the GLUE benchmark [29] and DialogSum dataset [6] for the evaluation of NLP tasks, which are widely used in the fine-tuning research for LLM [39, 25, 26]. GLUE benchmark comprises eight tasks, including linguistic acceptability (CoLA [31]), sentiment analysis (SST-2 [22]), similarity and paraphrase (MRPC [8], QQP [15], STS-B [4]), and natural language inference (MNLI [32], QNLI [20], RTE [2]). DialogSum includes summaries of real-world conversations on a diverse set of topics and scenarios to evaluate text-generation tasks. We use ROUGE scores (R1/R2/RL) as the accuracy metric. Following FedPETuning [39], we simulate non-IID data partitions using a Dirichlet distribution with concentration parameter α, where lower α values induce higher label distribution shift. (See Appendix B for dataset details.)

Parameters: Following FedPETuning [39], we set the number of communication rounds to 100 and the number of local training epochs to 1 for all baselines under the FL paradigm. All configurations deploy 100 clients with balanced device-type distribution in heterogeneous settings. Experiments are conducted in a semi-emulation manner, where on-device resource consumption and training time are measured on real devices. For EC-MobiLLM and those centralized fine-tuning baselines, the number of training epochs is set to 20. To ensure fair comparison, all experiments share the same configurations unless specified: FP16 precision, batch size 8, learning rate 5e-4, maximum sequence length 256, and 60 Mbps in-lab Wi-Fi transmission speed. Additionally, LoRA and EC-MobiLLM employ rank-64 low-rank trainable modules by default, while FwdLLM uses 300 global perturbations per iteration.

Baselines for Performance Comparison: We compare EC-MobiLLM with two groups of baselines:

Paradigm baselines: i) **FedPETuning** [39] (hereafter **FL**) ii) **SplitLoRA** [18] (hereafter **SFL**) iii) **FwdLLM** [34], which represent federated learning, split federated learning, and BP-free perturbation training, respectively. Each paradigm baseline is evaluated with two representative PEFT methods: i) **LoRA** [14]: Inserts trainable low-rank matrices into frozen backbone networks. ii) **BitFit** [37]: Fine-tunes exclusively bias terms while freezing other pre-trained weights.

Heterogeneity-aware baselines: **FlexLoRA** [1] and **FLoRA** [30], which support heterogeneous LoRA ranks under a shared backbone while still following the federated learning paradigm for collaboration.

5 Evaluation Results and Analysis

5.1 Comparative Analysis with Federated FT Baselines

We conduct comprehensive experiments to validate EC-MobiLLM's advantages in on-device resource efficiency, training efficiency, and fine-tuning performance across homogeneous and heterogeneous device environments. (See Appendix C for details of result computation.)

Table 2: Comparison across methods: i) Per-device resource efficiency (memory/computation/communication); ii) Fine-tuning accuracy (centralized training vs. distributed training); and iii) Time-to-Accuracy (TTA) under homogeneous (TX2/Xavier/AGX Xavier clusters) and heterogeneous (Mixed) device configurations. Task: RoBERTa-Base@MRPC. *Note: EC-MobiLLM uses uniform backbone models across devices in mixed settings to ensure a fair paradigm-level comparison.*

Methods	On-device memory (GB)	On-device comp. (TFLOPs)	On-device comm. (MB)	FT Performance (Acc.)		TTA@86.5(Mins)			
				Centralized	Distributed	TX2 (Hom.)	Xavier (Hom.)	AGX (Hom.)	Mixed (Heter.)
FL-LoRA	1.18	395.7	31.1	91.5	86.9	54.2	41.6	35.7	49.3
FL-BitFit	1.02	388.1	23.1	90.7	86.8	49.7	39.1	32.2	35.6
SFL-LoRA	0.54	58.3	5437.5	91.5	87.8	782.1	763.2	741.3	777.3
SFL-BitFit	0.49	56.5	5429.4	90.7	87.2	774.2	751.2	722.9	767.9
FwdLLM-LoRA	0.42	407.4	22.8	91.5	87.1	41.2	28.9	25.4	40.3
FwdLLM-BitFit	**0.38**	391.5	16.2	90.7	87.4	32.7	22.9	20.2	30.3
EC-MobiLLM	**0.38**	**2.7**	**1.1**	89.6	**88.1**	**5.7**	**6.0**	**5.1**	**5.9**

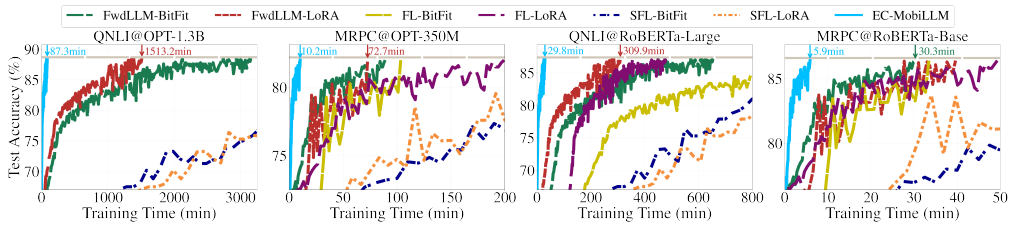


Figure 3: Convergence performance on various models and tasks under heterogeneous-device settings.

On-Device Resource Efficiency. As detailed in Table 2, EC-MobiLLM demonstrates superior resource efficiency across metrics in terms of on-device memory footprint, computation cost, and communication cost. Conventional FL-based PEFT paradigms incur substantial resource demands, while SFL approaches trade computation savings for significantly increased communication overhead (up to $175\times$). Similarly, FwdLLM trades memory savings for higher computation load. In contrast, EC-MobiLLM maintains an optimal balance, where devices perform only a single forward pass, reducing memory consumption to inference levels (e.g., $2.68\times$ reduction for RoBERTa-Base). Besides, it achieves at least 95.2% lower computation and 93.2% less communication by eliminating on-device backpropagation and leveraging server-side activation caching.

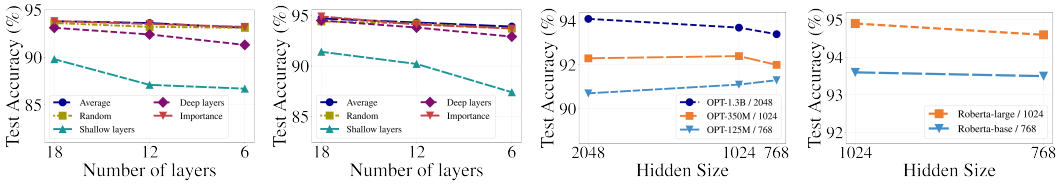
Training Efficiency. We evaluate EC-MobiLLM's training efficiency by measuring time-to-accuracy across diverse configurations, including two model architectures (OPT, RoBERTa) and two tasks (MRPC, QNLI), as shown in Fig. 3. To ensure fair comparison of different collaboration paradigms, all heterogeneous devices use identical backbone models across EC-MobiLLM and other paradigm baselines. The results show that EC-MobiLLM achieves at least a $5.1\times$ speedup across all tasks. This acceleration stems from our full-pipeline efficient design: during the initial phase, clients perform only one forward propagation per data sample, avoiding both on-device backpropagation in conventional FL and multi-pass perturbations in FwdLLM. After receiving activations from all devices, the server performs iterative training on the shared side-network independently, eliminating parameter synchronization with devices inherent in FL paradigms.

We further evaluate training efficiency across different client device setups, as shown in the right portion of Table 2, which validates EC-MobiLLM's superior straggler resilience. Under the heterogeneous-device setting, other collaboration paradigms suffer severe slowdowns due to synchronous waiting periods in federated aggregation protocols, resulting in training times approaching the all-TX2 (lowest-capacity devices) configuration. In contrast, EC-MobiLLM benefits from extremely lightweight on-device computation and non-blocking parallel device-server collaboration. As a result, computational speed variations across devices are effectively masked, yielding consistent training times across diverse client setups.

LLM FT Performance. We evaluate EC-MobiLLM against federated LLM FT methods and their centralized PEFT counterparts (LoRA, BitFit, side-network tuning). As Table 2 shows, while LoRA and BitFit outperform side-network tuning in centralized settings, their accuracy significantly degrades

Table 3: FT Performance under data heterogeneity: IID vs. non-IID ($\alpha = \{0.1, 1.0, 10.0\}$).

Methods	RoBERTa-Base@MRPC (Acc.)				OPT-1.3B@DialogSum (R1/R2/RL)		
	IID	non-IID (α= 10.0)	non-IID (α= 1.0)	non-IID (α= 0.1)	IID	non-IID (α= 10.0)	non-IID (α= 0.1)
FL-LoRA	86.9	85.1	84.5	82.1	19.2 / 6.1 / 15.1	17.9 / 5.2 / 14.3	16.1 / 4.6 / 12.7
FL-BitFit	86.8	84.9	84.6	81.5	19.0 / 6.2 / 14.9	17.7 / 5.1 / 14.2	16.4 / 4.2 / 12.9
SFL-LoRA	87.8	87.1	86.8	84.3	19.9 / 6.5 / 15.8	18.5 / 6.1 / 15.1	18.0 / 5.5 / 13.8
SFL-BitFit	87.2	87.0	86.1	85.2	19.7 / 6.2 / 15.5	18.7 / 6.0 / 15.2	17.8 / 5.4 / 13.4
FwdLLM-LoRA	87.1	86.7	86.5	84.1	19.3 / 6.3 / 15.7	18.2 / 6.2 / 15.1	17.8 / 5.2 / 13.7
FwdLLM-BitFit	87.4	87.1	86.6	84.7	19.6 / 5.9 / 15.1	18.6 / 6.1 / 14.9	17.3 / 5.1 / 14.1
EC-MobiLLM	**88.1**	**87.8**	**87.7**	**87.3**	**21.0 / 7.7 / 17.2**	**20.2 / 7.4 / 16.0**	**20.0 / 7.3 / 16.3**



(a) OPT-1.3B (b) RoBERTa-large (c) OPT Models (d) RoBERTa Models

Figure 4: Comparative accuracy across sampling methods (a-b) and hidden-size scaling (c-d) (Task: SST-2).

under distributed deployment with distributed data. Even the best federated baseline (FwdLLM-BitFit) exhibits at least 3.3% performance drop. In contrast, EC-MobiLLM maintains near-centralized performance with only a 1.5% accuracy drop, outperforming all federated FT baselines by at least 0.3%.

We further evaluate the impacts of data heterogeneity on EC-MobiLLM's performance. To build local datasets, we use three Dirichlet distributions ($\alpha \in \{0.1, 1.0, 10.0\}$) where a lower α indicates a higher non-IID level [39]. Table 3 demonstrates that while the performance of all methods degrades under data heterogeneity, EC-MobiLLM shows superior resilience. Taking the MRPC task as an example, at extreme heterogeneity ($\alpha = 0.1$), EC-MobiLLM's accuracy drop is 1.2% smaller than SFL-BitFit, validating enhanced robustness to cross-client data distribution shifts.

5.2 Cross-Model Alignment Performance

We evaluate the efficacy of our layer-wise activation sampling and hidden size scaling methods for heterogeneous model adaptation through comparative experiments with alternative approaches.

Layer Alignment Design. To determine optimal block configurations for layer alignment in EC-MobiLLM, we perform comparative experiments using OPT-1.3B and RoBERTa-Large models. Specifically, we feed sampled layer-wise backbone activations to the side network and analyze how layer-wise activation sampling strategies affect performance, as shown in Fig. 4 (a-b). We compare five strategies: shallow-only, deep-only, average-interval, random, and importance-based selection (identified via layer-wise ablation). Results indicate: i) Performance improves with more backbone activation layers; ii) The average-interval strategy achieves accuracy comparable to computationally intensive importance-based approaches. These findings support EC-MobiLLM's configuration: set block count to the most lightweight LLM backbone's layer depth across devices, partitioning larger-sized LLMs at equal intervals to ensure structural alignment while maintaining performance.

Hidden Size Alignment Design. For cross-model hidden size scaling, determining a unified dimension for side-network adapters is critical. We experiment with various OPT and RoBERTa models, and evaluate performance under different side-network hidden sizes. As shown in Fig. 4 (c-d), peak performance is achieved when side-network hidden sizes match backbone sizes, while dimensional mismatches degrade accuracy. For example, OPT-125M (hidden size =768) performs worse when forced dimension to 2048, which demonstrates that larger dimensions aren't always beneficial. These findings yield practical EC-MobiLLM configuration guidelines: for multi-device scenarios, select median dimensions (e.g., 1024 for 768/1024/2048); for dual-model scenarios, prioritize larger dimensions to preserve representational capacity.

Table 4: Performance comparison of heterogeneity-aware methods under heterogeneous devices. (*Single*: isolated side-network training on each single device's local data; *Global*: collaborative training of a shared side-network. Task: SST-2.)

Method	Devices	Local Backbone	LoRA Rank (r)	On-device Memory (GB)	Per-device Local Runtime(Mins)	FT Performance (Acc.)	
						Single	Global
FlexLoRA	AGX	OPT-125M	128	2.51	12.7	88.5	89.9
	Xavier	OPT-125M	64	2.48	15.8	88.5	89.7
	TX2	OPT-125M	16	2.46	17.9	88.4	89.4
FLoRA	AGX	OPT-125M	128	2.51	12.7	88.8	90.8
	Xavier	OPT-125M	64	2.48	15.8	88.6	90.7
	TX2	OPT-125M	16	2.46	17.9	88.6	90.4
EC-MobiLLM	AGX	OPT-1.3B	–	3.44	1.53	90.0	92.9
	Xavier	OPT-350M	–	1.10	1.46	89.9	92.4
	TX2	OPT-125M	–	0.56	1.41	88.4	92.1

5.3 Validation of Heterogeneous Backbone Adaptation

To validate the efficacy and necessity of EC-MobiLLM's heterogeneous-backbone design, we conduct systematic experiments. In EC-MobiLLM, each device loads a backbone model tailored to its hardware capabilities. In contrast, existing heterogeneous FL baselines are constrained to a single shared backbone (determined by the weakest device) and can only adjust the LoRA rank.

Device-Specific Workload Balancing. As shown in the results of the on-device workload in Table 4, device-specific backbone assignment optimizes hardware resource utilization compared to uniform model deployment. For example, AGX Xavier and Xavier run OPT-1.3B and OPT-350M, respectively, with memory usage both around 30% of their available capacities (12.4 GB and 4.6 GB). Computational loads similarly scale to device capacities, resulting in comparable execution times across heterogeneous devices. This helps ensure near-balanced contributions to side-network training and prevent representation drift toward data from those fast devices. In contrast, devices still exhibit pronounced cross-device imbalance in FlexLoRA and FLoRA.

Cross-Capacity Collaboration Effects. We investigate whether collaboration with lower-capacity devices in EC-MobiLLM compromises high-capacity device performance. To this end, we evaluate performance under two settings: i) training the side network using only each device's local data (denoted by *Single*), and ii) collaboratively training a shared side network across all devices (denoted by *Global*). As shown in Table 4, *Global* consistently outperforms *Single* across different backbone sizes, indicating that EC-MobiLLM enables all devices to benefit from a robust shared side network trained collaboratively on diverse cross-device data. In addition, compared to FlexLoRA and FLoRA, a high-resource device (e.g., AGX) can achieve significantly better performance for both itself and its collaborating devices when using a larger backbone model. This highlights the importance of supporting device-adaptive backbone selection in collaborative LLM fine-tuning.

6 Conclusion

This paper has presented EC-MobiLLM, an efficient framework for collaborative LLM fine-tuning across heterogeneous mobile devices. By pioneering an asynchronous server-assisted side-tuning paradigm, EC-MobiLLM decouples device responsibilities to forward-only propagation and activation uploading, while the server trains a shared side-network, which eliminates synchronization bottlenecks inherent in conventional FL-based fine-tuning approaches. Through layer-wise activation sampling and cross-architecture dimension alignment, EC-MobiLLM enables each device to load backbone models that match its hardware capacities while still maintaining the ability to collaboratively train a shared side network, ensuring robust support for device heterogeneity. Extensive experiments demonstrate EC-MobiLLM's efficacy and efficiency: achieving $2.68\times$ reduction in on-device memory usage, 95.2% reduction in computational cost, 93.2% lower communication overhead, and $5.1\times$ faster convergence compared to existing methods, while maintaining competitive accuracy. Additional discussions on server-side cache storage, network-speed sensitivity, and privacy considerations for transmitted activations are provided in Appendix A. These results collectively establish EC-MobiLLM as a practical and deployment-ready solution for real-world collaborative LLM fine-tuning on distributed mobile datasets.

Acknowledgments and Disclosure of Funding

Funding: The work of X. Yang and M. Pan were supported in part by the US National Science Foundation under grants CNS-2318664, CSR-2403249 and CNS-2431596. The work of X. Yang and X. Fu were supported in part by the US National Science Foundation under grants CNS-2107057, SSF-2504152. The work of L. Li was supported in part by the Natural Science Foundation of Guangdong Province under Grant 2026A1515011679, the Basic and Frontier Research Project of PCL under Grant 2025QYB014. The work of H. Wang at Stevens Institute of Technology was supported in part by the NSF under grants 2523997, 2534286, and 2332638.

Competing Interests: The authors declare no competing interests.

References

[1] Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous tasks and client resources. *Advances in Neural Information Processing Systems*, 37:14457–14483, 2024.

[2] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1, 2009.

[3] 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.

[4] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[5] Xiaopei Chen, Liang Li, Fei Ji, and Wen Wu. Memory-efficient split federated learning for llm fine-tuning on heterogeneous mobile devices. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2025.

[6] Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. Dialogsum: A real-life scenario dialogue summarization dataset. *arXiv preprint arXiv:2105.06762*, 2021.

[7] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 12903–12913, 2024.

[8] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.

[9] Wenliang Du and Mikhail J Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms*, pages 13–22, 2001.

[10] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[11] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

[12] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[13] 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 *Proc. of 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, June 2019.

[14] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

[15] Shankar Iyer, Nikhil Dandekar, Kornél Csernai, et al. First quora dataset release: Question pairs. *data. quora. com*, 2017.

[16] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International conference on machine learning*, pages 5201–5212. PMLR, 2021.

[17] Liang Li, Xingke Yang, Wen Wu, Hao Wang, Tomoaki Ohtsuki, Xin Fu, Miao Pan, and Xuemin Shen. Mobillm: Enabling llm fine-tuning on the mobile device via server assisted side tuning. *arXiv preprint arXiv:2502.20421*, 2025.

[18] Zheng Lin, Xuanjie Hu, Yuxin Zhang, Zhe Chen, Zihan Fang, Xianhao Chen, Ang Li, Praneeth Vepakomma, and Yue Gao. Splitlora: A split parameter-efficient fine-tuning framework for large language models. *arXiv preprint arXiv:2407.00952*, 2024.

[19] 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.

[20] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 2383–2392, 2016.

[21] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Representation learning with large language models for recommendation. In *Proceedings of the ACM Web Conference 2024*, pages 3464–3475, 2024.

[22] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[23] Timothy Stevens, Christian Skalka, Christelle Vincent, John Ring, Samuel Clark, and Joseph Near. Efficient differentially private secure aggregation for federated learning via hardness of learning with errors. In *31st USENIX security symposium (USENIX Security 22)*, pages 1379–1395, 2022.

[24] Huai-an Su, Pavana Prakash, Rui Chen, Yanmin Gong, Rong Yu, Xin Fu, and Miao Pan. Dafl: Device-to-device transmissions for delay efficient federated learning over mobile devices. *IEEE Internet of Things Journal*, 2024.

[25] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. *arXiv preprint arXiv:2403.12313*, 2024.

[26] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, Louisiana, December 2022.

[27] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. Fedbert: When federated learning meets pre-training. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022.

[28] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. Nopeek: Information leakage reduction to share activations in distributed deep learning. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 933–942. IEEE, 2020.

[29] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: a multi-task benchmark and analysis platform for natural language understanding. corr abs/1804.07461 (2018). *arXiv preprint arXiv:1804.07461*, 2018.

[30] Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *Advances in Neural Information Processing Systems*, 37:22513–22533, 2024.

[31] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[32] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[33] 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.

[34] Mengwei Xu, Dongqi Cai, Yaozong Wu, Xiang Li, and Shangguang Wang. {FwdLLM}: Efficient federated finetuning of large language models with perturbed inferences. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 579–596, 2024.

[35] Xingke Yang, Liang Li, Zhiyi Wan, Sicong Li, Xiaoqi Qi, Jiang Liu, Tomoaki Ohtsuki, Xin Fu, and Miao Pan. Pae mobillm: Privacy-aware and efficient llm fine-tuning on the mobile device via additive side-tuning. *arXiv preprint arXiv:2507.01216*, 2025.

[36] Qinyuan Ye. Cross-task generalization abilities of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pages 255–262, 2024.

[37] 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.

[38] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[39] Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Annual Meeting of the Association of Computational Linguistics 2023*, pages 9963–9977. Association for Computational Linguistics (ACL), 2023.

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

A Further Experimental Results and Discussion

A.1 Server Storage

Table 5: Server cache size across backbones and tasks (homogeneous Xavier clusters).

Backbone	Server cache size (GB)		
	RTE	SST2	MNLI
OPT-1.3B	0.23	6.21	36.77
RoBERTa-Base	0.08	2.40	14.21

The major focus of device-server collaborative LLM fine-tuning works is to reduce computation, memory, and/or communication costs on resource-constrained mobile devices. Most state-of-the-art studies [17, 35, 34] assume that the server has sufficient resources. We follow a similar setting and prioritize the analysis of task performance and system efficiency from the mobile-device side in the main paper.

Here, we provide quantitative measurements of EC-MobiLLM's server-side cache usage across different backbone sizes (from RoBERTa-Base to OPT-1.3B) and dataset scales (from RTE to MNLI). As shown in Table 5, under the 1.3B model setting, the server-side cache is about 36.77 GB for the largest dataset MNLI (about 393K examples), while it is only about 0.23 GB for a small-scale dataset such as RTE (about 2K examples). Since each device uploads only a small amount of selected activations, the storage overhead introduced by server-side caching does not make EC-MobiLLM impractical, supporting its deployability. Given current mobile memory constraints and the fact that per-user datasets are usually relatively limited, these results indicate that, in our target scenario, the server-side storage cost is acceptable and does not raise serious feasibility concerns for the current design.

However, when targeting larger backbones or higher data throughput, edge servers with tight storage budgets may require additional operational mechanisms, such as cache lifecycle management, eviction policies, and compression. We view these as meaningful directions for future work.

A.2 Impact of Network Speed on Training Efficiency

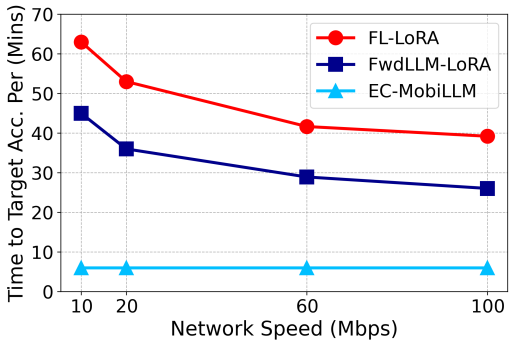


Figure 5: Impact of network speed on time-to-accuracy (homogeneous Xavier clusters, task: RoBERTa-Base@MRPC).

In practical collaborative LLM fine-tuning, device-server communication makes training efficiency sensitive to network conditions. Consequently, network fluctuations may impact overall training efficiency. Figure 5 illustrates the training efficiency across methods under different wireless transmission speeds. Since the total training time in FL-LoRA and FwdLLM-LoRA is affected by the communication time, they suffer severe performance degradation under low-speed transmission conditions. For example, FwdLLM-LoRA shows a $1.62\times$ slowdown at 10Mbps vs. 100Mbps. In contrast, EC-MobiLLM maintains stable total latency across different transmission speeds, demonstrating almost no variations from 10 Mbps to 100 Mbps. Such resilience stems from parallel scheduling that

overlaps communication with device- and server-side computation, masking transmission delays with on-device compute time that is unaffected by network-speed fluctuations.

A.3 Privacy Discussion on Activation Transmission

In this paper, the main contribution of EC-MobiLLM is to provide a collaborative LLM fine-tuning framework for heterogeneous mobile devices. It focuses on reducing on-device computation, memory, and communication cost, improving training efficiency, and enabling cross-backbone collaboration across devices. EC-MobiLLM involves the transmission of intermediate activations during training. This is aligned with the privacy assumptions commonly adopted in split learning related architectures [27] and federated learning (FL) [19] systems, where devices transmit gradients/activations instead of raw user text data, thereby avoiding the direct exposure of original user data.

However, when data security is strictly considered, prior studies [40, 11, 28] have shown that gradients or activations, as representations derived from local data, may still be used to infer original input information. To mitigate such risks, existing works [16, 23] have proposed additional protection mechanisms, such as differential privacy (DP) mechanisms [10] and secure multiparty computation (SMC) [9]. DP mechanisms can be applied by adding calibrated Gaussian or Laplace noise to uploaded gradients, activations, outputs, or objective functions, thereby reducing the risk of privacy leakage from transmitted information. SMC can be used as a complementary cryptographic mechanism to further protect the communication and computation process. These protection mechanisms can also be applied to EC-MobiLLM's activation transmission.

Compared with conventional FL/SFL-style methods, EC-MobiLLM has a favorable property from the perspective of privacy-loss composition. Conventional methods usually require multiple communication rounds, where repeated releases of noisy gradients or activations lead to accumulated privacy loss. In contrast, EC-MobiLLM uploads the selected activations for each local sample only once, and the server subsequently reuses the cached activations for iterative training. This reduces repeated data-dependent releases and can mitigate privacy-loss accumulation. Under the same overall privacy budget, one-time uploading can therefore reduce the required noise level and better preserve fine-tuning performance.

To evaluate this property, we conduct a sample-level DP experiment on SST-2 under the heterogeneous-device setting. We compare EC-MobiLLM without privacy protection, one-shot caching that uploads privatized activations once and reuses them for 20 training epochs, and a repeated-release variant that independently perturbs and uploads activations in every epoch. Both private configurations use the same overall DP budget $(\epsilon, \delta) = (4, 10^{-5})$.

Table 6: Fine-tuning performance under differential privacy on SST-2 across heterogeneous mobile devices. Accuracy is reported in percentage (%) ↑.

Method	Releases per Sample	DP Budget	AGX (OPT-1.3B)	Xavier (OPT-350M)	TX2 (OPT-125M)
EC MobiLLM (Non-private)	–	–	92.9	92.4	92.1
EC MobiLLM (One-shot Cache)	1	$(4, 10^{-5})$	89.2	88.5	88.1
Repeated Releases Variants	20	$(4, 10^{-5})$	86.5	85.3	84.4

As shown in Table 6, introducing DP protection leads to an average accuracy reduction of about 3.9 percentage points for EC-MobiLLM with one-shot caching compared to the non-private setting. However, under the same overall DP budget, one-shot caching consistently outperforms repeated releases by 2.7–3.7 percentage points across the three devices. These results demonstrate that EC-MobiLLM can incorporate DP protection while reducing the impact of privacy-preserving noise on fine-tuning performance through one-time activation uploading and server-side cache reuse.

A.4 Ablation Study of Key Components

To evaluate the contribution of each key component in EC-MobiLLM, we conduct an ablation study under the same experimental setting as Table 2. Specifically, we separately remove token selection (TS), server caching (SC), and non-blocking asynchronous collaboration (NBAC) to examine their effects on system efficiency and fine-tuning performance.

Table 7: Ablation study of the key components in EC-MobiLLM (RoBERTa-Base@MRPC, heterogeneous devices). TTA is measured at 86.5% accuracy.

Method	On-device Memory (GB)↓	On-device Comp. (TFLOPs)↓	On-device Comm. (MB)↓	FT Acc. (%)↑	TTA (min)↓
EC-MobiLLM	0.38	2.7	1.1	88.1	5.9
w/o. TS	0.38	2.7	281.6	88.1	45.7
w/o. SC	0.38	54.0	22.0	88.1	91.3
w/o. NBAC	0.38	2.7	1.1	88.1	9.6

As shown in Table 7, removing TS significantly increases communication cost and training time due to the transmission of full activations. Without SC, devices need to repeat forward propagation and activation transmission across training epochs, resulting in $20\times$ higher computation and communication costs. Removing NBAC increases TTA from 5.9 to 9.6 minutes due to synchronization delays. Meanwhile, all configurations maintain the same on-device memory usage and fine-tuning accuracy. These results demonstrate that the three components jointly improve the communication, computation, and training efficiency of EC-MobiLLM without sacrificing fine-tuning performance.

A.5 Inference-Time Overhead

To evaluate the inference overhead introduced by EC-MobiLLM, we compare backbone-only inference with inference using the additional projection modules and shared side-network on three heterogeneous mobile devices. We measure the inference latency, peak on-device memory, and computation cost under both configurations.

Table 8: Inference-time overhead of EC-MobiLLM on heterogeneous mobile devices. "+ Modules" denotes adding the projection modules and shared side-network to the backbone. The relative increases are shown in parentheses.

Device / Backbone	Configuration	Latency (s/batch)↓	Peak Memory (GB)↓	Comp. (TFLOPs/batch)↓
TX2 / OPT-125M	Backbone Only	0.98	0.56	0.36
	Backbone + Modules	1.03 (+5.1%)	0.57 (+1.8%)	0.37 (+2.8%)
Xavier NX / OPT-350M	Backbone Only	1.07	1.10	1.29
	Backbone + Modules	1.12 (+4.7%)	1.12 (+1.8%)	1.32 (+2.3%)
AGX Xavier / OPT-1.3B	Backbone Only	1.89	3.44	5.05
	Backbone + Modules	1.95 (+3.2%)	3.48 (+1.2%)	5.20 (+3.0%)

As shown in Table 8, the additional projection modules and side-network increase inference latency by only 3.2%–5.1%, peak on-device memory by 1.2%–1.8%, and computation cost by 2.3%–3.0% across the three devices. These results demonstrate that EC-MobiLLM introduces limited inference overhead while supporting collaborative fine-tuning across heterogeneous backbones.

B Datasets Statistics

Table 9: Datasets Statistics.

Dataset	**Description**	**Task**	**# Samples (train/validation)**
CoLA	Linguistic Acceptability	Classification	8551 / 1043
SST2	Sentiment Analysis	Classification	67349 / 872
MRPC	Sentence Equivalence	Classification	3668 / 408
STSB	Sentence Similarity	Regression	5749 / 1500
QQP	Paraphrase Recognition	Classification	363846 / 40430
RTE	Textual Entailment	Classification	2490 / 277
QNLI	Natural Language Inference	Classification	104743 / 5463
MNLI	Textual Entailment	Classification	392702 / 9815 (9832)
DialogSum	Abstract Summary	Generation	12460 / 500

C Computation of Reported Results

In this section, we use Table 2 as an example to present how we compute *on-device resource overhead* and *training efficiency* metrics for EC-MobiLLM and other baselines. We consider a setting with 100 homogeneous NVIDIA Xavier clients, each of which loads the same RoBERTa-Base backbone. The unified experimental setup is as follows: Dataset is evenly distributed across 100 clients according to the standard federated learning configuration. FP16 precision, batch size = 8, sequence length = 256, and an in-lab Wi-Fi throughput of 60 Mbps.

C.1 On-Device Memory footprint

- **FL-LoRA:** Performs full LoRA fine-tuning on the RoBERTa model on the device. The memory footprint consists of model weights + intermediate activation + optimizer states and hence is the largest among all methods.
- **SFL-LoRA:** To keep raw data and labels on the device, only offload the middle 10 layers of the 12-layer transformer to the server; the remaining layers are LoRA fine-tuned on the device. The memory footprint consists of partial weights + partial intermediate activation + partial optimizer states, thus lower than FL-LoRA.
- **FwdLLM-LoRA and EC-MobiLLM:** The device only performs forward propagation. The memory footprint consists of model weights + a small inference-time intermediate activation, achieving the smallest memory footprint.

C.2 On-Device computation

We report on-device computation as the total FLOPs executed on a single device over the entire training process until it reaches the target accuracy. Let the number of global rounds be R, the number of local iterations (epochs/passes) per round be E, and let $\mathrm{Cost}_{\text{local}}$ denote the computation for one complete local epoch under a given method.

- **FL-LoRA:** Each local epoch performs the full RoBERTa + LoRA (forward & backward).
$$\mathrm{Computation}_{\text{device}} \;=\; \mathrm{Cost}_{\text{FT(full backbone, LoRA)}} \times R \times E.$$
- **SFL-LoRA:** Only the non-offloaded layers performs LoRA on device (forward & backward).
$$\mathrm{Computation}_{\text{device}} \;=\; \mathrm{Cost}_{\text{FT(partial backbone, LoRA)}} \times R \times E.$$
- **FwdLLM-LoRA (forward-only with perturbations):** The device performs only forward propagation; each round uses K local perturbation forwards.
$$\mathrm{Computation}_{\text{device}} \;=\; \mathrm{Cost}_{\text{Infer(full backbone)}} \times R \times K.$$
- **EC-MobiLLM:** The device performs a single forward propagation traversal of its local data (no local replay).
$$\mathrm{Computation}_{\text{device}} \;=\; \mathrm{Cost}_{\text{Infer(full backbone)}}\,.$$

C.3 On-Device Communication

We report on-device communication as the total amount of data a single device exchanges with the server over the entire training process until it reaches the target accuracy. Let the number of global rounds be R, the number of local iterations (epochs/passes) per round be E.

- **FL-LoRA:** In standard FL, devices and server exchange trainable parameters in both directions (upload/downlink) each global round:
$$\mathrm{Communication}_{\text{device}} = 2 \times R \times \left|\theta_{\text{LoRA}}\right|,$$
where $\left|\theta_{\text{LoRA}}\right|$ is the size of the on-device LoRA trainable parameters.
- **SFL-LoRA:** Beyond the round-wise parameter exchange, split learning requires frequent exchange of intermediate activations and backward gradients during local forward/backward:
$$\mathrm{Communication}_{\text{device}} = 2 \times R \times \left|\theta_{\text{LoRA}}\right| \;+\; R \times E \times \mathrm{Comm}_{\text{act/grad-per epoch}},$$
where $\mathrm{Comm}_{\text{act/grad-per epoch}}$ denotes the activation/gradient traffic for one local forward+backward epoch.

- **FwdLLM-LoRA.** Similar to FL-LoRA, the standard communication rhythm for federated learning:
$$\text{Communication}_{\text{device}} = 2 \times R \times \left|\theta_{\text{LoRA}}\right|.$$
- **EC-MobiLLM:** The device performs a single forward propagation traversal of its local data and uploads only a small subset of layer-wise activations selected by the token selector; there is no parameter round-trip or gradient return:
$$\text{Communication}_{\text{device}} \approx \text{Comm}_{\text{selected-activations- single epoch}}.$$

C.4 Training Time to Target Accuracy

We report the training time to target accuracy for each LLM federated FT framework. Let the number of global rounds be R, and define: t_1 — per-round on-device runtime, t_2 — per-round device–server parameter communication time, t_3 — per-round server aggregation time.

For **FL-LoRA / SFL-LoRA / FwdLLM-LoRA**, the per-round steps run *serially*, so
$$T_{\text{train}} = (t_1 + t_2 + t_3) \times R.$$

The only difference lies in t_1:

- **FL-LoRA:** $t_1 =$ on-device LoRA fine-tuning (forward + backward).
- **SFL-LoRA:** $t_1 =$ split-learning based device–server co-training; frequent activation/gradient exchange makes t_1 typically longer.
- **FwdLLM-LoRA:** $t_1 =$ on device multiple forward propagations.

EC-MobiLLM does not follow a federated sequential rhythm. Instead, it overlaps single forward propagation time on the device side with communication time and server iteration time in parallel. The total training time approximates the maximum duration among these three components. In our testing, the server iteration typically dominates, i.e., $T_{\text{train}} \approx t_{\text{server-iter}}$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize the paper's main contributions: reducing on-device overhead, enabling asynchronous collaboration, and supporting collaborative fine-tuning across heterogeneous backbones.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses practical deployment limitations in Appendix A, including server-side cache storage, network-speed sensitivity, and privacy considerations for activation transmission.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results, formal theorems, or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper discloses the experimental testbed, models, datasets, baselines, data partitioning method, and training hyperparameters. Appendix C also provides the computation procedures for the main resource and training-efficiency metrics.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We do not release the experimental code at submission time. The paper uses publicly available datasets and provides detailed experimental settings and metric computation procedures.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the main experimental settings needed to understand the results, including hardware platforms, model backbones, datasets, data partitioning strategy, baselines, number of clients, training epochs, batch size, learning rate, sequence length, precision, and network speed.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper does not report error bars or statistical significance tests. Instead, we evaluate the method across multiple models, tasks, device configurations, and data heterogeneity levels to demonstrate consistent trends under different settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper reports the main compute platforms used in the experiments, including the server GPU and heterogeneous Jetson devices, and reports execution-time metrics such as time-to-accuracy. However, it does not provide a complete estimate of the total compute required for all experimental runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: There is no societal impact of the work performed.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper uses publicly available datasets, models, and software packages. The original creators are cited

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The research does not use LLMs as an external tool for core method development, data generation, annotation, or evaluation.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.