The Future of Large Language Model Pre-training is Federated

Lorenzo Sani[1,2,†] **Alex Iacob**[1,2] **Zeyu Cao**[1,*] **Bill Marino**[1,*]

Yan Gao[1,2] **Tomas Paulik**[1] **Wanru Zhao**[1] **William F. Shen**[1]

Preslav Aleksandrov[1] **Xinchi Qiu**[1] **Nicholas D. Lane**[1,2]

Abstract

Generative pre-trained large language models (LLMs) have demonstrated impressive performance over a wide range of tasks, thanks to the unprecedented amount of data they have been trained on. As established scaling laws indicate, LLMs' future performance improvement depends on the amount of computing and data sources they can leverage for pre-training. Federated learning (FL) has the potential to unleash the majority of the planet's data and computational resources, which are underutilized by the data-center-focused training methodology of current LLM practice. Our work presents a robust, flexible, reproducible FL approach that enables large-scale collaboration across institutions to train LLMs. We propose a scalable deployment system called *Photon* to enable the investigation and development of this new training paradigm for LLM pre-training. We show that *Photon* can be used by organizations interested in collaborating with their private data sources and computational resources for pre-training LLMs with billions of parameters. This paradigm would mobilize more computational and data resources while matching or potentially exceeding centralized performance. We further show the effectiveness of the federated training scales with model size and present our approach for training a billion-scale federated LLM using limited resources. Furthermore, we demonstrate that LLM training is highly resilient to the classical challenges of federated statistical and hardware heterogeneity. Finally, we show that convergence is robust to partial participation, opening the avenue for compute-efficient collaborative training. *Photon* will help data-rich actors to become the protagonists of LLMs pre-training instead of leaving the stage to the compute-rich alone.

1 Introduction

The impressive performance of generative pre-trained large language models (LLMs) and their multi-modal derivations largely owes to their capacity to learn representations at scale [1]. Thus, a handful of resourceful tech companies and institutions are using increasingly powerful computing facilities in the race to scale up LLMs and dataset sizes to achieve state-of-the-art (SOTA) performance. The thousands of hours of training on thousands of specialized and well-connected acceler-

[†]Lead and corresponding author: ls985@cam.ac.uk
[*]Equal contributors
[1]Department of Computer Science and Technology, University of Cambridge
[2]Flower Labs

International Workshop on Federated Foundation Models In Conjunction with NeurIPS 2024 (FL@FM-NeurIPS 2024), Vancouver.

ators in a single data center incur a high energy and monetary cost [2]. Distributing training across multiple data centers in sparse geographical locations, for those companies who could afford it, would drive the cost even higher due to communication overheads [3, 4].

Hoffmann et al. [5] showed that the effective performance improvement of increasingly large LLMs requires increasingly extensive training datasets. Most of the world's data sources are unevenly distributed among private actors who often don't want to share it, even if the data regulations of their respective jurisdictions permit such sharing. Since no organization independently owns the rights to a sufficient amount of text data, the multi-terabyte datasets used in these procedures must be obtained from publicly available sources. These may include materials potentially protected by intellectual property laws [6] or be otherwise problematic [7, 8, 9, 10]. The potential means of collecting non-public data may require arranging independent deals with data providers [11] or taking model training directly to these private data sources. It is anticipated that, by 2026, the amount of high-quality data that LLMs will require to continue to improve their performance will exceed what is publicly available [12]. Where datasets' sizes become constrained, but model growth is unbounded, this can lead to memorization and data leakage amongst current LLMs [13, 14].

The next generation of LLMs and foundation models (FMs) will benefit from effectively leveraging more data and computational resources than the centralized paradigm currently makes available. Therefore, we argue that making FMs' future even brighter requires shifting the dominant training paradigm to a collaborative federated approach, where nodes control considerable (but not immense) computational or data sources according to their abilities. Using federated learning (FL), we can expand the quality of our models by gaining access to previously untapped data and computational sources [15]. This broader data availability will, in turn, allow us to increase the size of models that can be efficiently trained compared to the centralized paradigm and to avoid both memorization and data leakage.

As shown in previous works [16, 17, 18], FL can relax the synchronization requirements of stochastic gradient descent (SGD) to accommodate such poorly connected nodes. Also, more recent works [19, 20, 21] showed that Local SGD could substantially reduce the communication overhead of training LLMs in data center settings with homogeneous and heterogeneous computational nodes. As a step forward, we argue for an entirely federated approach to LLM training that can automatically balance the workload of multiple actors and privately merge the knowledge derived from their local datasets without divulging them directly to other participants.

We are the **first** to fully utilize a practical and production-ready **geographically distributed** federation of **heterogeneous devices** and **data sources** for reproducible generative pre-training of a **billion-scale LLM**. This work presents *Photon*, a complete system for pre-training federated LLMs in a collaborative, reproducible, and scalable fashion that can address the main challenges for a flex-

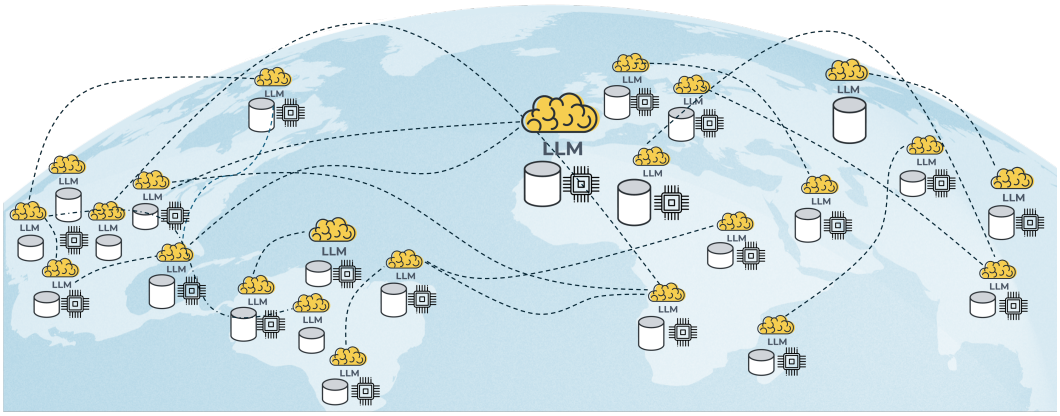


Figure 1: A hypothetical representation of the available data silos around the world. While scraping data from the web has taken foundation models quite far, most data remains under private entities' control. These organizations can collaborate in the federated generative pre-training of large language models to exploit their data towards the common goal of training LLMs they control. The collaborative nature of FL and its low communication requirements make this possible with only moderately powerful hardware, eliminating the prohibitive costs of pre-training.

ible execution in **arbitrary FL cross-silo settings**. We built it on the **open-source** FL framework *Flower* [22]. Furthermore, we make this available as a **training recipe** with a fully **transparent** experimental configuration accompanying our results. We emphasize that federated generative pre-training can be done with **affordable** hardware configurations by interlinking single nodes containing 1-8 GPUs from standard cloud providers —rather than renting multiple entire data centers concurrently. We also argue that federated systems can and must evolve beyond networks of clients holding both compute and data in a world of increasing hardware costs and competing privacy interests. To achieve this, *Photon* allows data providers (sources) to come together with compute providers in private partnerships among trusted parties without leaking any such data to other participants. We demonstrate that pre-training LLMs can be democratized through FL technologies, including data-rich actors **independent** of their **connectivity and computing resources**.

Developing this novel federated system led to numerous insights, including the fact that, in defiance of expectations, larger federated LLMs can more easily find a consensus across clients than their smaller counterparts. Our **results** show that:

1. Federated LLM training offers **competitive performance** with centralized training but with far **less communication overhead**.
2. Under our system, *Photon*, **larger models require less frequent communication** while receiving a more significant boost in generalization performance than smaller ones.
3. The classical challenges of statistical heterogeneity and partial participation are much less impactful for overparameterized LLMs than the small models typically used in the FL literature. This opens the door to compute-efficient collaboration at a massive scale.
4. By utilizing our local execution engine, *Pollen* [23], to balance loads across computational nodes, we can establish a training pipeline that is **faster and more robust than** its **centralized** counterpart.

2 Methods

We describe in Appendix F the system that we built to execute real federated learning (FL) orchestration on clients holding heterogeneous hardware. To test its efficacy, we imbue it with a series of experimental tools to enable modeling any potential federated configuration given available hardware and data using the same pipeline as a production scenario. This section describes how we have made *Photon* appropriate for scientific experimentation. Furthermore, it presents the experimental design of our evaluation meant to investigate if FL can match or even exceed the potential of centralized training.

2.1 Reproducibility by Design

As discussed in Appendix E, our most challenging aims are broad participation and inclusivity in the federated orchestration. We also target perfect reproducibility of both the system and results to allow for robust and meaningful research. The path towards democratizing the generative pre-training of LLMs inevitably involves disclosing training recipes, experimental details, and open-source and validated code. Then, inspired by other works, such as Zhang et al. [24], we rely on open-source resources as much as possible.

We used the popular open-source libraries maintained by Mosaic Research [25] for our local training pipeline and the centralized baselines. MosaicML NLP Team has released MPT (MosaicML Pretrained Transformer), a commercially usable, open-source combination of a language model and training pipeline to bundle the most popular software infrastructure and optimizations described in Appendix D.1. The standard MPT model [26] is a decoder-style transformer with 6.7B parameters, integrating Flash Attention [27] for efficient execution, ALiBi [28] for context length extrapolation and general stability improvements to mitigate loss spikes. In this work, we use MPT models of different scales and follow the best practices recommended by MosaicML for the *local* training pipeline unless stated otherwise. We seed every local training and the client selection mechanism to ensure experimental reproducibility.

2.1.1 Data Partitioning

To model scenarios where clients train on heterogeneous data, we provide scripts for partitioning *The Pile* [29] and Multilingual Cleaned Colossal Common Crawl (*mC4*) [30] into **naturally** heterogeneous subsets, unlike previous work [19] which relied on artificially partitioned texts within a dataset using the embeddings of a pre-trained model. While naturally heterogeneous partitions for LLM training have been previously considered by Charles et al. [31], their work focused on efficiently building fine-grained partitions for cross-device FL scenarios with potentially millions of clients based on individual URLs. Given our cross-silo setting, we are far more concerned with the degrees of heterogeneity likely to be exhibited across medium-scale and large organizations. Thus, we focus on building large and semantically meaningful heterogeneous partitions.

For example, the heterogeneous partitioning strategies we have thus far mentioned and constructed correspond to texts that differ by genre/originating website in the case of *The Pile*, simulating collaborations between publishers from different domains, and texts that vary by language in the case of *mC4*, simulating transnational cooperation.

We first partition each heterogeneous dataset by category (e.g., source dataset for *The Pile* and language for *mC4*) and then split each individual category into disjoint buckets. The number of buckets we create per category equals $J \times |C|$, where $|C|$ is the number of distinct clients in the federation, while J is the maximum number of categories a client may draw upon. For example, for a federation of 8 clients, each of which may be a combination of 3 categories, we would create 24 buckets per category. Each bucket can be mapped to at most one client in the federation, ensuring that even if two clients draw from the same source, they constantly sample from disjoint data subsets. This implementation allows us to build any arbitrarily complex topology without additional bookkeeping or runtime costs.

2.2 Experimental Design

To evaluate *Photon*, we consider several aspects of federated LLM optimization corresponding to what we perceive as the biggest challenges to adopting federated LLM pre-training.

Homogeneous Data Sources: First, we wish to isolate the impact of the optimization procedure itself by using a standard benchmark dataset partitioned across clients in an IID fashion. Since parameter averaging is known to inject noise into the optimization procedure [17], it may cause a change in the dynamics of LLM training. Thus, we want to observe whether it can substitute distributed data-parallel SGD for situations where data is mainly similar from client to client (e.g., in the same language and containing the same mix of genres). We use a version of C4 [32] randomly split into randomly constructed shards for our homogeneous data sources experiments.

Heterogeneous Data Sources: Second, we want to observe whether injecting *federated statistical heterogeneity* into the system harms convergence. We are particularly concerned with approximating a realistic cross-silo form of data heterogeneity similar to those encountered by institutions like publishers, which may specialize in different text genres. For this purpose, we choose to use *The Pile* partitioned into subsets representing general knowledge (*Wikipedia, English only*), general scientific publications (*ArXiv*), long-form books (*Project Gutenberg, alias PG-19*), technical news/discussions (*HackerNews*), medical publications (*PubMed Central*), law (*FreeLaw*), philosophical publications (*PhilPapers*), and Q/A websites (*StackExchange*). This coarse mapping allows us to directly determine the efficacy of federated training in reconciling different styles and sources of knowledge.

Model Scaling: Third, we want to show that federated optimization can train a large model in a cost-effective distributed fashion without access to the sprawling data centers traditionally tasked with such training. Given the prohibitive nature of such training, we initially focus on the IID $C4$ partition due to its usefulness as a generally agreed-upon benchmark in the community.

Partial Participation: One aspect of federated training with broad implications for the computational costs and data efficiency of the method is partial participation. As previously discussed in Appendix F.3, while in typical (i.e., cross-device) FL scenarios aggregating across a greater number of clients helps improve convergence by reducing the variance of client pseudo-gradients, this effect tends to saturate. Thus, we aim to discover if similar behaviors exist in federated settings.

Table 1: Architecture details and local training parameters for our 75M, 125M, 350M and 1.3B 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 [33] and the sequence length l.

Model Size	#Blocks	d	#Heads	Exp. Ratio	$(\beta_1,\ \beta_2)$	|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

2.3 Training Setup

All experiments use 500 local steps per round, executed by all the clients in the federation. We distinguish between *parallel* steps and *sequential* steps since the model parameters are averaged after each round, and each of the 500 local steps in a round is done in parallel, as in standard data-parallel training [34]. Our setting respects the conditions of standard cross-silo [35] FL in which the orchestrator samples the entire batch of equally capable clients in every round. Our experiments used diverse hardware resources whose specifications allow for reasonably fast execution; as an example, we used a combination of heterogeneous servers equipped with NVIDIA A40, A100, and H100 GPUs. Due to FL's lax synchronization requirements, these heterogeneous hardware accelerators were able to collaborate despite being located in different countries.

As shown in Table 1, we trained models ranging in size from 75 million parameters to 1.3 billion for the causal language modeling task. We used the tokenizer presented in [33] with a vocabulary size of 50 368. The local optimizer the clients use in our experiments is AdamW [36], while the server optimizer is FedMom [37]. The hyperparameters we used are reported in Table 2.

Table 2: Hyperparameters used in our experiments. The federated learning rate η_s and momentum μ_s [37] are applied by *Photon Aggregator*. The FL is performed across the number of rounds reported. $\mathbf{S_C}$ 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. We also report the batch size used in the local training by the *Photon LLM Nodes*.

Model Size	η_s	μ_s	α	η_{max}	T	Batch Size
75M	0.7	0.9	$10^{-1}, 10^{-6}$	4×10^{-4}	88 000	256
125M	$0.3, 0.5, 0.7$	0.9	$10^{-1}, 10^{-5}$	$\{3.0, 6.0\} \times 10^{-4}$	15 000	256
350M	0.1	0.9	10^{-1}	3×10^{-4}	13 400	256
1.3B	0.7	0.9	10^{-1}	2×10^{-4}	24 800	512

3 Evaluation

The results presented in this section indicate the viability of federated optimization for LLM pre-training and validate its ability to reach beyond the data and hardware constraints of centralized approaches.

3.1 Federated Optimization is Effective and Stable across Scales

Our initial experiments using the IID partition of the *C4* dataset reveal that federated optimization can reach a performance comparable to centralized training. As can be seen in Fig. 2, federated models perform comparatively well to centrally trained ones or even better at sufficiently large model sizes. Furthermore, while their convergence curves may initially look more unstable, the federated aggregation procedure quickly resolves this once the model settles into an appropriate compromise

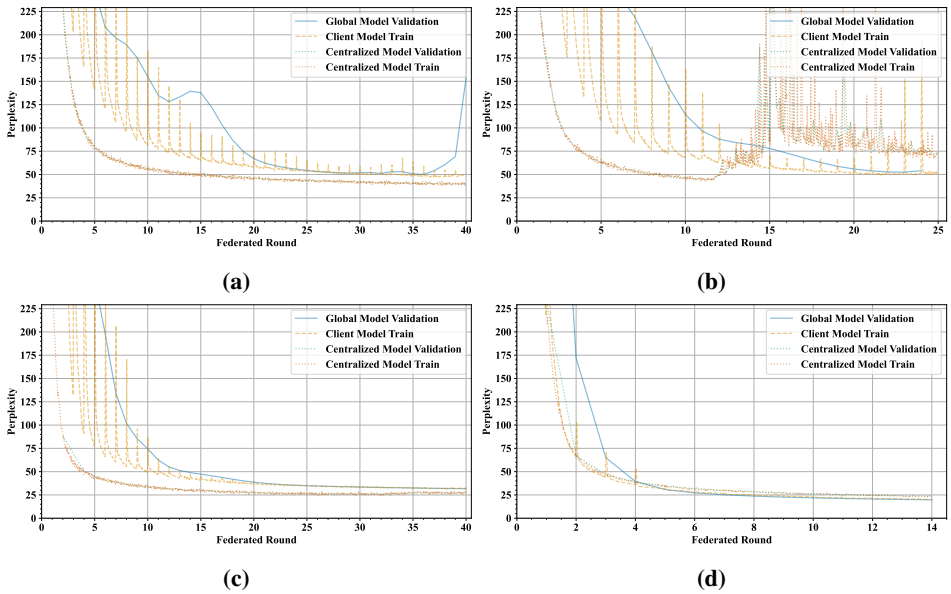


Figure 2: Comparison between the perplexity of the federated global model evaluated on the centralized validation set, the train perplexities of federated clients (averaged together), and the train and test perplexities for a centralized experiment. These metrics are reported for our 75M (a), 125M (b), 350M (c), and 1.3B (d) experiments. Crucially, the stability of federated training increases with model size. For example, the centralized model outperforms the 75M federated model while performing near-identically for the 1.3B models. While federated aggregation initially causes large spikes in client perplexity, these subside as the clients reach a consensus on the model parameters, which happens much quicker for larger models. Following this transitory phase, aggregation applies a regularizing effect on the model performance, allowing a better model to be trained than would be possible for a single client. The server validation perplexity is a soft upper bound for the spikes, with the gap between train and validation perplexities decreasing over time.

for the clients. Most importantly, **FL effectively trains the 1.3B parameters model**. The gap between centralized and federated performance becomes increasingly small as the model size increases, ranging between 10 and 2 points on the perplexity axis.

As such, we believe that federated optimization can effectively replace standard data-parallel techniques in data center settings when nodes are poorly connected, which *Photon* handles automatically. Furthermore, we will now argue that it is suitable for in-the-wild federated systems where data may also come from statistically heterogeneous distributions.

3.2 Federated Optimization is Robust to Heterogeneity

As previously discussed, *federated statistical heterogeneity* problem is inevitable for real federated learning scenarios as clients use naturally generated private data to train their models. Thus, we investigate how much heterogeneity impacts performance using our federated partition of *The Pile* composed of *Wikipedia (en)*, *ArXiv*, *Project Gutenberg (PG-19)*, *HackerNews*, *PubMed Central*, *FreeLaw*, *PhilPapers*, and *StackExchange*. We conduct these experiments on our smaller 75M and 125M models. We chose them because they are under-parameterized compared to our 1.3B model, making reconciling heterogeneous data particularly challenging. In Table 2, we report the hyperparameters we use for our experiments.

We observe a high degree of robustness towards this naturally heterogeneous partition. As shown in Fig. 3, while clients show much more significant variance in their loss, the server validation loss converges with a similar trend to the IID *C4* case. Compared to the previous set of experiments, the absolute performance improves from 50 to 35 for the 75M model and from 50 to 30 for the 125M model mainly because of the different datasets used. We believe that the highly effective

convergence is due to the regularization properties of FL, precisely because of FL's meta-learning properties [38, 39, 40].

Furthermore, as can be seen in Fig. 4, models trained in a federated fashion are not subject to the same tendency of rapidly increasing activations as centralized models are. Corroborating this against the perplexity in Fig. 3 shows that while the centralized model may diverge beyond repair, the federated model can quickly recover from temporary decreases in server validation accuracy. We argue that this is not a mere artifact of the model being less trained since Fig. 4 show that activations of the centralized model start higher than those of the federated one and remain so throughout the entire duration of the experiments. We attribute this to the noise injected by the aggregation procedure, given that we observe periodic reductions in activation norms at round boundaries. This confirms the robustness properties of federated aggregation [17].

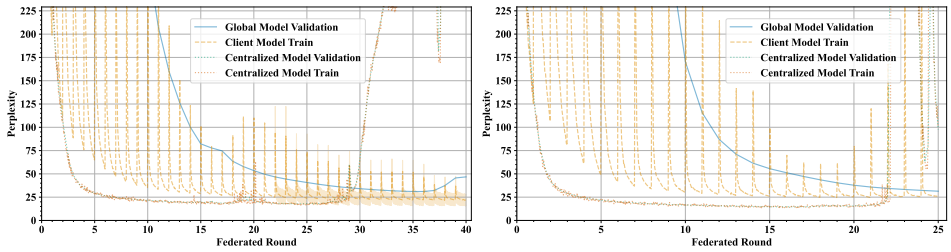


Figure 3: Perplexity comparison between the global model evaluated on the centralized validation set, the train and test perplexities for a centralized baseline, and the training perplexities of clients (averaged together) for our naturally heterogeneous partition of *The Pile* using either a 75M (a) or 125M (b) model size. Unlike the homogeneous partition, the natural heterogeneity of the underlying datasets makes an initial consensus harder to reach for the federated model, as can be observed from the very high initial clients and server perplexities. However, like the IID partition shown in Fig. 2, once clients reach consensus, performance becomes comparable to a centralized baseline.

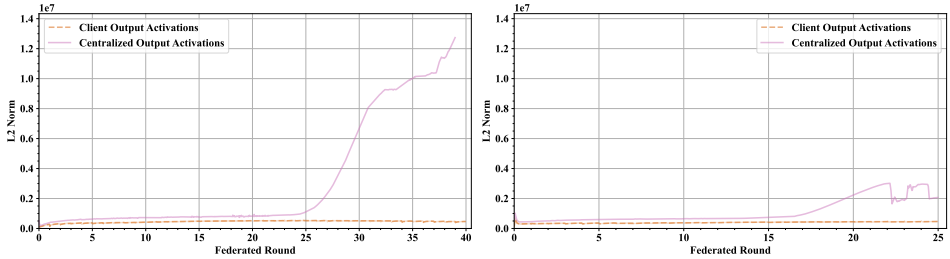


Figure 4: The l_2 norms of the output activations of our 75M (a) and 125M (b) models trained on a naturally heterogeneous partition of *The Pile*. The norm of the activations is a well-known indicator of future model divergence [24]. As can be observed, the activations of the centralized model outpace those of the federated clients right from the start and experience a massive increase towards the end of training. The aggregation procedure keeps the federated clients in check by reducing the norm of the activations round to round.

3.3 Large Models Reach Consensus

Our experiments focused on scaling the model size to determine if federated optimization can train models approximating the size of those commonly used today [41, 42].

We observed that the stability of the training procedure improves directly with the model size, with federated optimization reaching a better consensus across client models. As shown in Fig. 2, while the centralized model's final performance is slightly greater for small model sizes, it becomes almost identical for the 1.3B model. Furthermore, client training perplexities showcase a shift from a

transient oscillatory phase to a convergence phase where federated optimization acts as a regularizer, improving performance. Thus, while aggregation increases local perplexity in the early stages, it decreases it at convergence. This transition is much quicker for larger models, with the 1.3B model spending only 4 rounds in the pre-convergence phase, while the 75M model requires over 20 rounds to reach a consensus across client models. Similarly, while the 75M model continues to experience occasional turbulences in convergence after round 20, the larger models, such as the 350M model, obtain **full** client model alignment.

3.4 Federated Optimization is Robust to Partial Participation

Our experiments have shown that meaningfully-sized LLMs can be trained in a federated fashion when using full client participation. However, one particularly effective way of reducing overall training costs in terms of parallel computing while still having the regularization benefits of training on more data is to subsample a smaller portion of the entire federated population. The experiments in Fig. 5 show how subsampling only 6.25% of clients from our 64-client partition results in the same performance as our previous full-participation experiments, despite training only 4 clients per round instead of 8. Such robustness to partial participation opens the door to highly efficient LLM training schedules in which the scalable addition of data, as needed, improves convergence whilst minimizing computing resource utilization. Furthermore, it enables applications where multiple federated workloads may be executed simultaneously by sampling different clients from the population.

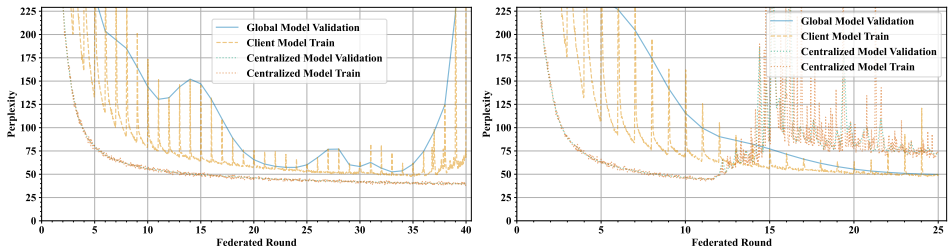


Figure 5: Perplexity comparison, 75M (a) and 125M (b), between the server model evaluated on the centralized validation set, the train and test perplexities for a centralized baseline, and the training perplexities of clients for our 64-client partition of *C4* with 4 client participating per round for a 6.25% participation rate. Despite using half of the parallel compute of the full participation baselines, partial participation converges to the same performance at the expense of more turbulences early in the training process. Thus, subsampling cohorts can substantially reduce hardware utilization for a sufficiently large cross-silo population, allowing for parallel training of several models.

4 Related Work

In the past few years, the research community has investigated the problem of executing the expensive training of LLMs assuming a distributed infrastructure with restricted communication capabilities. Unlike standard settings, which possess extremely well-connected hardware accelerators, these worldwide distributed scenarios are assumed to communicate through the Internet with bandwidth in the range of 10-100Mbps. In these situations, the amount and rate of data transferred between workers is paramount because it can lead to suboptimal execution, eventually wasting precious GPU hours on the nodes participating in the training.

Yuan et al. [43] were the first to try to answer the following very interesting research question: *can we instead leverage the much greater amount of decentralized, heterogeneous, and lower-bandwidth interconnected compute?* They defined a graph problem based on the concept of "tasklets" to optimize for minimal communication overheads during the distributed training using a set of widely distributed data centers with heterogeneous bandwidth but homogeneous hardware accelerators. Their approach, however, assumes that the minimal requirements for executing such "tasklets" are satisfied in any place the training can take place, e.g., data availability, power, and compute availability. Our approach, instead, supports stragglers, partial participation, heterogeneous data sources, heterogeneous hardware resources, and not only heterogeneous networks, as it supports any federated

optimizer and makes broad inclusivity one of its basic principles. Douillard et al. [19] proposes using parameter averaging, known as Local-SGD [16] in centralized settings and FedAvg [18] in federated ones, as a communication-efficient replacement for standard data-parallel techniques [44] used in centralized pre-training. While their results show significant reductions in communication frequency, their methodology is only suitable for centralized pre-training. This is because it relies on homogeneous hardware and stateful optimizers for each participating node, which requires full participation to avoid training with stale optimizer states. Such constraints make it impossible to use in a federated setting and make any data center training where the hardware does not have 100% uptime impossible, as drop-outs would automatically invalidate the local optimizer states. Charles et al. [31] train LLMs in federated settings comprised of large client populations with heterogeneous data and weak hardware. To tackle the very large degree of heterogeneity present in such a scenario, their methodology relies on using a very low number of local steps coupled with a small batch size, making their algorithm far closer to FedSGD [18] than FedAvg with the accompanying high communication frequency and costs. More recently, Research [45] published a preliminary technical report claiming impressive communication reduction in training an LLM with more than one billion parameters using a set of 32 Nvidia H100 in a data center. Interestingly, they assume each GPU acts as a single worker training the entire model on just a shard of the batch processed per step, recalling the Distributed Data-Parallel (DDP) technique. However, they maintain a peer-to-peer topology that may not be sufficiently private in most federated settings. Given our assumptions and the optimization procedure, our proposal further reduces communication by a factor of 32x compared to DisTrO without assuming any specific optimization, such as RingAllReduce, instead of standard AllReduce.

Compared to theirs, the strength of our work is that we are taking a completely new perspective regarding the problem of pre-training LLMs. Instead of trying to combine the standard algorithms used for centralized training to these uncommon distributed infrastructures, we devise a setting that strictly depends on the reality of the data sources and the silo of computing resources in the wild. We also take full advantage of the federated optimization to reduce as much as possible the number of communication steps that are usually one of the most critical factors to minimize for obtaining an efficient federated learning execution.

5 Future Work

This work presents a complete system for generative pre-training of LLMs across a federation of clients, each participating with its own computing and data sources. In this section, we present a summary of the future direction and possibilities this work will allow. We plan to fine-tune these models on established benchmark tasks to assess further their performance in the broad range of downstream applications the participants may be interested in. We also plan to scale up the federated setting presented in this work regarding the population and model size. In addition, we will conduct further investigations into the heterogeneity of data sources, such as different languages or genres, and how this inherent property of FL impacts the capabilities of these federated models. These investigations will illuminate the potential advantages of collaboration compared to the isolated centralized paradigm. We expect FL's benefits to grow only as we incorporate more extensive private datasets into the training procedure. Since FL has been introduced as a privacy-preserving by-design technique, we will analyze this paradigm's potential privacy-preservation strengths and weaknesses. As we advance the proposed system, we will delineate and develop further optimizations to mitigate the few overheads remaining in the execution runtime. This will eventually make our proposal more efficient and effective for allowing an even broader participation of the FL.

6 Conclusions

We successfully demonstrate the potential of LLM federated generative pre-training by obtaining the first federated billion-scale model fully trained in a heterogeneous FL setting and showing how federated training is robust to both data heterogeneity and partial participation. The complete system presented in this work allows for collaborative, reproducible, and scalable pre-training of LLMs, releasing the computational and data resources spread across the planet. Our system is built on an **open-source** framework [22] and will soon be made publicly available. Furthermore, we fully disclose our training recipe to enable future development and collaborations. Our work advances LLM pre-training by democratizing it through federated technologies. This allows data-rich actors

to pool resources together, whether extensive or limited. Given current trends, we believe that the future of LLM pre-training lies in the data sources that FL can bring together.

References

[1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

[2] R. F. Berriel, A. T. Lopes, A. Rodrigues, F. M. Varejão, and T. Oliveira-Santos. Monthly energy consumption forecast: A deep learning approach. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4283–4290, 2017.

[3] Xuanyu Cao, Tamer Basar, Suhas N. Diggavi, Yonina C. Eldar, Khaled B. Letaief, H. Vincent Poor, and Junshan Zhang. Communication-efficient distributed learning: An overview. *IEEE J. Sel. Areas Commun.*, 41(4):851–873, 2023. doi: 10.1109/JSAC.2023.3242710. URL https://doi.org/10.1109/JSAC.2023.3242710.

[4] Elia Guerra, Francesc Wilhelmi, Marco Miozzo, and Paolo Dini. The cost of training machine learning models over distributed data sources. *IEEE Open J. Commun. Soc.*, 4:1111–1126, 2023. doi: 10.1109/OJCOMS.2023.3274394. URL https://doi.org/10.1109/OJCOMS.2023.3274394.

[5] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022. doi: 10.48550/ARXIV.2203.15556. URL https://doi.org/10.48550/arXiv.2203.15556.

[6] Michael M Grynbaum and Ryan Mac. The times sues openai and microsoft over a.i. use of copyrighted work, Dec 2023. URL https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html.

[7] Vasilisa Bashlovkina, Zhaobin Kuang, Riley Matthews, Edward Clifford, Yennie Jun, William W. Cohen, and Simon Baumgartner. Trusted source alignment in large language models. *CoRR*, abs/2311.06697, 2023. doi: 10.48550/ARXIV.2311.06697. URL https://doi.org/10.48550/arXiv.2311.06697.

[8] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross J. Anderson. The curse of recursion: Training on generated data makes models forget. *CoRR*, abs/2305.17493, 2023. doi: 10.48550/ARXIV.2305.17493. URL https://doi.org/10.48550/arXiv.2305.17493.

[9] Meera A Desai, Irene V Pasquetto, Abigail Z Jacobs, and Dallas Card. An archival perspective on pretraining data. *Patterns (N Y)*, 5(4):100966, March 2024.

[10] Florian Tramèr, Gautam Kamath, and Nicholas Carlini. Considerations for differentially private learning with large-scale public pretraining. *CoRR*, abs/2212.06470, 2022. doi: 10.48550/ARXIV.2212.06470. URL https://doi.org/10.48550/arXiv.2212.06470.

[11] OpenAI, Dec 2023. URL https://openai.com/blog/axel-springer-partnership.

[12] Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *CoRR*, abs/2211.04325, 2022. doi: 10.48550/ARXIV.2211.04325. URL https://doi.org/10.48550/arXiv.2211.04325.

[13] Sara Abdali, Richard Anarfi, C. J. Barberan, and Jia He. Securing large language models: Threats, vulnerabilities and responsible practices. *CoRR*, abs/2403.12503, 2024. doi: 10.48550/ARXIV.2403.12503. URL https://doi.org/10.48550/arXiv.2403.12503.

[14] Jaydeep Borkar. What can we learn from data leakage and unlearning for law? *CoRR*, abs/2307.10476, 2023. doi: 10.48550/ARXIV.2307.10476. URL https://doi.org/10.48550/arXiv.2307.10476.

[15] Sixing Yu, J. Pablo Muñoz, and Ali Jannesari. Federated foundation models: Privacy-preserving and collaborative learning for large models. *CoRR*, abs/2305.11414, 2023. doi: 10.48550/ARXIV.2305.11414. URL https://doi.org/10.48550/arXiv.2305.11414.

[16] Sebastian U. Stich. Local SGD converges fast and communicates little. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=S1g2JnRcFX.

[17] Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local SGD. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=B1eyO1BFPr.

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 2017.

[19] Arthur Douillard, Qixuang Feng, Andrei A. Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *CoRR*, abs/2311.08105, 2023. doi: 10.48550/ARXIV.2311.08105. URL https://doi.org/10.48550/arXiv.2311.08105.

[20] Bo Liu, Rachita Chhaparia, Arthur Douillard, Satyen Kale, Andrei A. Rusu, Jiajun Shen, Arthur Szlam, and Marc'Aurelio Ranzato. Asynchronous local-sgd training for language modeling. *CoRR*, abs/2401.09135, 2024. doi: 10.48550/ARXIV.2401.09135. URL https://doi.org/10.48550/arXiv.2401.09135.

[21] Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Adhiguna Kuncoro, Yani Donchev, Rachita Chhaparia, Ionel Gog, Marc'Aurelio Ranzato, Jiajun Shen, and Arthur Szlam. Dipaco: Distributed path composition. *CoRR*, abs/2403.10616, 2024. doi: 10.48550/ARXIV.2403.10616. URL https://doi.org/10.48550/arXiv.2403.10616.

[22] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. Flower: A friendly federated learning research framework. *CoRR*, abs/2007.14390, 2022. URL https://arxiv.org/abs/2007.14390.

[23] Lorenzo Sani, Pedro Porto Buarque de Gusmão, Alex Iacob, Wanru Zhao, Xinchi Qiu, Yan Gao, Javier Fernández-Marquás, and Nicholas Donald Lane. High-throughput simulation of federated learning via resource-aware client placement. *CoRR*, abs/2306.17453, 2023. doi: 10.48550/ARXIV.2306.17453. URL https://doi.org/10.48550/arXiv.2306.17453.

[24] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068, 2022. doi: 10.48550/ARXIV.2205.01068. URL https://doi.org/10.48550/arXiv.2205.01068.

[25] Databricks. mosaic research, 2024. URL https://www.databricks.com/research/mosaic.

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

[27] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

[28] Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

[29] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL https://arxiv.org/abs/2101.00027.

[30] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 483–498. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.41. URL https://doi.org/10.18653/v1/2021.naacl-main.41.

[31] Zachary Charles, Nicole Mitchell, Krishna Pillutla, Michael Reneer, and Zachary Garrett. Towards federated foundation models: Scalable dataset pipelines for group-structured learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/662bb9c4dcc96aeaac8e7cd3fc6a0add-Abstract-Datasets_and_Benchmarks.html.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[33] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. Gpt-neox-20b: An open-source autoregressive language model. *CoRR*, abs/2204.06745, 2022. doi: 10.48550/ARXIV.2204.06745. URL https://doi.org/10.48550/arXiv.2204.06745.

[34] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: Experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12): 3005–3018, aug 2020. ISSN 2150-8097. doi: 10.14778/3415478.3415530. URL https://doi.org/10.14778/3415478.3415530.

[35] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021. doi: 10.1561/2200000083. URL https://doi.org/10.1561/2200000083.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[37] Zhouyuan Huo, Qian Yang, Bin Gu, Lawrence Carin, and Heng Huang. Faster on-device training using new federated momentum algorithm. *CoRR*, abs/2002.02090, 2020. URL https://arxiv.org/abs/2002.02090.

[38] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. URL http://arxiv.org/abs/1803.02999.

[39] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/24389bfe4fe2eba8bf9aa9203a44cdad-Abstract.html.

[40] Royson Lee, Minyoung Kim, Da Li, Xinchi Qiu, Timothy M. Hospedales, Ferenc Huszar, and Nicholas D. Lane. Fedl2p: Federated learning to personalize. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/2fb57276bfbaf1b832d7bfcba36bb41c-Abstract-Conference.html.

[41] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[42] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[43] Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy Liang, Christopher Ré, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/a37d615b61f999a5fa276adb14643476-Abstract-Conference.html.

[44] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In Christine Cuicchi, Irene Qualters, and William T. Kramer, editors, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM, 2020. doi: 10.1109/SC41405.2020.00024. URL https://doi.org/10.1109/SC41405.2020.00024.

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

[46] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[47] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

[48] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023. doi: 10.48550/ARXIV.2312.11805. URL https://doi.org/10.48550/arXiv.2312.11805.

[49] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100, 2022. doi: 10.48550/ARXIV.2211.05100. URL https://doi.org/10.48550/arXiv.2211.05100.

[50] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only, 2023.

[51] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *CoRR*, abs/1812.06162, 2018. URL http://arxiv.org/abs/1812.06162.

[52] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL http://arxiv.org/abs/1912.06680.

[53] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *CoRR*, abs/1802.05799, 2018. URL http://arxiv.org/abs/1802.05799.

[54] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using

model parallelism. *CoRR*, abs/1909.08053, 2019. URL http://arxiv.org/abs/1909.08053.

[55] Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake A. Hechtman. Mesh-tensorflow: Deep learning for supercomputers. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10435–10444, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3a37abdeefe1dab1b30f7c5c7e581b93-Abstract.html.

[56] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch FSDP: experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16 (12):3848–3860, 2023. doi: 10.14778/3611540.3611569. URL https://www.vldb.org/pvldb/vol16/p3848-huang.pdf.

[57] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174, 2016. URL http://arxiv.org/abs/1604.06174.

[58] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. Zero-offload: Democratizing billion-scale model training. In Irina Calciu and Geoff Kuenning, editors, *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*, pages 551–564. USENIX Association, 2021. URL https://www.usenix.org/conference/atc21/presentation/ren-jie.

[59] Yingzhou Lu, Huazheng Wang, and Wenqi Wei. Machine learning for synthetic data generation: a review. *CoRR*, abs/2302.04062, 2023. doi: 10.48550/ARXIV.2302.04062. URL https://doi.org/10.48550/arXiv.2302.04062.

[60] La Javaness. https://lajavaness.medium.com/llm-large-language-model-cost-analysis-d5022bb43e9e, 2023.

[61] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *CoRR*, abs/2402.17764, 2024. doi: 10.48550/ARXIV.2402.17764. URL https://doi.org/10.48550/arXiv.2402.17764.

[62] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[63] Alexander Borzunov, Max Ryabinin, Artem Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, and Colin Raffel. Distributed inference and fine-tuning of large language models over the internet, 2023.

[64] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=BJ0hF1Z0b.

[65] K. A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. URL https://arxiv.org/abs/1611.04482.

[66] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In Ameet Talwalkar, Virginia Smith, and Matei Zaharia, editors, *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. mlsys.org, 2019. URL https://proceedings.mlsys.org/book/271.pdf.

[67] Jose Javier Gonzalez Ortiz, Jonathan Frankle, Mike Rabbat, Ari S. Morcos, and Nicolas Ballas. Trade-offs of local SGD at scale: An empirical study. *CoRR*, abs/2110.08133, 2021. URL https://arxiv.org/abs/2110.08133.

[68] Agrin Hilmkil, Sebastian Callh, Matteo Barbieri, Leon René Sütfeld, Edvin Listo Zec, and Olof Mogren. Scaling federated learning for fine-tuning of large language models. In Elisabeth Métais, Farid Meziane, Helmut Horacek, and Epaminondas Kapetanios, editors, *Natural Language Processing and Information Systems - 26th International Conference on Applications of Natural Language to Information Systems, NLDB 2021, Saarbrücken, Germany, June 23-25, 2021, Proceedings*, volume 12801 of *Lecture Notes in Computer Science*, pages 15–23. Springer, 2021. doi: 10.1007/978-3-030-80599-9_2. URL https://doi.org/10.1007/978-3-030-80599-9_2.

[69] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

[70] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1423. URL https://doi.org/10.18653/v1/n19-1423.

[71] Pascal Riedel, Manfred Reichert, Reinhold von Schwerin, Alexander Hafner, Daniel Schaudt, and Gaurav Singh. Performance analysis of federated learning algorithms for multilingual protest news detection using pre-trained distilbert and BERT. *IEEE Access*, 11:134009–134022, 2023. doi: 10.1109/ACCESS.2023.3334910. URL https://doi.org/10.1109/ACCESS.2023.3334910.

[72] Boxin Wang, Jacky Yibo Zhang, Yuan Cao, Bo Li, H. Brendan McMahan, Sewoong Oh, Zheng Xu, and Manzil Zaheer. Can public large language models help private cross-device federated learning? *CoRR*, abs/2305.12132, 2023. doi: 10.48550/ARXIV.2305.12132. URL https://doi.org/10.48550/arXiv.2305.12132.

[73] Orion Weller, Marc Marone, Vladimir Braverman, Dawn J. Lawrie, and Benjamin Van Durme. Pretrained models for multilingual federated learning. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1413–1421. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.NAACL-MAIN.101. URL https://doi.org/10.18653/v1/2022.naacl-main.101.

[74] Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Guoyin Wang, and Yiran Chen. Towards building the federated GPT: federated instruction tuning. *CoRR*, abs/2305.05644, 2023. doi: 10.48550/ARXIV.2305.05644. URL https://doi.org/10.48550/arXiv.2305.05644.

[75] Tao Fan, Yan Kang, Guoqiang Ma, Weijing Chen, Wenbin Wei, Lixin Fan, and Qiang Yang. FATE-LLM: A industrial grade federated learning framework for large language models. *CoRR*, abs/2310.10049, 2023. doi: 10.48550/ARXIV.2310.10049. URL https://doi.org/10.48550/arXiv.2310.10049.

[76] Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. *CoRR*, abs/2309.00363, 2023. doi: 10.48550/ARXIV.2309.00363. URL https://doi.org/10.48550/arXiv.2309.00363.

[77] Jingang Jiang, Xiangyang Liu, and Chenyou Fan. Low-parameter federated learning with large language models. *CoRR*, abs/2307.13896, 2023. doi: 10.48550/ARXIV.2307.13896. URL https://doi.org/10.48550/arXiv.2307.13896.

[78] Shubham Malaviya, Manish Shukla, and Sachin Lodha. Reducing communication overhead in federated learning for pre-trained language models using parameter-efficient finetuning. In Sarath Chandar, Razvan Pascanu, Hanie Sedghi, and Doina Precup, editors, *Conference on Lifelong Learning Agents, 22-25 August 2023, McGill University, Montréal, Québec, Canada*, volume 232 of *Proceedings of Machine Learning Research*, pages 456–469. PMLR, 2023. URL https://proceedings.mlr.press/v232/malaviya23a.html.

[79] Mingbin Xu, Congzheng Song, Ye Tian, Neha Agrawal, Filip Granqvist, Rogier C. van Dalen, Xiao Zhang, Arturo Argueta, Shiyi Han, Yaqiao Deng, Leo Liu, Anmol Walia, and Alex Jin. Training large-vocabulary neural language models by private federated learning for resource-constrained devices. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pages 1–5. IEEE, 2023. doi: 10.1109/ICASSP49357.2023.10096570. URL https://doi.org/10.1109/ICASSP49357.2023.10096570.

[80] Mengwei Xu, Dongqi Cai, Yaozong Wu, Xiang Li, and Shangguang Wang. Fwdllm: Efficient fedllm using forward gradient, 2024.

[81] Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. Slora: Federated parameter efficient fine-tuning of language models. *CoRR*, abs/2308.06522, 2023. doi: 10.48550/ARXIV.2308.06522. URL https://doi.org/10.48550/arXiv.2308.06522.

[82] Yeachan Kim, Junho Kim, Wing-Lam Mok, Jun-Hyung Park, and SangKeun Lee. Client-customized adaptation for parameter-efficient federated learning. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1159–1172. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.75. URL https://doi.org/10.18653/v1/2023.findings-acl.75.

[83] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.243. URL https://doi.org/10.18653/v1/2021.emnlp-main.243.

[84] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pages 1–5. IEEE, 2023. doi: 10.1109/ICASSP49357.2023.10095356. URL https://doi.org/10.1109/ICASSP49357.2023.10095356.

[85] Tianshi Che, Ji Liu, Yang Zhou, Jiaxiang Ren, Jiwen Zhou, Victor S. Sheng, Huaiyu Dai, and Dejing Dou. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7871–7888. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.488. URL https://doi.org/10.18653/v1/2023.emnlp-main.488.

[86] Wanru Zhao, Royson Lee, Yihong Chen, Xinchi Qiu, Yan Gao, Hongxiang Fan, and Nicholas Donald Lane. Breaking physical and linguistic borders: Multilingual federated prompt tuning for low-resource languages. In *The Twelfth International Conference on Learning Representations*, 2024.

[87] Sahil Patel and Stephanie Palazzolo. OpenAI offers publishers as little as $1 million a year — the information, Jan 2024. URL https://www.theinformation.com/articles/openai-offers-publishers-as-little-as-1-million-a-year.

[88] Alexandre Magueresse, Vincent Carles, and Evan Heetderks. Low-resource languages: A review of past work and future challenges. *CoRR*, abs/2006.07264, 2020. URL https://arxiv.org/abs/2006.07264.

[89] Surangika Ranathunga, En-Shiun Annie Lee, Marjana Prifti Skenduli, Ravi Shekhar, Mehreen Alam, and Rishemjit Kaur. Neural machine translation for low-resource languages: A survey. *ACM Comput. Surv.*, 55(11):229:1–229:37, 2023. doi: 10.1145/3567592. URL https://doi.org/10.1145/3567592.

[90] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *CoRR*, abs/1611.04482, 2016. URL http://arxiv.org/abs/1611.04482.

[91] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze, editors, *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020. URL https://proceedings.mlsys.org/book/316.pdf.

[92] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=LkFG3lB13U5.

[93] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 17455–17466, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/91cff01af640a24e7f9f7a5ab407889f-Abstract.html.

[94] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proc. IEEE*, 108(4):485–532, 2020. doi: 10.1109/JPROC.2020.2976475. URL https://doi.org/10.1109/JPROC.2020.2976475.

[95] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=ByexElSYDr.

[96] Tian Li, Ahmad Beirami, Maziar Sanjabi, and Virginia Smith. Tilted empirical risk minimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=K5YasWXZT3O.

[97] Tailin Zhou, Zehong Lin, Jun Zhang, and Danny H.K. Tsang. Understanding and improving model averaging in federated learning on heterogeneous data. *IEEE Transactions on Mobile Computing*, page 1–16, 2024. ISSN 2161-9875. doi: 10.1109/tmc.2024.3406554. URL http://dx.doi.org/10.1109/TMC.2024.3406554.

[98] Kilian Pfeiffer, Martin Rapp, Ramin Khalili, and Jörg Henkel. Federated learning for computationally constrained heterogeneous devices: A survey. *ACM Comput. Surv.*, 55(14s): 334:1–334:27, 2023. doi: 10.1145/3596907. URL https://doi.org/10.1145/3596907.

[99] Zachary Charles, Zachary Garrett, Zhouyuan Huo, Sergei Shmulyian, and Virginia Smith. On large-cohort training for federated learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 20461–20475, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/ab9ebd57177b5106ad7879f0896685d4-Abstract.html.

[100] Hulya Vural, Murat Koyuncu, and Sinem Guney. A systematic literature review on microservices. In Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Giuseppe Borruso, Carmelo Maria Torre, Ana Maria A. C. Rocha, David Taniar, Bernady O. Apduhan, Elena N. Stankova, and Alfredo Cuzzocrea, editors, *Computational Science and Its Applications - ICCSA 2017 - 17th International Conference, Trieste, Italy, July 3-6, 2017, Proceedings, Part VI*, volume 10409 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 2017. doi: 10.1007/978-3-319-62407-5_14. URL https://doi.org/10.1007/978-3-319-62407-5_14.

[101] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[102] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

[103] Inc. MinIO. The Object Store for AI Data Infrastructure, 2024. URL https://min.io/.

[104] Amazon. Amazon S3, 2024. URL https://aws.amazon.com/s3/.

[105] the boto project. Boto3 - The AWS SDK for Python, 2024. URL https://github.com/boto/boto3.

[106] Fahim Faisal, Yinkai Wang, and Antonios Anastasopoulos. Dataset geography: Mapping language data to language users. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3381–3411. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.239. URL https://doi.org/10.18653/v1/2022.acl-long.239.

A Additional Experimental Tools

Unlike other FL applications, the incredible computational resources and time necessary for training LLMs require constant monitoring, checkpoints, and potential manual interventions to restart training from an appropriate point. Thus, we use the MosaicML tools to track running statistics on the norms of optimizer states, gradient updates, and activations. These generally serve as leading indicators of model divergence in centralized settings [24]. Our extensions allow the monitors to track per-model and per-layer statistics efficiently, a necessary addition given the added complexity of FL and the need for client-level monitoring. Besides this, we allow for the easy integration of federated metrics that cannot be captured locally, such as the pairwise distance or cosine similarity between client models, client pseudo-gradient norms, and server-side momentum norms. Furthermore, we ensure that all the resources controlling the local pipeline are correctly cleaned up or reused as appropriate to avoid interference between different rounds or concurrent experiments.

To enable proper experimental configuration, tracking, and resumption, we provide a set of quality-of-life improvements, such as automatic federated training resumption from the most recent round and typed experimental schemas for all federated hyperparameters. Crucially, since federated execution may occur on various devices, we provide an efficient procedure for automatically choosing a micro-batch size appropriate for your GPU type. The procedure produces an estimate for the micro-batch size based on the model's memory consumption and a micro-batch of 1 by finding the power of 2 that most closely approaches the limits of the VRAM. It then iteratively improves upon this guess by binary searching over powers of two for the largest batch size, which does not cause an out-of-memory condition (OOM). We only search over powers of 2 as they are most likely to match the assumptions of both underlying algorithms and the hardware configuration of most machines, which have $1, 2, 4$ or 8 GPUs.

B Additional Hyperparameters

Table 3: The table reports the hyperparameters we used for our federated experiments. For each model size we evaluated, we report the number of total rounds executed, the population size $\boldsymbol{P}$, the number of clients sampled per round (cohort size) $\boldsymbol{K}$, the datasets we have been training on $\boldsymbol{D}$ and the number of local steps every sampled client performs in every federated round $\boldsymbol{\tau}$.

Model Size	#Rounds	P	K	D	τ
75M	40	$8, 64$	$8, 4$	C4 [32], The Pile [29]	500
125M	$10, 25$	$8, 64$	$8, 4$	C4 [32], The Pile [29]	$250, 500$
350M	40	8	8	C4 [32]	500
1.3B	14	8	8	C4 [32]	500

C Additional Results

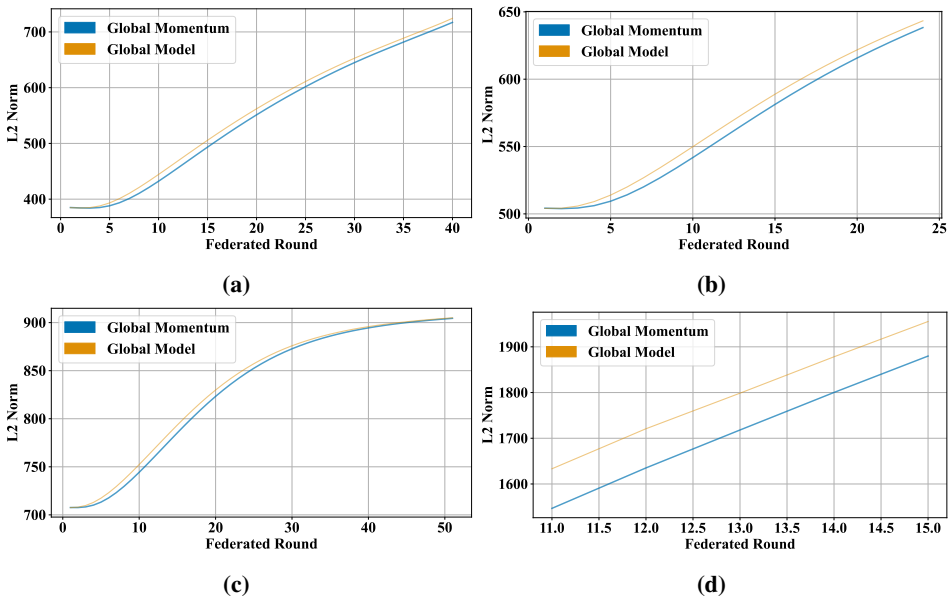


Figure 6: The l_2 norm of the global model versus the server-side Nesterov momentum, which seeks to track an exponential moving average of the model using $\beta = 0.7$, for our 75M (a), 125M (b), 350M (c), and 1.3B (d) experiments.

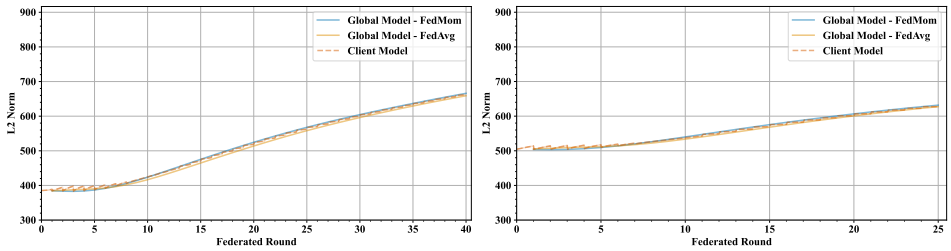


Figure 7: The l_2 norms of the global model, client models, and the average of client models for our 75M (a) and 125M (b) experiments on our naturally heterogeneous partition of *The Pile*. Despite the heterogeneous data, the momentum mechanism of the server and the decaying learning rate allow it to enforce a consensus effectively.

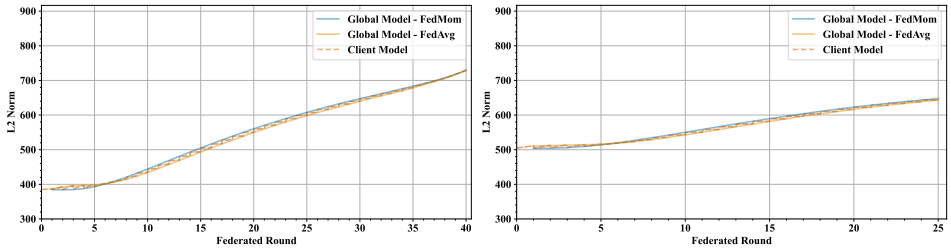
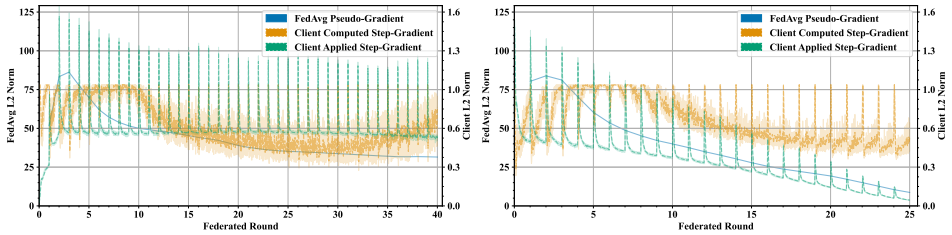
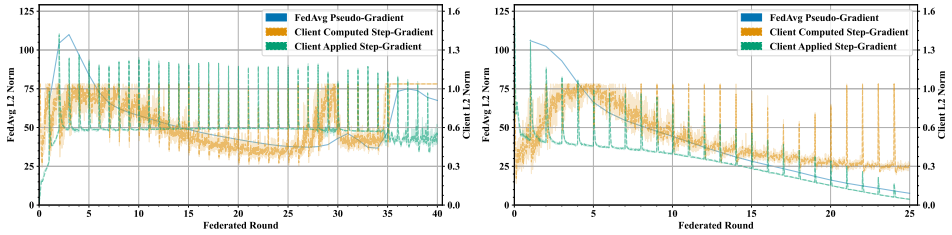


Figure 8: The l_2 norms of the global model, client models, and the average of client models for our 75M (a) and 125M (b) experiments on our partial participation experiments using 4 clients per round out of 64 total *C4* clients, i.e., the 6% of the federated population. Despite the FL using only a small sample of clients every round, the global model follows the exact same trend as in the full participation scenario.



(a) **(b)**

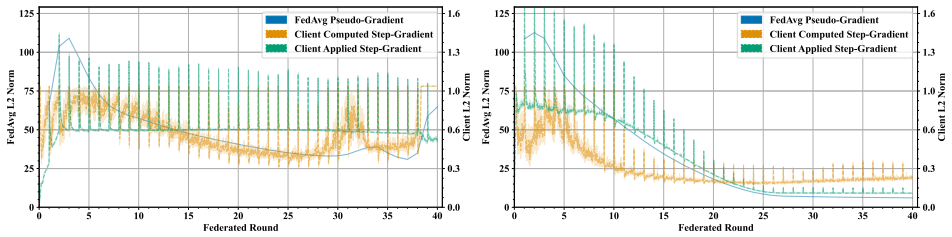
Figure 9: The l_2 norms, for our 75M (a) and 125M (b) experiments on *The Pile*, of the FedAvg Pseudo-Gradient (average of client deltas relative to the server model of the previous round), the client model gradients computed on a per-step basis, and the client gradients applied to the model when considering learning rate, weight decay, and clipping. Unlike the IID case, the FedAvg Pseudo-Gradient decays much faster than local client gradients, indicating that its decrease results from the model adapting to data heterogeneity rather than being a consequence of learning rate decay.



(a) **(b)**

Figure 10: The l_2 norms, for our 75M (a) and 125M (b) partial participation experiments on *C4*, of the FedAvg Pseudo-Gradient (average of client deltas relative to the server model of the previous round), the client model gradients computed on a per-step basis, and the client gradients applied to the model when considering learning rate, weight decay, and clipping. The relationship between the FedAvg Pseudo-Gradient and the local steps is the same as in the full-participation case despite a very small sample of clients being considered every round.

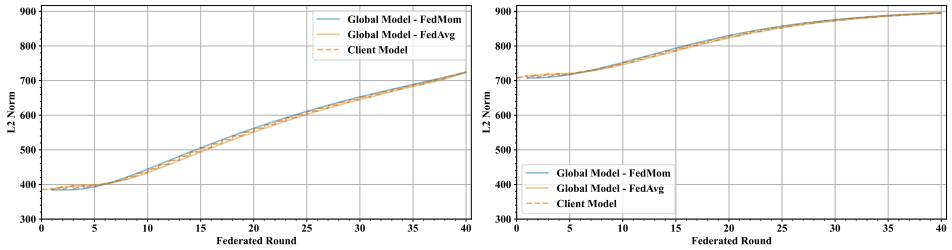
Federated Optimization Aligns Client Gradients: We observe the cause of client model convergence in Fig. 11, which showcases the relationship between the norms of the per-round **pseudo-gradient** (i.e., average client update) and the local client gradients applied at every SGD step. While the pseudo-gradient starts much larger than the applied local gradients, it reaches a similar or much smaller size as the client models converge. The same trends hold for our results on heterogeneous data, shown in Fig. 9, and for our partial-participation experiments, shown in Fig. 10.



(a) **(b)**

Figure 11: The l_2 norms, for our 75M (a) and 350M (b) experiments, of the FedAvg Pseudo-Gradient (average of client deltas relative to the server model of the previous round), the client model gradients computed on a per-step basis, and the client gradients applied to the model when considering learning rate, weight decay, and clipping. The decay in the pseudo-gradient is faster than the decay of the scheduler for the 75M model, being data-driven, and comparable to that of the step-gradient for the 125M model.

Interplay of Client and Server Models: We further investigate the training dynamics of the models of size 75M and 350M parameters. In particular, we are interested in understanding how the client and server models interplay as the FL converges. As Fig. 12 shows, the server model initially "pulls back" the clients' model norms through the aggregation. After a few rounds, the federated aggregation starts incrementing the norm of the averaged clients' models until the global and local models converge to the same norm. Figure 12 showcases how the client and server optimizations interplay to agree on an initial set of global model parameters and then converge to an optimum. The same trends are present for our heterogeneous partition of The Pile (Fig. 7) and for our partial-participation experiments (Fig. 8).



(a) **(b)**

Figure 12: The l_2 norms of the global model, client models, and the average of client models for our 75M (a) and 350M (b) experiments. While in the transitory early stage, the server model grows slower than the aggregate of client models, it is consistently larger in later stages. These figures reflect both clients reaching a consensus and the ability of the momentum mechanism to stabilize the optimization trajectory.

D The Landscape of LLM Training

Generative pre-trained large language models (LLMs) have demonstrated powerful performance across various natural language processing tasks, leading to rapid and widespread adoption. They are trained on massive corpora of text, incorporating mixtures of low-quality web-scraped data and high-quality curated datasets [46, 47]. Recently, several institutions have released their pre-trained LLMs,

either in the closed form, such as GPT-4 [47], Chinchilla [5], and Gemini [48], or the open-sourced form, such as BLOOM [49], LLaMa [41, 42], and Falcon [50]. The scaling laws identified by Kaplan et al. [1], Hoffmann et al. [5] dictate that model size and dataset size should be increased in equal measure to improve model performance best. These suggest a future race between entities interested in developing state-of-the-art LLMs to grab as many compute and data sources as possible. Thus, LLMs are headed in a promising direction that can become even more luminous by gaining the trust of private entities possessing an unprecedented breadth of knowledge and computing resources [15].

In Appendix D.1, we describe the current landscape for generative pre-training of LLMs with particular attention to the techniques for centralized distributed training. We discuss federated learning (FL) and its impact as a communication efficient technique in Appendix D.2. Appendix D.3, for completeness, presents the effort the community has put into matching FL with LLMs fine-tuning. However, we highlight that our work tackles the far more challenging problem of federated pre-training of LLMs.

D.1 Centralized Distributed Optimization

LLM pre-training is based on two denoising pillars: leveraging huge batch sizes and very long contexts, i.e., sequence length of a single input sample. The combination of model and batch sizes forces training LLMs to scale SGD beyond the confines of a single GPU. Thus, it is often necessary (a) to process more training samples in parallel and (b) to split the model across GPUs if it cannot fit within the memory of one GPU. In the following, we present a subset of optimizations and techniques the research community proposed to make this challenging training recipe possible. As we will see in Appendix D.2, FL follows as a natural next step in the sequence of distributed training optimizations previously adopted. Given the importance of these optimizations even for FL's local training step, our work supports most of them, as described in Appendix G.1 and Section 2.1.

D.1.1 Data and Model Parallelism

The number of trainable parameters and the size of the datasets make LLM training very sensitive to the stochastic fluctuations of the optimizer used, thus requiring a solid and robust regularization achieved by the denoising properties of enormous batch sizes. Distributed Data Parallelism (DDP) replicates the model N_d (number of devices) times across different devices to enable training with sufficiently large batch sizes. Then, DDP splits each enormous batch into micro-batches, subsequently fed into each replica in parallel. The number of samples in every micro-batch depends on the available memory in the hardware accelerator; as such, it is often referred to as device batch size. Thus, each replica produces the gradients of its training micro-batch locally and accumulates them via simple summation. This accumulation is eventually followed by a synchronization step where these gradients are averaged and applied to the model. Since the batch size of an ML algorithm plays a crucial role in its convergence [51, 52], this synchronization step happens after each replica has accumulated sufficient micro-batches to match the desired true batch size. Modern DDP implementations such as the one used by `PyTorch Distributed` [34] use the `Ring AllReduce` algorithm popularized by `Horovod` [53] to reduce the gradients across replicas. The algorithm is implemented with low-level collective communication libraries like Nvidia's `NCCL` or Facebook's `Gloo`. `Ring AllReduce` scales proportionally to the amount of data being transmitted and the latency of the slowest connection between workers in the ring, making it highly sensitive to the network topology connecting GPUs.

In the case of LLM training, the model size often mandates computing gradients over batch sizes larger than what GPUs can support, even with DDP. Thus, practitioners are forced to use a micro-batch size dictated by the number of training samples that fit in the GPU, which is, in turn, limited by the VRAM occupancy due to model parameters and activations. These factors result in a complex interplay between the model's memory consumption and the training pipeline's efficiency. The resulting memory-efficiency trade-off is even more complex when considering how many ways the models may be partitioned across GPUs to reduce their VRAM consumption.

For sufficiently large models, the parameters must be split across GPU workers so that they fit in VRAM. Traditionally, this was achieved via Model Parallelism (MP) [54, 55], splitting individual tensors and their computation vertically across GPUs and communicating as needed. While model parallelism can reduce the per-GPU memory requirements linearly by the model parallelism degree

N_m (number of model splits), it induces communication overheads that are tolerable in a single-machine context with high inter-GPU bandwidth but cannot scale to multi-machine contexts.

Algorithm 1 Distributed Data Parallel (DDP) Training Algorithm

Require: N: Number of devices (workers), f_θ: Model with parameters θ
Require: 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

D.1.2 Fully Sharded Data Parallelism

An alternative approach is to shard the model into equally-sized units amongst GPUs, with units potentially containing multiple layers, and then materialize the units, as necessary, to compute the activations during the forward pass via collective communication. This form of fully-sharded data parallelism [44, 56] reduces memory consumption linearly in the data-parallelism degree N_d while increasing communication by $1.5\times$ compared to standard DDP. It is also possible to combine this methodology with other techniques for reducing memory consumption, such as model parallelism, activation checkpointing [57], or CPU offloading [58]. Activation checkpointing functions by re-computing activations during the backward pass rather than saving them in memory. CPU offloading refers to offloading either data to system RAM or operations to the CPU.

Given the memory requirements, it is crucial to consider the minimum number of GPUs necessary to train one model with reasonable efficiency, e.g., without extreme CPU offloading. This estimate provides a lower bound on the hardware that an organization requires to participate in any distributed training of an LLM and is generally determined by the model size and the micro-batch size. The micro-batch size is also known as the device batch size since it is bounded by the number of samples that can fit on a GPU device together with the model. By accounting for each organization's independent resources and manipulating the amount of local computation they do together with the micro-batch size, our method can relax the lower bound within reasonable limits, thus allowing even organizations with weak hardware to participate in distributed training.

D.1.3 Bottlenecks for generative pre-training of LLMs

High-quality public language data is liable for exhaustion within the next decade, while low-quality language data may be exhausted in a few decades [12]. When taken together with a growing interest from both individuals and corporations in constraining what data can be scraped from the Internet or used to train an LLM, this limits the size of models that can be efficiently trained. Circumventing this either requires costly independent deals with data providers [11], leaps in the effectiveness of synthetic data generation for model training [59], or significant improvements in the data efficiency of ML optimization.

Similarly, hardware accelerators with enough memory and throughput to support LLM training are scarce and increasingly unaffordable to anyone interested except for the best-funded organizations. Hundreds to thousands of such accelerators are required with extremely high monetary costs for training and inference [49, 60]. Moreover, such accelerators must be largely uniform in specifications to avoid stragglers and need to be extremely well-connected both within a computational node and across nodes due to the synchronization requirements of data parallel or fully shared data parallel SGD [34, 44, 56]. These constraints pose significant system challenges that can be mitigated by extremely expensive data center configurations or by developing heterogeneity-aware pipelines that exploit the resources available despite their heterogeneity. The difficulties described above scale with model size, as splitting the model across the memory of several GPUs further increases communication demands [44, 56] and the optimization gaps.

D.1.4 Mitigation of LLMs demands

The wider AI community has gone to great lengths to circumvent the high requirements of LLM utilization and thus expand the data pool we can draw upon. Such endeavors have widely focused on locally running efficient inference with pre-trained model weights [41, 42], quantization [61], or parameter-efficient fine-tuning [62]. The recent work proposes Petal [63], which enables wide-scale collaboration for inference and parameter-efficient fine-tuning over the Internet by joining the resources of multiple parties. Their work assumes that clients provide inference jobs or data for fine-tuning, and servers execute the LLM inference/fine-tuning in a distributed pipeline parallel fashion. Their system makes clients responsible for holding the trainable parameters to enable efficient fine-tuning, with servers merely running forward passes and returning gradients for the pre-trained weights they store.

We argue that while methods exploiting pre-trained weights are highly beneficial to the broader community, they do not resolve the bottleneck of pre-training. Thus, the community is bound to the decisions made by organizations capable of training LLMs regarding their structure and the data they use, and they may suffer downstream consequences for such reliance. Performance degradation may happen since the pre-trained model, further fine-tuned for a specific downstream task, is the dominant upper bound for the downstream model's performances. Thus, we propose developing systems capable of the distributed *pre-training* of LLMs that can accommodate the variety of hardware and data available in the AI community. Nevertheless, this new paradigm may include training from scratch if the model's size permits, starting from pre-initialized weights and retraining the entire model instead of only a limited subset of parameters.

D.2 Federated Learning and Local SGD

Traditional machine learning involves using a central server that hosts the machine learning models and all the data in one place. In contrast, when using federated learning (FL) [18] algorithms, client devices collaboratively learn a shared global model using their local compute and data.

FL aims to collaboratively learn a global model while keeping private data on the device. Such a training pipeline occurs over multiple communication rounds. During each round, a fraction of the clients are selected and receive the global model from the server. Those selected clients then perform local training with their local data before sending the updated models back to the central server. Finally, the central server aggregates these updates via averaging [18] into a **pseudo-gradient**. It then uses a federated optimizer [37] to update its model based on the pseudo-gradient, creating a new global model. Then, this three-stage process is repeated.

Federated optimization has several properties that make it suitable as a new paradigm for LLM training: (a) it does not require the private data of participants to be directly shared, (b) it can naturally incorporate Differential Privacy [64] or Secure Aggregation [65] to compile with privacy regulations at an actor level, (c) it allows for more control over the optimization and has less restriction on the connectivity as each data-source can be associated with a series of updates. Crucially, since FL allows previously unseen data to be accessed during training, it reduces the likelihood of data memorization and leakage, which have become increasingly common as model size has increased [13, 14].

Despite these advantages, FL comes with two major challenges in the form of data and systems heterogeneity [35]. Data heterogeneity refers to the tendency of naturally generated and partitioned data to fail the IID assumption, which is common in centralized ML optimization. Systems heterogene-

Algorithm 2 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
Require: `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

ity refers to the ability of client hardware to vary in terms of computational ability, communication efficiency, or availability [66]. Both forms of heterogeneity are highly relevant for the distributed training of LLMs. Data heterogeneity may arise from participants holding texts in different languages, belonging to various genres, or varying in complexity. Systems heterogeneity is primarily present in the computational ability of the GPUs held by a specific client, their VRAM, and number, as well as the communication efficiency of said client.

Local SGD [16, 67] is a data-parallel training paradigm where each replica applies independent gradient updates to its parameters for several local steps before averaging parameters rather than gradients. While mathematically equivalent to the FedAvg [18], the context in which it is applied, lowering the communication costs in centralized training, lacks the hardware and potential data heterogeneity specific to FL clients. Unlike previous work [19], we intend to go beyond local SGD towards fully federated training inclusive of the broadest possible range of participants.

D.3 Federated Fine-tuning and Parameter Efficient Fine-tuning of LLMs

Until now, full federated pre-trained LLMs have not been accomplished because researchers could not solve the dual challenges of its communication overhead and pre-training large models on resource-challenged devices. That said, researchers, eager to reap the benefits of federated learning, have nonetheless made progress on federating downstream LLM training tasks whose computational and communication demands are lower, such as fine-tuning, parameter-efficient fine-tuning (PEFT), and prompt-tuning. We outline these developments here for a complete reference.

For example, Hilmkil et al. [68] use FL to fine-tune all the model parameters of ALBERT [69] and BERT [70], reaching 90% of the accuracy achieved by a centrally trained model on text classification tasks. Meanwhile, Riedel et al. [71] found that BERT fine-tuned in an FL setting could perform as well as a centralized model on multilingual text classification tasks. Wang et al. [72] and Weller et al. [73] also conducted federated fine-tuning on private data but preceded it with centralized pre-training on public data. They found that this workflow improved fine-tuning accuracy, even when pre-training only on a 1% sampling of a large public corpus or if the client's data was non-IID.

Much progress has also been made on federated PEFT, whose computational and communication hurdles are lower than those of federated fine-tuning. Researchers have shown that a model that

has been subject to federated PEFT can outperform the original pre-trained model [74], outperform siloed client models [75], and even outperform federated fine-tuning [76, 77], including in non-IID scenarios [78], but with far lower computation and communication costs because clients only need to update and transmit the smaller set of parameters. Federated LoRA, for example, may consume as little as 0.058% of the communication cost of federated full fine-tuning [75]. To reduce these costs even further, Xu et al. [79] add Noise Contrastive Estimation to reduce memory demands and Xu et al. [80] add backpropagation-free training to improve memory and time efficiency. Meanwhile, to address the more significant impact that non-IID client distributions can have on federated PEFT performance, Babakniya et al. [81] precede federated LoRA with a round of federated efficient sparse fine-tuning, reducing the performance gap while keeping combined training time low. Differently, Kim et al. [82] abandon a global model altogether, instead using FL to train client-specific models that benefit from sharing data via FL but are more resilient to client drift.

Federated prompt tuning, wherein clients tune a set of continuous soft prompts appended to input prompts, has also demonstrated its effectiveness. This technique can significantly reduce the number of model parameters that clients must train and transmit compared with full fine-tuning. Models fine-tuned this way perform better than siloed client transfer learning [76], better than centralized training by benefiting from improved generalisation [83, 84], and also reduce the impact of client drift due to non-IID data [85, 86] by leveraging an adaptive optimization method.

E Design Principles for Federated Generative Pre-Training of LLMs

We propose a federated LLM generative pre-training paradigm to create new pre-trained models over which the previously mentioned fine-tuning techniques may be applied. Thus, we aim to disentangle the broad, distributed community of researchers and practitioners that build upon LLMs from the willingness of large organizations to provide open-source weights.

This section proposes a series of inclusive principles necessary for such federations to be effective and incorporate them into the design of our system, *Photon*. The principles we chose are meant to tackle the foundational issues of federated LLM pre-training with a particular focus on data and hardware inclusivity, robustness, and efficiency.

Broad Access to Data and Compute: The ability to train an LLM should depend on the data that a participant or a group possesses rather than unrestricted access to hardware. Thus, we aim to transform organizations that right now can serve as data providers only [11, 87], when their ownership of the data is respected, into active participants in the LLM training process. We believe that incorporating such contributors directly into the federated learning process and offering them an incentive to participate, obtaining a model performing well on their data, is the natural next step in the proliferation of generative AI generally and LLMs in particular. While some data-rich organizations may be unable or unwilling to invest in computing power, we believe that **voluntarily** partnering with a compute-rich yet data-poor one in a federated training context provides an excellent avenue for their participation. Thus, *Photon* aims to create a broad network of data producers and consumers where a single client may offer computing resources, data sources, or both to the training process.

Limited Communication Requirements: Pre-training should be possible without the strong synchronization requirements of standard data-parallel training [44] to accommodate geographically distributed and poorly connected participants. Our federated learning solutions allow orders-of-magnitude reductions in communication frequency compared to centralized solutions. While such improvements boost efficiency for all participants in the federation, they also offer particular benefits to data sources that may have been underrepresented in the past. For example, incorporating private data silos from regions with a lower internet presence may help alleviate the challenge of NLP for low-resource languages [88, 89].

Broad Hardware Inclusivity: Organizations with valuable data must be able to participate in the federated network, even with poorly connected or limited hardware resources. In the case of clients that may hold distributed compute nodes lacking the connectivity necessary to support the high-bandwidth `Ring AllReduce` algorithm necessary for classical data-parallel training (e.g., Infiniband), we perform local federated training over their compute nodes and transparently aggregate the node updates before sending results to the main server. Suppose a client suffers from either com-

putational or communication limitations. In that case, we account for it by adjusting the maximum round duration or allowing them to modulate the amount of local training they undertake. Thus, our system can accommodate a broad set of potential participants with hardware varying from a few medium-powered GPUs to small data centers.

Scalable Local Training Pipelines: Given that the participants in federated training are likely to vary in terms of model-size requirements, for example, if they prefer smaller models capable of faster training more specific to their data or more appropriate for the amount of data held by each of them, a federated training system must be effective across a wide range of model sizes. Thus, we build our system to support local training pipelines appropriate for all scales from billions-sized models with fully-shaded data parallelism [44, 56] to simple data parallel training [34] and CPU offloading [58]. While this work focuses on the most challenging large-scale setting, our codebase is highly adaptable to all requirements.

F *Photon* Design

Photon is a system for federated generative pre-training of large language models (LLMs), allowing organizations with *limited communication capabilities* to collaboratively train a federated learning (FL) model using their private data sources. True to FL principles, *Photon* guarantees that private data is never exchanged or moved without the consent of the participants, which is a necessary condition as opposed to the centralized paradigm that uses public data. Following our principle of *broad hardware inclusivity* (Appendix E), *Photon* allows FL participants to execute *highly scalable local training pipelines* with heterogeneous computing resources in amount and type (e.g., different generations of GPU), although sufficient (e.g., just one server with a few data center-level GPUs). Furthermore, to offer *broad access to data and compute*, *Photon* allows clients that do not hold both data and compute to self-organize into a distributed network of voluntary data producers and consumers while keeping their data private from all other participants. *Photon* aims to satisfy the system requirements and tackle the challenges for generative pre-training of LLMs, which FL can sometimes exacerbate. This training paradigm requires infrequent communication of unprecedentedly big payloads (for FL) through unreliable heterogeneous networks. Every FL system is prone to performance degradation due to dynamic client availability, stragglers, hardware heterogeneity, and unexpected dropouts. Although some of these are present even in the centralized paradigm (e.g., stragglers and dropouts), their impact on this FL application is more severe as they may result in wasting several precious GPU hours.

The features described above make *Photon* the first of its kind to successfully address the system challenges for federated generative pre-training of LLMs, paving the way for a more democratic training environment for federated LLMs. Our system provides three building blocks to execute such a federated pipeline: one central server orchestrating the FL, called *Photon Aggregator*; at least one distributed *Photon LLM Node* supporting scalable local training; and one distributed *Photon Data Source* for each participating client running on the edge. We describe these components and their interaction in Appendix F.1, while we further detail *Photon*'s scalability properties in Appendix F.3.

F.1 *Photon* Architecture

As discussed below, *Photon*'s three core components are co-designed to account for system requirements and challenges of the specific setting featured in this work, following the principles extensively described in Appendix E. The main objective of this discussion is to promote reproducibility and further developments of the federated pre-training of LLMs. Fig. 13 presents a high-level overview of the architecture of *Photon*, comprehensive of its three core components and their sub-elements.

***Photon Aggregator*:** The central component of standard FL systems is the server responsible for *orchestrating the training* procedure and maintaining the *global training state* up to date. *Photon Aggregator* plays this role by using the sub-components described here. Before starting any distributed step in the federated pipeline, the *Photon Aggregator* summons the *client sampler*, which assesses how many *Photon LLM Nodes* are available and selects a number of them depending on the requirements of the optimization algorithm. After such selection, the *Photon Aggregator* broadcasts the current global model and the training instructions to the appointed *Photon LLM Nodes*. Any

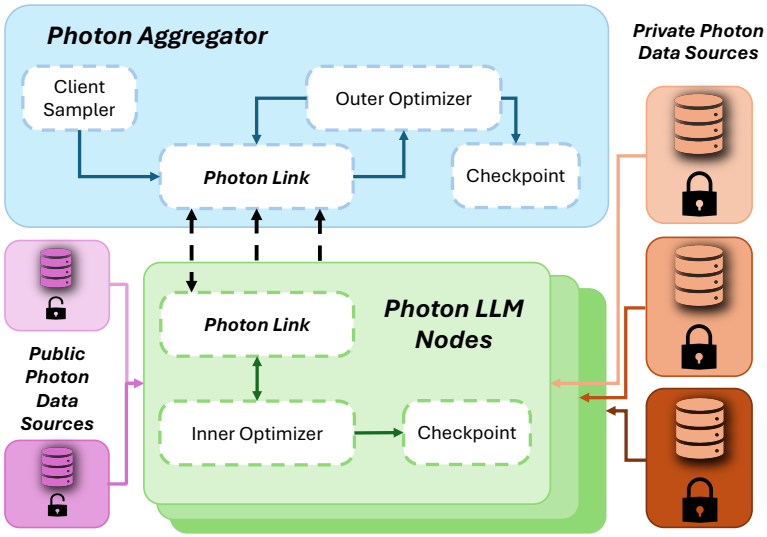


Figure 13: The diagram describes the *Photon*'s three principal components - *Photon Aggregator*, *Photon LLM Node*, *Private Photon Data Sources* and *Public Photon Data Sources* - and their sub-components. Arrows describe how such elements work together or exchange messages. The *Photon Aggregator* can communicate with the *Photon LLM Nodes* only through the *Photon Link*. The instances responsible for storing the data samples, the *Photon Data Source*, can uniquely stream to the *Photon LLM Node* bonded to them.

message between the *Photon Aggregator* and the *Photon LLM Nodes* travels through the *Photon Link*, which acts as the communication gateway for both components. *Photon Link* also supports secure communication protocols, such as HTTPS and the more complex secure aggregation [90]. The message payload includes, but is not limited to, model updates, training and evaluation instructions, client training or evaluation metrics, and side information (when considered safe to share). The *Photon Link* receives the results of the distributed task executing at the *Photon LLM Nodes*, such as model updates. The *outer optimizer* applies the gradient updates to the global model – aggregation step – using the appointed optimization algorithms, such as FedAvg [18], FedProx [91] or FedOPT [92]. When the method is associative, the *outer optimizer* further improves its efficiency by taking advantage of asynchronous partial aggregation of the client updates. The *Photon Aggregator* guarantees robustness in case of failures by keeping the state of the FL continuously *checkpointed*. This state contains the global model parameters, the snapshot of the appointed outer optimizer's attributes, and bookkeeping metrics such as the timestamp and the elapsed time. We assume that the *Photon Aggregator* is well-connected to the *Photon LLM Nodes* through the Internet and doesn't require particularly powerful computing capabilities, i.e., a commercial-level CPU is sufficient. However, a *Photon Aggregator* requires sufficient storage capacity to maintain the training state, which can be on a local disk or in the cloud.

***Photon LLM Node*:** The *Photon LLM Node* is the distributed component of *Photon*, which executes the *local training pipeline* of the federated optimization. Each client in the federated population has a *Photon LLM Node* that can arbitrarily connect or disconnect to *Photon*'s runtime at any time during training. A *Photon LLM Node* leverages several sub-components to fulfill the requested training tasks robustly. The *inner optimizer* trains the latest model received by the *Photon LLM Node* on the local client data using a selection of distributed optimization methods [34, 44], which allow increasingly high memory savings at the cost of additional inter-GPU communication. Given that our clients may range from single machines with a few GPUs to clusters of machines, the *Photon LLM Node* selects the preferred local training method following the client's hardware resources, their connectivity, and the model's size. The *Photon Link* can exchange model parameter updates and training instructions with *Photon Aggregator*. Given the large size of such models and the abundance of pruning techniques available, the *Photon Link* also handles model compression.

We do not prune the model by default and only use lossless compression. Similarly to the *Photon Aggregator*, the *Photon LLM Node* has a *checkpointing* component that guarantees speedy recovery in case of disruption. In addition to the model parameters, this local state must track the optimizer and data loading index states and the number of full epochs the client has completed. Since federated quantity skew may result in clients having very different local dataset sizes, the checkpoints save the dataset state privately without any server control that could impact the optimization's fairness. Our assumptions for the minimal hardware capabilities of *Photon LLM Nodes* require them to have at least a few data center-level GPUs sufficiently supported by the RAM and CPU. We also assume the *Photon LLM Nodes* tightly connect to their coupled *Photon Data Source* through the Internet.

***Photon Data Source*:** FL requires strict guarantees regarding where to store data samples and how to exchange them. To systematically satisfy such requirements, we designed *Photon*'s data storage component, *Photon Data Source*, to tie uniquely to a *Photon LLM Node*. *Photon Data Source* provides the training pipeline executing on the *Photon LLM Node* with *continuous streaming of data samples* respecting the pipeline's throughput demands. Exploiting this architecture, an institution possessing a large silo of data samples that may want to participate in the FL can either acquire computing resources to build a *Photon LLM Node* or reach an agreement with a trusted third party with extensive computational resources. *Photon*'s design permits such confined collaborations that are easier to achieve than sharing global data and computing resources.

Algorithm 3 *Photon* execution pipeline

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

procedure PHOTONSERVER(T, K, H, P)
  $\theta^0 \leftarrow \texttt{InitModel}(H)$ ▷ Init on the server or sample a client and extract model weights
  for each round $t = 1, 2, 3, \ldots, T$ **do**
    $\mathcal{C} \sim \mathcal{U}(P, K)$ ▷ Sample K clients at random from the population
    for $k \in \mathcal{C}$ **do in parallel** ▷ Each sampled client in parallel executes the local training
      $\theta_k^t, \mathcal{M}_k^t \leftarrow$ PHOTONCLIENT(k, θ^t, H)
      $\Delta_k^t \leftarrow \theta^t - \theta_k^t$
    $\Delta^t \leftarrow \frac{1}{|C|} \sum_{k \in C} \Delta_k^t$ ▷ Aggregate pseudo-gradients Δ_k^t from clients
    $\theta^{t+1} \leftarrow \text{ServerOpt}(\theta^t, -\Delta^t, t)$ ▷ Apply pseudo-gradient to the global model
    $\mathcal{M}_k^{t+1} \leftarrow \texttt{AggMetrics}(\mathcal{M}_k^t \; \forall k \in \mathcal{C})$ ▷ Aggregate metrics across clients
    $\texttt{Checkpoint}(\theta_k^{t+1})$ ▷ Checkpoint model
  return θ_k^{T+1}

procedure PHOTONCLIENT(k, θ^t, H)
  $\mathcal{D}_k \leftarrow \texttt{BindStream}(k)$ ▷ Bind *Photon Data Sources* to a merged data stream $\mathcal{D}_k$
  $I_k \leftarrow \texttt{GetNodes}(k)$ ▷ Extract hardware configuration I_k
  if $\texttt{HasInfiniband}(I_k)$ **then**
    $B_k \leftarrow \texttt{CalcBatchSize}(I_k)$ ▷ Binary search for batch size B_k with static initial guess
    $\theta_k^t, \mathcal{M}_k^t \leftarrow \texttt{TrainClient}(\theta^t, \mathcal{D}_k, B_k, H)$ ▷ Use DDP or FSDP based on model size
  else
    for node $i \in I$ **do in parallel** ▷ In every node of the current client do FL
      $B_k^i \leftarrow \texttt{CalcBatchSize}(i, I_k)$
      $\mathcal{D}_k^i \leftarrow \texttt{PartitionStream}(i, \mathcal{D}_k)$ ▷ Split the client data into $|I|$ shards
      $\theta_i^t, \mathcal{M}_i^t \leftarrow \texttt{TrainClient}(\theta^t, \mathcal{D}_k^i, B_k^i, H)$ ▷ Use DDP or FSDP based on model size.
    $\theta_k^t \leftarrow \frac{1}{|I|} \sum_{i \in I} \theta_i^t$ ▷ Partially aggregate node models
    $\mathcal{M}_k^t \leftarrow \texttt{AggMetrics}(\mathcal{M}_i^t \; \forall i \in I_k)$ ▷ Partially aggregate metrics across nodes
  $\texttt{Checkpoint}(\theta_k^t, \mathcal{D}_k)$ ▷ Checkpoint model and dataset state
  $\theta_k^t \leftarrow \texttt{PostProcess}(\theta_k^t, \mathcal{M}_k^t)$ ▷ E.g., apply differential privacy or compress the model
  return $\theta_k^t, \mathcal{M}_k^t$

F.2 Workflow Overview

We are now ready to discuss the workflow of using *Photon*, which is succinctly illustrated in Algorithm 3. A standard application for *Photon* is that of multiple independent institutions possessing siloed data sources and computational resources. Each of these institutions will need to instantiate a *Photon LLM Node*. The *Photon Aggregator* could be one of the collaborating institutions or an appointed third party to provide further privacy protection and reliability. Regardless of the hosting policy, the server is responsible for initializing a model or sourcing it from a participating client (L.1), coordinating rounds and sampling participating clients (L.3 – 4), of sending the model parameters to the clients and collecting results (L.5 – 7), and finally building new federated models (L.8 – 9). Each institution participating with data sources must set up *Photon Data Source* to supply the *Photon LLM Node* with training and evaluation samples. There might be private agreements between organizations holding data sources without possessing any computing resources and other organizations in the opposite situation. These two organizations will collaborate to provide one coupling of *Photon Data Source* and *Photon LLM Nodes*. Public *Photon Data Source* is also permitted, which can be used for initializing the global model in a pre-training fashion, regularizing the FL, and mitigating the impact of the heterogeneity of *Photon Data Sources*.

Once all the parties have set up *Photon*'s components, the *Photon Aggregator* starts the FL by initializing the global model (L.2) and sampling the *Photon LLM Nodes* (L.4) participating in the first iteration. The following steps describe such iteration, repeating until the global model converges. The *Photon Aggregator* broadcasts the model parameters to the sampled clients to execute the described local training (L.6). Then, the *Photon LLM Nodes* start the local training using the data samples they query from the *Photon Data Source* (L.13). To efficiently complete the training task, *Photon LLM Nodes* assess the hardware resources available (L.14) and choose the optimal execution strategy. They take advantage of the fast connectivity between GPUs on different nodes if available (L.15), or they execute another level of federated optimization if connectivity between nodes is poor (L.19). In the second case, an additional partial aggregation step (L.23 – 24) between nodes is necessary to complete the local training. At the end, clients produce model updates they send back to the *Photon Aggregator* (L.25). Before sending these updates (L.27), clients apply local post-processing steps such as gradient clipping and noise injection in the case of differential privacy [64, 93], or model compression [94] for lower server-client communication costs. After all models have been transmitted, the iteration comes to the last step: the aggregation. The *Photon Aggregator* produces a new version of the global model using the updates received (L.9). The next iteration will use this new version of the global model parameters. The validation of the training procedure is always performed on a held-out split of the samples available in any *Photon Data Source*. *Photon Data Source* ensures this split is preserved and streamed to the *Photon LLM Nodes* when asked to validate the current version of the model. As such, *Photon* allows for distributed evaluation of the local and global models at any time during the FL. The FL can also be validated with public *Photon Data Source* to take advantage of well-defined benchmarks, such as perplexity on Cleaned Colossal Common Crawl (*C4*) [32], which allows meaningfully comparing the capabilities of the FL models.

This ability to choose between public and private data enables *Photon* to assess performance in a global context (i.e., on public data or the average performance across client test sets) or a personalized one (i.e., on one client's specific private test set). The ability to account for local performance opens the door for various fair aggregation algorithms [95, 96] and allows for easy evaluation of bias when accounting for AI regulation.

F.3 Communication-computation Scalability

Photon is built for a setting that crucially differs from the settings of previous proposals and applications [18, 66]. Prior system designs have mostly focused on cross-device FL, which often has opposite demands compared to the cross-silo setting discussed in this work. While the very first FL application [18] has been deployed for training a model with 860K parameters, *Photon* aims to train scalable LLMs of unprecedentedly large sizes (i.e., several billions of parameters). While sampling from populations of millions of devices with heterogeneous data and hardware brings many challenges [51, 97], *Photon* is intended for cross-organizational use cases where the number of participants may reach the hundreds. Furthermore, although FL is often challenged by executing ML pipelines on constrained devices (such as smartphones, laptops, or other edge devices) [98], *Photon*

assumes clients either have data center-level hardware accelerators, though limited in number, or that some clients have such accelerators but insufficient data and may be willing to join a private voluntary partnership with a data producer which does not own such hardware. These differences with previous works implicitly reinstate how the *Photon*'s scalability properties impact the FL.

The communication step is considered the major bottleneck in cross-device FL settings, as local training is performed by constrained devices with small models on a few data samples. This is not necessarily the case for *Photon*'s use cases - we assume the *Photon Aggregator*'s connectivity with the *Photon LLM Nodes* is fast and stable based on industry-level access to the Internet. One may expect the communication steps to become even more problematic when training an LLM, as each communication has a payload size several times greater than the usual models in cross-device settings. However, in compute-intensive tasks such as LLM training, the bottleneck introduced by external communication between client and server is negligible compared to the computational time. Indeed, such computationally intensive tasks represent the predominant factor in the runtime for *Photon*'s execution, as long as the number of local steps chosen is the range proposed in this work. This does not apply to cross-device situations, where the limited hardware and data available on edge devices restrict the number of possible local steps that can produce a meaningful aggregation on the server. This cross-silo approach differentiates *Photon* from previous work [31], which trained clients using 4 local steps instead of 500 with a batch size of 16 rather than 512 as *Photon* uses. Given that this amount of local steps approaches FedSGD [18], which is known to be impractical in wall-clock terms even for small CNNs due to communication inefficiency, we argue that *Photon*'s approach for cross-silo represents a far more promising avenue.

Optimizing the efficiency of the compute-intensive part of the system requires more attention than optimizing the communication part, even though the latter can still take advantage of the several optimizations the community has proposed [94].

For an FL pipeline with a fixed-size model, the local training step takes a linear time with the number of local steps performed by the client, assuming that no particular system optimization has been performed and the workload for a single step fits the hardware resources. The transmission of a model's parameters with fixed precision takes linear time in the number of parameters, assuming a constant transmission speed without chunking or compression.

Parallel and Sequential Computation: Given the paramount nature of clients' computing capabilities, compute efficiency is the most important aspect to consider when evaluating the efficiency of federated learning. To properly understand the scaling properties of FL, it is first important to distinguish between *parallel* and *sequential* optimization steps. In stochastic gradient descent (SGD) context, each sequential step (batch gradient) changes the model, causing future sequential steps to calculate gradients w.r.t. a different set of parameters. In contrast, samples within a batch are evaluated in parallel and thus produce a gradient w.r.t. the same parameters before they are aggregated and applied to the model. As such, for standard data-parallel techniques [34, 56], the parallelization unit is a batch, and each batch completed moves the model forward. Once federated optimization is considered, each client will perform a series of local updates in parallel and thus produce different models. However, these models are all computed w.r.t. the same model received at the start of the federated round. Thus, when the server averages client updates, it averages over all the local steps of the clients, moving in the approximate mean direction of all clients and thus benefiting from access to more data and lower variance [97]. Thus, FL serves to provide access to a practically larger batch size [17] with a far greater variety of data available, enabling us to satisfy the data demand of scaling laws [12] at the cost of having to train on more samples in parallel rather than sequentially.

There is strong evidence from the cross-device FL literature [99, 97], which our evaluation supports in the case of LLMs, that training on a larger proportion of the total client population during a round has diminishing returns as long as unbiased random sampling is applied. Thus, for a sufficiently large training population of size P (e.g., between 32 and 128 clients) training for a fixed number of local steps, it may be possible to boost the statistical efficiency of training by $5 - 10\times$ over full participation by simply sampling only 10% of clients each round without hurting model quality. While a client's data may be overlooked in a given round, it will eventually be seen and incorporated into the model.

G *Photon* Implementation

This section discusses the details of *Photon*'s implementation starting from the three principal components described in Appendix F. As for its software infrastructure, *Photon* is a distributed system consisting of a set of microservices [100], where each microservice tackles a well-defined component of the LLM training pipeline. We hope this description allows any group of organizations interested in the collaborative training of LLMs to replicate and build upon our design effectively.

Both *Photon Aggregator* and *Photon LLM Node* are microservices implemented using `Python`, `Pollen` [23], `Flower` [22], and `PyTorch` [101]. We have chosen `Flower` and `PyTorch` due to their technical maturity and ease of customization. In particular, `Flower` provides *Photon* with the underlying communication layer and the basic FLOps for orchestrating the FL training. We use the latest version of this framework, recently released and named `Flower Next`[1]. We added some customizations to the framework to enable checkpointing operations and improve reproducibility, e.g., reproducible sampling. `PyTorch` is the machine learning framework on which all the local training operations of the *Photon LLM Nodes* are based. We opted for `Pollen`'s edge components for managing internal communication and collaboration at *Photon LLM Node* level.

The entire configuration of each given execution (i.e., the training session) is structured as a hierarchical set of `YAML` files, which are parsed and processed by the *Photon Aggregator* and the *Photon LLM Nodes* using `Hydra` [102] – an open-source `Python` framework that simplifies the development complex applications.

Photon Data Source is implemented on a server with high storage capacity using an instance of `MinIO` [103] – a high-performance open-source object store – that is supported with `Amazon S3`-compatible [104] API. For *Photon Data Source* to be sufficiently compelling with the requirements of a *Photon* execution, the server is backed by an array of solid-state disks (SSDs), providing the best cost-random IO performance trade-off. As various management tools and `RESTful API` client libraries are available to take full advantage of `S3`-based solutions, we built a `Python S3 client` built on top of the `boto3` [105] library tied to the *Photon LLM Node*.

The checkpointing sub-components in *Photon* require storing large binary objects with a frequency that depends on the setting utilized. Such binaries are mainly composed of model parameters that cannot be lossy-compressed to maintain the consistency of the execution. Since `MinIO` provides persistent storage of binary blobs, we also use it to store training checkpoints of the *Photon Aggregator* and the *Photon LLM Nodes*. These components have a `Python S3 client` to communicate with the checkpoint `S3` bucket similarly to that implemented for *Photon Data Source*.

G.1 Parallel Execution within *Photon LLM Nodes*

The local execution engine used in our solution is based on `Pollen`, which can adaptively take advantage of the underlying GPU configuration at a *Photon LLM Node* level. Depending on the model size, a single computer with one GPU per *Photon LLM Node* might provide insufficient hardware resources to execute the pre-training. `Pollen` orchestrates the collaboration of the single training processes on the GPUs available with `PyTorch Distributed` [34]. Thanks to `Pollen`, a *Photon LLM Node* can dynamically set arbitrary collaboration topologies of GPUs through single or multiple `PyTorch Distributed` environments, with support for a range of optimization strategies appropriate to the hardware of a client and the size of the model.

Single-GPU Training In scenarios where the model fits within the VRAM of one GPU with a sufficient device batch size to keep pace with the rest of the federation, *Photon LLM Node* defaults to single-GPU training. While this strategy may seem inappropriate for all but the smallest model sizes, FL's high parallelization potential allows the efficiency of single-GPU training to scale linearly with the total number of GPUs available in the federation. Since FL requires no intra-round communication, this is possible even when poorly connected.

Multi-GPU Training For scenarios where a single client has multiple well-connected GPUs available, *Photon* can exploit the `PyTorch` implementations of Distributed Data Parallel (DDP) [34] or Fully Sharded Data Parallel (FSDP) [56] training. DDP constructs replicas across GPUs, computes

[1] `Flower Next` is the name given to `Flower`'s latest version (`1.9.0`) at the time of writing this manuscript.

gradients independently w.r.t. separated batches, and then synthesizes these gradients on a per-batch basis at a high communication cost. For situations where a model replica is too large to fit on a single GPU with a reasonable batch size, FSDP shards the model together with its optimizer states and gradients by splitting the layers into units that are distributed across GPUs and materialized as needed at $1.5\times$ the communication cost [44] of DDP.

Multi-Machine Training For settings where clients may own more than one GPU-equipped machine, *Photon LLM Node* can optimize the communication cost further than standard `PyTorch Distributed` by adaptively deciding what technique to apply. For very well-connected clusters, *Photon LLM Node* sets up a `PyTorch Distributed` process group containing all GPUs and uses DDP or FSDP (if the model is too large for one GPU). However, when inter-machine connectivity is poor (i.e., it cannot match the speed of high-bandwidth interconnection such as Infiniband NDR or RoCEv2), *Photon LLM Node* creates disjoint dataset partitions of *Photon LLM Node*'s *Photon Data Sources*. Then, it maps each disjoint partition to one of the nodes and treats all of them as separate clients in a sub-federation. Each node then produces a model trained on its shard. These models are then partially aggregated by the client's lead node and transparently sent to the federated server as a single client update. In distributed settings, islands of nodes with high-bandwidth connections or even complex hierarchical structures are common [56, see sec.1, Hardware Heterogeneity]. For such scenarios, *Photon* applies the same procedure over islands of well-connected machines rather than single nodes, using DDP or FSDP across the nodes in the island.

The partially aggregated result is then transparently sent to the *Photon Aggregator*, which combines it with the other clients' results. Thus, while *Photon* is intended for federated training, it can also enhance training efficiency within single organizations when they possess multiple nodes or entire data centers that are geographically distributed and/or poorly connected.

G.2 Streaming Data Sources

Given the heterogeneous nature of naturally distributed data [106], we built our system to support creating clients with heterogeneous data. Our implementation, based on the underlying MosaicML `StreamingDataset` abstraction, does not enforce a one-to-one mapping between clients and data sources; rather, it enables clients to draw upon arbitrary data streams with complete control over the sampling procedure within and across streams. This flexibility enables the creation of a distributed system of data and compute providers, which extends the federated network. Rather than having a fixed set of clients, all of which hold data and compute, the topology becomes enriched with data nodes that provide streams to compute-enabled clients in exchange for access to the final model. To serve as data producers, clients must offer a means of accessing and locating their data buckets to the compute clients willing to use them during training.

By streaming, caching, and optionally pre-tokenizing or compressing the data, we avoid the large storage overheads of transferring it from data-producing clients to data-consuming clients. Furthermore, we allow data sources to be arbitrarily swapped out between rounds and for newly generated data to be incorporated into the federated training process. With pre-tokenization enabled, the computing resources of data-producing clients may also be effectively utilized in the training process, freeing up the computing clients' resources.