PHOTON: FEDERATED LLM PRE-TRAINING

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

ABSTRACT

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

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

Proceedings of the 8^{th} MLSys Conference, Santa Clara, CA, USA, 2025.

Appendix

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

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

A Hyper-parameters

As shown in Table 1, we trained models ranging in size from 125 million parameters to 7 billion for the causal language modeling task. We used the tokenizer presented in (Black et al., 2022) with a vocabulary size of 50 368. The local optimizer the clients use in our experiments is AdamW (Loshchilov & Hutter, 2019), while the server optimizer is FedMom (Huo et al., 2020). For all of our non-DiLoCo experiments, we default to FedAvg with server learning rate 1.0 and server momentum 0.0. The hyperparameters we used are reported in Table 2. We chose to train decoder-only models, although our system could train any LLM architecture because they have become the de-facto standard in language modeling and text generation owed to their sample efficiency.

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

Table 1. Architecture details and local training parameters for our 125M, 350M, 1.3B, 3B, and 7B models. We report the number of transformer blocks, hidden model dimension d, number of attention heads, the linear layer expansion ratio, and Adam's parameters (β_1 and β_2). We also report the vocabulary size of the tokenizer we used (Black et al., 2022) and the sequence length l.

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

Table 2. Hyperparameters used in our experiments. The federated learning rate $\boldsymbol{\eta_s}$ and momentum $\boldsymbol{\mu_s}$ (Huo et al., 2020) are applied by the Photon Agg. $\mathbf{S_C}$ are the parameters of the learning rate scheduler synchronized across **sequential** steps. α is the factor to be applied to the maximum learning rate η_{max} to obtain the minimum learning rate for the cosine scheduler, i.e., $\eta_{min} = \alpha \times \eta_{max}$. T is the duration, in steps, of the cosine scheduler for fed/cent variants. We also report the batch size used in the local training by the Photon clients and the centralized batch size.

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

B Implementation Details

B.1 Modeling Wall Time

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

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

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

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

Table 3. Hyperparameters for our federated experiments. P represents the total number of clients per federations, K the number of clients sampled per round, D the dataset, τ the number of steps per round.

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

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

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

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

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

where:

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

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

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

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

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

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

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

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

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

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

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

where R is the total number of federated learning rounds.

The aggregation time T_{agg} is calculated by:

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

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

B.2 Performance impact of the *Link* component

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

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

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

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

B.3 Overlapping communication and cleaning up

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

B.4 Advanced sharing for reducing memory footprint

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

C Extended Discussion

C.1 Federated Optimization of LLM Pre-training

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

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

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

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

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

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

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

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

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

C.2 Recent Advances in Federated Optimization

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

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

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

D Additional Evaluation

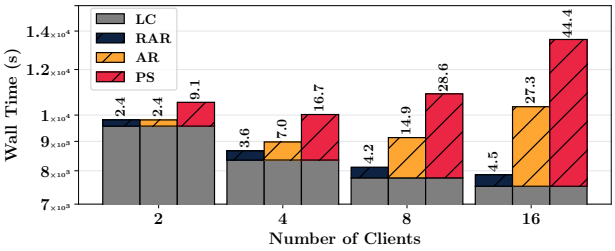

Figure 1. We report the split of the total wall time in two parts: the local compute time (LC) the clients endure to achieve the desired perplexity value and the communication time. The communication time is reported for three different aggregation implementations: *parameter server* (PS), which is necessary when privacy constraints are present; *AllReduce* (AR), more scalable than PS; *Ring-AllReduce* (RAR), the most scalable approach bounded by the slowest link across the ring topology. As we expected, communication represents a more important cost as the number of clients increases. Still, when implemented efficiently (RAR), the wall time benefits of scaling the computing resources are maintained. At the top of each bar, we report the percentage of time spent communicating for the respective experiments and implementation.

D.1 Communication Efficiency and Scalability

We present additional results on the communication efficiency and scalability of Photon when using 64 local steps per round(Fig. 1) and 128 local steps per round(Fig. 2). While using 128 steps results in a slightly increased total computational load due to minor reductions in convergence speed, reducing communication frequency by half significantly lowers the communication burden, particularly with higher numbers of clients per round. This trend is especially pronounced in communication-inefficient PS implementations and also applies to the faster RAR and AR methods.

D.2 Photon Robustness to Node Failures

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

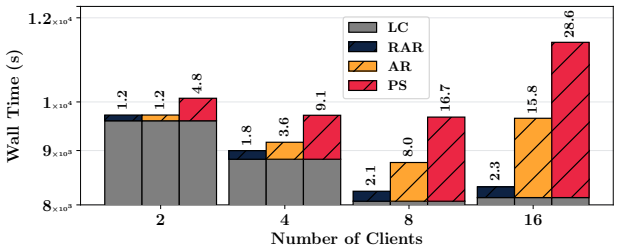

Figure 2. We report the split of the total wall time in two parts: the local compute time (LC) the clients endure to achieve the desired perplexity value and the communication time. The communication time is reported for three different aggregation implementations: *parameter server* (PS), which is necessary when privacy constraints are present; *AllReduce* (AR), more scalable than PS; *Ring-AllReduce* (RAR), the most scalable approach bounded by the slowest link across the ring topology. As we expected, communication represents a more important cost as the number of clients increases. Still, when implemented efficiently (RAR), the wall time benefits of scaling the computing resources are maintained. At the top of each bar, we report the percentage of time spent communicating for the respective experiments and implementation.

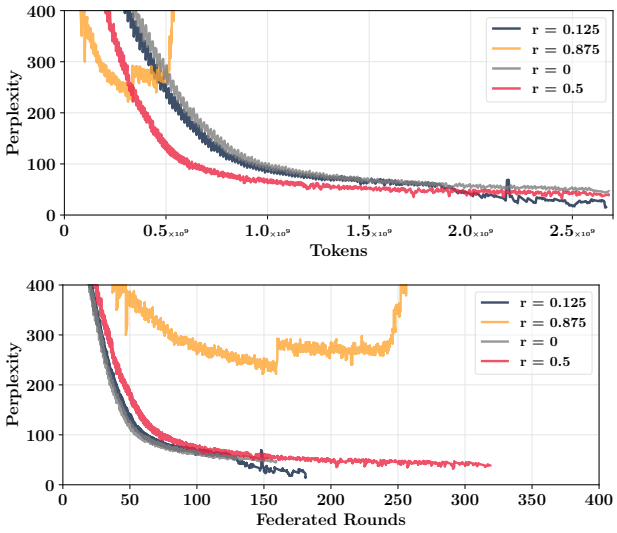

Figure 3. The robustness of Photon for IID data distributions across clients. We show the training perplexity against (top) the number of tokens trained and (bottom) the number of federated rounds for different dropout ratios $r \in \{0, 0.125, 0.5, 0.875\}$ corresponding to $\{0, 1, 4, 7\}$ clients dropping out at every federated round respectively. With the highest value of r, the training procedure fails to converge as there is not sufficient training data per round to leverage the hyperparameter setting. For all other values of r, the federated training succeeds, potentially reaching a final better perplexity with the same number of total tokens. However, values of $r > 0$ result in longer training executions as the number of federated rounds to reach the target number of total tokens increases proportionally to the number of clients dropping out.

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

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

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

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

The relevant comparisons we show in this evaluation relate to how the convergence, in terms of local training perplexity, is impacted by the absence of updates due to clients dropping out. Figures 3 and 4 show that only extreme and unrealistic dropout ratios ($r = 0.875$) can completely dis-

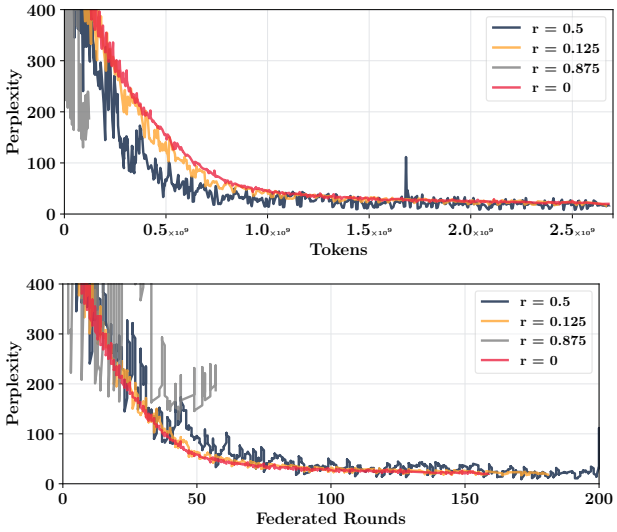

Figure 4. The robustness of Photon for non-IID data distributions across clients. We show the training perplexity against (top) the number of tokens trained and (bottom) the number of federated rounds for different dropout ratios $r \in \{0, 0.125, 0.5, 0.875\}$ corresponding to $\{0, 1, 4, 7\}$ clients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

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

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

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

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

Table 5. In-context learning comparison between Photon models. Our biggest model wins 3 out of 3 comparisons in this group.

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

D.3 Downstream evaluation of Photon's models

To evaluate the downstream task performance of our models, we test across a series of in-context learning benchmarks. Our results, shown in Tables 4 to 7, demonstrate that the downstream performance of models trained with Photon scales as expected with model size, with our largest model winning 10 out of 14 comparisons. This proves the downstream utility of Photon models even when using a pre-training dataset not optimized for downstream performance. We expect that as we increase the model size and incorporate a broader and more qualitative data mixture, the downstream performance of Photon models will keep improving.

Table 6. In-context learning comparison between Photon models. Our biggest model wins 2 out of 3 comparisons in this group.

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

Table 7. In-context learning comparison between Photon models. Our biggest model wins 3 out of 4 comparisons in this group.

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

E FULL ALGORITHMS

Algorithm 1 Distributed Data Parallel (DDP) Training Algorithm

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

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

F ARTIFACT APPENDIX

F.1 Abstract

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

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

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

F.3 Description

F.3.1 How delivered

The artifact is provided in a zipped repository containing:

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

F.3.2 Hardware dependencies

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

F.3.3 Software dependencies

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

F.3.4 Data sets

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

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

F.4 Installation and Usage

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

System prep and environment:

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

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

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

3. **Install dependencies:**

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

Download, prepare/convert dataset with the provided script.

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

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

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

This command executes the following steps internally:

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

F.5 Evaluation and expected result

Targets of interest:

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

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

- **Checkpoints:** Partial and final checkpoints are saved in the `checkpoints/` folder.

F.6 Experiment customization

- **Config override:** Edit `scripts/fed_125m_example.sh` or pass Hydra overrides to change client count or hyperparameters.

- **Hardware scaling:** By default, the script spawns multiple clients on a single node. For multi-node, adapt the aggregator IP and client addresses in `scripts/fed_125m_example.sh`.

- **Batch sizes / epochs:** Controlled by Hydra configs in the `configs/` folder.

- **Dataset:** Replace the small C4 path with your own local data for more extended training.

F.7 Notes

- **Partial or intermittent clients:** If a client crashes or is not reachable, the aggregator continues with remaining clients in subsequent rounds.

- **Performance considerations:** For minimal overhead, ensure a stable GPU environment. Larger-scale runs (1.3B+) require more disk space, memory, and multi-GPU setups.

F.8 Methodology

We adhere to artifact evaluation guidelines:

- Single-blind AE with emphasis on reproducibility and clarity.

- Clear *build* (ffrom the scripts `scripts/system_setup.sh` and `scripts/install_env.sh`), *run* (using the script `scripts/fed_125m_example.sh`), and *analysis* (logs, final checkpoint) phases.

- **ACM Artifact Badging** best practices: code will be made public, well-documented, and tested on a standard environment.

APPENDIX REFERENCES

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

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

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

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

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

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

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

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved

question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Lozhkov, A., Li, R., Allal, L. B., Cassano, F., Lamy-Poirier, J., Tazi, N., Tang, A., Pykhtar, D., Liu, J., Wei, Y., Liu, T., Tian, M., Kocetkov, D., Zucker, A., Belkada, Y., Wang, Z., Liu, Q., Abulkhanov, D., Paul, I., Li, Z., Li, W.-D., Risdal, M., Li, J., Zhu, J., Zhuo, T. Y., Zheltonozhskii, E., Dade, N. O. O., Yu, W., Krauß, L., Jain, N., Su, Y., He, X., Dey, M., Abati, E., Chai, Y., Muennighoff, N., Tang, X., Oblokulov, M., Akiki, C., Marone, M., Mou, C., Mishra, M., Gu, A., Hui, B., Dao, T., Zebaze, A., Dehaene, O., Patry, N., Xu, C., McAuley, J., Hu, H.,

Scholak, T., Paquet, S., Robinson, J., Anderson, C. J., Chapados, N., Patwary, M., Tajbakhsh, N., Jernite, Y., Ferrandis, C. M., Zhang, L., Hughes, S., Wolf, T., Guha, A., von Werra, L., and de Vries, H. Starcoder 2 and the stack v2: The next generation, 2024.

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

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

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

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

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

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

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

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

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., Agarwal, A., Power, A., Ray, A., Warstadt, A., Kocurek, A. W., Safaya, A., Tazarv, A., Xiang, A., Parrish, A., Nie, A., Hussain, A., Askell, A., Dsouza, A., Slone, A., Rahane, A., Iyer, A. S., Andreassen, A., Madotto, A., Santilli, A., Stuhlmüller, A., Dai, A. M., La, A., Lampinen, A. K., Zou, A., Jiang, A., Chen, A., Vuong, A., Gupta, A., Gottardi, A., Norelli, A., Venkatesh, A., Gholamidavoodi, A., Tabassum, A., Menezes, A., Kirubarajan, A., Mullokandov, A., Sabharwal, A., Herrick, A., Efrat, A., Erdem, A., Karakas, A., Roberts, B. R., Loe, B. S., Zoph, B., Bojanowski, B., Özyurt, B., Hedayatnia, B., Neyshabur, B., Inden, B., Stein, B., Ekmekci, B., Lin, B. Y., Howald, B., Orinion, B., Diao, C., Dour, C., Stinson, C., Argueta, C., Ramírez, C. F., Singh, C., Rathkopf, C., Meng, C., Baral, C., Wu, C., Callison-Burch, C., Waites, C., Voigt, C., Manning, C. D., Potts, C., Ramirez, C., Rivera, C. E., Siro, C., Raffel, C., Ashcraft, C., Garbacea, C., Sileo, D., Garrette, D., Hendrycks, D., Kilman, D., Roth, D., Freeman, D., Khashabi, D., Levy, D., González, D. M., Perszyk, D., Hernandez, D., Chen, D., Ippolito, D., Gilboa, D., Dohan, D., Drakard, D., Jurgens, D., Datta, D., Ganguli, D., Emelin, D., Kleyko, D., Yuret, D., Chen, D., Tam, D., Hupkes, D., Misra, D., Buzan, D., Mollo, D. C., Yang, D., Lee, D., Schrader, D., Shutova, E., Cubuk, E. D., Segal, E., Hagerman, E., Barnes, E., Donoway, E., Pavlick, E., Rodolà, E., Lam, E., Chu, E., Tang, E., Erdem, E., Chang, E., Chi, E. A., Dyer, E., Jerzak, E. J., Kim, E., Manyasi, E. E., Zheltonozhskii, E., Xia, F., Siar, F., Martínez-Plumed, F., Happé, F., Chollet, F., Rong, F., Mishra, G., Winata, G. I., de Melo, G., Kruszewski, G., Parascandolo, G., Mariani, G., Wang, G., Jaimovitch-López, G., Betz, G., Gur-Ari, G., Galijasevic, H., Kim, H., Rashkin, H., Hajishirzi, H., Mehta, H., Bogar, H., Shevlin, H., Schütze, H., Yakura, H., Zhang, H., Wong, H. M., Ng, I., Noble, I., Jumelet, J., Geissinger, J., Kernion, J., Hilton, J., Lee, J., Fisac, J. F., Simon, J. B., Koppel, J., Zheng, J., Zou, J., Kocon, J., Thompson, J., Wingfield, J., Kaplan, J., Radom, J., Sohl-Dickstein, J., Phang, J., Wei, J., Yosinski, J., Novikova, J., Bosscher, J., Marsh, J., Kim, J., Taal, J., Engel, J. H., Alabi, J., Xu, J., Song, J., Tang, J., Waweru, J., Burden, J., Miller, J., Balis, J. U., Batchelder, J., Berant, J., Frohberg, J., Rozen, J., Hernández-Orallo, J., Boudeman, J., Guerr, J., Jones, J., Tenenbaum, J. B., Rule, J. S., Chua, J., Kanclerz, K., Livescu, K., Krauth, K., Gopalakrishnan, K., Ignatyeva, K., Markert, K., Dhole, K. D., Gimpel, K., Omondi, K., Mathewson, K., Chiafullo, K., Shkaruta, K.,

Shridhar, K., McDonell, K., Richardson, K., Reynolds, L., Gao, L., Zhang, L., Dugan, L., Qin, L., Ochando, L. C., Morency, L., Moschella, L., Lam, L., Noble, L., Schmidt, L., He, L., Colón, L. O., Metz, L., Senel, L. K., Bosma, M., Sap, M., ter Hoeve, M., Farooqi, M., Faruqui, M., Mazeika, M., Baturan, M., Marelli, M., Maru, M., Ramírez-Quintana, M. J., Tolkiehn, M., Giulianelli, M., Lewis, M., Potthast, M., Leavitt, M. L., Hagen, M., Schubert, M., Baitemirova, M., Arnaud, M., McElrath, M., Yee, M. A., Cohen, M., Gu, M., Ivanitskiy, M. I., Starritt, M., Strube, M., Swedrowski, M., Bevilacqua, M., Yasunaga, M., Kale, M., Cain, M., Xu, M., Suzgun, M., Walker, M., Tiwari, M., Bansal, M., Aminnaseri, M., Geva, M., Gheini, M., T., M. V., Peng, N., Chi, N. A., Lee, N., Krakover, N. G., Cameron, N., Roberts, N., Doiron, N., Martinez, N., Nangia, N., Deckers, N., Muennighoff, N., Keskar, N. S., Iyer, N., Constant, N., Fiedel, N., Wen, N., Zhang, O., Agha, O., Elbaghdadi, O., Levy, O., Evans, O., Casares, P. A. M., Doshi, P., Fung, P., Liang, P. P., Vicol, P., Alipoormolabashi, P., Liao, P., Liang, P., Chang, P., Eckersley, P., Htut, P. M., Hwang, P., Milkowski, P., Patil, P., Pezeshkpour, P., Oli, P., Mei, Q., Lyu, Q., Chen, Q., Banjade, R., Rudolph, R. E., Gabriel, R., Habacker, R., Risco, R., Millière, R., Garg, R., Barnes, R., Saurous, R. A., Arakawa, R., Raymaekers, R., Frank, R., Sikand, R., Novak, R., Sitelew, R., LeBras, R., Liu, R., Jacobs, R., Zhang, R., Salakhutdinov, R., Chi, R., Lee, R., Stovall, R., Teehan, R., Yang, R., Singh, S., Mohammad, S. M., Anand, S., Dillavou, S., Shleifer, S., Wiseman, S., Gruetter, S., Bowman, S. R., Schoenholz, S. S., Han, S., Kwatra, S., Rous, S. A., Ghazarian, S., Ghosh, S., Casey, S., Bischoff, S., Gehrmann, S., Schuster, S., Sadeghi, S., Hamdan, S., Zhou, S., Srivastava, S., Shi, S., Singh, S., Asaadi, S., Gu, S. S., Pachchigar, S., Toshniwal, S., Upadhyay, S., Debnath, S. S., Shakeri, S., Thormeyer, S., Melzi, S., Reddy, S., Makini, S. P., Lee, S., Torene, S., Hatwar, S., Dehaene, S., Divic, S., Ermon, S., Biderman, S., Lin, S., Prasad, S., Piantadosi, S. T., Shieber, S. M., Misherghi, S., Kiritchenko, S., Mishra, S., Linzen, T., Schuster, T., Li, T., Yu, T., Ali, T., Hashimoto, T., Wu, T., Desbordes, T., Rothschild, T., Phan, T., Wang, T., Nkinyili, T., Schick, T., Kornev, T., Tunduny, T., Gerstenberg, T., Chang, T., Neeraj, T., Khot, T., Shultz, T., Shaham, U., Misra, V., Demberg, V., Nyamai, V., Raunak, V., Ramasesh, V. V., Prabhu, V. U., Padmakumar, V., Srikumar, V., Fedus, W., Saunders, W., Zhang, W., Vossen, W., Ren, X., Tong, X., Zhao, X., Wu, X., Shen, X., Yaghoobzadeh, Y., Lakretz, Y., Song, Y., Bahri, Y., Choi, Y., Yang, Y., Hao, Y., Chen, Y., Belinkov, Y., Hou, Y., Hou, Y., Bai, Y., Seid, Z., Zhao, Z., Wang, Z., Wang, Z. J., Wang, Z., and Wu, Z. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Trans. Mach. Learn. Res.*, 2023, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472.