


PHOTON: FEDERATED LLM PRE-TRAINING

Lorenzo Sani [1 2] **Alexandru-Andrei Iacob*** [1 2] **Zeyu Cao*** [1] **Royson Lee** [1] **Bill Marino** [1] **Yan Gao** [1 2]
Dongqi Cai [1 3] **Zexi Li** [1 4] **Wanru Zhao** [1] **Xinchi Qiu** [1] **Nicholas D. Lane** [1 2]

ABSTRACT

Scaling large language models (LLMs) demands extensive data and computing resources, which are traditionally constrained to data centers by the high-bandwidth requirements of distributed training. Low-bandwidth methods like federated learning (FL) could enable collaborative training of larger models across weakly connected GPUs or weakly connected clusters of GPUs if they can effectively be used for pre-training. Building robust low-bandwidth training systems can: (a) significantly reduce communication infrastructure costs, (b) minimize the impact of hardware failures, (c) widen the pool of usable GPUs, (d) enable collaborative training over the internet, and (e) allow dynamic compute sourcing based on factors like electricity prices. Such advancements would lessen the dependence on specialized data centers, making large-scale AI training more accessible, cost-effective, and adaptable to real-time demands. To achieve this, we introduce Photon, the first complete system for federated end-to-end LLM training, leveraging cross-silo FL for global-scale training with minimal communication overheads. Using Photon, we train the first federated family of decoder-only LLMs from scratch. We show that: (1) Photon can train model sizes up to 7B in a federated fashion while reaching an even better perplexity than centralized pre-training; (2) Photon model training time decreases with available compute, achieving a similar compute-time trade-off to centralized; and (3) Photon outperforms the wall-time of baseline distributed training methods by 35% via communicating $64\times$–$512\times$ less. Our proposal is robust to data heterogeneity and converges twice as fast as previous methods like DiLoCo. This surprising data efficiency stems from a unique approach combining small client batch sizes with extremely high learning rates, enabled by federated averaging's robustness to hyperparameters. Photon thus represents the first economical system for global internet-wide LLM pre-training.

1 INTRODUCTION

Trends in developing state-of-the-art large language models (LLMs) suggest training ever-larger models on expanding datasets with growing compute resources (Kaplan et al., 2020; Jiang et al., 2024). The standard approach involves using a mix of distributed learning algorithms with high-bandwidth communication requirements deployed in single data center, e.g., distributed data parallelism across racks, pipeline parallelism across servers in a rack, and tensor parallelism across GPUs in a server (Rasley et al., 2020; Lee et al., 2024a; Qian et al., 2024; Dubey et al., 2024). Thus, increasing model size requires extending computing facilities to exploit high-bandwidth distributed training algorithms (Hu et al., 2024; Yang et al., 2024).

Recently, a small but growing interest in low-bandwidth distributed training algorithms has developed to exploit the worldwide distribution of computing facilities connected through the Internet (Douillard et al., 2023; Tang et al., 2024; Borzunov et al., 2024; Mi et al., 2020; Chang et al., 2023; Sani et al., 2024). If successful, low-bandwidth distributed training could overcome the need to build more extensive data centers. The federated learning (FL) approach (McMahan et al., 2017) is appealing as an additional layer of parallelism across poorly connected nodes, such as data centers distributed in different regions (Douillard et al., 2023; Nous Research, 2024; He et al., 2024; Marfoq et al., 2020) or internet-wide collaborative training. The reasons for its appeal are three-fold: (1) FL optimizers derive from LocalSGD (Stich, 2019), which allows more infrequent synchronization compared to distributed data parallelism, reducing demands on the communication infrastructure; (2) the size of text datasets makes it challenging to replicate across data centers (Choudhury et al., 2024), which can be alleviated by bringing training to the data; and (3) federations scale seamlessly as participants join (Xu et al., 2024), i.e., as the total available compute expands, without needing to build additional costly infrastructure or reconfigure existing systems.

*Equal contribution [1]Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom [2]Flower Labs [3]Beijing University of Posts and Telecommunications [4]Zhejiang University. Correspondence to: Lorenzo Sani <ls985@cam.ac.uk>.

Proceedings of the 8^{th} MLSys Conference, Santa Clara, CA, USA, 2025. Copyright 2025 by the author(s).

In this work, we present Photon, the first open-source FL system for executing pre-training of LLMs across a distributed setting - composed of individuals privately owing a handful of hardware accelerators - communicating through the Internet. We show that Photon effectively navigates the trade-off between performance and efficiency and fills the gap for researchers and practitioners to federatedly pre-train high-performance LLMs off the shelf. Notably, Photon has been used for pre-training the *first* family of federated decoder-only large language models *from scratch*, scaling model size up to 7B. Moreover, academic and industry researchers have used Photon to execute 1811 experiments and submit six papers to international machine-learning venues. We built Photon on the `Flower` framework, and the code will be made publicly available.

The contributions of this work are the following:

1. We introduce Photon, the first open-source system for federated LLM pre-training over the Internet, enabling collaboration across private GPUs or distributed subsets of data centers worldwide. Photon has successfully trained the first federated family of decoder-only LLMs from scratch, reaching lower perplexities than centralized training for models up to 7B parameters.
2. We show that Photon achieves up to 20% higher throughput (samples/sec) than centralized distributed training, requiring $64\times$–$512\times$ less communication. Furthermore, Photon is significantly more robust than standard data-parallel approaches since workers can continue training if another participant in the federation fails.
3. We propose a novel federated pre-training approach exploiting the robustness to hyperparameters of federated averaging to combine small device batch sizes with high learning rates. This allows models trained with Photon to converge twice as fast as previous methods, such as DiLoCo (Douillard et al., 2023).
4. By combining high throughput with our optimization method, we demonstrate that the training time with Photon reduces as more compute resources are added up to a certain batch size limit. This achieves a compute-time trade-off similar to centralized pre-training.

Photon enables scaling pre-training infrastructure to a global scale. Beyond allowing a potentially unprecedented number of GPUs to train a model effectively and robustly collaboratively, it could be used for novel applications. For example, distributing compute across nodes based on energy costs, carbon emissions, a fairness policy, or the proportion of unique private data.

2 LLM PRE-TRAINING UNDER DECENTRALIZED SYSTEM CONDITIONS

We focus on low-bandwidth distributed settings for training LLM. In such scenarios, the standard distributed training method of synchronizing gradients at every batch step would incur very high overheads. For example, the inter-worker communication costs of using distributed data parallelism (DDP) with *Ring-AllReduce* (Sergeev & Balso, 2018) would be $\mathcal{O}(|\theta| \times T)$ where $|\theta|$ is the model size and T is the number of training steps. Moving to federated training allows us to reduce the communications costs (Stich, 2019; Kairouz et al., 2021) to $\mathcal{O}(|\theta| \times \frac{T}{T_{\text{local}}})$ by performing T_{local} steps on a "worker" prior to synchronization.

While highly beneficial from a communication perspective, infrequent synchronization significantly alters the optimization procedure (Stich, 2019; Ortiz et al., 2021; Lin et al., 2020). Every gradient descent step in standard distributed training is executed on a fully up-to-date model. At the same time, in federated learning, participants operate with stale parameters for T_{local} steps. This poses challenges to reaching a similar level of data efficiency as standard distributed training (Wang et al., 2021). In addition to this fundamental challenge, various FL settings can pose additional difficulties, such as hardware and data heterogeneity (Kairouz et al., 2021).

2.1 Emerging Decentralized Scenarios

Our design space has three particular settings for which low-bandwidth training methods are likely beneficial.

Cross Data-center: This scenario resembles standard distributed training, where multiple data centers collaborate to train models even larger than the current SOTA. Typically, distributed training approaches cannot operate over the low-bandwidth connection across data centers, forcing corporations to build ever-larger facilities.

Cross-silo: Here, we assume collaboration among several small organizations, each equipped with one to eight high-performance accelerators. In such cases, not only is the bandwidth across silos low, but silos may have an insufficient number of GPUs to saturate the batch-size requirements of even modest-sized models.

Collaboration via Commodity Hardware: In this setup, individuals with a small number of consumer-grad GPUs collaborate in model training. This setting presents the above challenges in a harsher form due to strong VRAM constraints on commodity GPUs, making it difficult even to train a model without extreme CPU offloading.

Collaborating in such federated scenarios aims to benefit from *more computing power and data sources* by *achieving the machine learning (ML) optimization objective* quicker

than what standard training can do in a single location. Given our available resources, we focus on the standard cross-silo setting. Thus, we *assume* that every participant possesses the following minimal requirements: (a) one or many well-connected hardware accelerators, which can be sporadically available throughout a full training cycle; (b) sufficient memory to train the full model with a pre-defined small (local) batch size; (c) access to a pre-tokenized text corpus, either stored in the same facility or streamed through the Internet from a private data silo; and (d) a stable connection to the Internet with an average bandwidth of 2.5Gbps.

2.2 Computation Efficiency

LLM pre-training has presented many challenges to systems and architecture designers, as it has unprecedented memory footprints (VRAM) and requires extensive computing capabilities (FLOPs/s) (Hoffmann et al., 2022b). Most of these are mitigated by pooling extensive hardware accelerators and adopting distributed training algorithms. Standard distributed training algorithms are based on 3D parallelism, which applies data parallelism (DP) across racks in a data center (Dean et al., 2012), pipeline parallelism (PP) across servers in a rack (Narayanan et al., 2019), and tensor parallelism (TP) across GPUs in a server (Narayanan et al., 2021). The common practice is designing computing facilities to fully exploit 3D parallelism for optimal resource utilization.

Achieving optimal resource utilization requires thoughtful configuration of the hyperparameters and the mixture of distributed algorithms, which becomes more challenging as the scale increases. This tuning involves choosing the most appropriate batch size that will result in the least expensive gradient accumulation (ideally, none). We assume the participants in the distributed settings discussed here thoroughly understand their available hardware and their interplay with the 3D parallelism of the target model size. We construct our evaluation on settings running full batch steps matching their resources without any gradient accumulation.

2.3 Sourcing and Moving Data

ML workloads, such as LLM pre-training, require massive training data and computational resources, naturally distributed across several regions. To achieve reasonable efficiency, ML infrastructures are designed to follow the *data-GPU collocation* principle, i.e., data warehouses are colocated in the same region with GPU clusters to avoid relying on the cross-region network bandwidth (usually 10 times lower than intra-region network bandwidth). Large-scale infrastructures face the challenge of satisfying *data-GPU collocation* for *exabytes* of data, tens of regions with thousands of GPUs. Data is also continuously produced and removed, making the collocation task even more challenging. A worldwide scheduler, such as MAST (Choudhury et al., 2024), optimizes the cross-regional data placement daily, leveraging algorithms that can take up to 5 hours to complete their task. In this context, our work tackles a setting, usually referred to as *training-at-home*. Our Photon takes advantage of the available computing power at clients, where data is stored, resulting in the following benefits: (a) it doesn't require particular data placement optimization; (b) it can leverage low-hanging fruit local storage optimizations, such as data pre-tokenization, (c) it is compliant with privacy constraints as it doesn't move data.

2.4 Cross-silo Communication

Training procedures based on 3D parallelism leverage high-bandwidth networks supported by *intra-datacenter* networking solutions such as `RoCE` and `InfiniBand` with the typical link speed from 100Gbps up to 400Gbps per link. (Sergeev & Del Balso, 2018; Gangidi et al., 2024; Li et al., 2024). Their communication efficiency of the standard distributed training approach is heavily impacted by slow network links, which makes them unsuitable for *cross-region* applications we are interested in as their network bandwidth ranges from 0.8Gbps to 40Gbps. In this work, we use distributed training algorithms based on LocalSGD, which requires less frequent communication across workers (Lee et al., 2020) and strongly reduces the overheads of possessing slower links across workers.

3 Architecture and Design of Photon

To enable collaborative and effective cross-silo FL pre-training of LLMs with limited inter-data center communication, Photon follows three core principles: *broad inclusivity of data and compute sources*, *minimal compute requirements*, and *scalable local training pipelines*. These principles maximize client resource utilization and ensure a robust design. We now present Photon 's architecture, beginning with a brief overview of its core innovations. The main components are summarized in Figure 1.

Adaptive Local Parallelism: Photon integrates standard distributed training techniques with federated learning, optimizing training data storage, transfer, parameter communication, and aggregation. It adapts to each client's connectivity and topology, allowing automatic selection between standard distributed training and low-bandwidth LocalSGD.

Improved Model Generalization: The federated optimization we adopt produces robust model minima and is resilient to hyperparameter variations due to noise injection (Lin et al., 2020) and meta-learning effects (Nichol, 2018), ensuring convergence across varied client participation levels, data heterogeneity, and local training hyperparameters. Since robust model minima, defined as model parameters

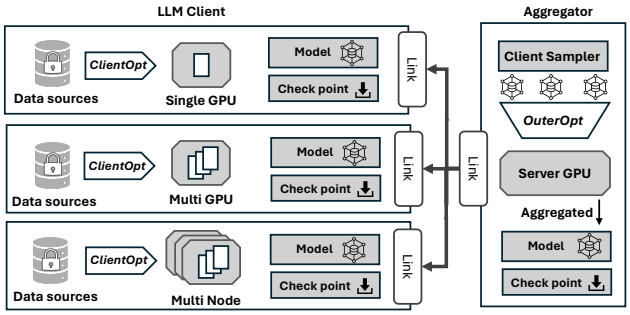


Figure 1. **Systematic diagram of Photon's three principal components - Photon aggregator, Photon LLM clients, and Photon data sources.** Arrows describe interactions and message exchanges. The Photon aggregator can only communicate with the Photon LLM nodes through the Photon link. The instances responsible for storing the data samples, the Photon data sources, can uniquely stream to the Photon client bound to them.

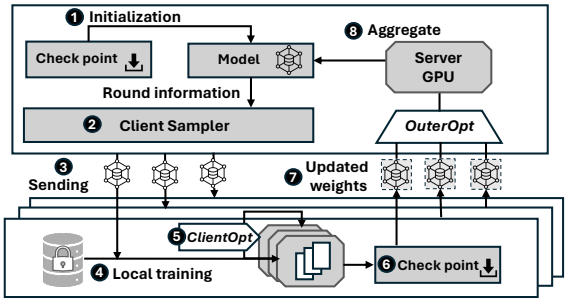


Figure 2. **Information flows between LLM clients, the aggregator, and data sources.** Following initialization (1), An LLM client selected by the client sampler (2) receives model parameters from the server (3), trains on data from a data source (4,5), checkpoints (6), and then returns the updated parameters (7) to the aggregator for federated optimization (8).

whose loss does not significantly change upon perturbation, are known to generalize better and produce lower validation loss (Keskar et al., 2017; Lin et al., 2020), this partially explains the outperformance that models trained with Photon show over centralized pre-training.

Exploiting Small Batches and High Learning Rates: Photon's robustness to hyperparameter choices enables the use of **small** (hardware-determined) local batch sizes, which promote flat minimizers of the loss that generalize better (Keskar et al., 2017), along with **high learning rates** decayed over an extended period—typically unstable with small batches—allowing us to maintain data efficiency. For example, if centralized training uses a decay period T with batch size $\mathcal{B}$, federated learning enables us to extend it to $T \times \frac{\mathcal{B}}{\mathcal{B}_{\text{small}}}$. In our experiments, using small batch sizes $\mathcal{B}_{\text{small}}$ in centralized training always resulted in model divergence unless the maximal learning rate was reduced linearly w.r.t the batch size. Full details are in Appendix C.1.

3.1 Architecture

Photon consists of the following core components.

Aggregator (Agg): Agg serves as the central server orchestrating the federated training process. At the start of each round, it activates the *Client Sampler* to access and select LLM-C instances according to the optimization algorithm's requirements. Agg then uses Link to relay messages between LLM-C clients. Once results are received from LLM-C, `OuterOpt` aggregates updates and applies the optimization to the global model, followed by checkpointing.

LLM Client (LLM-C): LLM-C is the distributed client in Photon responsible for the local training pipeline within the federated optimization process. Each LLM-C can connect to Agg at any point during training. The `ClientOpt` trains the model received from Agg on local data, utilizing various distributed algorithms suited to each LLM-C's hardware capabilities. Model updates and training metadata are then exchanged with Agg through Link. The training state is regularly checkpointed for fast recovery in case of failure.

Data Sources (DS): Data Sources serve as the data storage for Photon, meeting the federated learning requirements regarding data location and exchange protocols. Each private DS is linked to an individual LLM-C within the same client's data domain, generating a continuous data stream that matches `ClientOpt`'s training throughput. This decoupled structure allows institutions with large data silos to obtain paired computation through LLM-C without sharing data globally. Additionally, public DS can be configured for data sharing among LLM-C clients to support collaboration or data-sharing agreements between participants.

3.2 Operation

We detail Photon's workflow, as illustrated in Algorithm 1, and present a workflow visualization in Figure 2.

Photon assumes collaboration among multiple independent institutions, each with distinct data and compute silos, to pre-train LLMs with their own LLM-C. For enhanced data protection, the Aggregator (Agg) can be hosted by one of the institutions or a trusted third party. At the start of training, Agg initializes the model or sources it from one of the LLM-C (L.2). After initialization, Agg coordinates round information for client sampling (L.3 – 4), sends the model to each sampled client, and collects results (L.5 – 7). It then aggregates the local models to construct a new federated model (L.8 – 10), checkpointing the global model at the end of each round to enhance robustness (L.11).

At each round, sampled LLM-C instances initiate their local

training pipeline by acquiring data streams from their respective private DS (L.14). For enhanced evaluation, Photon allows DS to use public data sources when configured. Each LLM-C evaluates available hardware resources to select the optimal execution strategy (L.15), either utilizing fast interconnection between GPUs (L.16 – 18) or applying an additional level of federated optimization if inter-GPU connectivity is limited (L.19 – 23). In the latter case, LLM-C performs an extra level of local aggregation (L.24 – 25). A local checkpoint is also maintained for quick recovery in case of failure (L.27). After completing the local training pipeline, LLM-C applies post-processing (e.g., gradient clipping, compression, or differential privacy noise injection) before returning updates to Agg (L.28).

Algorithm 1 Photon execution pipeline

Require: Number of rounds T, training population P
Require: Number of clients per round K, hyperparameters H

1: **procedure** AGGREGATOR(T, K, H, P)
2: $\theta^0 \leftarrow \texttt{InitModel}(H)$
3: **for** each round $t = 1, 2, 3, \ldots, T$ **do**
4: $\mathcal{C} \sim \mathcal{U}(P, K)$
5: **for** $k \in \mathcal{C}$ **do in parallel**
6: $\theta_k^t, \mathcal{M}_k^t \leftarrow$ LLM CLIENT(k, θ^t, H)
7: $\Delta_k^t \leftarrow \theta^t - \theta_k^t$
8: $\Delta^t \leftarrow \frac{1}{|C|} \sum_{k \in C} \Delta_k^t$
9: $\theta^{t+1} \leftarrow \text{ServerOpt}(\theta^t, -\Delta^t, t)$
10: $\mathcal{M}_k^{t+1} \leftarrow \texttt{AggMetrics}(\mathcal{M}_k^t | \forall k \in \mathcal{C})$
11: $\texttt{Checkpoint}(\theta_k^{t+1})$
12: **return** θ_k^{T+1}

13: **procedure** LLM CLIENT(k, θ^t, H)
14: $\mathcal{D}_k \leftarrow \texttt{BindStream}(k)$
15: $I_k \leftarrow \texttt{GetNodes}(k)$
16: **if** $\texttt{HasRDMA}(I_k)$ **then**
17: $B_k \leftarrow \texttt{CalcBatchSize}(I_k)$
18: $\theta_k^t, \mathcal{M}_k^t \leftarrow \texttt{TrainClient}(\theta^t, \mathcal{D}_k, B_k, H)$
19: **else**
20: **for** node $i \in I$ **do in parallel**
21: $B_k^i \leftarrow \texttt{CalcBatchSize}(i, I_k)$
22: $\mathcal{D}_k^i \leftarrow \texttt{PartitionStream}(i, \mathcal{D}_k)$
23: $\theta_i^t, \mathcal{M}_i^t \leftarrow \texttt{TrainClient}(\theta^t, \mathcal{D}_k^i, B_k^i, H)$
24: $\theta_k^t \leftarrow \frac{1}{|I|} \sum_{i \in I} \theta_i^t$
25: $\mathcal{M}_k^t \leftarrow \texttt{AggMetrics}(\mathcal{M}_i^t | \forall i \in I_k)$
26: $\texttt{Checkpoint}(\theta_k^t, \mathcal{D}_k)$
27: $\theta_k^t \leftarrow \texttt{PostProcess}(\theta_k^t, \mathcal{M}_k^t)$
28: **return** $\theta_k^t, \mathcal{M}_k^t$

4 PHOTON IMPLEMENTATION

The Photon implementation is highly optimized for the unique challenges of federated pre-training. Thus, it has the following objectives: (a) allow for efficient, intermittent data transfer between clients, data sources, and the aggregator; (b) effectively select the distributed training algorithm for a given client given their local GPU topology; and (c) exploit the federated communication topology to select the fastest aggregation method. We now discuss how our implementation, consisting of approximately 16 273 lines of code, achieves these aims.

Link between Agg and LLM-C: To enable efficient communication between Agg and LLM-C, Photon includes a dedicated communication module, *Link*. This module assumes a relatively fast and stable internet connection of at least 1Gbps between Agg and LLM-C, which is appropriate for the cross-silo federated setting. Serving as the communication gateway, Link uses secure TLS encryption and supports secure aggregation (Bonawitz et al., 2016) for enhanced privacy, if needed. Beyond model updates, message payloads carry metadata, including training and evaluation instructions, metrics, and global instructions. Link provides an extensible post-processing pipeline by leveraging model compression and pruning techniques. By default, Photon uses lossless compression techniques without pruning.

Data Streaming for DS: Data is naturally distributed in cross-silo settings; therefore, our DS implementation breaks from the traditional one-to-one mapping between compute and data resources. Treating data as streams from one or multiple DS elements, Photon enables mixing arbitrary data streams with precise control over sampling across such streams. This decoupling allows data providers to operate independently of compute providers, broadening our federation. Moreover, this design reduces core network utilization if a DS can communicate with an LLM-C over a separate network. To optimize data streaming, Photon DS employs caching alongside optional data pre-tokenization and compression. These optimizations heuristically minimize storage overhead during the transfer from data producers to data consumers, reduce compute demands on data consumers, and maintain the throughput required by each LLM-C.

Optimal Training Strategy Selection for LLM-C: To accommodate the heterogeneous nature of the hardware, Photon 's LLM-C can support a wide range of configurations, provided the model fits within the available total VRAM with a batch size of at least one sample. An LLM-C hardware setup can include a single GPU, multiple GPUs within a server node, or multiple servers connected by high-bandwidth interconnects. Photon aims to maximize throughput for each LLM-C by selecting an optimal training strategy through a heuristic-based approach that is summarized as

follows:

1. If a model and sufficient batch size fit within a single GPU, enabling the client to keep pace with the federation, LLM-C assigns a dedicated GPU to each client.
2. For nodes with multiple GPUs, LLM-C uses either DDP or FSDP, depending on whether a model with a viable batch size fits within a single GPU.
3. When clients have a cluster of GPU-equipped machines, LLM-C selects a strategy based on the cluster interconnection speed. High-bandwidth interconnects lead to DDP or FSDP, similar to the previous variant. In contrast, lower bandwidth may necessitate constructing sub-federations with further data sub-partitioning, with each partition trained independently.

In each case, the strategy ensures optimal compute utilization within the LLM-C topology. All LLM-C training strategies are transparent to Agg, enhancing the federation's scalability and extensibility. Our experiments primarily use common cross-silo settings, such as allocating one GPU per client, reflecting typical institutional bandwidth constraints between servers. However, as demonstrated later, Photon is flexible and supports better-interconnected topologies.

Topology Between Clients: For Photon to efficiently pretrain an LLM in a federated setting, parameter aggregation must be effectively managed within current communication constraints. In general, a model with size $M = |\theta|$ trained across N workers can be aggregated in three variants, each with other distinct use cases, advantages, and limitations:

1. A *parameter server* (PS) receives all updates from participating workers. This variant is ideal for relatively small N, as the data received by the server scales in $\mathcal{O}(NM)$. It handles worker dropouts well by providing a partial update derived from surviving workers and is the only viable option when privacy restrictions prohibit peer-to-peer communication.
2. Workers communicate directly via *AllReduce* (AR). In this setup, each worker sends its model to all peers and receives models from all others, resulting in $\mathcal{O}(N^2M)$ data transmission per worker. Like PS, AllReduce tolerates dropouts well, enabling partial updates from remaining workers; however, privacy limitations may restrict peer-to-peer communication.
3. Workers use *Ring-AllReduce* (RAR), communicating over a ring topology for efficient aggregation. This bandwidth-optimal method requires each worker to send/receive $\mathcal{O}(M)$ data, with the bottleneck being the slowest link in the ring. RAR does not tolerate dropouts and has similar privacy considerations to AllReduce.

Each method has its constraints, and Photon adapts to select the most efficient option for each scenario.

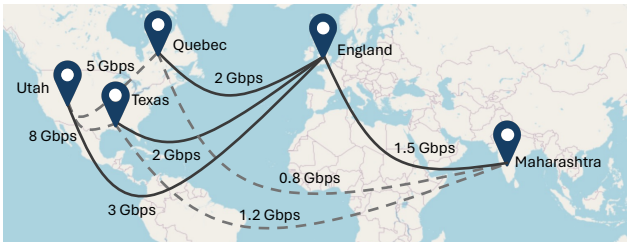


Figure 3. **The locations and bandwidth of participating clients in the Federation**, with multiple nodes equipped with H100s at each site. More details are available in Table 1. Bandwidth between regions varies significantly, impacting the efficiency of Photon's aggregation procedures. The map shows the RAR topology (gray dashed line) and the PS topology (black solide line). The slowest link in the RAR topology, between Maharashtra and Quebec, acts as a bottleneck. In the PS topology, the connection speed to England limits each update's communication.

Table 1. **Computational resources of different regions.** For each region, "num. of clients **x** num. of GPUs held by each client" is shown, e.g., 1 x 8 H100. Photon enabled this globally distributed setup, overcoming challenges of low average bandwidth.

Size	Agg	England	Utah	Texas	Quebec	Maharashtra
7B	England	-	1 x 8 H100	1 x 8 H100	1 x 8 H100	1 x 8 H100
3B	England	-	1 x 4 H100	1 x 4 H100	1 x 4 H100	1 x 4 H100
1B	England	1 x 2 H100	2 x 2 H100	2 x 2 H100	2 x 4 H100	1 x 4 H100
125M	England	2 x 1 H100	2 x 1 H100	2 x 1 H100	2 x 1 H100	2 x 1 H100

5 EVALUATION

This section demonstrates the effectiveness of Photon in training large language models (LLMs) from scratch in federated settings, compared to models trained using standard centralized methods and prior low-bandwidth distributed approaches. We evaluate models based on final performance, compute efficiency, and communication scalability.

5.1 Experimental Setup

Our experiments employ a cross-silo FL setup where clients are equipped with one or more high-end GPUs, such as Nvidia H100s, and are interconnected via the Internet. The computational resources used on each client vary based on model size: million-parameter models are trained using a single GPU, whereas billion-scale models may require up to 8 GPUs per client. We present the five locations used during training for our distributed setup in Fig. 3, with details on the number of clients and GPUs available per client provided in Table 1.

The client's local batch size is determined by its VRAM, model size, and optimal throughput, leveraging heuristics similar to those proposed by the Microsoft DeepSpeed AutoTuner (Microsoft DeepSpeed Team, 2024b;a). For instance, clients training a 125M parameter model use 1 Nvidia H100, processing a hardware-determined local batch size $B_l = 32$, without gradient accumulation or

activation checkpointing, to maximize throughput. Given the assumption of homogeneous client resources, all clients independently employ the same local throughput optimization strategies. Experiments are conducted using PyTorch (v2.4.0) with mixed precision context `bfloat16` (BF16). The model architecture is based on the MPT (Mosaic Pre-Trained) family of decoder-only transformers (MosaicML NLP Team, 2023), with the MPT centralized training recipe used as a baseline configuration.

Datasets and Training Recipe. We model data distribution across clients by randomly partitioning the `C4` (Raffel et al., 2020) dataset uniformly into 64 equally sized shards. N clients, hence, refer to a subset of N shards from these 64 shards. The local training step for clients uses AdamW (Loshchilov & Hutter, 2019) as the optimizer, with a cosine learning rate schedule and an initial linear warm-up phase. We report all hyperparameters in Appendix A. The learning rate schedule varies by model size and is adjusted to match the batch size and total token count seen in centralized training. We experiment with different numbers of local steps per round, specifically $\{62, 128, 512\}$, which defines the amount of local work done by clients. Model performance is evaluated using perplexity on the full `C4` validation set.

To explore the robustness of Photon to data heterogeneity, we also explore a setting where clients hold data from a variety of text sources representing diverse categories. Specifically, we use The Pile (Gao et al., 2020) dataset, which includes text sources from `ArXiv` (academic), `C4` and `Wikipedia` (internet), and `Project Gutenberg` (prose) to capture a range of language styles while providing sufficient data for scaling. For our experiments with full participation, we explore three configurations: four clients (one text source per client), eight clients (two text sources split into two clients), and sixteen clients (four text sources split into four clients). For partial participation, we adopt the sixteen-client configuration, sampling 25%, 50%, and 100% of clients per round, with evaluation on `C4`.

5.2 Photon trains Billion-sized LLMs

Using Photon, we train models up to 7B parameters from scratch in federations of 16 to 4 clients, using full participation every round and prove they outperform their centralized counterparts. While previous results suggest that federated optimization is generally inferior to standard SGD in terms of loss (Wang et al., 2021) and that LLM pre-training may be less sample-efficient (Douillard et al., 2023; Sani et al., 2024), we find it to be quite competitive in terms of wall-clock time and, comparable if not slightly superior in terms of loss. As shown in Fig. 4 and Table 2, the 3B and 7B models achieve 13.8% to 16.9% lower perplexity than models trained using centralized training. As shown in Table 3, such models can perform similarly in terms of perplexity

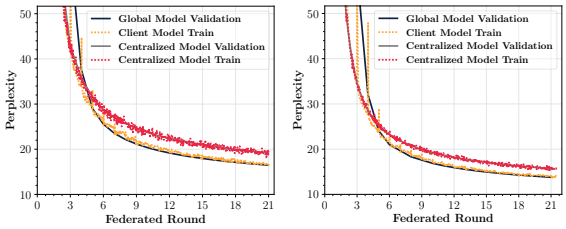


Figure 4. **Comparison of perplexity convergence (*lower is better*) for Photon and centralized training with 3B (left) and 7B (right) models.** The federated global model was evaluated on the `C4` test set, with averaged train perplexities across clients and centralized train/test perplexities presented for both models. These large federated models show lower perplexity than centralized models and remain stable during aggregation, with minimal perplexity spikes after early rounds.

Table 2. **Our results show federated models to be comparable to centralized and potentially superior as they obtain lower perplexity (PP) given the same computational resources.** Their perplexity gains grow with model size.

Size	Fed PP	Cent PP	Gain (%)
1.3B	**20.1**	23.2	**13.4**%
3B	**15.7**	18.2	**13.7**%
7B	**13.8**	16.6	**16.9**%

after sufficient training. For instance, when utilizing identical computational resources with a 10 Gbps connection and Ring-AllReduce, Table 3 demonstrates that federated optimization for the 7B model to reach the same perplexity requires only 95.6 total hours whereas centralized methods require over 147 hours. Despite the compute time of the 7B model being almost 2× higher, the 500× communication reduction of Photon can bring an overall reduction in wall time. Higher-bandwidth connections such as `InfiniBand` would bring the total time closer to the compute time. Moving beyond perplexity, we also show that the models produced by Photon are effective for downstream tasks in Appendix D.3.

We observe that the perplexity gap between federated and centralized models widens at larger scales; smaller models converge to similar performance levels across methods. Additionally, as model size grows, training stability improves, as we observe perplexity spikes decreasing in magnitude compared to smaller models. Our observed performance improvements stem from the noise-injecting (Lin et al., 2020) and meta-learning (Nichol, 2018) of our federated optimization. As we will later see, federated optimization supports high learning rates with small batch sizes, enhancing generalization via noise injection (Keskar et al., 2017).

Table 3. **We report system metrics for the billion-scale models trained with Photon and their centralized baselines**, including total wall time with compute and communication time breakdowns. We provide GPU efficiency metrics such as average GPU utilization and Model FLOPs Utilization (MFU) during local computation.While federated optimization may increase compute time, Photon 's federated approach shortens overall training time by reducing communication steps, assuming consistent setups: the number of federated clients matches data-parallel workers, aggregation uses *Ring-AllReduce* with a fixed 10Gbps bandwidth for the slowest link, and both clients and workers maintain the same mini-batch throughput.

Model	Wall Time [h]	Compute Time [h]	Comm. Time [h]	Local Util. GPU [%]	Local MFU per device
Cen-1.3B	26.7 (1×)	6.5 (**1×**)	20.2 (1×)	74	0.8027
Fed-1.3B	18.02 (**0.67×**)	18.0 (2.8×)	0.02 (**0.001×**)	83	1.1245
Cen-3B	56.6 (1×)	16.1 (**1×**)	40.48 (1×)	81	0.165
Fed-3B	25.2 (**0.45×**)	25.1 (1.6×)	0.05 (**0.001×**)	78	0.240
Cen-7B	147.9 (1×)	50.7 (**1×**)	97.2 (1×)	88	0.335
Fed-7B	95.6 (**0.65×**)	95.5 (1.9×)	0.1 (**0.001×**)	90	0.224

5.3 Photon Model Performance Scales with Federation Size

Previous works (Charles et al., 2021; Douillard et al., 2023) raised concerns about the wall time benefits of increasing the federated population size. Using Photon, we show that client contributions to the global model convergence depend on the amount of local work, based on the following hyperparameters: the number of local training steps per round and the global batch size $B_g = NB_l \in \{32, 64, 128, 256, 512\}$, where $N \in \{1, 2, 4, 8, 16\}$ is the number of clients per round, and $B_l = 32$ is the local batch size.

As illustrated in Figure 5, more frequent communication (smaller local steps per round) leads to greater wall time reductions with larger B_g. The trends are smoother for a higher target perplexity of 42 than at a lower target perplexity. On the other hand, at a target perplexity of 35, the gains in wall time drop as B_g increases, especially at 128 steps per round. This phenomenon is also observed in McCandlish et al. (2018), where increasing the batch size in the centralized setting led to diminishing returns in wall time. For more details, see Appendix C.1. Hence, depending on the target perplexity and local steps per round, a compute-optimal regime exists where the global batch size B_g must be carefully tuned to maximize the number of available clients.

Comparison with State-of-the-Art. Through utilizing the aforementioned compute-optimal regime, Photon significantly outperforms previous state-of-the-art LLM distributed pre-training framework, DiLoCo (Douillard et al., 2023). In Table 4, we train a model with 125M parameters, varying the number of clients $N \in \{2, 4, 8\}$, and using a local batch size $B_l = 32$. We show that, within an appropriate global batch size B_g regime, DiLoCo yields limited returns as the number of clients per round increases and requires

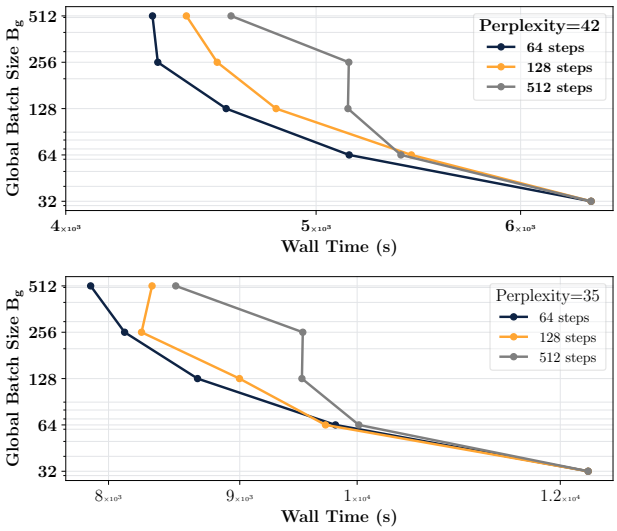


Figure 5. **The tradeoff between time and compute resources (the larger batch size, the more resources) spent to train a model by Photon to target perplexities (top 42 and bottom 35).** We measure the impact of the global batch size $B_g = NB_l$, where $N \in \{1, 2, 4, 8, 16\}$ (number of clients per round) and $B_l = 32$ (local batch size), on the wall time needed to reach two target perplexities: 42 (top, near the centralized baseline) and 35 (bottom, near optimum). Fewer local steps per round (64) show clear benefits from increasing B_g for both perplexity targets, but with more local work (128 and 512 steps), the returns on reduced wall time diminish.

roughly 2× more time to reach the same perplexity for both target values, 42 and 35. We observe similar trends across different B_g settings used in DiLoCo.

In Douillard et al. (2023), the authors adopted a much higher compute regime.DiLoCo trains a *smaller* model (75M parameter) with $B_l = 512$, $N = 8$, and $B_g = 4096$ for 88 000 steps, which corresponds to 46B tokens per worker and 369B tokens in total, far beyond the compute-optimal 1.5B estimated by Hoffmann et al. (2022a)[1]. In contrast, for the **larger** $125M$ model, Photon automatically tunes the local batch size $B_l = 32$ to suit the hardware capabilities and achieve 9000 cumulative training steps per client when using four clients per round, reaching 2.32B token processed in total, close to the compute-optimal 2.5B tokens estimated by Hoffmann et al. (2022a).

5.4 Photon's Comms. Efficiency and Scalability

Communication is an overhead in FL, often scaling poorly as the number of clients grows. In this section, we evaluate the communication scalability of Photon, considering its nontrivial relationship with the faster convergence speed

[1]The original work is not explicit if 512 is global or local. Substituting $B_l = \frac{512}{8}$ gives similar conclusions, with 5.75B tokens per worker and 46B in total, far beyond the optimal 1.5B.

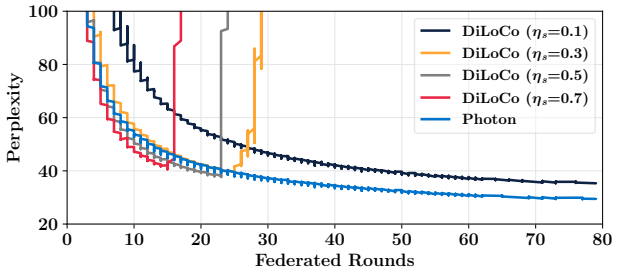


Figure 6. **Perplexity convergence comparison between Photon and DiLoCo.** We tuned DiLoCo's recommended server optimizer (`OuterOpt`), `SGD` with Nesterov momentum, using learning rates $\eta_s \in \{0.1, 0.3, 0.5, 0.7\}$, while keeping the momentum coefficient fixed at 0.9. A 125M-parameter model was trained with a global batch size $B_g = 128$ and $N = 4$ clients per round. Higher η_s values accelerated training but hindered achieving the desired perplexity due to early divergent activations. Consequently, we set $\eta_s = 0.1$ in later experiments to reach lower perplexities.

Table 4. **Photon consistently reaches a satisfactory perplexity twice as fast as DiLoCo ($\eta = 0.1$) across three client counts.** We evaluated the impact of the global batch size $B_g = NB_l$, where $N \in \{2, 4, 8\}$ (clients per round) and $B_l = 32$ (local batch size), on the wall time required to reach target perplexities of 42 (near the centralized baseline) and 35 (near optimal). Results show the wall time gap between Photon and DiLoCo when tuning the server learning rate $\eta_s = 0.1$ to meet the target perplexities.

N	Method	Wall Time [s]	
		PPL = 42	PPL = 35
2	DiLoCo ($\eta_s = 0.1$)	10528.8 (1×)	19516.8 (1×)
	Photon	5392.8 (**0.51×**)	10015.2 (**0.51×**)
4	DiLoCo ($\eta_s = 0.1$)	10545.2 (1×)	19032.8 (1×)
	Photon	5144.0 (**0.49×**)	9516.4 (**0.5×**)
8	DiLoCo ($\eta_s = 0.1$)	9523.8 (1×)	20334.6 (1×)
	Photon	5148.0 (**0.54×**)	9523.8 (**0.47×**)

brought by using more clients per round (Fig. 5). Thus, we examine three aggregation implementations for Photon: *parameter server* (PS), *AllReduce* (AR), and *Ring-AllReduce* (RAR). These approaches are detailed in Section 4.

Increasing the number of clients per round speeds up convergence, reducing the required communication steps. Thus, we evaluate communication scalability by varying the number of clients per round, $N \in \{2, 4, 8, 16\}$ (excluding $N = 1$, as no communication occurs). Since the number of pseudo-gradients communicated and aggregated scales linearly with the number of clients and model size, we use the most communication-efficient setup in our experiments. Specifically, clients perform 512 local steps per round, training a 125M parameter model targeting a perplexity of 35.

Figure 7 shows that communication overhead increases with N, especially under the PS implementation due to its limited scalability. However, using more clients accelerates convergence, reducing the required communication steps

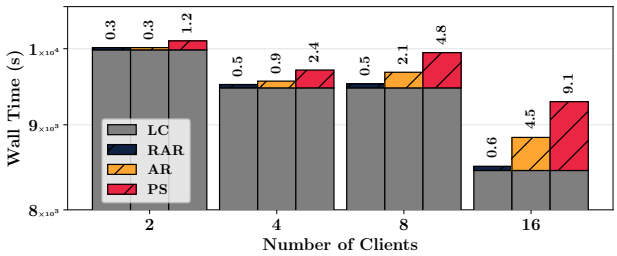


Figure 7. **Wall time comparison among different topologies.** We report total wall time divided into local compute (LC) and communication time. Communication time is evaluated for three aggregation methods: *parameter server* (PS) for privacy-constrained settings, *AllReduce* (AR) for better scalability, and *Ring-AllReduce* (RAR), the most scalable but limited by the slowest link. As expected, communication costs rise with more clients. However, efficient implementations like RAR maintain the wall time reduction from scaling compute resources. The top indicates the percentage of time spent on communication for each method.

and mitigating the costlier communication. Except at the largest cohort size (16 clients), the less scalable implementations (PS and AR) account for a minor portion of wall time relative to the local computation (LC) time at clients. Figure 7 highlights RAR as the most scalable option when minimizing communication time is crucial.

5.5 Photon's Robustness to Data Heterogeneity

In typical federated learning scenarios, clients possess heterogeneous data distributions, raising concerns about the robustness of federated optimization. Non-coherent pseudo-gradient updates across clients can impact model convergence by slowing it down or reducing performance. This challenge is compounded by privacy restrictions that prevent observing client data distributions directly.

We assess Photon's robustness to data heterogeneity by training on different data sources from The Pile dataset, distributed across clients as described in Section 5.1. The most challenging scenario involves partial participation of the client population, where the global federated model is only intermittently exposed to diverse data distributions. Figure 8 (top) shows that higher client sampling ratios improve convergence speed, final performance, and convergence smoothness under partial participation, demonstrating Photon's robustness to heterogeneous data.

Under full participation, Figure 8 (bottom) indicates that larger client cohorts reduce wall-clock training time, though less effectively than with IID data due to conflicting gradients from diverse sources (Charles et al., 2021). Aggregation methods designed for heterogeneous data, as in (Yadav et al., 2023), could further enhance convergence in such cases.

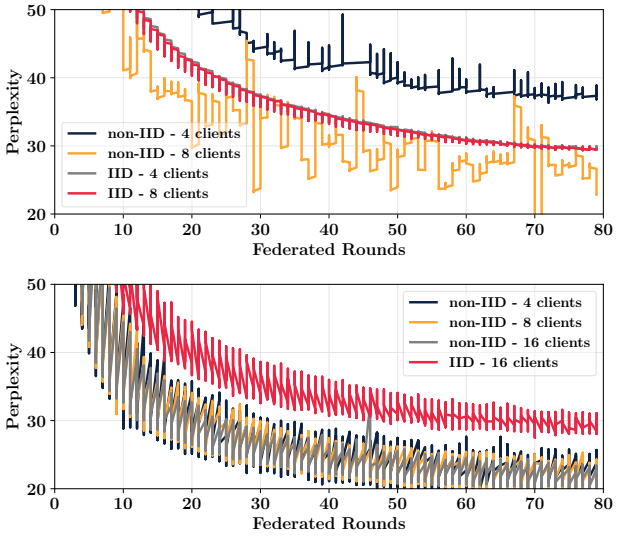


Figure 8. **We assess Photon's robustness to data heterogeneity by training with (top) and without (bottom) partial participation.** Federation of clients using data from The Pile dataset, as described in Section 5.1. Results from homogeneous data distribution (IID) experiments for reference. While partial participation causes larger fluctuations across rounds, full participation behaves similarly to the IID case. In all settings, increasing the number of clients per round accelerates achieving the target perplexity, with the effect more pronounced under partial participation.

6 Opportunities and Limitations

Our proposed Photon introduces both algorithmic and system optimization techniques to achieve state-of-the-art LLM pre-training across distributed heterogeneous data. In this section, we reiterate some of our work's limitations and expand on some of our research's potential opportunities and applications.

Federated Hyperparameter Tuning. In Section 5.3, we emphasize the importance of selecting the global batch size to maximize computational resources and performance. A natural extension of our work would be to investigate other critical hyperparameters, such as the learning rate, learning rate scheduler, and their interaction with batch size. Photon 's significant reduction in pre-training costs for LLMs makes it feasible to leverage existing federated hyperparameter optimization algorithms (Zhou et al., 2023; Khodak et al., 2021) to explore optimal global or per-client hyperparameters.

Addressing Data Heterogeneity. Our experiments in Section 5.5 demonstrate Photon's robustness in handling data heterogeneity. However, further exploration of alternative aggregation strategies, loss functions, and client selection methods could enhance performance under such conditions. Techniques include minimizing the Euclidean distance or maximizing agreement between global and local models (Li et al., 2020b; 2021), reducing local model divergence from the global model (Acar et al., 2021; Karimireddy et al., 2020), and measuring client contributions (Wang et al., 2019; 2020), with client selection based on their value to the global model (Cho et al., 2020; Huang et al., 2021). Most of these methods can be directly integrated into Photon.

Continual Pre-training & Personalization. A key advantage of using Photon for pre-training LLMs is improved model convergence and performance, offering a stronger initialization for continual pre-training or personalization. This aligns with findings from previous studies (Nguyen et al., 2023; Chen et al., 2023), which highlight the importance of starting with a strong pre-trained model to stabilize federated training and enhance global model performance. A robust pre-trained model also benefits LLM personalization by enabling further fine-tuning or per-client learning strategies (Lee et al., 2023).

Cross-device Federated Scenarios. In this work, we demonstrate how Photon effectively addresses the challenges of cross-silo FL. Another common context is cross-device FL, where clients are often lower-compute devices like mobile phones and IoT devices. To handle the diverse system heterogeneity in these scenarios, Photon can be extended with existing methods proven successful in cross-device FL, such as parameter-efficient fine-tuning (Sun et al., 2022; Zhang et al., 2023), quantization (Yoon et al., 2022), low-rank decomposition (Yao et al., 2021), pruning (Caldas et al., 2018; Jiang et al., 2022), and early exits (Lee et al., 2024b; Kim et al., 2023; Ilhan et al., 2023).

7 Related Work

Recent advances in large-scale FL and distributed deep learning focus on improving scalability, efficiency, and privacy while addressing challenges like device heterogeneity and system constraints. Early efforts, such as (Bonawitz et al., 2019), emphasize system design in FL for mobile devices, addressing resource limitations and connectivity in applications like Google's Gboard. Asynchronous FL systems like PAPAYA (Huba et al., 2022) improve scalability by allowing clients to update models independently, boosting convergence speed and reducing communication overhead. Platforms like FLINT (Wang et al., 2023a) simulate real-world FL constraints, benefiting large-scale applications like LinkedIn. In distributed deep learning, PyTorch's Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023) and Distributed Data Parallel (DDP) (Li et al., 2020a) are crucial for scalable training, while frameworks like Horovod (Sergeev & Balso, 2018) use *Ring-AllReduce* to simplify distributed learning. Tools like Dataset Grouper (Charles et al., 2023) help create large-scale datasets, and systems like MAST (Choudhury et al., 2024) optimize workload placement across data centers. ZeRO (Wang et al., 2023b; Ren et al., 2021; Rajbhandari et al., 2020) and PETALS

(Borzunov et al., 2023a;b) extend scalability, enabling training and inference for models with trillions of parameters and decentralized deployment, respectively, providing robust solutions for large-scale ML.

In federated or collaborative LLM training, most efforts have focused on fine-tuning (Kuang et al., 2024; Ye et al., 2024) or inference (Borzunov et al., 2023a;b), with pre-training only recently gaining attention (Douillard et al., 2023; Sani et al., 2024; Nous Research, 2024; Iacob et al., 2025; Cheng & Glasgow, 2025). Previously, federated learning primarily centered on cross-device settings, which are unsuited to pre-training's substantial hardware demands. Douillard et al. (2023), discussed in the main work, achieves notable communication reductions in a high-compute training regime. Sani et al. (2024) introduces a library for LLM pre-training and provides initial results on large models. Nous Research (2024) offers early findings on a peer-to-peer AllReduce approach to Federated Learning.

8 Conclusion

This work introduces Photon, the first federated system for decentralized end-to-end LLM pre-training in low-bandwidth, globally distributed settings. Photon enables collaborative training of models up to 7B parameters, outperforming centralized training in perplexity. By pooling client resources, Photon accelerates training as compute scales, and in low-bandwidth conditions, it surpasses standard distributed training by increasing throughput and reducing communication costs. This is achieved through adaptive local parallelism, which dynamically switches between distributed algorithms and low-bandwidth Local SGD based on client connectivity. With small local batch sizes and high learning rates, Photon supports an aggressive training strategy, making it the first cost-effective solution to scale LLM pre-training beyond data centers.

Acknowledgements

All costs for the computation used for this work was funded by Flower Labs, and the research conducted by a team of researchers from Flower Labs and The University of Cambridge. Support for university-based researchers came from a variety of sources, but in particular, the following funding organizations are acknowledged: the European Research Council (REDIAL), the Royal Academy of Engineering (DANTE), and the Ministry of Education of Romania through the Credit and Scholarship Agency. We additionally express our immense gratitude to Pedro Porto Buarque de Gusmão, who helped construct the foundations of this project; Javier Fernandez-Marques for supporting us with invaluable insights; and Christopher Irwin and Andrej Jovanovic for the insightful feedback.

References

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

Allal, L. B., Lozhkov, A., Bakouch, E., Blázquez, G. M., Penedo, G., Tunstall, L., Marafioti, A., Kydlícek, H., Lajarín, A. P., Srivastav, V., Lochner, J., Fahlgren, C., Nguyen, X., Fourrier, C., Burtenshaw, B., Larcher, H., Zhao, H., Zakka, C., Morlon, M., Raffel, C., von Werra, L., and Wolf, T. Smollm2: When smol goes big - data-centric training of a small language model. *CoRR*, abs/2502.02737, 2025.

Ben Allal, L., Lozhkov, A., Penedo, G., Wolf, T., and von Werra, L. Cosmopedia, 2024. URL https://huggingface.co/datasets/HuggingFaceTB/cosmopedia.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B., and Weinbach, S. Gpt-neox-20b: An open-source autoregressive language model. *CoRR*, abs/2204.06745, 2022. doi: 10.48550/ARXIV.2204.06745.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

Bonawitz, K. A., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, B., Overveldt, T. V., Petrou, D., Ramage, D., and Roselander, J. Towards federated learning at scale: System design. In *Proceedings of Machine Learning and Systems 2019, MLSys*, 2019.

Borzunov, A., Baranchuk, D., Dettmers, T., Riabinin, M., Belkada, Y., Chumachenko, A., Samygin, P., and Raffel, C. Petals: Collaborative inference and fine-tuning of large models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: System*

Demonstrations, ACL 2023, Toronto, Canada, July 10-12, 2023, pp. 558–568. Association for Computational Linguistics, 2023a. doi: 10.18653/V1/2023.ACL-DEMO.54.

Borzunov, A., Ryabinin, M., Chumachenko, A., Baranchuk, D., Dettmers, T., Belkada, Y., Samygin, P., and Raffel, C. A. Distributed inference and fine-tuning of large language models over the internet. In *Conference on Neural Information Processing Systems*, 2023b.

Borzunov, A., Ryabinin, M., Chumachenko, A., Baranchuk, D., Dettmers, T., Belkada, Y., Samygin, P., and Raffel, C. A. Distributed inference and fine-tuning of large language models over the internet. *Conference on Neural Information Processing Systems*, 36, 2024.

Caldas, S., Konečny, J., McMahan, H. B., and Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

Chang, Q., Yan, Z., Zhou, M., et al. Mining multi-center heterogeneous medical data with distributed synthetic learning. *Nature Communications*, 14(5510):5510, 2023. doi: 10.1038/s41467-023-40687-y.

Charles, Z., Garrett, Z., Huo, Z., Shmulyian, S., and Smith, V. On large-cohort training for federated learning. In *Conference on Neural Information Processing Systems*, pp. 20461–20475, 2021.

Charles, Z., Mitchell, N., Pillutla, K., Reneer, M., and Garrett, Z. Towards federated foundation models: Scalable dataset pipelines for group-structured learning. In *Conference on Neural Information Processing Systems*, 2023.

Chen, H.-Y., Tu, C.-H., Li, Z., Shen, H. W., and Chao, W.-L. On the importance and applicability of pre-training for federated learning. In *International Conference on Learning Representations*, 2023.

Cheng, Z. and Glasgow, M. Convergence of distributed adaptive optimization with local updates. In *The Thirteenth International Conference on Learning Representations*, 2025.

Cho, Y. J., Wang, J., and Joshi, G. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

Choudhury, A., Wang, Y., Pelkonen, T., Srinivasan, K., Jain, A., Lin, S., David, D., Soleimanifard, S., Chen, M., Yadav, A., Tijoriwala, R., Samoylov, D., and Tang, C. MAST: global scheduling of ML training across geo-distributed datacenters at hyperscale. In *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024*, pp. 563–580. USENIX Association, 2024.

Clark, C., Lee, K., Chang, M., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1300.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M. a., Senior, A., Tucker, P., Yang, K., Le, Q., and Ng, A. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

Douillard, A., Feng, Q., Rusu, A. A., Chhaparia, R., Donchev, Y., Kuncoro, A., Ranzato, M., Szlam, A., and Shen, J. Diloco: Distributed low-communication training of language models. *CoRR*, abs/2311.08105, 2023. doi: 10.48550/ARXIV.2311.08105.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I. M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Gangidi, A., Miao, R., Zheng, S., Bondu, S. J., Goes, G., Morsy, H., Puri, R., Riftadi, M., Shetty, A. J., Yang, J., Zhang, S., Fernandez, M. J., Gandham, S., and Zeng, H. Rdma over ethernet for distributed training at meta

scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, pp. 57–70, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706141. doi: 10.1145/3651890.3672233.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Granziol, D., Zohren, S., and Roberts, S. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *J. Mach. Learn. Res.*, 23: 173:1–173:65, 2022a.

Granziol, D., Zohren, S., and Roberts, S. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *J. Mach. Learn. Res.*, 23: 173:1–173:65, 2022b.

Han, X., Jian, Y., Hu, X., Liu, H., Wang, Y., Fan, Q., Ai, Y., Huang, H., He, R., Yang, Z., and You, Q. Infimm-webmath-40b: Advancing multimodal pre-training for enhanced mathematical reasoning, 2024. URL https://arxiv.org/abs/2409.12568.

He, Q., Zhuang, X., and Wu, Z. Exploring scaling laws for local sgd in large language model training. *arXiv preprint arXiv:2409.13198*, 2024.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *ICLR*. OpenReview.net, 2021.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022a.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J., and Sifre, L. An empirical analysis of compute-optimal large language model training. In *Conference on Neural Information Processing Systems*, volume 35, pp. 30016–30030, 2022b.

Hu, Q., Ye, Z., Wang, Z., Wang, G., Zhang, M., Chen, Q., Sun, P., Lin, D., Wang, X., Luo, Y., Wen, Y., and Zhang, T. Characterization of large language model development in the datacenter. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 709–729. USENIX Association, 2024. ISBN 978-1-939133-39-7.

Huang, J., Hong, C., Chen, L. Y., and Roos, S. Is shapley value fair? improving client selection for mavericks in federated learning. *arXiv preprint arXiv:2106.10734*, 2021.

Huba, D., Nguyen, J., Malik, K., Zhu, R., Rabbat, M., Yousefpour, A., Wu, C., Zhan, H., Ustinov, P., Srinivas, H., Wang, K., Shoumikhin, A., Min, J., and Malek, M. PAPAYA: practical, private, and scalable federated learning. In *Proceedings of the Fifth Conference on Machine Learning and Systems,MLSys 2022*, 2022.

Huo, Z., Yang, Q., Gu, B., Carin, L., and Huang, H. Faster on-device training using new federated momentum algorithm. *CoRR*, abs/2002.02090, 2020.

Iacob, A., Sani, L., Kurmanji, M., Shen, W. F., Qiu, X., Cai, D., Gao, Y., and Lane, N. D. DEPT: Decoupled embeddings for pre-training language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ilhan, F., Su, G., and Liu, L. Scalefl: Resource-adaptive federated learning with heterogeneous clients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24532–24541, 2023.

Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K., and Tassiulas, L. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Jiang, Z., Lin, H., Zhong, Y., Huang, Q., Chen, Y., Zhang, Z., Peng, Y., Li, X., Xie, C., Nong, S., Jia, Y., He, S., Chen, H., Bai, Z., Hou, Q., Yan, S., Zhou, D., Sheng, Y., Jiang, Z., Xu, H., Wei, H., Zhang, Z., Nie, P., Zou, L., Zhao, S., Xiang, L., Liu, Z., Li, Z., Jia, X., Ye, J., Jin, X., and Liu, X. Megascale: Scaling large language model training to more than 10, 000 gpus. In *21st USENIX Symposium on Networked Systems Design and Implementation, NSDI 2024*, pp. 745–760. USENIX Association, 2024.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K. A., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021. doi: 10.1561/2200000083.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Khodak, M., Tu, R., Li, T., Li, L., Balcan, M.-F. F., Smith, V., and Talwalkar, A. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. *Advances in Neural Information Processing Systems*, 34, 2021.

Kim, M., Yu, S., Kim, S., and Moon, S.-M. DepthFL : Depthwise federated learning for heterogeneous clients. In *The Eleventh International Conference on Learning Representations*, 2023.

Kuang, W., Qian, B., Li, Z., Chen, D., Gao, D., Pan, X., Xie, Y., Li, Y., Ding, B., and Zhou, J. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In *KDD*, pp. 5260–5271. ACM, 2024.

Lee, K., Gangidi, A., and Oldham, M. Building Meta's GenAI infrastructure, March 2024a. Categories: AI Research, Data Center Engineering, ML Applications.

Lee, R., Kim, M., Li, D., Qiu, X., Hospedales, T. M., Huszar, F., and Lane, N. D. Fedl2p: Federated learning to personalize. In *Conference on Neural Information Processing Systems*, 2023.

Lee, R., Fernandez-Marques, J., Hu, S. X., Li, D., Laskaridis, S., Dudziak, Ł., Hospedales, T., Huszár, F., and Lane, N. D. Recurrent early exits for federated learning with heterogeneous clients. In *The 41st International Conference on Machine Learning*, 2024b.

Lee, S., Kang, Q., Agrawal, A., Choudhary, A., and Liao, W.-k. Communication-efficient local stochastic gradient descent for scalable deep learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 718–727, 2020. doi: 10.1109/BigData50022.2020.9378178.

Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., and Chintala, S. Pytorch distributed: Experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12):3005–3018, aug 2020a. ISSN 2150-8097. doi: 10.14778/3415478.3415530.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020*, 2020b.

Li, W., Liu, X., Li, Y., Jin, Y., Tian, H., Zhong, Z., Liu, G., Zhang, Y., and Chen, K. Understanding communication characteristics of distributed training. In *Proceedings of the 8th Asia-Pacific Workshop on Networking*, APNet '24, pp. 1–8, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400717581. doi: 10.1145/3663408.3663409.

Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don't use large mini-batches, use local SGD. In *International Conference on Learning Representations*, 2020.

Lo, K. I., Sadrzadeh, M., and Mansfield, S. Generalised winograd schema and its contextuality. In Mansfield, S., Valiron, B., and Zamdzhiev, V. (eds.), *Proceedings of the Twentieth International Conference on Quantum Physics and Logic, QPL 2023, Paris, France, 17-21st July 2023*, volume 384 of *EPTCS*, pp. 187–202, 2023. doi: 10.4204/EPTCS.384.11.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Lozhkov, A., Li, R., Allal, L. B., Cassano, F., Lamy-Poirier, J., Tazi, N., Tang, A., Pykhtar, D., Liu, J., Wei, Y., Liu, T., Tian, M., Kocetkov, D., Zucker, A., Belkada, Y., Wang, Z., Liu, Q., Abulkhanov, D., Paul, I., Li, Z., Li, W.-D., Risdal, M., Li, J., Zhu, J., Zhuo, T. Y., Zheltonozhskii, E., Dade, N. O. O., Yu, W., Krauß, L., Jain, N., Su, Y., He, X., Dey, M., Abati, E., Chai, Y., Muennighoff, N., Tang, X., Oblokulov, M., Akiki, C., Marone, M., Mou, C., Mishra, M., Gu, A., Hui, B., Dao, T., Zebaze, A., Dehaene, O., Patry, N., Xu, C., McAuley, J., Hu, H., Scholak, T., Paquet, S., Robinson, J., Anderson, C. J., Chapados, N., Patwary, M., Tajbakhsh, N., Jernite, Y., Ferrandis, C. M., Zhang, L., Hughes, S., Wolf, T., Guha, A., von Werra, L., and de Vries, H. Starcoder 2 and the stack v2: The next generation, 2024.

Marfoq, O., Xu, C., Neglia, G., and Vidal, R. Throughput-optimal topology design for cross-silo federated learning. *Advances in Neural Information Processing Systems*, 33: 19478–19487, 2020.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training. *CoRR*, abs/1812.06162, 2018.

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

Mi, H., Xu, K., Feng, D., Wang, H., Zhang, Y., Zheng, Z., Chen, C., and Lan, X. Collaborative deep learning across multiple data centers. *Sci. China Inf. Sci.*, 63(8):1–11, 2020.

Microsoft DeepSpeed Team. DeepSpeed Autotuning, 2024a. URL https://github.com/microsoft/DeepSpeed/tree/master/deepspeed/autotuning.

Microsoft DeepSpeed Team. Microsoft DeepSpeed, 2024b. URL https://www.deepspeed.ai/.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1260.

MosaicML NLP Team. Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs, 2023. URL https://www.databricks.com/blog/mpt-7b.

Narayanan, D., Harlap, A., Phanishayee, A., Seshadri, V., Devanur, N. R., Ganger, G. R., Gibbons, P. B., and Zaharia, M. Pipedream: generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pp. 1–15, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368735. doi: 10.1145/3341301.3359646.

Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., Phanishayee, A., and Zaharia, M. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3476209.

Nguyen, J., Wang, J., Malik, K., Sanjabi, M., and Rabbat, M. Where to begin? on the impact of pre-training and initialization in federated learning. In *International Conference on Learning Representations*, 2023.

Nichol, A. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Nous Research. DisTrO, 2024. URL https://github.com/NousResearch/DisTrO/blob/main/A_Preliminary_Report_on_DisTrO.pdf.

Ortiz, J. J. G., Frankle, J., Rabbat, M., Morcos, A. S., and Ballas, N. Trade-offs of local SGD at scale: An empirical study. *CoRR*, abs/2110.08133, 2021.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/V1/P16-1144.

Penedo, G., Kydlícek, H., Allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C. A., von Werra, L., and Wolf, T. The fineweb datasets: Decanting the web for the finest text data at scale. In *NeurIPS*, 2024.

Qian, K., Xi, Y., Cao, J., Gao, J., Xu, Y., Guan, Y., Fu, B., Shi, X., Zhu, F., Miao, R., Wang, C., Wang, P., Zhang, P., Zeng, X., Ruan, E., Yao, Z., Zhai, E., and Cai, D. Alibaba HPN: A data center network for large language model training. In *Proceedings of the ACM SIGCOMM 2024 Conference, ACM SIGCOMM 2024, Sydney, NSW, Australia, August 4-8, 2024*, pp. 691–706. ACM, 2024. doi: 10.1145/3651890.3672265.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event Atlanta, Georgia, USA, November 9-19, 2020*, pp. 20. IEEE/ACM, 2020. doi: 10.1109/SC41405.2020.00024.

Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference*

on Knowledge Discovery & Data Mining, KDD '20, pp. 3505–3506. Association for Computing Machinery, 2020. ISBN 9781450379984. doi: 10.1145/3394486.3406703.

Ren, J., Rajbhandari, S., Aminabadi, R. Y., Ruwase, O., Yang, S., Zhang, M., Li, D., and He, Y. Zero-offload: Democratizing billion-scale model training. In *2021 USENIX Annual Technical Conference, USENIX ATC 2021*, pp. 551–564. USENIX Association, 2021.

Roemmele, M., Bejan, C. A., and Gordon, A. S. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI, 2011.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6399.

Sani, L., Iacob, A., Cao, Z., Marino, B., Gao, Y., Paulik, T., Zhao, W., Shen, W. F., Aleksandrov, P., Qiu, X., et al. The future of large language model pre-training is federated. *arXiv preprint arXiv:2405.10853*, 2024.

Sergeev, A. and Balso, M. D. Horovod: fast and easy distributed deep learning in tensorflow. *CoRR*, abs/1802.05799, 2018.

Sergeev, A. and Del Balso, M. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., Agarwal, A., Power, A., Ray, A., Warstadt, A., Kocurek, A. W., Safaya, A., Tazarv, A., Xiang, A., Parrish, A., Nie, A., Hussain, A., Askell, A., Dsouza, A., Slone, A., Rahane, A., Iyer, A. S., Andreassen, A., Madotto, A., Santilli, A., Stuhlmüller, A., Dai, A. M., La, A., Lampinen, A. K., Zou, A., Jiang, A., Chen, A., Vuong, A., Gupta, A., Gottardi, A., Norelli, A., Venkatesh, A., Gholamidavoodi, A., Tabassum, A., Menezes, A., Kirubarajan, A., Mullokandov, A., Sabharwal, A., Herrick, A., Efrat, A., Erdem, A., Karakas, A., Roberts, B. R., Loe, B. S., Zoph, B., Bojanowski, B., Özyurt, B., Hedayatnia, B., Neyshabur, B., Inden, B., Stein, B., Ekmekci, B., Lin, B. Y., Howald, B., Orinion, B., Diao, C., Dour, C., Stinson, C., Argueta, C., Ramírez, C. F., Singh, C., Rathkopf, C., Meng, C., Baral, C., Wu, C., Callison-Burch, C., Waites, C., Voigt, C., Manning, C. D., Potts, C., Ramirez, C., Rivera, C. E., Siro, C., Raffel, C., Ashcraft, C., Garbacea, C., Sileo, D., Garrette, D., Hendrycks, D., Kilman, D., Roth, D., Freeman, D., Khashabi, D., Levy, D., González, D. M., Perszyk, D., Hernandez, D., Chen, D., Ippolito, D., Gilboa, D., Dohan, D., Drakard, D., Jurgens, D., Datta, D., Ganguli, D., Emelin, D., Kleyko, D., Yuret, D., Chen, D., Tam, D., Hupkes, D., Misra, D., Buzan, D., Mollo, D. C., Yang, D., Lee, D., Schrader, D., Shutova, E., Cubuk, E. D., Segal, E., Hagerman, E., Barnes, E., Donoway, E., Pavlick, E., Rodolà, E., Lam, E., Chu, E., Tang, E., Erdem, E., Chang, E., Chi, E. A., Dyer, E., Jerzak, E. J., Kim, E., Manyasi, E. E., Zheltonozhskii, E., Xia, F., Siar, F., Martínez-Plumed, F., Happé, F., Chollet, F., Rong, F., Mishra, G., Winata, G. I., de Melo, G., Kruszewski, G., Parascandolo, G., Mariani, G., Wang, G., Jaimovitch-López, G., Betz, G., Gur-Ari, G., Galijasevic, H., Kim, H., Rashkin, H., Hajishirzi, H., Mehta, H., Bogar, H., Shevlin, H., Schütze, H., Yakura, H., Zhang, H., Wong, H. M., Ng, I., Noble, I., Jumelet, J., Geissinger, J., Kernion, J., Hilton, J., Lee, J., Fisac, J. F., Simon, J. B., Koppel, J., Zheng, J., Zou, J., Kocon, J., Thompson, J., Wingfield, J., Kaplan, J., Radom, J., Sohl-Dickstein, J., Phang, J., Wei, J., Yosinski, J., Novikova, J., Bosscher, J., Marsh, J., Kim, J., Taal, J., Engel, J. H., Alabi, J., Xu, J., Song, J., Tang, J., Waweru, J., Burden, J., Miller, J., Balis, J. U., Batchelder, J., Berant, J., Frohberg, J., Rozen, J., Hernández-Orallo, J., Boudeman, J., Guerr, J., Jones, J., Tenenbaum, J. B., Rule, J. S., Chua, J., Kanclerz, K., Livescu, K., Krauth, K., Gopalakrishnan, K., Ignatyeva, K., Markert, K., Dhole, K. D., Gimpel, K., Omondi, K., Mathewson, K., Chiafullo, K., Shkaruta, K., Shridhar, K., McDonell, K., Richardson, K., Reynolds, L., Gao, L., Zhang, L., Dugan, L., Qin, L., Ochando, L. C., Morency, L., Moschella, L., Lam, L., Noble, L., Schmidt, L., He, L., Colón, L. O., Metz, L., Senel, L. K., Bosma, M., Sap, M., ter Hoeve, M., Farooqi, M., Faruqui, M., Mazeika, M., Baturan, M., Marelli, M., Maru, M., Ramírez-Quintana, M. J., Tolkiehn, M., Giulianelli, M., Lewis, M., Potthast, M., Leavitt, M. L., Hagen, M., Schubert, M., Baitemirova, M., Arnaud, M., McElrath, M., Yee, M. A., Cohen, M., Gu, M., Ivanitskiy, M. I., Starritt, M., Strube, M., Swedrowski, M., Bevilacqua, M., Yasunaga, M., Kale, M., Cain, M., Xu, M., Suzgun, M., Walker, M., Tiwari, M., Bansal, M., Aminnaseri, M., Geva, M., Gheini, M., T., M. V., Peng, N., Chi, N. A., Lee, N., Krakover, N. G., Cameron, N., Roberts, N., Doiron, N., Martinez, N., Nangia, N., Deckers, N., Muennighoff, N., Keskar, N. S., Iyer, N., Constant, N., Fiedel, N., Wen, N., Zhang, O., Agha, O., Elbaghdadi, O., Levy, O., Evans, O., Casares, P. A. M., Doshi, P., Fung, P., Liang, P. P., Vicol, P., Alipoormolabashi, P., Liao, P., Liang, P., Chang, P., Eckersley, P., Htut, P. M., Hwang, P., Milkowski, P.,

Patil, P., Pezeshkpour, P., Oli, P., Mei, Q., Lyu, Q., Chen, Q., Banjade, R., Rudolph, R. E., Gabriel, R., Habacker, R., Risco, R., Millière, R., Garg, R., Barnes, R., Saurous, R. A., Arakawa, R., Raymaekers, R., Frank, R., Sikand, R., Novak, R., Sitelew, R., LeBras, R., Liu, R., Jacobs, R., Zhang, R., Salakhutdinov, R., Chi, R., Lee, R., Stovall, R., Teehan, R., Yang, R., Singh, S., Mohammad, S. M., Anand, S., Dillavou, S., Shleifer, S., Wiseman, S., Gruetter, S., Bowman, S. R., Schoenholz, S. S., Han, S., Kwatra, S., Rous, S. A., Ghazarian, S., Ghosh, S., Casey, S., Bischoff, S., Gehrmann, S., Schuster, S., Sadeghi, S., Hamdan, S., Zhou, S., Srivastava, S., Shi, S., Singh, S., Asaadi, S., Gu, S. S., Pachchigar, S., Toshniwal, S., Upadhyay, S., Debnath, S. S., Shakeri, S., Thormeyer, S., Melzi, S., Reddy, S., Makini, S. P., Lee, S., Torene, S., Hatwar, S., Dehaene, S., Divic, S., Ermon, S., Biderman, S., Lin, S., Prasad, S., Piantadosi, S. T., Shieber, S. M., Misherghi, S., Kiritchenko, S., Mishra, S., Linzen, T., Schuster, T., Li, T., Yu, T., Ali, T., Hashimoto, T., Wu, T., Desbordes, T., Rothschild, T., Phan, T., Wang, T., Nkinyili, T., Schick, T., Kornev, T., Tunduny, T., Gerstenberg, T., Chang, T., Neeraj, T., Khot, T., Shultz, T., Shaham, U., Misra, V., Demberg, V., Nyamai, V., Raunak, V., Ramasesh, V. V., Prabhu, V. U., Padmakumar, V., Srikumar, V., Fedus, W., Saunders, W., Zhang, W., Vossen, W., Ren, X., Tong, X., Zhao, X., Wu, X., Shen, X., Yaghoobzadeh, Y., Lakretz, Y., Song, Y., Bahri, Y., Choi, Y., Yang, Y., Hao, Y., Chen, Y., Belinkov, Y., Hou, Y., Hou, Y., Bai, Y., Seid, Z., Zhao, Z., Wang, Z., Wang, Z. J., Wang, Z., and Wu, Z. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Trans. Mach. Learn. Res.*, 2023, 2023.

Stich, S. U. Local SGD converges fast and communicates little. In *International Conference on Learning Representations*, 2019.

Sun, G., Mendieta, M., Yang, T., and Chen, C. Conquering the communication constraints to enable large pre-trained models in federated learning. *arXiv*, 2022.

Tang, Z., Kang, X., Yin, Y., Pan, X., Wang, Y., He, X., Wang, Q., Zeng, R., Zhao, K., Shi, S., et al. Fusionllm: A decentralized llm training system on geo-distributed gpus with adaptive compression. *arXiv preprint arXiv:2410.12707*, 2024.

Wang, E., Chen, B., Chowdhury, M., Kannan, A., and Liang, F. FLINT: A platform for federated learning integration. In *Proceedings of the Sixth Conference on Machine Learning and Systems, MLSys 2023*. mlsys.org, 2023a.

Wang, G., Dang, C. X., and Zhou, Z. Measure contribution of participants in federated learning. In *IEEE International Conference on Big Data*. IEEE, 2019.

Wang, G., Qin, H., Jacobs, S. A., Holmes, C., Rajbhandari, S., Ruwase, O., Yan, F., Yang, L., and He, Y. Zero++: Extremely efficient collective communication for giant model training. *arXiv preprint arXiv:2306.10209*, 2023b.

Wang, J., Charles, Z., Xu, Z., Joshi, G., McMahan, H. B., y Arcas, B. A., Al-Shedivat, M., Andrew, G., Avestimehr, S., Daly, K., Data, D., Diggavi, S. N., Eichner, H., Gadhikar, A., Garrett, Z., Girgis, A. M., Hanzely, F., Hard, A., He, C., Horváth, S., Huo, Z., Ingerman, A., Jaggi, M., Javidi, T., Kairouz, P., Kale, S., Karimireddy, S. P., Konečný, J., Koyejo, S., Li, T., Liu, L., Mohri, M., Qi, H., Reddi, S. J., Richtárik, P., Singhal, K., Smith, V., Soltanolkotabi, M., Song, W., Suresh, A. T., Stich, S. U., Talwalkar, A., Wang, H., Woodworth, B. E., Wu, S., Yu, F. X., Yuan, H., Zaheer, M., Zhang, M., Zhang, T., Zheng, C., Zhu, C., and Zhu, W. A field guide to federated optimization. *CoRR*, abs/2107.06917, 2021.

Wang, T., Rausch, J., Zhang, C., Jia, R., and Song, D. A principled approach to data valuation for federated learning. *Federated Learning: Privacy and Incentive*, 2020.

Xu, M., Cai, D., Wu, Y., Li, X., and Wang, S. {FwdLLM}: Efficient federated finetuning of large language models with perturbed inferences. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 579–596, 2024.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. Ties-merging: Resolving interference when merging models. In *NeurIPS*, 2023.

Yang, F., Peng, S., Sun, N., Wang, F., Wang, Y., Wu, F., Qiu, J., and Pan, A. Holmes: Towards distributed training across clusters with heterogeneous NIC environment. In *Proceedings of the 53rd International Conference on Parallel Processing ICPP 2024*, pp. 514–523. ACM, 2024.

Yao, D., Pan, W., O'Neill, M. J., Dai, Y., Wan, Y., Jin, H., and Sun, L. Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization. *arXiv preprint arXiv:2111.14655*, 2021.

Ye, R., Wang, W., Chai, J., Li, D., Li, Z., Xu, Y., Du, Y., Wang, Y., and Chen, S. Openfedllm: Training large language models on decentralized private data via federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6137–6147, 2024.

Yoon, J., Park, G., Jeong, W., and Hwang, S. J. Bitwidth heterogeneous federated learning with progressive weight dequantization. In *International Conference on Machine Learning*, pp. 25552–25565. PMLR, 2022.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence?

In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472.

Zhang, Z., Yang, Y., Dai, Y., Wang, Q., Yu, Y., Qu, L., and Xu, Z. 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*, pp. 9963–9977. Association for Computational Linguistics (ACL), 2023.

Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., Desmaison, A., Balioglu, C., Damania, P., Nguyen, B., Chauhan, G., Hao, Y., Mathews, A., and Li, S. Pytorch FSDP: experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, 2023. doi: 10.14778/3611540.3611569.

Zhou, Y., Ram, P., Salonidis, T., Baracaldo, N., Samulowitz, H., and Ludwig, H. Single-shot general hyper-parameter optimization for federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Appendix

In this appendix, we provide the details omitted in the main paper and more analyses and discussions.

- Appendix A: Hyperparameters we used for various experiments in our paper, including architectural details and both centralized and federated hyperparameters.
- Appendix B: Implementation details, which include i) full algorithms (pseudo-codes) of the proposed methods (Appendix E); ii) implementation of wall time in the paper (subsection B.1).
- Appendix C: Additional discussions that are helpful for the readers to better understand the background, including federated optimization of LLM pre-training (subsection C.1).
- Appendix D: Additional evaluations of the systems, e.g., the downstream evaluations.
- Appendix E: Full algorithms for Distributed Data Parallelism (DDP) and cross-silo federated learning.

A Hyper-parameters

As shown in Table 5, we trained models ranging in size from 125 million parameters to 7 billion for the causal language modeling task. We used the tokenizer presented in (Black et al., 2022) with a vocabulary size of 50 368. The local optimizer the clients use in our experiments is AdamW (Loshchilov & Hutter, 2019), while the server optimizer is FedMom (Huo et al., 2020). For all of our non-DiLoCo experiments, we default to FedAvg with server learning rate 1.0 and server momentum 0.0. The hyperparameters we used are reported in Table 6. We chose to train decoder-only models, although our system could train any LLM architecture because they have become the de-facto standard in language modeling and text generation owed to their sample efficiency.

We also note that the billion-scale experiments assume intermittent client availability, reflecting real-world scenarios in which participants may occasionally allocate free computing resources to federated pre-training. To accommodate this, we employ a stateless local optimization procedure, and resetting optimizer states each round. This enables Photon to operate seamlessly in sparse-compute scenarios, unlike standard distributed data parallelism (DDP), which requires fully dedicated and synchronized GPU workers. Stateless local optimization also eliminates the communication costs of synchronizing optimizer states, making it easier to ensure that federated pre-training remains compute-bound.

Table 5. Architecture details and local training parameters for our 125M, 350M, 1.3B, 3B, and 7B models. We report the number of transformer blocks, hidden model dimension d, number of attention heads, the linear layer expansion ratio, and Adam's parameters (β_1 and β_2). We also report the vocabulary size of the tokenizer we used (Black et al., 2022) and the sequence length l.

Model Size	#Blocks	d	#Heads	Exp. Ratio	(β_1, β_2)	|Vocab|	l
75M	3	896	16	4	(0.9, 0.95)	50 368	1024
125M	12	768	12	4	(0.9, 0.95)	50 368	2048
350M	24	1024	16	4	(0.9, 0.95)	50 368	2048
1.3B	24	2048	16	4	(0.9, 0.95)	50 368	2048
3B	32	2560	20	4	(0.9, 0.95)	50 368	2048
7B	32	4096	32	4	(0.9, 0.95)	50 368	2048

Table 6. Hyperparameters used in our experiments. The federated learning rate $\boldsymbol{\eta_s}$ and momentum $\boldsymbol{\mu_s}$ (Huo et al., 2020) are applied by the Photon Agg. $\mathbf{S_C}$ are the parameters of the learning rate scheduler synchronized across **sequential** steps. α is the factor to be applied to the maximum learning rate η_{max} to obtain the minimum learning rate for the cosine scheduler, i.e., $\eta_{min} = \alpha \times \eta_{max}$. T is the duration, in steps, of the cosine scheduler for fed/cent variants. We also report the batch size used in the local training by the Photon clients and the centralized batch size.

Model Size	η_s	μ_s	α	η_{max}	T	T_{cent}	Batch Size	Batch Size Cent
125M	$\{0.0, 0.1, 0.3, 0.5, 0.7, 1.0\}$	$\{0.9, 0.0\}$	10^{-1}	6.0×10^{-4}	40 960	5120	32	256
1.3B	1.0	0.0	10^{-1}	2×10^{-4}	24 800	24 800	512	512
3B	1.0	0.0	10^{-1}	1.6×10^{-4}	51 500	51 500	512	512
7B	1.0	0.0	10^{-1}	1.2×10^{-4}	63 900	63 900	1024	1024

B Implementation Details

B.1 Modeling Wall Time

We implement a comprehensive wall time model to analyze the temporal efficiency of our federated learning system across different communication architectures. The wall time calculations account for both computation and communication costs, considering factors such as local training time, model broadcast time, gradient collection time, and aggregation overhead.

Local Training Time The local training time (T_L) for each client is determined by the number of local training steps and the client's computational throughput:

$$T_L = \frac{\tau}{\nu}, \tag{1}$$

where τ represents the number of local training steps and ν is the local throughput measured in batches per second. Notably, T_L doesn't scale with the number of clients per round K as we assume the ideal case where they all execute the same local training recipe in parallel on equipollent hardware. In our experiments, τ represents a hyperparameter that we vary to observe its influence on the final performance. During deployment, τ is one of the most important

Table 7. Hyperparameters for our federated experiments. P represents the total number of clients per federations, K the number of clients sampled per round, D the dataset, τ the number of steps per round.

Model Size	P	K	D	τ
125M	$\{1, 2, 4, 8, 16\}$	$\{1, 2, 4, 8, 16\}$	C4 (Raffel et al., 2020), The Pile (Gao et al., 2020)	$64, 128, 512$
1.3B	8	8	C4 (Raffel et al., 2020)	500
3B	4	4	C4 (Raffel et al., 2020)	500
7B	4	4	C4 (Raffel et al., 2020)	500

hyperparameters to tune to achieve a pre-defined objective, i.e., target perplexity value at some target wall time. The value of ν depends on the computing resources available and the distributed strategy that Photon adopts at local clients. Throughout our evaluation of the 125M parameter model, we used an empirical value of $\nu = 2$ batches per second for both centralized and federated models. For the 1B model, we used an empirical ν value of 0.147 for federated models and 0.839 for centralized models. For the 3B model, we used $\nu = 0.144$ and 0.395 respectively, and for the 7B model, we used $\nu = 0.032$ and $\nu = 0.12$.

Communication Time The communication overhead varies based on the chosen architecture. We implement a bandwidth scaling factor for systems with more than θ channels (default: 100) to account for network congestion.

For Parameter Server (PS) architecture, the total communication time (T_C^P) is:

$$T_C^{PS} = \begin{cases} \frac{KS}{B} & \text{if } K \leq \theta; \\ \frac{KS}{B} & \text{if } K > \theta, \end{cases} \tag{2}$$

where:

- K is the number of clients per round;
- S is the model size in megabytes;
- B is the server bandwidth in MBps.

For AllReduce (AR) architecture, the communication time (T_C^{AR}) is:

$$T_C^{AR} = \frac{(K-1)S}{B}, \tag{3}$$

For Ring AllReduce (RAR), we optimize the communication pattern, resulting in:

$$T_C^{RAR} = \frac{2S(K-1)}{KB}. \tag{4}$$

We admit that accounting for congestion and real-world measurements could further improve these models. However, we find them to provide sufficiently accurate results.

Total Wall Time The total wall time for one round (T_r) combines local computation and communication costs:

$$T_r^\alpha = T_L + T_C^\alpha, \tag{5}$$

where $\alpha \in \{PS, AR, RAR\}$ represents the chosen architecture.

The total wall time for the entire training process (T) is:

$$T^\alpha = RT_r^\alpha, \tag{6}$$

where R is the total number of federated learning rounds.

The aggregation time T_{agg} is calculated by:

$$T_{agg} = \frac{KS}{\zeta}, \tag{7}$$

where ζ is the server computational capacity. The default value of ζ is 5TFLOPS per second. For simplicity, the aggregation time at the server is currently considered negligible compared to communication costs, as shown in Equation 6. Still, the model allows for future extensions to include server-side computational overhead in cases where its computational capabilities are highly constrained. Our implementation accounts for exceptional cases as well, such as single-client scenarios without communication.

B.2 Performance impact of the *Link* component

The *Link* component provides the crucial connection between the aggregator (Agg) and client (LLM-C). The bandwidth available to the link of each LLM-C dictates how quickly the model parameters can be exchanged for aggregation. As discussed in Appendix B.1, in standard Distributed Data Parallel (DDP) training, gradients have to be synchronized before every gradient descent step. The most efficient implementations use the *Ring-AllReduce* algorithm to synchronize gradients while having workers communicate in a peer-to-peer fashion using RDMA network, such as `InfiniBand` and `RoCE`. When using the same aggregation/synchronization algorithm, e.g., *Ring-AllReduce*, for both DDP and Photon, the *Link* bandwidth determines the gap in communication time between the two methodologies. Using the methodology presented in Appendix B.1 and assuming a given aggregation algorithm, one can estimate the communication time for a given model size and bandwidth. Factoring in that Photon communicates less frequently by a constant factor τ (e.g., $500\times$), the minimal *Link* bandwidth

B_{Photon} required for Photon to match DDP's communication time at bandwidth B_{DDP} follows from:

$$B_{\text{Photon}} \geq \frac{B_{\text{DDP}}}{\tau}.$$

This ignores the optimization aspects of model training, which may increase the compute time of Photon, which is why it represents a mere minimum bandwidth. If the above inequality does not hold, it is necessary to either increase the available bandwidth or to make the communication of Photon more infrequent, which may impact the machine learning performance.

B.3 Overlapping communication and cleaning up

When partial participation is involved in the federated setting, clients may sporadically become available or drop out of the federation at any time. When they disconnect after they finish executing their local work for a specific federated round, the Photon LLM Client can offload the communication process and simultaneously clean up the memory allocated by the training pipeline to allow for the prompt return to idle state. This routine quickly makes computing resources available for another workload, which is particularly important when using shared computing facilities.

B.4 Advanced sharing for reducing memory footprint

Every Photon LLM Client comprises a multiprocessing stack managed by a leader process that coordinates subordinate processes handling the hardware accelerators. Such a leader process is also in charge of the communication endpoint, so it receives and sends model parameters as the algorithm requires. To minimize the RAM footprint up to $8\times$, the model parameters exchanged are stored in shared memory, accessible by all subordinate processes.

C Extended Discussion

C.1 Federated Optimization of LLM Pre-training

Federated optimization differs significantly from standard data-parallel training due to infrequent synchronization, which affects the multitude of assumptions upon which centralized pre-training of LLMs is built. In a centralized context, previous works have shown the following: (a) the number of parameters $|\theta|$ and the number of tokens T seen by the model should be scaled roughly equally (Hoffmann et al., 2022a) for compute-optimal training; (b) the batch size $\mathcal{B}$ should be chosen based on the available hardware resources, with larger batch sizes providing benefits until a critical batch size $\mathcal{B}_{\text{crit}}$, is reached (McCandlish et al., 2018); (c) the learning rate should be scheduled using cosine decay with a period equal to the total number of optimization-steps/batches $T/\mathcal{B}$. All of these components need to be modified for effective federated optimization.

From a theoretical perspective, infrequent parameter averaging methods such as Local SGD (Lin et al., 2020) or FedAvg (McMahan et al., 2017) are expected to provide an effect similar to scaling the batch size in a centralized setting (Lee et al., 2020) when scaling the number of clients per round, however, given the many moving parts of the centralized recipe obtaining such improvements requires successfully adapting it to a federated context. We need to distinguish between the batch size of a given client $\mathcal{B}_c$ and the effective batch size of a given round $\mathcal{B}_{\text{eff}} = \sum_{c \in C_r} B_c$, which depends on the batch size of all sampled clients. While smaller batch sizes are known to provide generalization benefits (Keskar et al., 2017), for the sake of efficiency, using the largest batch size that can fit inside a given accelerator is preferable. Thus, we assume that each client uses a fixed B_c determined by their hardware and that, for the sake of simplicity, all clients have access to the same hardware

$$\mathcal{B}_i = B_j \,, \forall i, j \in C \times C.$$

In cases where clients have sufficiently powerful hardware, we assume that they use a batch size $\mathcal{B}_c = \mathcal{B}_{\text{crit}}$ to avoid wasting compute. With this simplifying assumption, the compute-time trade-off in federated optimization depends only on the number of clients sampled in a given round $|C_r|$ and the number of local iterations T_c performed on each client, both assumed constant. In the case of $T_c = 1$, this coincides, when assuming full participation, with the centralized setting, allowing the critical batch size $\mathcal{B}_{\text{crit}}$ to be determined using the gradient noise scale as done in the work of McCandlish et al. (2018). We perform numerous experiments to understand how the number of local steps T_c changes the Pareto-frontier of the compute-time trade-off.

Assuming that the findings of Hoffmann et al. (2022a) hold in a federated context, the compute-optimal number of total steps, $T = R \times T_c$, that should be performed depends on the number of tokens appropriate for the given model size, roughly $20 \times |\theta|$ according to the work of Hoffmann et al. (2022a), and on the effective batch size as follows:

$$R \times T_c = \frac{20 \times |\theta|}{\mathcal{B}_{\text{eff}}}, \tag{8}$$

with the large caveat that this compute-optimal point was chosen, assuming that training would be conducted using the centralized critical batch size. However, accounting for this in the learning rate schedule is not trivial for two reasons. First, the averaging-based aggregation procedure will limit the impact of any individual update. This is true both from a simple mathematical perspective, the norm of the average update being less than the average of the update norms, and because client updates in federated learning

have been shown to produce near-orthogonal updates which tend to result in small pseudo-gradient norms (Charles et al., 2021). Second, clients take several optimization steps using their hardware-determined batch size before aggregation, with smaller batch sizes being known to require smaller learning rates (McCandlish et al., 2018) in centralized settings. Since we expect the client hardware batch size to be generally smaller than $\mathcal{B}_{\text{crit}}$, they are likely to fall in the regime of small-batch training. Thus, in our work, we propose decaying the learning rate following a cosine scheduler appropriate for the hardware batch size B_c, using Eq. (8) to obtain the period by replacing $\mathcal{B}_{\text{eff}}$ with $\mathcal{B}_c$. Having fixed this period, we only have to tune one hyperparameter in the form of the maximum learning rate $\eta_{\max}$ with the minimum learning rate being computed as $\eta_{\min} = \frac{\eta_{\max}}{10}$. In contrast, we find that neither square root nor linear learning rate scaling (Granziol et al., 2022a;b) sufficiently stabilize centralized training across varying batch sizes.

The momenta stabilize the local optimization direction for a given local momentum-based optimizer, such as AdamW (Loshchilov & Hutter, 2019). Since they are generally implemented using exponential decay, the impact of any individual gradient step is reduced. In the case of federated optimization, this poses a challenge as we aim to update the momentum vectors to reflect the information received during the aggregation step. Directly communicating and averaging the momenta of all clients would increase the communication costs by the number of momenta of the optimizer relative to transmitting only the parameters, e.g., it would be three times higher for AdamW. To avoid such increases in communication costs, we keep the momenta local and rely on only the parameter update to regularize training.

C.2 Recent Advances in Federated Optimization

Recent works such as Cheng & Glasgow (2025) and Iacob et al. (2025) have shown that federated optimization algorithms can be competitive with standard Distributed Data-Parallel methods in specific circumstances.

Cheng & Glasgow (2025) prove the convergence of a Local AdamW variant that averages both model parameters and optimizer states every round. They further show that, depending on the task and data distribution, local-update optimizers can converge faster than standard minibatch SGD, particularly under IID data across workers/clients (where update variance is low).

Iacob et al. (2025) demonstrate the effectiveness of federated optimization even when data heterogeneity is high. They observe that the noise introduced by averaging model updates from diverse data distributions can yield a more robust set of parameters for the transformer body, potentially improving generalization or adaptation to new data distributions.

D Additional Evaluation

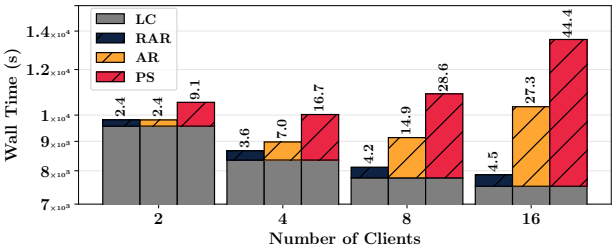


Figure 9. We report the split of the total wall time in two parts: the local compute time (LC) the clients endure to achieve the desired perplexity value and the communication time. The communication time is reported for three different aggregation implementations: *parameter server* (PS), which is necessary when privacy constraints are present; *AllReduce* (AR), more scalable than PS; *Ring-AllReduce* (RAR), the most scalable approach bounded by the slowest link across the ring topology. As we expected, communication represents a more important cost as the number of clients increases. Still, when implemented efficiently (RAR), the wall time benefits of scaling the computing resources are maintained. At the top of each bar, we report the percentage of time spent communicating for the respective experiments and implementation.

D.1 Communication Efficiency and Scalability

We present additional results on the communication efficiency and scalability of Photon when using 64 local steps per round(Fig. 9) and 128 local steps per round(Fig. 10). While using 128 steps results in a slightly increased total computational load due to minor reductions in convergence speed, reducing communication frequency by half significantly lowers the communication burden, particularly with higher numbers of clients per round. This trend is especially pronounced in communication-inefficient PS implementations and also applies to the faster RAR and AR methods.

D.2 Photon Robustness to Node Failures

In centralized data center training, strong synchronization and a fixed communication topology mean that a single hardware failure can halt training, requiring a complete restart from a past checkpoint. Such hardware failures, even limited to one accelerator, are common and account for 98% of training restarts (Dubey et al., 2024). In contrast, our federated approach offers a more robust and communication-efficient alternative to distributed data parallelism (DDP). Photon only needs one pseudo-gradient update to progress a federated round while asynchronously restarting edge components (LLM Clients), unlike centralized systems that require a full restart to reinitialize the distributed process group (Li et al., 2020a). Thus, Photon is completely robust to any fail-

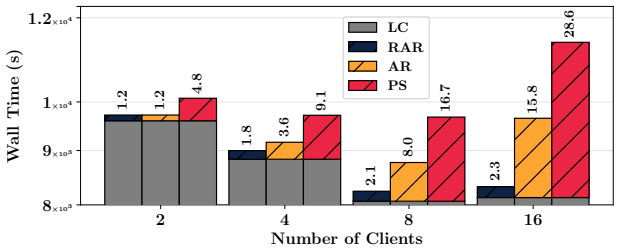


Figure 10. We report the split of the total wall time in two parts: the local compute time (LC) the clients endure to achieve the desired perplexity value and the communication time. The communication time is reported for three different aggregation implementations: *parameter server* (PS), which is necessary when privacy constraints are present; *AllReduce* (AR), more scalable than PS; *Ring-AllReduce* (RAR), the most scalable approach bounded by the slowest link across the ring topology. As we expected, communication represents a more important cost as the number of clients increases. Still, when implemented efficiently (RAR), the wall time benefits of scaling the computing resources are maintained. At the top of each bar, we report the percentage of time spent communicating for the respective experiments and implementation.

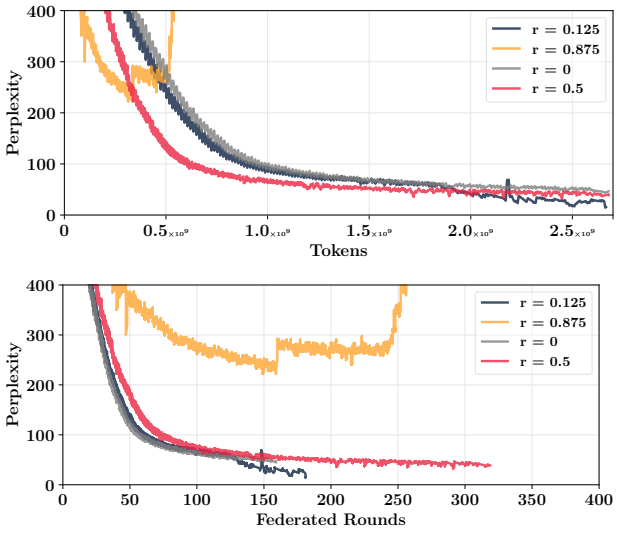


Figure 11. The robustness of Photon for IID data distributions across clients. We show the training perplexity against (top) the number of tokens trained and (bottom) the number of federated rounds for different dropout ratios $r \in \{0, 0.125, 0.5, 0.875\}$ corresponding to $\{0, 1, 4, 7\}$ clients dropping out at every federated round respectively. With the highest value of r, the training procedure fails to converge as there is not sufficient training data per round to leverage the hyperparameter setting. For all other values of r, the federated training succeeds, potentially reaching a final better perplexity with the same number of total tokens. However, values of $r > 0$ result in longer training executions as the number of federated rounds to reach the target number of total tokens increases proportionally to the number of clients dropping out.

ure affecting less than 100% of the system. The decoupling between LLM Clients and the data source, along with the Aggregator's ability to seamlessly add new Clients, allows Photon to maintain the same process group even as Clients are added or removed.

To test the robustness of Photon against node failures, we run a series of experiments simulating various node dropout ratios. We configured the federated pre-training of a 125M parameters model as if we were not expecting any failure between our clients: 8 clients per round, 32 samples in each local batch, 32 local steps per round, and a target number of total tokens to train on equal to $\sim 2.5 \times 10^9$ (5120 sequential steps and 160 federated rounds at full client participation), i.e., 20 token per parameter considering that we used a model with 125M parameters. The other training hyperparameters were the same as the main paper experiments referring to the 125M parameter models unless stated otherwise.

This setting corresponds to the centralized environment where at least a GPU in any node fails every 32 training steps, for a total of $N_{\text{failures}} = \frac{5120}{32} = 160$, i.e., a rate of 3.125% failures per step. Notably, for the standard centralized approach, different numbers of GPUs or nodes failing may have the same impact on the training procedure as the entire process group often needs to be restarted.

We model IID data distribution across clients by randomly partitioning the `C4` (Raffel et al., 2020) dataset uniformly into 8 equally sized shards. For nonIID experiments, unlike the main work, which uses the well-known `C4` dataset, we adopt the newer state-of-the-art data mixture used by `SmolLM-V2` (Allal et al., 2025), specifically we randomly partition each of the following datasets into 16 IID shards: (1) `FineWeb-EDU` (Penedo et al., 2024), a high-quality general language dataset sourced from `Common Crawl`, (2) `Cosmopedia V2` (Ben Allal et al., 2024), a synthetic dataset generated by the `Mixtral-8x7B-Instruct-v0.1` model, (3) `Python-EDU`, a high-quality subset of `The Stack V2` (Lozhkov et al., 2024) code dataset, (4) `FineMath 4+` (Allal et al., 2025), a high-quality math subset of `Common Crawl`, (5) `Infi-WebMath 4+`, a high-quality variant `Infi-WebMath` (Han et al., 2024) released by Allal et al. (2025). Following the recipe of Allal et al. (2025), we then compose shards to construct clients whose data is comprised of: 70% `FineWeb-EDU` data, 10% `Cosmopedia V2`, 10% `Python-EDU`, 5% `FineMath 4+`, and 5% `Infi-WebMath 4+`.

The relevant comparisons we show in this evaluation relate to how the convergence, in terms of local training perplexity, is impacted by the absence of updates due to clients dropping out. Figures 11 and 12 show that only extreme and unrealistic dropout ratios ($r = 0.875$) can completely

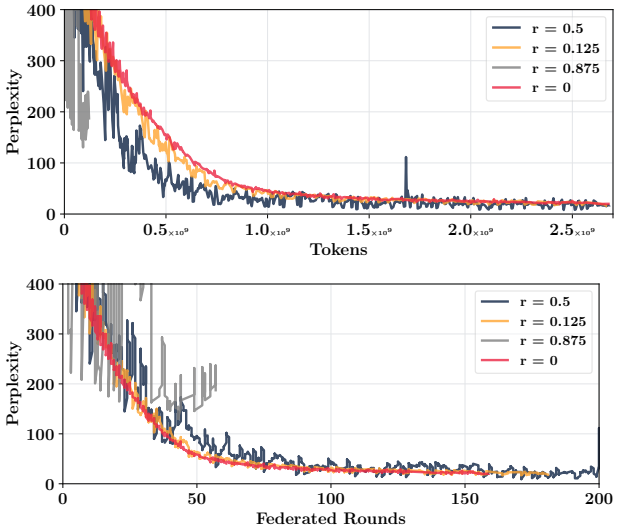


Figure 12. The robustness of Photon for non-IID data distributions across clients. We show the training perplexity against (top) the number of tokens trained and (bottom) the number of federated rounds for different dropout ratios $r \in \{0, 0.125, 0.5, 0.875\}$ corresponding to $\{0, 1, 4, 7\}$ clients dropping out at every federated round respectively. With the highest value of r, the training procedure fails to converge as there is not sufficient training data per round to leverage the hyperparameter setting. For all other values of r, the federated training succeeds, potentially reaching a final better perplexity with the same number of total tokens. However, values of $r > 0$ result in longer training executions as the number of federated rounds to reach the target number of total tokens increases proportionally to the number of clients dropping out.

disrupt the training independently on the heterogeneity of the data distributions. Notably, for the other values of r and for both IID and non-IID data, more dropouts correspond to better final perplexity when effectively training on the same number of total tokens, i.e., executing far more federated rounds (taking more time). When comparing the final perplexity at different values of r with the number of federated rounds, which are directly proportional to the real wall time, more clients dropping out result in longer training times to achieve the target number of total tokens, as expected.

The takeaways of this evaluation are: (1) federated training converges for all dropout ratios $r < 0.875$, making it suitable for highly unreliable hardware configurations, (2) since nodes train in isolation, a node failure does not require interrupting the entire federated round, rather it only reduces the number of pseudo-gradients used for an update, (3) to compensate for such failures, it is sufficient to extend training until the target number of tokens is reached, and (4) configurations with higher dropout ratios correspond to a reduction in the effective batch size of the training, which may improve the final performance at the cost of longer training times.

Table 8. In-context learning comparison between Photon models. Our biggest model wins 3 out of 3 comparisons in this group.

Name	ARC-Challenge (Clark et al., 2018)	BigBench QA Wikidata (Srivastava et al., 2023)	HellaSwag (Zellers et al., 2019)
Photon-7B	**0.265**	**0.447**	**0.524**
Photon-3B	0.247	0.360	0.455
Photon-1B	0.243	0.215	0.349

Table 9. In-context learning comparison between Photon models. Our biggest model wins 3 out of 3 comparisons in this group.

Name	PIQA (Bisk et al., 2020)	Winogrande (Sakaguchi et al., 2020)	ARC-Easy (Clark et al., 2018)
Photon-7B	**0.729**	**0.522**	**0.508**
Photon-3B	0.705	0.512	0.461
Photon-1B	0.676	0.516	0.390

D.3 Downstream evaluation of Photon's models

To evaluate the downstream task performance of our models, we test across a series of in-context learning benchmarks. Our results, shown in Tables 8 to 11, demonstrate that the downstream performance of models trained with Photon scales as expected with model size, with our largest model winning 10 out of 14 comparisons. This proves the downstream utility of Photon models even when using a pre-training dataset not optimized for downstream performance. We expect that as we increase the model size and incorporate a broader and more qualitative data mixture, the downstream performance of Photon models will keep improving.

Table 10. In-context learning comparison between Photon models. Our biggest model wins 2 out of 3 comparisons in this group.

Name	BoolQ (Clark et al., 2019)	Openbook QA (Mihaylov et al., 2018)	Winograd (Lo et al., 2023)
Photon-7B	0.530	**0.358**	**0.681**
Photon-3B	0.591	0.316	0.656
Photon-1B	**0.612**	0.274	0.604

Table 11. In-context learning comparison between Photon models. Our biggest model wins 3 out of 4 comparisons in this group.

Name	LAMBADA (OpenAI) (Paperno et al., 2016)	Bigbench Strategy QA (Srivastava et al., 2023)	COPA (Roemmele et al., 2011)	MMLU (Hendrycks et al., 2021)
Photon-7B	**0.457**	0.466	**0.710**	**0.263**
Photon-3B	0.381	0.464	0.620	0.252
Photon-1B	0.298	**0.470**	0.630	0.248

E FULL ALGORITHMS

Algorithm 2 Distributed Data Parallel (DDP) Training Algorithm

Require: N: Number of devices (workers), f_θ: Model with parameters θ,T: Number of epochs
Require: $\mathcal{D}$: Dataset partitioned across devices $\mathcal{D}_i$ where $i \in \{1, 2, \ldots, N\}$
Require: `RingAllReduce`: All-reduce operation to aggregate across devices on a ring
Require: `Opt`: Optimizer for updating θ with gradients
Initialize: Randomly initialize model parameters θ_0 on each device
for $t = 1$ to T **do**
    Step 1: Parallel Local Training
    for each device $i \in \{1, 2, \ldots, N\}$ **in parallel do**
        Compute local mini-batch loss $\mathcal{L}_i(\theta_{t-1}, \mathcal{D}_i)$
        Compute local gradients $\nabla_{\theta_{t-1}} \mathcal{L}_i(\theta_{t-1})$
    Step 2: Distributed `RingAllReduce` Gradient Aggregation
    $\nabla_{\theta_{t-1}} \mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\theta_{t-1}} \mathcal{L}_i$
    Each device now possesses the global gradient $\nabla_{\theta_{t-1}} \mathcal{L}$
    Step 3: Parallel Model Update
    for each device $i \in \{1, 2, \ldots, N\}$ **in parallel do**
        $\theta_t = \texttt{Opt}(\theta_{t-1}, \nabla_{\theta_{t-1}} \mathcal{L})$
Output: Trained model parameters θ_T

Algorithm 3 Cross-silo Federated Learning (FL) Algorithm

Require: N: Number of clients, f_θ: Model with parameters θ
Require: T: Number of federated rounds, K: Number of local steps
Require: $\{\mathcal{D}_i\}$: Federated dataset, i.e., a set of private $\mathcal{D}_i$, $i \in \{1, \ldots, N\}$
Require: `ClientOpt`: local client optimizer, `ServerOpt`: server optimizer
Initialize: Randomly initialize global model parameters θ_0 on the server
for $t = 1$ to T **do**
    Step 1: Broadcast model parameters
    Server sends θ_t to all N clients
    Step 2: Parallel Local Training
    for each client $i \in \{1, 2, \ldots, N\}$ **in parallel do**
        $\omega_{i,0} \leftarrow \theta_{t-1}$
        for each local iteration $k \in \{1, 2, \ldots, K\}$ **do**
            Compute local mini-batch loss $\mathcal{L}_i(\omega_{i,k-1}, \mathcal{D}_i)$
            Compute local gradients $\nabla_{\omega_{i,k-1}} \mathcal{L}_i(\omega_{i,k-1})$
            $\omega_{i,k} \leftarrow \texttt{ClientOpt}(\omega_{i,k-1}, \nabla_{\omega_{i,k-1}} \mathcal{L}_i(\omega_{i,k-1}))$
        $\Delta\theta_{t-1,i} \leftarrow \omega_{i,K} - \theta_{t-1}$
    Step 3: Global Model Update (on the server)
    $\theta_t = \texttt{ServerOpt}(\theta_{t-1}, \{\Delta\theta_{t-1,i}\})$
Output: Trained model parameters θ_T

F ARTIFACT APPENDIX

F.1 Abstract

This Artifact Appendix provides the instructions, scripts, and configurations necessary to run the experiments of our paper on federated large language model (LLM) pre-training using the *Photon* system. We focus on the script, `scripts/fed_125m_example.sh`, that orchestrates the entire process: downloading dependencies, launching the federated server, spinning up clients, and training a 125M-parameter model end to end. However, we recommend following carefully the `README.md` file and the provided example scripts for a more detailed understanding of the setup and execution. By running the `scripts/fed_125m_example.sh` script, users can witness how Photon handles Hydra-based configuration resolution, aggregator (server) bootstrapping, and client participation.

F.2 Artifact check-list (meta-information)

- **Algorithm:** LocalSGD-based federated optimization with integrated distributed data-parallel (DDP) or fully sharded data parallel (FSDP) when applicable.
- **Program:** Python scripts employing PyTorch, integrated with Flower (for federated coordination) and Ray for model updates communication.
- **Compilation:** No explicit compilation. A Python-based environment setup is mandatory.
- **Transformations:** Data tokenization, normalization, optional data pre-processing (compression), and partitioning in client shards.
- **Binary:** No direct binaries; entire artifact is Python-based.
- **Data set:** A small subset of C4 is included for demonstration. For larger training, full C4 or The Pile can be substituted (scripts not included here).
- **Run-time environment:** Linux system (Ubuntu 22.04 recommended), Python 3.11, CUDA(12.4)-enabled PyTorch 2.1.5, plus Hydra for configuration resolution.
- **Hardware:** At least one NVIDIA GPU (NVIDIA A40, RTX2080Ti, V100, A100, H100, etc.), stable network links (1–10Gbps) if multiple machines are used.
- **Run-time state:** Users can run everything on a single machine with multiple GPUs, or distribute across multiple nodes.
- **Execution:** A single script `scripts/fed_125m_example.sh` that performs the entire flow (setup, server launch, client launches, local training).
- **Metrics:** Primary metric is validation perplexity, with secondary metrics including GPU utilization, throughput, and communication overhead. Wandb logging is supported but requires custom configuration for which guidelines are provided in the code docstrings.
- **Output:** Model checkpoints, logs of training progress, final perplexity.
- **Experiments:** Demonstration of the federated pre-training and centralized training of a 125M-parameter decoder-only LLM, which can be scaled up if desired.
- **Disk space required:** Approximately $5/15$GB for the small subset of C4 plus checkpoints. (Larger experiments may require $300/1000$GB).
- **Time needed to prepare workflow:** Approximately 1 hour for environment setup, $30/60$ minutes to download and preprocess the small dataset.
- **Time needed to complete experiments:** A few hours for the 125M demonstration. Larger-scale runs can take days.
- **Publicly available:** Yes, code repository is licensed (Apache-2.0 license) and will be made public.
- **Code licenses (if publicly available):** Apache License 2.0.
- **Data licenses (if publicly available):** C4 is under the ODC-BY license.
- **Workflow framework used:** Flower + Ray + PyTorch + Hydra, plus a single orchestrating shell script.
- **Archived:** DOI on Zenodo.
- **Public permalink:** Flower Labs Research.

F.3 Description

F.3.1 How delivered

The artifact is provided in a zipped repository containing:

- `README.md`: A quick overview and key instructions.
- `scripts/system_setup.sh`: Installs base dependencies, sets up the environment.
- `scripts/convert_c4_dataset.sh`: Acquires a small version of C4 for demonstration. Prepare the dataset for training.
- `scripts/fed_125m_example.sh`:
  The single script that launches everything for a 125M-parameter model. It internally invokes Hydra-based configs for server and clients, then orchestrates the run.
- `scripts/cen_125m_example.sh`:
  The single script that launches centralized training of a 125M-parameter model. It internally invokes Hydra-based configs. It is prepared to operate on a single machine setup launching a parallelized training on the available GPUs.
- `configs/`: YAML files specifying hyperparameters (learning rate, batch size, etc.), aggregator properties, and Hydra overrides.

F.3.2 Hardware dependencies

- **GPU:**
  - For the 125M example, a single GPU with $\geq$12GB memory is sufficient, even though a larger memory ($\geq$40GB) is recommended.
  - For multi-node, each node should have a CUDA-capable GPU and at least 1–10Gbps network connectivity.

F.3.3 Software dependencies

- **OS:** Linux (Ubuntu 22.04+).
- **Python:** 3.11 or higher.
- **CUDA/CuDNN:** Version 12.4 is recommended, being compatible with PyTorch 2.1.5 and your specific GPU driver. These can be installed automatically via `scripts/system_setup.sh`
- **Package managers:** Poetry is supported for dependency management.
- **Libraries:** PyTorch 2.1.5, Flower (custom version), Ray, Hydra, and standard Python utilities (NumPy, Pandas, etc.). Installed automatically via the `scripts/system_setup.sh` and `scripts/install_env.sh` scripts.

F.3.4 Data sets

- A small subset of C4 is included for demonstration.
- It is fetched, unpacked locally, and tokenized by `scripts/convert_c4_dataset.sh`.

Users can later replace this with the full C4 or other corpora by adjusting parts of the code and configuration files.

F.4 Installation and Usage

Refer to the `README.md` file for a more detailed guide. Below is a quick start guide to run the federated pre-training of a 125M-parameter model.

System prep and environment:

1. **Download the code:** The code is maintained and made available through the Flower Labs Research webpage.
2. **Run the setup script:** Once the repository has been obtained, run the setup script to install the necessary dependencies and prepare the environment.

```
cd <path>/<to>/<photon>
cd scripts
. system_setup.sh
```

   This can install build tools, CUDA drivers (Ubuntu-based).

3. **Install dependencies:**

```
cd scripts
. install_env.sh
```

Download, prepare/convert dataset with the provided script.

```
bash scripts/convert_c4_dataset.sh
```

Run the single script for federate pre-training of the 125M model:

```
bash scripts/fed_125m_example.sh
```

This command executes the following steps internally:

- **Hydra configs interpretation:** Hydra interprets the configs and dumps them to a file that is read by the other processes. The file `photon/hydra_resolver.py` is used.
- **Launch Flower Superlink:** The command used is `poetry run flower-superlink`.
- **Launch Flower ServerApp:** The command used is `poetry run flower-server-app photon.server_app:app`.
- **Launch Flower ClientApps:** The command used is `poetry run flower-client-app photon.client_app:app`
- **Federated rounds:** The aggregator orchestrates local training (LocalSGD) across clients, synchronizes updates after each round.
- **Checkpoints and logs:** Intermediate global checkpoints and logs are saved in `checkpoints/` and `runs/` respectively.
- **Completion:** The script logs periodically several metrics, e.g., perplexity and throughput.

F.5 Evaluation and expected result

Targets of interest:

- **Validation perplexity:** For the 125M demo, you should observe perplexity dropping towards the low 40s or upper 30s after sufficient rounds, depending on configuration.

- **Runtime logs:** Both aggregator and client logs are found under `runs/`, indicating the number of tokens processed, average GPU utilization, and steps per round.
- **Checkpoints:** Partial and final checkpoints are saved in the `checkpoints/` folder.

F.6 Experiment customization

- **Config override:** Edit `scripts/fed_125m_example.sh` or pass Hydra overrides to change client count or hyperparameters.
- **Hardware scaling:** By default, the script spawns multiple clients on a single node. For multi-node, adapt the aggregator IP and client addresses in `scripts/fed_125m_example.sh`.
- **Batch sizes / epochs:** Controlled by Hydra configs in the `configs/` folder.
- **Dataset:** Replace the small C4 path with your own local data for more extended training.

F.7 Notes

- **Partial or intermittent clients:** If a client crashes or is not reachable, the aggregator continues with remaining clients in subsequent rounds.
- **Performance considerations:** For minimal overhead, ensure a stable GPU environment. Larger-scale runs (1.3B+) require more disk space, memory, and multi-GPU setups.

F.8 Methodology

We adhere to artifact evaluation guidelines:

- Single-blind AE with emphasis on reproducibility and clarity.
- Clear *build* (ffrom the scripts `scripts/system_setup.sh` and `scripts/install_env.sh`), *run* (using the script `scripts/fed_125m_example.sh`), and *analysis* (logs, final checkpoint) phases.
- **ACM Artifact Badging** best practices: code will be made public, well-documented, and tested on a standard environment.