On Harnessing Idle Compute at the Edge for Foundation Model Training

Leyang Xue, Eren Mendi, Ismaeel Bashir and Mahesh K. Marina

The University of Edinburgh

Foundation models (FMs) have become central to modern AI, powering breakthroughs in language, vision, biology, networking, and software. Models like GPT-4, Stable Diffusion, and AlphaFold are trained on massive datasets and fine-tuned for diverse downstream tasks. As demand grows for domain specific FMs such as DeepSeek-Coder so does the computing need for training them. Even with techniques like LoRA and quantized pretraining (e.g., DeepSeek-R1), training these models still requires near-terabyte GPU memory and costly infrastructure, accessible only to a few large cloud providers, steering us further towards an already centralized AI ecosystem.

Given this backdrop, foundation model training using edge devices presents a decentralized alternative to the traditional cloud-based approach. A particularly compelling opportunity is to leverage volunteered compute from edge devices like laptops and smartphones à la SETI@home. These devices offer growing AI capabilities, are designed to be energy efficient and have high idle time, making them viable for collaborative training among such edge devices and thereby reduce the cloud dependence. Consumer hardware such as MacBooks are capable of model finetuning, with Apple's M4 Max delivering 38 TFLOPS, within an order of magnitude of datacenter GPUs. With edge devices staying in idle state majority of the time (as high as 75% of the time) including while charging, this collective compute pool can rival that of the centralized cloud compute infrastructure.

Decentralized foundation model training that harnesses idle compute on edge devices, while promising, needs to meet several key requirements in order to be viable. These include: (i) it must deliver training performance comparable to cloud based training in terms of time-to-train, while preserving model accuracy by supporting standard architectures, optimizers, and hyperparameters (e.g., batch size, sequence length); (ii) it must scale effectively with both model and dataset size to provide the benefits predicted by modern scaling laws; (iii) it must operate reliably over heterogeneous and highly dynamic edge devices to ensure that training performance remains robust.

However, current decentralized training methods fall short of satisfying the core requirements outlined above. Even with access to an unlimited number of edge devices, existing approaches are unable to match the performance and efficiency of cloud-based GPU training. While edge-oriented optimizations such as gradient compression, asynchronous updates, or homographic computation can mitigate some bottlenecks, they often come at the cost of reduced model accuracy and architectural compatibility. Fundamentally, existing methods have two key limitations: (i) High communication overhead: The use of peer-to-peer communication for data parallelism or pipeline parallelism results in excessive bandwidth demands due to frequent synchronization operations, making it impractical for low-bandwidth, high-latency environments typical of edge networks. (ii) Coarse-grained fault tolerance: Existing systems rely on global checkpoints or complete layer recomputation, which are ill-suited for highly dynamic settings where devices frequently join or leave. These approaches are not scalable and introduce significant delays under device churn.

We propose a novel selective hybrid tensor parallelism (TP) technique, along with a system that flexibly leverages the scale of edge devices while accounting for their resource constraints and dynamic characteristics. Unlike prior peer-to-peer communication based schemes, we adopt a parameter server (PS) centric framework, where the PS acts as a central aggregator for synchronization. This design eliminates redundant data exchange among devices and reduces communication complexity. To adapt tensor parallelism to the edge setting without sacrificing accuracy, our key insight is that FM training computation is dominated by generalized matrix multiplication (GEMM) operations and that those GEMM operations can be partitioned into smaller sub-matrix level operations, such that each device is only responsible for computing a minimal amount of units. This dramatically lowers per-device memory requirements and significantly reduces communication latency. Each sub-matrix becomes a self-contained unit of work, which can be independently scheduled, executed, and streamed back to the PS asynchronously. Moreover, this granularity enables fine-grained fault tolerance, as completed sub-matrix results can be checkpointed and verified independently. Each unit of computation can be validated upon return, ensuring robustness in the face of device churn or failures while maintaining high throughput and consistency in the overall training process.

We evaluate our proposed training method against SOTA edge (e.g., DTFM) and cloud based (e.g., Alpa) training methods through simulation initially, considering training two foundation language models of various sizes of OPT and Llama2. Our results show that the proposed method not only enables training of larger models than SOTA edge training methods but also achieves the same training runtime as that of a cloud GPU. Furthermore, it is 4-10X faster than the current edge training methods. In addition, the design of our system leveraging TP and PS-based architecture allows it to scale to thousands of devices, supporting 2-8X more devices than the baseline methods. We then perform real world validation with a testbed with small number of smartphones as well as larger scale realistic evaluation through emulation. For the latter, we built a custom device emulator with a measurement-based performance model to reflect the runtime accurately, together with a real backend of parameter server. We have designed the interaction with the PS to allow seamless use of either real or emulated devices.