arXiv:2606.19025v2 [cs.LG] 20 Jun 2026

FoMoE: Breaking the Full-Replica Barrier with a Federation of MoEs

Lorenzo Sani[1,2], Zeyu Cao[1], Meghdad Kurmanji[1], Alex Iacob[1,2], Andrej Jovanović[1,2], Yan Gao[1,2], Wanru Zhao[1], Nicholas D. Lane[1,2]

[1]Cambridge ML Systems Lab (CaMLSys), University of Cambridge [2]Flower Labs

Abstract. Pre-training Large Language Models (LLMs) typically demands large-scale infrastructure with tightly coupled hardware accelerators. Mixture-of-Experts (MoEs) architectures partially decouple model capacity from per-token compute. This efficiency alone does not make MoE training feasible over ordinary Internet links or loosely connected commodity hardware since active expert routing still assumes high-speed datacenter fabrics. Low-communication methods such as DiLoCo and Photon reduce synchronization frequency across distributed sites, mitigating bandwidth constraints, yet still require *full model replicas* at every site. This creates a mismatch: modern MoEs have sparse data paths, but their distributed training infrastructure remains communication-dense and memory-inefficient, limiting attempts to pool geographically distributed compute. In this work, we introduce FoMoE, a system that breaks the full-replica paradigm by partitioning expert layers across workers and skipping non-resident experts during local training. We demonstrate that FoMoE: (I) reduces communication costs by up to 1.42× over efficient baselines and 45.44× over Distributed Data Parallelism (DDP) via partial expert replication in controlled regimes; (II) achieves empirical throughput speedups of up to 1.4× through the "skip-token" mechanism; and (III) shows stable routing in the trained regimes and projects the communication/memory benefits to 100B-scale configurations through system modeling.

Correspondence: Lorenzo Sani <ls985@cam.ac.uk>
Keywords: machine learning systems, mixture of experts, federated learning, large language models

1 Introduction

State-of-the-art Large Language Models (LLMs) continue to grow in scale, driven by the relationship between parameter count and model quality (Kaplan et al., 2020; Hoffmann et al., 2022). Training frontier *dense* Transformer models has therefore become increasingly expensive, pushing compute, memory, and high-bandwidth interconnect limits in specialized training clusters (Hoffmann et al., 2022; Rajbhandari et al., 2020; Athlur et al., 2022; Palak et al., 2024). To address this computational bottleneck, the field has increasingly adopted the *Mixture-of-Experts* (MoEs) architecture (Shazeer et al., 2017; Fedus et al., 2022). By activating only a *sparse* subset of parameters per token, models like DeepSeek-V2/V3 (DeepSeek-AI et al., 2024; DeepSeek-AI, 2024) partially decouple computational cost from model capacity. This modularity reduces per-token compute, but it does not remove the need for fast links between hardware accelerators: conventional MoE training dispatches token activations to active experts during the forward and backward passes, so those experts are typically co-located in large data centers with high-bandwidth interconnects. As a result, MoE pre-training remains poorly matched to ordinary Internet links, volunteer or consumer-device pools (Ryabinin & Gusev, 2020; Ryabinin et al., 2023), and other settings that could pool accelerators across sites rather than within a single tightly coupled cluster.

To aggregate more compute, frontier labs have shown interest in training across geographically distributed, weakly connected sites, effectively imitating a single supercomputer (Comanici et al., 2025; Guthrie, 2025; Sevilla & Troynikov, 2025). For example, Google's Gemini 2.5 was trained using connected data centers (Comanici et al., 2025). However, the current understanding of this training paradigm remains in its infancy. In search of more efficiency in such cross-datacenter training, algorithms and systems like DiLoCo or Photon have emerged that reduce bandwidth overhead by limiting the synchronization *frequency* during training (Douillard et al., 2023; Sani et al., 2025). This makes WAN-connected training less bandwidth-hungry, but it leaves a second bottleneck untouched: how much model state must move per round and how much model memory each site must hold.

However, a critical inefficiency remains: these methods typically treat the model as a **dense, monolithic replica** and do not reduce **payload size**. This approach forces every participating site to store the entire model state and exchange full parameter payloads, which often amount to hundreds of gigabytes, at every synchronization step. Consequently, cross-datacenter training faces two prohibitive barriers: (1)

Preprint.

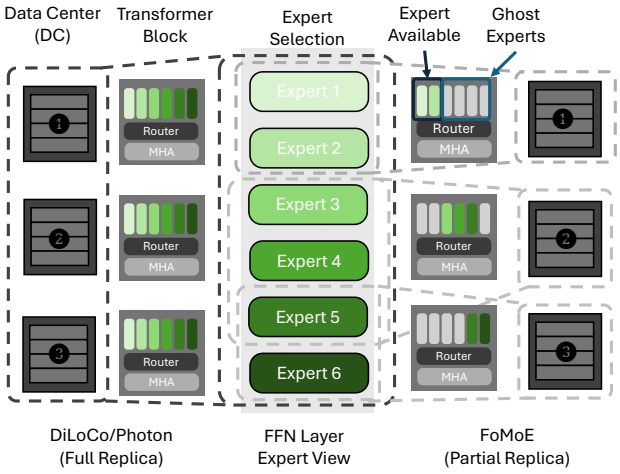

Figure 1: FoMoE breaking the "full-replica" barrier for WAN-connected MoE training. Methods such as DiLoCo and Photon maintain a full MoE replica at each site and communicate this object at round boundaries. FoMoE instead leverages MoE sparsity so each worker trains and communicates only the experts assigned to it. This design targets cross-site WAN settings where activation dispatch is too expensive.

Bandwidth: full-payload transfers create massive latency spikes on WAN links, rendering training extremely challenging for truly large models, and (2) **Memory**: the maximum model size is capped by the storage capacity of the most memory-constrained site. This creates a mismatch: while modern MoE models possess **sparse data paths**, their distributed training infrastructure remains **communication-dense** and **memory-inefficient**. We argue that leveraging **communication sparsity** complements the computational efficiency of architectures like DeepSeek, enabling the pooling of **large-scale distributed resources** and further closing the gap between open and proprietary models.

In this work, we present **FoMoE**, a cross-site training system that breaks the **"full-replica" barrier** by co-designing the training strategy with the model's structural sparsity. Rather than maintaining identical replicas at every site, FoMoE partitions each expert layer and assigns subsets of expert modules to different data centers. As a result, FoMoE recovers the bandwidth-efficiency properties of previous methods (Douillard et al., 2023; Sani et al., 2025) by reducing synchronization frequency, but also reduces the *payload size* sent during each synchronization. Since a worker does not fetch missing experts over the WAN during local training, FoMoE also requires a path for tokens routed to non-resident experts; our skip-token mechanism bypasses those expert paths rather than dispatching activations across sites.

Relative to data-center approaches such as expert parallelism (EP), FoMoE moves a different object across the network. EP shards experts within a high-bandwidth cluster and sends token activations to remote experts during every forward and backward pass (Rajbhandari et al., 2022; Hwang et al., 2023; Gale et al., 2023; DeepSeek-AI et al., 2024; DeepSeek-AI, 2024). FoMoE assumes workers never exchange token activations over the WAN; they train with local experts and synchronize selected model state only at round boundaries. EP can still be used within a site, while FoMoE targets the cross-site payload and memory bottleneck left by full-replica low-communication training.

Our evaluation studies a controlled setting with homogeneous workers, i.i.d. data partitions, and controlled WAN bandwidth and latency. This isolates the effect of expert partitioning and payload reduction from orthogonal concerns such as asymmetric links, jitter, stragglers, and stale synchronization.

FoMoE enables communication-efficient training of MoE models across multiple data centers without sacrificing machine learning performance. Our contributions are:

1. **The FoMoE System.** We introduce a framework for cross-site MoE training that supports partial expert replication. We define the design space of *partitioning* (balancing memory vs. bandwidth) and *placement* (mapping experts to workers), exploring both fixed and random strategies to trade off expert specialization against generalization.
2. **Partitioned FFN Optimization.** We demonstrate that partial expert replication linearly reduces communication volume, outperforming communication-efficient baselines by up to $1.42\times$ and DDP by up to $45.44\times$ in communication costs while maintaining perplexity in our controlled static-link evaluation. This shifts the empirical non-dominated trade-off frontier, enabling ML performance at lower system costs in the studied regimes.
3. **"Skip-Token" Throughput Gains.** We introduce the skip-token mechanism required by partial expert placement: when a token routes to a non-local expert (a "ghost expert"), the worker bypasses that expert path instead of sending the token over the WAN. We show that this improves throughput linearly in theory and achieves empirical speedups up to **1.4**$\times$.
4. **Scaling and Stability.** We separate the evidence by scale: trained proxy configurations provide empirical routing and convergence evidence, while the larger configurations in Tab. 1 are analyzed through communication, memory, FLOP, and wall-clock system models. Our results show that FoMoE maintains high routing entropy and avoids expert collapse in the trained regimes, and that the same partial-replication mechanism projects favorable payload and memory scaling under the modeled large-scale configurations.

FoMoE bridges the gap between high-bandwidth, tightly-coupled MoE training and low-bandwidth, loosely-coupled

distributed paradigms. By reducing the payload size per synchronization, it complements existing low-frequency methods (Douillard et al., 2023; Sani et al., 2025), paving the way for training next-generation LLMs on globally sourced compute infrastructure.

2 BACKGROUND

Large-scale model training. Distributing computation is essential for large-scale model training. Standard data parallelism replicates the full model and synchronizes gradients per step (Li et al., 2020a), often saturating interconnects (Narayanan et al., 2019). For memory constraints, model parallelism partitions the network: pipeline parallelism divides layers into sequential stages (Huang et al., 2019; Narayanan et al., 2019), while tensor parallelism splits layers across devices (Shoeybi et al., 2019). The *Zero Redundancy Optimizer* (ZeRO), a form of data parallelism, eliminates memory redundancy by sharding parameters and optimizer states (Rajbhandari et al., 2020). While these methods assume high-bandwidth fabrics, *Varuna* adapts pipeline parallelism to commodity cloud VMs with limited WAN bandwidth (Athlur et al., 2022).

MoE for scalable LLMs. Mixture-of-Experts (MoE) architectures decouple parameter count from compute cost by activating sparse expert MLPs per token (Shazeer et al., 2017). Implementations like GShard (Top-2) and *Switch Transformers* (Top-1) achieve trillion-parameter capacities efficiently (Lepikhin et al., 2021; Fedus et al., 2022). Recent advances reinforce this efficiency (GLaM (Du et al., 2022)) and optimize architectural design (DeepSeek-V2/V3 (DeepSeek-AI et al., 2024; DeepSeek-AI, 2024)). Theoretical works unify MoE perspectives via σ-MoE (Csordás et al., 2023) and establish fine-grained scaling laws (Ludziejewski et al., 2024). Alternative routing strategies, including BASE layers (Lewis et al., 2021), Expert-Choice (Zhou et al., 2022), and Hash Layers (Roller et al., 2021), address load balancing, while V-MoE addresses vision (Riquelme et al., 2021). Expert pruning and skipping methods remove or bypass low-utility experts to improve deployment efficiency (Lu et al., 2024); FoMoE uses different skip-token semantics during training, in which missing remote experts are deliberately not executed across the WAN.

Systems support for MoE and expert parallelism. Commodity MoE training leverages specialized runtimes (e.g., DeepSpeed-MoE, Tutel, MegaBlocks, FasterMoE, SmartMoE) to optimize All-to-All communication, memory management, and kernel execution (Rajbhandari et al., 2022; Hwang et al., 2023; Gale et al., 2023; He et al., 2022; Zhai et al., 2023). These systems typically rely on *expert parallelism* (EP): experts are sharded across devices, while token activations are dispatched to the owning experts over a fast intra-DC fabric. EP can reduce per-device memory and enable large MoE within a cluster, but it does not remove the need for high-bandwidth activation exchange during each forward and backward pass. DeepSeek-style node-limited routing narrows this dispatch scope within a cluster (DeepSeek-AI, 2024); FoMoE instead treats the WAN as a hard boundary, avoids cross-site activation dispatch entirely, and communicates only selected model state at synchronization boundaries. Thus, our design is orthogonal to intra-DC EP and targets the payload and memory overheads that remain in full-replica cross-site training.

Low-bandwidth distributed training. Training across weakly connected sites builds upon distributed optimization and local-update foundations (McMahan et al., 2017; Reddi et al., 2021; Li et al., 2020b; Stich, 2019). Standard data parallelism synchronizes a full model every step; DiLoCo (Douillard et al., 2023) and Photon (Sani et al., 2025) instead perform multiple local steps before each synchronization, reducing the frequency-dependent bandwidth cost by roughly the local-step interval. Theoretically, (Cheng & Glasgow, 2025) proves convergence for adaptive optimizers in this regime, provided optimizer states are synchronized. Further communication reductions are achieved by decoupling model components (DEPT (Iacob et al., 2025a)), desynchronizing optimizer states (DES-LOC (Iacob et al., 2025b)), co-designing modular paths (DiPaCo (Douillard et al., 2024)), or combining local updates with compression and sparse peer communication (SparseLoCo, DiLoCoX, GASLoC) (Sarfi et al., 2025; Qi et al., 2025; Cagnasso et al., 2026). Related systems work studies geo-distributed and weakly connected training through workload-aware WAN scheduling, pipeline parallelism, and volunteer-device MoE, including Atlas/BubbleTea, SWARM Parallelism, and Learning@home/DMoE (Palak et al., 2024; Ryabinin et al., 2023; Ryabinin & Gusev, 2020). However, a direct mapping of these methods to MoE pre-training still materializes the full dense backbone and all experts at each site, or targets different axes such as pipeline scheduling, peer topology, or compression. FoMoE addresses the complementary dimension of *which* MoE experts must be present and synchronized at each site, thereby reducing per-round payload size and steady-state memory, rather than only the synchronization frequency.

3 FOMOE: THE FEDERATED MOE SYSTEM

Although current low-communication pre-training methods reduce the frequency of synchronization, and thus the cumulative communication cost, they still require transmitting full parameter payloads during updates (Douillard et al., 2023; Sani et al., 2025). The straightforward adoption of MoE architectures in this setting achieves a FLOP reduction

for each worker by leveraging the structural **sparsity** of the model – only $k \ll E$ of the total number of experts (E) are activated per token (Fedus et al., 2022; DeepSeek-AI et al., 2024; Ludziejewski et al., 2024; Shazeer et al., 2017). However, when applied directly to methods such as Photon or DiLoCo, the communication costs remain unchanged compared to dense models. By design, these methods require that the full parameter payload be communicated at every synchronization step, despite the inherent sparsity observed in these MoE architectures.

To alleviate this issue, we propose a federated MoE framework tailored to bandwidth-constrained wide-area network environments (WAN), leveraging the **sparsity** present in MoE architectures to reduce **both computational and communication costs** by **only training, and thus communicating, a subset of experts in each worker**.

We now present the FoMoE system and its design space, covering: (a) **model partitioning**, which determines the optimal degree of expert replication; (b) **expert placement**, which orchestrates the distribution of experts across workers; and (c) **machine learning optimization**, which ensures stability and performance parity with high-communication baselines. Fig. 1 shows a simplified system view of FoMoE.

3.1 Architecture and design

Our system architecture is optimized for communication-constrained federated environments. By breaking the "full-replica" barrier, we decouple the model size from synchronization costs. We detail the setting below and in Sec. 3.2.

Architecture and distributed setting. We target cross-datacenter training environments characterized by a distinct bandwidth hierarchy typical of real-world deployments: high intra-datacenter bandwidth B_{intra} and severely constrained cross-datacenter bandwidth B_{cross} (Palak et al., 2024). We explicitly focus on the regime where $B_{\text{intra}} \gg B_{\text{cross}}$; the utility of local training strategies decreases when bandwidths are comparable ($B_{\text{intra}} \approx B_{\text{cross}}$). Formally, we consider a set W of homogeneous data-parallel workers $M = |W|$, where each worker functions as a datacenter-class node with accelerators interconnected at B_{intra} (e.g., $\approx 200.00\,\text{Gb}ps$). Each worker samples i.i.d. data partitions and connects to peers through WAN links with effective bandwidth B_{cross} (e.g., $\approx 1.00\,\text{Gb}ps$). In this setting, standard computation-communication overlap offers minimal benefit as transfer times dominate the latency. To mitigate this, we perform K local optimization steps between fully synchronous rounds (Douillard et al., 2023; Sani et al., 2025). We employ a central coordinator to manage initialization, broadcasting, and aggregation, restricting our study to synchronous training. We leave asynchronous extensions to future work due to the complexity of managing stale updates (Douillard et al., 2025) and note that these methods are fully **compatible** and **orthogonal** to our work.

3.2 Model partitioning design

Although our system can flexibly accommodate arbitrary MoE architectures (DeepSeek-AI et al., 2024; Team et al., 2025), we fix the architecture design in this work. Specifically, our system trains decoder-only Transformer models (Vaswani et al., 2017) in which we replace the standard feed-forward network (FFN) with σ-MoE-FFN modules (Csordás et al., 2023; Ludziejewski et al., 2024) whose expert and router parameters are, *potentially*, only **partially** replicated across **subsets** of workers (we provide full details of the architecture design in Sec. B). Our architectural choice also supports capturing shared knowledge among experts (DeepSeek-AI et al., 2024). Conceptually, FoMoE categorizes the model parameters θ into the following.

1. **Dense components** (θ_{dense}): *Fully replicated* parameters including attention weights, layer norms, shared experts, and input/output embeddings (tied by default).
2. **Sparse components** (θ_{sparse}): *Potentially partitioned* parameters including expert weights and routing networks. We detail the worker partitioning and assignment of sparse components in Secs. 3.3.1 and 3.3.2.

To illustrate this breakdown, we consider a typical Transformer configuration. The token embedding matrix comprises $d_{\text{model}} \times |\mathcal{V}|$ parameters where $\mathcal{V}$ is the vocabulary. Each attention layer contains four weight matrices totaling $4d^2_{\text{model}}$ parameters (assuming standard head dimensions). In contrast, an MoE layer adds a routing network with $d_{\text{model}} \times N_e$ parameters (where N_e is the total expert count) and the experts' FFNs. For a dense expansion ratio e, we allocate a total FFN hidden dimension of $e \cdot d_{\text{model}}$ per layer, divided among N_e experts (each with hidden size $d_{\text{ff,expert}} = \frac{e\, d_{\text{model}}}{N_e}$). Consequently, the combined expert parameter count is $e \times d^2_{\text{model}}$, comparable to a dense FFN.

As model scale increases, the ratio $\theta_{\text{sparse}}/\theta$ shifts significantly. While scaling depth primarily adds dense parameters, scaling width (i.e., increasing d_{model}) primarily adds experts. Adopting a standard width-to-depth ratio of $d_{\text{model}} : L \approx 32$ (Dey et al., 2025), θ_{sparse} grows from $\approx 20\%$ of total parameters at the 150M scale ($d_{\text{model}} = 512$) to $> 60\%$ at the 26B scale ($d_{\text{model}} = 4096$), as shown in Fig. 2. This relationship shows that the efficient synchronization of θ_{sparse} is the primary lever for enhancing communication efficiency as model size increases. This scaling design is well established in the literature (Csordás et al., 2023; Malasnicki et al., 2025; Ludziejewski et al., 2024) and facilitates the study of the broader design space of expert placement options as the number of expert modules increases.

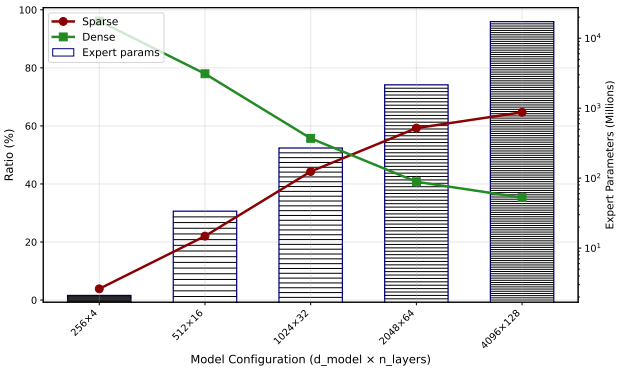

Figure 2: Parameter distribution trends. Scaling width (d_{model}) dramatically increases the proportion of sparse parameters (θ_{sparse}), motivating the focus of our strategy.

3.3 Partitioning and placement strategies

FoMoE enables the distribution of sparse parameters controlled via two coupled design decisions: **partitioning**, which determines expert replication and balances memory and bandwidth; and **placement**, which maps experts to workers. These mechanisms jointly control communication costs and effective model capacity. We define participation in Sec. 3.3.1 and the placement algorithms in Sec. 3.3.2.

3.3.1 Experts partitioning and scalability

To analyze the impact of partitioning at scale, consider a layer ℓ with $N_{e,\ell}$ experts distributed across M workers. We introduce the *overlapping factor* $\mathcal{O}_e \in \{1, \dots, M\}$, defined as the number of distinct workers hosting any specific expert. This parameter controls the design spectrum from completely disjoint partitions ($\mathcal{O}_e = 1$) to full replication ($\mathcal{O}_e = M$). Consequently, each worker hosts a reduced local subset of $N_{\text{le},\ell} = \frac{\mathcal{O}_e}{M} N_{e,\ell}$ experts. During training, local routing is bounded by $\min(k, N_{\text{le},\ell})$ experts (where $k = 2$ by default). Partitioning ($\mathcal{O}_e < M$) yields threefold advantages over full-replicas methods:

1. **Communication:** the payload scales with the fraction of globally shared parameters $\mathcal{O}_e/M$, saving the remaining $1 - \frac{\mathcal{O}_e}{M}$ from cross-site transfer.
2. **Compute:** computation can be reduced in two ways. First, the router scores only $N_{\text{le},\ell}$ experts, reducing router FLOPs by $\approx \mathcal{O}_e/M$. Second, if $N_{\text{le},\ell} < k$, the system activates fewer experts per token, thus reducing FFN FLOPs. Otherwise, the per-token compute matches standard MoE ($N_{\text{le},\ell} \geq k$).
3. **Memory:** the per-worker memory footprint for expert weights decreases by a factor of $M/\mathcal{O}_e$, enabling training where naïve full replication would be too expensive, due to bandwidth and memory constraints.

Algorithm 1 Expert Placement & Migration Strategy

Require: M (workers), N_e (total experts), O_e (overlap)
Require: T (shuffle period), t (round), $\mathcal{S}_m^{(t-1)}$ (prev. set)
Require: Mode (Fixed or Random)

$\mathcal{S}_m^{(t)} \leftarrow \mathcal{S}_m^{(t-1)}$ // Default: retain set
Phase 1: Placement Logic
if $t = 0$ **or** (Mode = Rand **and** $t \pmod T = 0$) **then**
  $\mathcal{P} \leftarrow \texttt{GenPlacement}(N_e, O_e, M)$
  $\mathcal{S}_m^{(t)} \leftarrow \texttt{GetLocalSubset}(\mathcal{P}, m)$
  ▷ Assigns $N_{le} = \frac{O_e}{M} N_e$ experts to worker m
end if
Phase 2: Migration & State Mgmt.
$\Delta_{\text{in}} \leftarrow \mathcal{S}_m^{(t)} \setminus \mathcal{S}_m^{(t-1)}$ // Identify new experts
if $\Delta_{\text{in}} \neq \emptyset$ **then**
  Broadcast(Δ_{in})
  ▷ Fetch missing weights (prob. $p_{miss} \approx 1 - \frac{O_e}{M}$)
else
  ▷ Maintain local momentum (no migration)
end if
return $\mathcal{S}_m^{(t)}$

3.3.2 Expert placement modes

Once the expert partitioning configuration (defined by $\mathcal{O}_e$ and $N_{e,\ell}$) is set, FoMoE enables flexible expert-to-worker assignment. We investigate two modes, detailed in Alg. 1:

Fixed placement. Upon initialization of the M workers, the central server deterministically maps each expert in layer ℓ to $\mathcal{O}_e$ workers. Consequently, each worker is assigned a static set of $N_{\text{le},\ell}$ experts. These assignments remain fixed throughout training, with each worker responsible solely for updating its local experts. In the limiting case where $\mathcal{O}_e = 1$, this scheme reduces to disjoint, fully local experts; workers collaborate exclusively on the dense parameters θ_{dense}. This effectively specializes each site to a specific subset of expert modules, analogous to decoupled training methods such as DEPT (Iacob et al., 2025a). While DeepSeek's Expert Parallelism (EP) treats the cluster as a unified memory pool by dynamically sharding experts to maximize capacity (DeepSeek-AI et al., 2024), FoMoE's fixed placement treats the WAN as a hard constraint, strictly pinning experts to specific workers to enforce data locality and eliminate weight synchronization during local steps.

Random placement. To improve expert generalization, this mode reassigns experts to workers at a cadence of T synchronization rounds. The server randomly redistributes experts subject to the constraint that every expert must reside on exactly $\mathcal{O}_e$ workers, and every worker must host $N_{\text{le},\ell}$ **distinct** experts. Under this scheme, the specific expert set resident on a worker evolves over time. A small reshuffling period ensures experts are trained across worker data partitions, potentially enhancing generalization. However, this introduces trade-offs: it incurs network overhead for migrat-

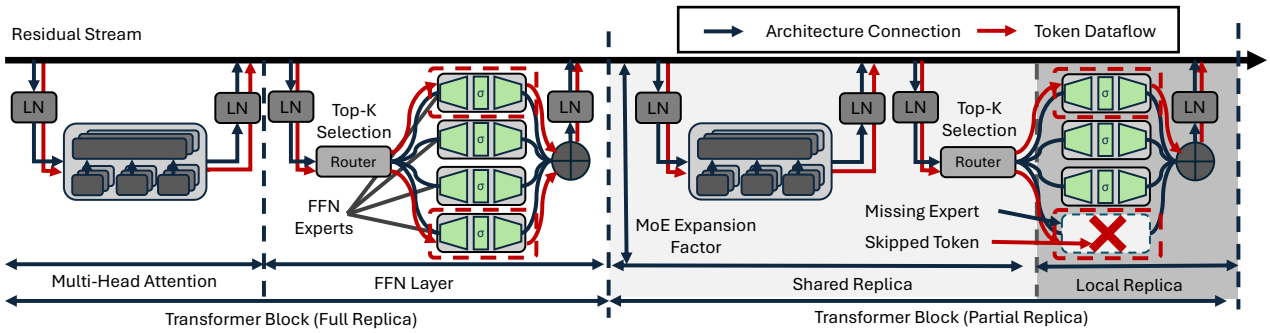

Figure 3: Comparison between a transformer MoE block as a full versus partial replica. In the partial-replica form, the same experts can be selected per token as in the full-replica case, but missing experts are ignored by the local computational graph, yielding savings in FLOP, communication, and memory.

ing model states and may affect local optimizer momentum (which we discuss in Sec. 3.4). In practice, the reshuffling period is tuned to balance data mixing against stability and costs. Eq. (8) demonstrates that while reshuffling can, in the worst case, necessitate migrating every expert, this overhead decreases linearly as $\mathcal{O}_e \to M$ and as the reshuffling interval increases relative to the communication period.

System context and routing logic. We emphasize that, in our federated setting, token routing is **strictly local**: tokens processed at a site are dispatched exclusively to experts hosted at that site, with no inter-site exchange of activations. This design maximizes locality, avoiding high-latency WAN communication during the forward and backward passes. This distinguishes FoMoE from systems like DeepSpeed-MoE, Tutel, MegaBlocks, FasterMoE, SmartMoE, which optimize execution *within* high-bandwidth clusters via all-to-all communication (Rajbhandari et al., 2022; Hwang et al., 2023; He et al., 2022; Zhai et al., 2023). Our placement strategies thus operate at a distinct granularity from typical intra-layer load balancing. While we assume standard mechanisms (e.g., BASE layers (Lewis et al., 2021), Expert-Choice (Zhou et al., 2022), or capacity scaling (Fedus et al., 2022)) are employed to balance loads among *available* local experts, our work addresses the WAN-constrained challenge of determining *which* experts are available at each site.

Placement trade-off. Random expert placement may outperform fixed placement when data or hardware heterogeneity makes additional expert mixing valuable. First, while a fixed topology restricts a worker's router to a static support set of N_{le} experts, limiting the routing combinations to $\binom{N_{\text{le}}}{k}$, random placement reshuffles assignments every synchronization interval. Over R rounds, a worker explores up to $\min(R, \binom{N_e}{N_{\text{le}}})$ distinct support sets. Second, randomization can act as a regularizer when the i.i.d. assumption fails, since it decouples experts from specific data shards and exposes them to a broader range of data distributions. In homogeneous i.i.d. settings evaluated in Sec. 6.2, however, the migration and optimizer-state costs dominate this hypothesized mixing benefit.

3.3.3 *"Ghost Experts" and "Skip-Token"*

For most of our work, we partition each router, along with the experts, so that individual tokens are not routed to experts that are unavailable to the local worker. However, we investigate the opportunity to go beyond such biased routing and to further leverage the sparsity and partitioning of experts, reducing computation. We propose an alternative partitioning, as seen in Fig. 3, that does not partition the routing matrices, allowing each router to assign individual tokens to experts who are not locally available to the workers. The cost of not partitioning the routing matrices and communicating them in full, as if they are part of the dense components θ_{dense}, is negligible compared to that of the expert modules. We refer to those experts that can receive token assignments but are not available as "ghost experts". The token assignments to the "ghost experts" provide an opportunity to further reduce FLOPs, since we cannot follow the computational path of these tokens due to missing expert weights. When the router's assignments are perfectly balanced, the executed expert FLOPs scale with the ratio of local experts to total experts, so the skipped computation grows with the complementary ghost-assignment fraction. This should be interpreted as an approximation whose quality impact must be measured, not as an assumption that the router has a large top-1/top-2 score margin or that experts are non-redundant. In the present experiments, we do not use router-score margins to justify skipping; instead, we evaluate the downstream effects directly via validation perplexity and routing stability in Fig. 7. In contrast to DeepSeek's dispatch mechanism, which relies on high-bandwidth interconnects to route tokens to remote experts for full computation (DeepSeek-AI et al., 2024), FoMoE uses its "ghost expert" strategy to skip computation for remote experts, trad-

ing a potential reduction in active expert computation for the complete elimination of cross-node activation transfer.

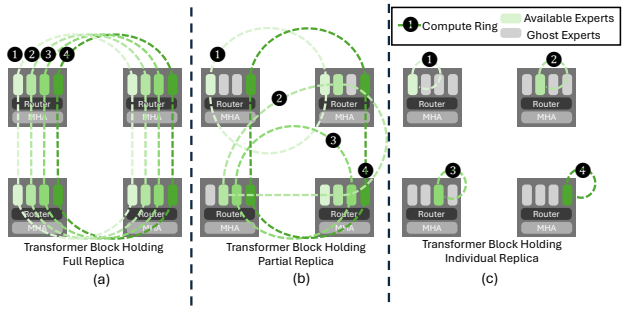

Figure 4: Expert placement and ring synchronization. In a full-replica regime, each available expert incurs its own full ring across workers. As $O_e \to 1$, the communication cost $R(O_e)$ decreases because each expert ring spans fewer workers; at $O_e = 1$, each ring is localized to the worker that owns the expert.

3.4 Synchronization and model optimization

FoMoE training (as we show in Alg. 2) proceeds in rounds (McMahan et al., 2017): the server broadcasts parameters, workers perform K local steps (this implies a synchronization frequency reduction of $1/K$), and updates are aggregated. We make the following choices regarding the training process.

Round structure and collectives. We utilize bandwidth-optimal Ring-AllReduce for aggregation. Averaging a tensor of size S across M peers entails transmitting $\frac{2(M-1)}{M}S$ bytes per peer (Rabenseifner, 2004; Patarasuk & Yuan, 2009). While we employ a flat reduction, the framework supports hierarchical strategies for heterogeneous networks (Kairouz et al., 2021). Furthermore, asynchronous techniques like Streaming DiLoCo (Douillard et al., 2025) can overlap communication with computation; while we have not implemented such overlapping in FoMoE, it can be readily incorporated into our performance modeling (e.g., as a compression factor).

Outer optimization. We combine infrequent synchronization ($K \gg 1$) with adaptive optimization. Workers execute AdamW (Loshchilov & Hutter, 2019) locally on their parameter subsets, while the server applies momentum (FedOPT) to the aggregated model to maintain convergence (Reddi et al., 2021; Khaled et al., 2025). Crucially, the local optimizer states are preserved rather than reset. For the global θ_{dense}, states are tracked by all workers; for θ_{sparse}, workers maintain history only for their own experts.

A challenge arises in the *random placement* regime, where expert migration separates weights from their optimizer history. To avoid the overhead of transferring states when experts are reassigned, we reinitialize the optimizers for reassigned experts with a learning-rate warmup applied only to the reassigned weights. In contrast, *fixed placement* continuously retains the state. Empirically, we observe that preserving local momentum prevents the divergence observed in stateless approaches. To ensure scalability without retuning, we transfer the following hyperparameters: global learning rate schedules (leveraging μP/CompleteP (Yang & Hu, 2021; Dey et al., 2025; Malasnicki et al., 2025) transfer), AdamW settings ($\beta_1, \beta_2, \epsilon$), warmup duration, and router temperature. See Sec. C for more details.

Bandwidth regimes and trends. Unlike high-speed intra-datacenter settings ($100.00\,\text{Gb}ps$ – $400.00\,\text{Gb}ps$) where communication overlaps computation (Rajbhandari et al., 2022), WAN constraints ($1.00\,\text{Gb}ps$ – $20.00\,\text{Gb}ps$) necessitate minimizing volume ($\mathcal{O}_e \ll M$) and frequency ($K \gg 1$). Although increasing M or K exacerbates model drift, the server momentum effectively stabilizes training (Stich, 2019). Recent works confirm that with tuned hyperparameters, infrequent synchronization maintains perplexity parity (Douillard et al., 2023; Sani et al., 2025) while reducing communication overhead (Iacob et al., 2025b). Gradient compression (Alistarh et al., 2017) remains orthogonal.

4 FoMoE Scalability & Resource Consumption

Building on the system design in Sec. 3, we define a quantitative cost model to optimize FoMoE configurations and hyperparameters under resource constraints. Given M weakly connected sites with effective WAN bandwidth B_{eff} and fixed hardware, we determine the expert overlap $\mathcal{O}_e$, synchronization interval K, and expert count N_e that maximize performance per wall-clock time. These models help determine how to set up FoMoE for the specific requirements of the underlying training infrastructure.

The model comprises three components: (1) computational cost based on FLOPs (Sec. 4.1); (2) communication cost based on data volume (Sec. 4.2); and (3) a wall-clock time model synthesizing these factors within the distributed environment (Sec. 4.3). This framework identifies the partitioning methodology of Sec. 3, synchronization interval K, expert overlap factor $\mathcal{O}_e$, and worker count M as the primary levers. While prior work (Douillard et al., 2023; Sani et al., 2025) examined general communication-computation trade-offs, they neglected MoE-specific partitioning. Our model explicitly captures the trade-offs of partial expert replication in bandwidth-constrained multi-datacenter settings, demonstrating how tuning $\mathcal{O}_e$ and K overcomes the "full-replica" barrier to efficient distributed training.

4.1 Compute and Memory Scaling

Training a dense Transformer with N parameters on D tokens requires approximately $6ND$ FLOPs (Kaplan et al., 2020), obeying power-law scaling between compute and model quality (Hoffmann et al., 2022). Sparse models MoE partially decouple the parameter count from the compute cost by activating only $k \ll N_e$ experts per layer. The effective parameter count for compute becomes $N_{\text{dense}} + \phi N_{\text{exp}}$; $\phi = k/N_e$ is the active fraction (Du et al., 2022) of the total expert parameters. Since $\phi \ll 1$, MoEs achieve a trillion-parameter scale with per-token cost of significantly smaller dense models (Riquelme et al., 2021; Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022).

In our federated topology, local compute costs depend on the available experts. If a worker holds $N_{\text{le},\ell} \geq k$ experts, the costs match that of full-replica training. However, if $N_{\text{le},\ell} < k$ (e.g., each worker has a single expert when $k = 2$), the worker activates only $\min(k, N_{\text{le},\ell})$ experts, reducing FFN FLOPs (Fedus et al., 2022). Notably, the gating network projects only to local experts, scaling router operations by $N_{\text{le},\ell}/N_{e,\ell} = \mathcal{O}_e/M$. Reducing expert overlap $\mathcal{O}_e$ lowers per-token compute, especially when forcing sparse activation (e.g., top-1) (Gale et al., 2023).

We estimate the per-token FLOPs F_{token} for a worker with overlap $\mathcal{O}_e$ and local expert count $N_{\text{le},\ell} = \frac{\mathcal{O}_e}{M} N_{e,\ell}$ as:

$$F_{\text{token}} \approx F_{\text{dense}} + \sum_{\ell=1}^{L} \left(\frac{\min(k, N_{\text{le},\ell})}{k} F_{\text{FFN},\ell} + \frac{N_{\text{le},\ell}}{N_{e,\ell}} F_{\text{gate},\ell} \right) \quad (1)$$

where F_{dense}, $F_{\text{FFN},\ell}$, and $F_{\text{gate},\ell}$ denote the per-token FLOPs of the dense components, unpartitioned expert FFNs, and full gating networks, respectively.

In the presence of "ghost experts" adopting the "skip-token" procedure, the formula requires applying a scaling factor that accounts for the possibility of skipping the FFN computation for a token. With a perfectly balanced routing network, such a scaling factor can be written as:

$$\sigma_{\text{st}} = \frac{N_{\text{le},\ell}}{N_{e,\ell}} \leq 1 \quad (2)$$

and, consequently, the total flops per token can be expressed, more generally, as:

$$F_{\text{token}} \approx F_{\text{dense}} + \sum_{\ell=1}^{L} \left(\sigma_{\text{st}} \frac{\min(k, N_{\text{le},\ell})}{k} F_{\text{FFN},\ell} + \frac{N_{\text{le},\ell}}{N_{e,\ell}} F_{\text{gate},\ell} \right) \quad (3)$$

We note that routing decisions are not always perfectly balanced; introducing a load-balancing loss ensures they converge to a balanced state. In the presence of a severe imbalance in the assignment, the scaling factor above should be replaced with the actual frequency of the token assignments to "ghost experts" against the total number of assignments to provide more precise cost estimation. In our experiments, we observe an overall balance, with the empirical scaling factor always being very close to its ideal counterpart. We recommend running a few warm-up iterations in full-replica mode to detect excessive imbalance before it becomes too problematic to maintain machine learning performance.

Memory Footprint. Large-scale training memory comprises *steady-state* allocations for parameters and optimizer states, and *dynamic* allocations for activations and temporary buffers (Rajbhandari et al., 2020; Micikevicius et al., 2018). A standard mixed-precision AdamW requires ≈ 18.00 B per parameter (2 for FP16 weights, 4 for FP32 master weights, 8 for moments, and 4 for gradients). Activation memory scales with batch size and sequence length, and activation checkpointing can trade additional compute for lower activation storage (Chen et al., 2016).

Our FoMoE architecture reduces both memory components. Partitioning experts across M sites with overlap $\mathcal{O}_e$ reduces the stored expert parameters per site to a fraction $\frac{\mathcal{O}_e}{M}$. The expected per-site steady-state memory is approximately $\frac{\mathcal{O}_e}{M} \sum_{\ell,e} s_{\ell,e} + S_{\text{dense}}$, where $s_{\ell,e}$ and S_{dense} denote the size of expert and dense parameters, respectively (Rajbhandari et al., 2022). Reducing $\mathcal{O}_e$ toward 1 yields nearly linearly proportional memory savings, particularly for models where sparse experts dominate the parameter count (e.g., $> 60\%$ in DeepSeek-V2 (DeepSeek-AI et al., 2024)). Furthermore, fewer local experts reduce the dimensionality of the gating network, providing additional steady-state savings.

Dynamic memory also decreases with lower $\mathcal{O}_e$: when a worker holds fewer experts than the global top-k (i.e., $N_{\text{le},\ell} < k$), it computes activations for only $\min(k, N_{\text{le},\ell})$ experts per token. A local top-1 execution in a global top-2 regime halves the stored FFN activation tensors: fewer local experts reduce activation memory and gradient buffers. Such savings are additive to standard memory-optimization techniques (Fedus et al., 2022; Rajbhandari et al., 2020).

4.2 Communication Scaling

We quantify the communication overhead by deriving the *bytes-per-token* (BPT) exchanged between sites. This metric directly links the synchronization interval K and the expert overlap $\mathcal{O}_e$ to WAN bandwidth requirements.

Setting and Aggregation. As detailed in Alg. 2, training proceeds in synchronous rounds where M sites perform K local steps before aggregating parameters (McMahan et al., 2017; Reddi et al., 2021; Li et al., 2020b). We partition the model into fully replicated dense parameters θ_{dense} and sparse expert weights $\{\theta_{\text{sparse}}\}$ distributed according to the overlap factor $\mathcal{O}_e$ (where $\mathcal{O}_e$ denotes the number of workers hosting a specific expert). Various strategies modify the synchronization payload and frequency: LocalSGD (Stich, 2019) and DiLoCo (Douillard et al., 2023) sync only model

parameters, keeping optimizer states local; `LocalAdamW` (Cheng & Glasgow, 2025) syncs both, roughly tripling traffic; and `DES-LOC` (Iacob et al., 2025b) decouples parameter and state synchronization schedules, enabling flexible trade-offs in parameter vs optimizer state synchronization. Outer-optimizer analyses further explain the roles of learning rates, momentum, and acceleration in local-update regimes (Khaled et al., 2025). We consider a subset of these strategies for our parameter-partitioned `MoE` regime.

Modeling Communication Costs. Synchronizing fully replicated components (θ_{dense} of size S_{dense} in bytes) via `Ring-AllReduce` (Rabenseifner, 2004; Patarasuk & Yuan, 2009) incurs a per-site cost of:

$$B_{\text{dense}}^{\text{per-site}}(M) = \frac{2(M-1)}{M} S_{\text{dense}} \tag{4}$$

For sparse components, communication depends on the overlap $\mathcal{O}_e$. Assuming a uniform assignment where each worker holds $N_{\text{le},\ell} = \frac{\mathcal{O}_e}{M} N_{e,\ell}$ experts, the per-site communication cost depend on two steps: (1) synchronization and aggregation of experts' weights which can be performed with `Ring-AllReduce`; and (2) the broadcast due to potential re-shuffling of the experts across workers (only occurs for the random placement strategy). During step (1), a worker will need to exchange data for each local expert with a cost weighted by a ring factor:

$$R(\mathcal{O}_e) = \frac{2(\mathcal{O}_e - 1)}{\mathcal{O}_e} \tag{5}$$

Notably, when $\mathcal{O}_e = 1$, this cost disappears since no synchronization and aggregation occur. During step (2), the communication cost is determined by the set difference between the experts assigned at round $t+1$ and those already present at round t. We model this as a probabilistic process.

Assuming a uniform random assignment, the probability that a specific expert e assigned to a worker in the new placement was already present in its local partition S_t is equal to the fraction of global experts held locally:

$$p_{\text{hit}} = \frac{N_{\text{le},\ell}}{N_{e,\ell}} = \frac{(\frac{\mathcal{O}_e}{M} N_{e,\ell})}{N_{e,\ell}} = \frac{\mathcal{O}_e}{M} \tag{6}$$

The probability that an assigned expert is missing and requires migration (a "broadcast" event) is $p_{\text{miss}} = 1 - \frac{\mathcal{O}_e}{M}$.

We define the weighting factor $P(\mathcal{O}_e, M)$ as this miss probability, scaled by the reshuffling frequency γ (where $\gamma = 1$ implies reshuffling every synchronization round, and $\gamma < 1$ implies amortized periodic reshuffling). As discussed in Sec. 3.3.2, we investigate two placement modes, fixed and random, which correspond to the setting $\gamma = 0$ and $\gamma = 1$, respectively. Substituting into Eq. (7):

$$P(\mathcal{O}_e, M, \gamma) = \gamma \left(1 - \frac{\mathcal{O}_e}{M}\right) \tag{7}$$

This derivation aligns with the boundary conditions discussed in Sec. 4.2:

- **Full Replica ($\mathcal{O}_e = M$):** The term becomes $1 - 1 = 0$. Since workers hold all the experts, no migration is ever required, and the synchronization term also accounts for the cost of weight updates.
- **Disjoint Partition ($\mathcal{O}_e = 1$):** The term becomes $1 - \frac{1}{M} \approx 1$. A worker holds a negligible fraction of the global experts, so nearly every reshuffling event requires replacing the local expert set entirely.

Adding these components together, the communication cost due to sparse components of the model is:

$$B_{\text{sparse}}^{\text{per-site}}(\mathcal{O}_e, M, \gamma) = (R(\mathcal{O}_e) + P(\mathcal{O}_e, M, \gamma)) \sum_\ell \left(N_{le,\ell}\, s_{\text{expert},\ell}\right) \tag{8}$$

where $N_{le,\ell}$ is the number of local experts, $s_{\text{expert},\ell}$ is the single expert size in bytes, which includes up and down projection, and its router projector if not using the "skip-token" procedure. This formulation recovers the full-replica cost when $\mathcal{O}_e = M$ (matching Eq. (4)). Notably, with $\mathcal{O}_e = 1$, the system requires one worker to act as an orchestrator to handle the reassignment of distinct sparse components that are not shared among workers. In such cases, the ring factor $\frac{2(\mathcal{O}_e - 1)}{\mathcal{O}_e}$ disappears, and the cost uniquely depends on the summation. Strategies synchronizing optimizer states (e.g., `LocalAdamW`) multiply $B_{\text{AR}}^{\text{per-site}}$ by a constant factor (typically $3\times$) (Cheng & Glasgow, 2025).

Bytes-Per-Token (`BPT`). For a site processing $T_{\text{local}} = KB_l$ tokens per round, where K is the number of local optimization steps and $B_l = \frac{B_g}{M}$ is the local batch size (global batch size B_g split across the M parallel workers), the total `BPT` is derived from the sum of dense and sparse costs ($B^{\text{per-site}} = B_{\text{dense}}^{\text{per-site}} + B_{\text{sparse}}^{\text{per-site}}$):

$$\text{BPT} = \frac{B^{\text{per-site}}}{KB_l} = \frac{MB^{\text{per-site}}}{KB_g} \tag{9}$$

This relation highlights three key levers: increasing K or batch size B amortizes communication; minimizing $\mathcal{O}_e$ caps the sparse term.

4.3 Wall-Clock Time Model

We synthesize the computational (Sec. 4.1) and communication (Sec. 4.2) models to estimate the total training time T_{tot}. This unified framework serves as a "Roofline" model for federated `MoE` training, identifying bottlenecks to guide the selection of the synchronization interval K, the expert overlap $\mathcal{O}_e$, and the reshuffling frequency γ.

Computational Time (t_{comp}). The local optimization step time, t_{comp}, depends on the dynamic FLOP count and hardware efficiency. Unlike dense baselines, t_{comp} scales with the partitioning strategy. Let Φ_{peak} be the theoretical peak FLOPs and $\text{MFU}(\cdot)$ be the Model FLOPs Utilization, which is sensitive to local batch size $B_l = \frac{B_g}{M}$ and arithmetic intensity of the reduced expert kernels. Using $F_{\text{token}}(\mathcal{O}_e)$ from Eq. (3):

$$t_{\text{comp}}(\mathcal{O}_e, M) = \frac{F_{\text{token}}(\mathcal{O}_e) \cdot \frac{B_g}{M}}{\text{MFU}(\frac{B_g}{M}, N_{\text{le},\ell}) \cdot \Phi_{\text{peak}}} \tag{10}$$

Reducing $\mathcal{O}_e$ lowers F_{token} due to skipping tokens, reducing t_{comp}. However, this is bounded by the fragmentation limit: if the local expert count $N_{\text{le},\ell}$ becomes too small relative to B_l, MFU degrades due to insufficient tensor core saturation.

Communication Time (t_{comm}). Communication time is governed by the effective WAN bandwidth B_{eff}. While intra-datacenter interconnects (e.g., NVLink) offer 100–400 Gbps (Rajbhandari et al., 2022), cross-site WAN links typically operate at 1–20 Gbps (Jaghouar et al., 2024), making transfer time bandwidth-dominated. We extend the payload formulation to account for gradient compression (e.g., QSGD (Alistarh et al., 2017)), which is an orthogonal addition to our work, via a factor $\rho \in (0, 1]$, which is $\rho = 1.0$ by default:

$$t_{\text{comm}}(M, \mathcal{O}_e, \gamma) \approx \frac{\rho \cdot \left[B_{\text{dense}}^{\text{per-site}}(M) + B_{\text{sparse}}^{\text{per-site}}(\mathcal{O}_e, M, \gamma)\right]}{B_{\text{eff}}} \tag{11}$$

This term captures the trade-off of our strategy: lowering $\mathcal{O}_e$ minimizes gradient synchronization (the aggregation cost), but utilizing random placement ($\gamma > 0$) introduces a state migration penalty (the broadcast factor $P \to \gamma$).

Total Training Time. The total time T_{tot} for a token budget of N_{tok} sums the costs of K compute steps and one communication step per round, subject to an overlap factor α. With global batch size $B_g = M \cdot B_l$ and total rounds $\frac{N_{\text{tok}}}{K \cdot B_g}$:

$$T_{\text{tot}} \approx \frac{N_{\text{tok}}}{M \cdot B_l \cdot K} \left[K \cdot t_{\text{comp}}(M, \mathcal{O}_e) + (1 - \alpha) t_{\text{comm}}(M, \mathcal{O}_e, \gamma)\right] \tag{12}$$

Model Dynamics. This framework highlights three critical dynamics captured by the model:

1. **The Federated Compute-Bound Regime:** Ideally, training satisfies $K \cdot t_{\text{comp}} \gg (1 - \alpha) t_{\text{comm}}$. If inter-site bandwidth is comparable to intra-site, partitioning offers minimal benefit. However, on commodity WAN, full-replica approaches ($\mathcal{O}_e = M$) may require excessively large K to satisfy this inequality. FoMoE achieves this regime with moderate K by aggressively reducing payload size via partial replication ($\mathcal{O}_e \to 1$). This improvement is multiplicative with other optimizations; for example, maximal overlap ($\alpha \to 1$) and compression ($\rho \ll 1$) can make low-bandwidth training more attainable even for 100B+ modeled configurations on 1.00 Gb*p*s links (Qi et al., 2025).
2. **The Memory-Time Coupling:** Eq. (12) reveals a quadratic efficiency gain from reducing $\mathcal{O}_e$. Lowering overlap reduces memory pressure (Sec. 4.1), enabling a larger local batch size B_l. Since the communication overhead ratio is proportional to $\frac{1}{K \cdot B_l}$, increasing B_l dilutes the fixed communication cost over more tokens. Thus, reducing $\mathcal{O}_e$ linearly reduces the payload ($B^{\text{per-site}} \downarrow$) increases throughput per sync ($B_l \uparrow$).
3. **The Placement Convexity:** Under random placement, the interaction between aggregation and migration costs dictates an optimal overlap $\mathcal{O}_e^*$. If reshuffling is frequent ($\gamma \approx 1$), the migration penalty makes disjoint partitioning ($\mathcal{O}_e = 1$) suboptimal. FoMoE targets a "Semi-Static" regime ($\gamma \ll 1$) to amortize migration costs, allowing the system to reduce bandwidth via disjoint experts.

Solving Eq. (12) for a specific hardware profile ($B_{\text{eff}}, \Phi_{\text{peak}}$) yields the theoretical optimal configuration $(\mathcal{O}_e^*, K^*)$ that minimizes wall-clock time. In practice, we generate a set of hardware-efficient configurations ranked by increasing training time and empirically evaluate their task performance (e.g., language modeling loss). This sweep optimizes the trade-off between system efficiency and task performance, aiming to maximize performance per time unit.

5 Experimental Setting

We evaluate FoMoE across a range of model scales to quantify trade-offs among communication efficiency, computational throughput, and machine learning quality. The experiments focus on the modeling and cost effects of partial expert replication: convergence, routing stability, communication volume, and throughput. They deliberately avoid characterizing every production-system concern, such as time-varying jitter, asymmetric links, stragglers, or heterogeneous accelerators.

Model Configurations. We define five model configurations: **Small Proxy (54M)**, **Medium (150M)**, **Large (2B)**, **XL (13B)**, and **XXL (100B)**. These span five operational regimes: rapid prototyping, entry-level modeling, compact open-weight scaling, standard open-weight deployment, and memory-bound scaling. All models utilize a decoder-only Transformer backbone as described in Sec. B. All comparisons are made on an iso-parameter basis: each sparse MoE configuration matches the parameter count of its dense counterpart while reducing inference FLOPs by a factor of 0.25. To configure the larger model sizes, internal components are scaled from the **Small Proxy (54M)** model, as in (Dey et al., 2025; Malasnicki et al., 2025; Yang & Hu, 2021). The specifics of the architectures are detailed in Table 1.

Table 1: Overview of the model configurations used in the "ghost experts" studies. Configurations range from a small proxy model to a massive 100B parameter model, maintaining the same head dimension ($d_{head} = 64$) and expansion factor (4).

	Model Architecture			Model Configuration			Scale
Configuration	d_{model}	n_{layers}	n_{heads}	$N_{experts}$	d_{expert}	N_{active}	**Total Params**
Small Proxy	256	4	4	8	128	2	∼54M
Medium (150M)	512	16	8	16	128	4	∼153M
Large (2B)	1024	128	16	32	128	8	∼1.8B
XL (13B)	2048	256	32	64	128	16	∼13.3B
XXL (100B)	4096	512	64	128	128	32	∼104B

Machine Learning Performance. We evaluate performance stability and convergence speed. To circumvent the prohibitive cost of hyperparameter tuning at the 100B scale, we adopt the **CompleteP** parameterization framework (Dey et al., 2025; Malasnicki et al., 2025; Yang & Hu, 2021). We first validate the optimal learning rates across cross-site configurations derived at the **Small Proxy (54M)** scale, and then transfer them to large-model settings. All models are trained on a large pre-training corpus comprising a mixture of general web-scraped text and code, inspired by the SmolLM Corpus (Ben Allal et al., 2024). We adopted the GPTOss tokenizer for pre-processing the raw text (OpenAI, 2025). The training duration has been chosen to reach at least the Chinchilla compute optimality, i.e., 20 tokens per parameter, and slightly adjusted to allow cross-site experiments to terminate at a round boundary (Hoffmann et al., 2022). Subsequently, we conduct ablation studies on the overlapping factor $\mathcal{O}_e$ to quantify the resulting perplexity degradation due to reduced expert visibility. Finally, we analyze the routing dynamics of the "skip-token" mechanism to ensure that dropping tokens for "ghost experts" does not destabilize convergence relative to forced local routing. Each trained model was evaluated on the entire validation set of the `en` C4 corpus (Raffel et al., 2020), and we report the mean and standard deviation of evaluation perplexity across the dataset.

Communication Efficiency. We quantify network overhead using **Bytes Per Token (BPT)** and total synchronization volume per round. We compare FoMoE against full-replica baselines (e.g., DiLoCo/Photon), modeling the communication cost as a function of expert overlap. Theoretically, setting $\mathcal{O}_e = 1$ (disjoint experts) restricts synchronization to the dense backbone (θ_{dense}), so the payload reduction scales with the share of parameters in sparse experts. In the XL and XXL configurations of Tab. 1, where sparse experts dominate the model state, this removes most expert payload from each cross-site round.

Computational Efficiency. Efficiency is measured via **Model FLOPs Utilization (MFU)** and wall-clock throughput (tokens/sec). While standard partial replication maintains an iso-FLOP count per token, the "skip-token" mechanism dynamically reduces the effective computational graph. We model the theoretical throughput of this mechanism and compare it with empirical results, identifying overheads due to router divergence and memory fragmentation.

Model Size Scaling and Placement. We explore the high-dimensional design space defined by the interactions among model size (N), expert count (N_e), and cluster size (M). By varying the number of experts and workers, we analyze the fragmentation limit: the threshold where the reduction in local batch size per expert (due to high N_e/M ratios) can degrade GPU tensor core saturation.

Training Environment. Experiments are executed on GPU cluster nodes with NVIDIA accelerators. Unless a result is explicitly identified as an empirical throughput measurement, WAN bandwidth and latency values are treated as controlled cost-model parameters rather than a claim of packet-level network emulation. In particular, our modeled cross-site setting uses $1.00\,\mathrm{Gb}ps$ and $10.00\,\mathrm{Gb}ps$ links with $50.00\,\mathrm{ms}$ latency, without time-varying jitter, asymmetric bandwidth, or persistent stragglers. Since the present implementation is synchronous, a slow site would delay the corresponding synchronization round. Stale-synchronous or asynchronous variants, such as Streaming DiLoCo, are compatible with the cost model but left to future work.

Implementation. We implement FoMoE using PyTorch collective communication primitives and a lightweight coordination layer. The coordinator initializes dense parameters globally, assigns sparse expert subsets according to the placement policy, triggers local training for K steps, and aggregates only the parameters owned by each expert replica group. Standard `Distributed Data Parallel` and FSDP assume static full-model replication or high-bandwidth intra-cluster sharding, so they cannot directly express the variable expert subset synchronization used by FoMoE without an additional placement and aggregation layer.

Reproducibility. Sec. F reports the full reproducibility summary for the experiments. The appendix records the training settings, sweep grids, dataset mixture, evaluation protocol, artifact status, and known gaps.

6 Evaluation and Results

We systematically evaluate FoMoE to quantify the trade-offs between communication, throughput, and quality. The evidence combines three dimensions: trained small-proxy quality runs for convergence and routing stability, measured skip-token throughput on accelerator hardware, and analytical/modeled communication, FLOP, memory, and wall-clock scaling for larger configurations.

Sec. 6.1 benchmarks full-replica baselines and identifies

which outer optimizer yields stronger communication-quality trade-offs than methods such as DiLoCo. Sec. 6.2 and Sec. 6.3 evaluate expert overlap, placement, and token skipping. The main takeaway is that FoMoE shifts the empirical Pareto frontier relative to DDP and full-replica, low-communication baselines, thereby providing improved performance at a given communication or compute cost in the regimes studied. Sec. 6.4 evaluates modeled scalability, demonstrating that reducing expert overlap yields linear reductions in communication and wall-clock components under our cost model, while "skip-token" improves measured throughput without specialized kernels. To ensure that our baselines are well-tuned across all trained model scales, we adopt the CompleteP transfer framework (Dey et al., 2025; Yang & Hu, 2021; Malasnicki et al., 2025). We provide further context and details of the sweeps in Sec. C.

6.1 Baseline Evaluation for Full Replica

We first calibrate two "full replica" baselines – dense and MoE implementations – on the perplexity-versus-communication Pareto front. Specifically, we evaluate full-replica runs across varying synchronization intervals $K \in \{32, 64, 128\}$ and worker counts $M \in \{2, 4, 8\}$, utilizing both DiLoCo and LocalAdamW outer optimizers. Experiments employ our *small proxy* configuration (see Tab. 1) and an *iso-token* budget, training all models to compute-optimal parameter counts. In these plots, *centralized* denotes the single-site high-bandwidth reference training run with no WAN throttling or low-frequency cross-site synchronization; it is a quality reference and not a practical baseline. *Full replica* denotes a cross-site configuration that keeps all dense and sparse parameters at every site and reduces only synchronization frequency. Fig. 5 therefore identifies the stronger full-replica outer-optimizer baseline used for the subsequent FoMoE comparisons.

Taken only as baseline calibration, Fig. 5 identifies LocalAdamW as the stronger full-replica optimizer: despite the extra optimizer-state traffic, it tracks the centralized quality reference more closely than DiLoCo at comparable communication budgets. The appendix table in Tab. 3 reports the full K and M grid and, consistent with prior work (Sani et al., 2025; Douillard et al., 2023), shows that increasing either value degrades perplexity through local drift and gradient noise. We therefore carry LocalAdamW forward as the main optimizer baseline for the FoMoE trade-off comparison in Fig. 6.

6.2 $\mathcal{O}_e$ and Expert Placement Strategies

Fig. 6 gives the corresponding FoMoE trade-off comparison, ablating the impact of O_e and the placement strategy. Here, LocalAdamW yields lower-perplexity configurations with comparable communication budgets be-

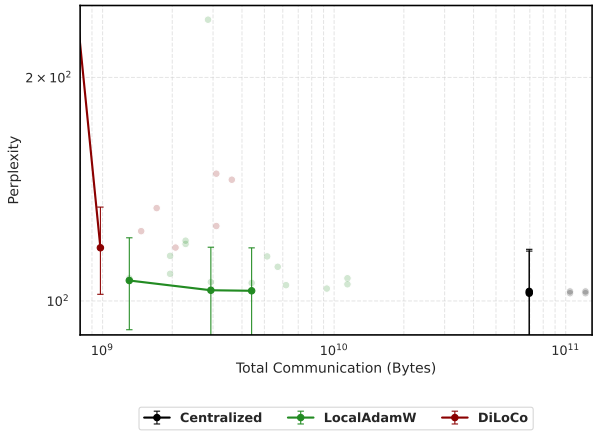

Figure 5: Pareto frontier of communication costs versus perplexity for full-replica baselines across outer optimizers. "Centralized" is the high-bandwidth single-site quality reference. LocalAdamW offers a better trade-off than DiLoCo within a given communication budget, motivating the use of the stateful optimizer in subsequent FoMoE configurations.

cause synchronizing optimizer state improves stability in low-frequency synchronization regimes (Cheng & Glasgow, 2025; Iacob et al., 2025b). Across both DiLoCo and LocalAdamW, fixed placement yields the best configurations for both communication-quality and compute-quality trade-offs. Random placement does not shift this frontier in the homogeneous, i.i.d. sweep: additional mixing doesn't bring sufficient benefits considering the migration costs; under heterogeneous or non-i.i.d. data, where mixing may improve quality, this conclusion may change.

Addressing the impact of O_e: across all combinations of synchronization intervals K, placement strategies, and workers M, we find that as $O_e \to M$, performance converges towards full-replica performance as seen in Tab. 6. Smaller values of O_e achieve higher perplexity under an iso-token training budget, as the models have less active capacity per site to capture the underlying data distribution. However, larger O_e values incur higher communication and computation costs; $O_e \approx 4$ yields the configurations that best balance quality against these costs.

6.3 "Skip-Token" and "Ghost Experts"

We now determine the influence of the "skip-token" and "ghost expert" mechanisms on the empirical compute-quality frontier, as seen in Fig. 7. Across both outer optimizers, introducing these mechanisms shifts the frontier by adding configurations with lower perplexity at comparable or lower compute budgets. This shift arises because each worker skips tokens routed to missing local experts rather than spending compute on unavailable expert paths. Additionally, these results show that adding the "skip-token" mechanism does not significantly impact training convergence.

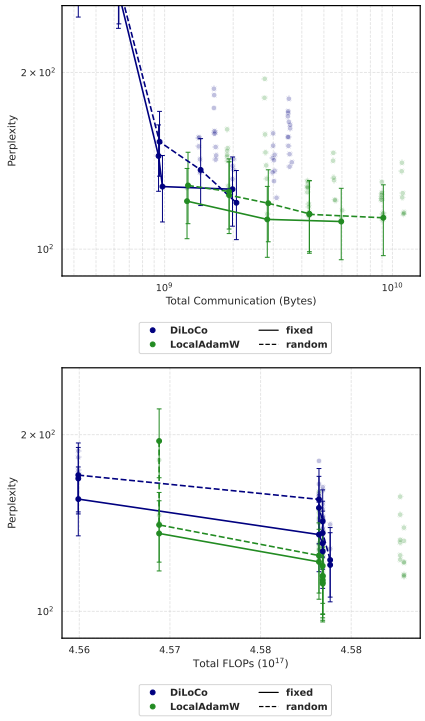

Figure 6: Empirical Pareto frontiers comparing communication cost (top) and computation cost (bottom) against perplexity for FoMoE across different outer optimizers and placement strategies. In this sweep, the strongest observed non-dominated configurations occur with LocalAdamW and fixed expert placement. Momentum-vector averaging and expert reassignment affect the trade-off between regularization, data mixing, and additional migration overhead.

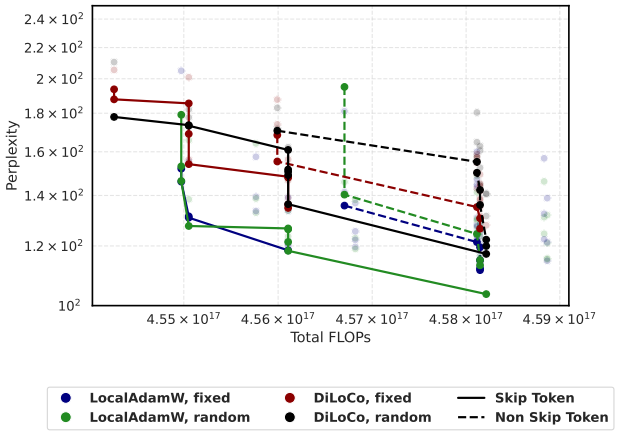

Figure 7: Impact of "skip-token" and "ghost experts" on the perplexity versus FLOPs Pareto frontier. Across outer optimizer configurations, these mechanisms shift the empirical compute-quality frontier toward lower FLOP cost at comparable perplexity, and lower perplexity at comparable FLOP cost, relative to full-replica baselines.

Quality interpretation. It is crucial to note that "skip-token" is not equivalent to full-replica training or to EP with remote activation dispatch, because it removes expert paths that are unavailable at the local site for a subset of the tokens in the batch. In this paper, we test its machine-learning effectiveness and stability through validation perplexity; here, routing stability should be interpreted as a supporting diagnostic. Future work must test whether, for bigger and more capable models, downstream task performance remains unchanged, especially for reasoning-heavy tasks where the router's ability to choose experts is critical, and assess longer training trajectories.

6.4 FoMoE at Scale

We assess FoMoE's scalability using the theoretical framework from Sec. 4. We analyze FLOPs, communication costs, and wall-clock time as analytical/modeled quantities, not as full large-scale training runs. Results show that minimizing expert overlap (O_e) significantly reduces compute, communication, and time in the modeled configurations, enabling FoMoE to outperform DDP and full-replica baselines under those assumptions.

FLOPs Scalability. Fig. 8 instantiates predictions from Sec. 4: (a) partitioning reduces router FLOPs, with linear FFN savings when $O_e < k$; (b) the "skip-token" mechanism further linearly reduces FLOPs by bypassing non-local experts; and (c) these savings grow with model scale as sparse parameters increasingly dominate.

Communication Scalability. Fig. 9 shows modeled cumulative communication in which FoMoE reduces bytes versus DDP and full-replica baselines. Unlike DDP's continuous overhead, FoMoE employs periodic, smaller bursts.

Wall-clock Time Scalability. Fig. 10 demonstrates that minimizing communication via lower O_e reduces modeled training time across the plotted bandwidths (1–1000 Gbps). Speedups are most critical in low-bandwidth regimes (e.g., 1 Gbps), where minimizing overlap is essential. In these modeled bandwidth regimes, FoMoE provides a benefit over the full-replica variant. Since the gains provided by FoMoE are multiplicative with communication-efficient methods, their relative magnitude holds across bandwidths under the cost-model assumptions.

Measured Skip-Token Throughput. Fig. 11 confirms that the "skip-token" mechanism enables the linear throughput gains predicted in Sec. 4 by bypassing computation for ghost experts. While reducing expert overlap generally improves throughput, skipping-token scales nearly linearly and outperforms the baseline. The current implementation of the mechanism is limited by a lack of a dedicated kernel; however, we expect that the theoretical gains will be at least partially transferable to any optimized implementation.

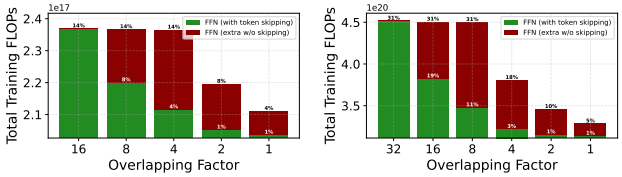

Figure 8: Analytical/modeled total training FLOPs versus expert overlap (O_e). Left: Small model ($d_{model} = 512$, 16 layers, 16 workers). **Right:** Large model ($d_{model} = 8192$, 512 layers, 32 workers). Green bars show modeled FLOPs with token skipping; red bars indicate avoided computation. Lowering overlap reduces compute, with greater relative savings for larger models.

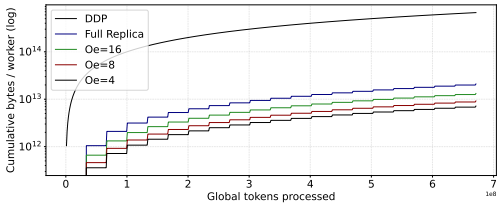

Figure 9: Analytical/modeled cumulative communication overhead versus global tokens. Communication cost (log scale) for a large model ($d_{model} = 8192$, 512 layers) on 32 workers using DiLoCo with random placement. FoMoE achieves increasing bandwidth savings over DDP and Full Replica as expert overlap decreases from $O_e = 16$ (green) to $O_e = 4$ under the model assumptions.

7 Limitations

Our work is limited in scale. We train proxy-size models and measure skip-token throughput on accelerator hardware; the XL/XXL and 100B-scale results are projections from the communication, memory, FLOP, and wall-clock models, not end-to-end pre-training runs. Skip-token is also not equivalent to full-replica training or EP with remote activation dispatch: tokens routed to non-local experts are skipped rather than sent over the WAN. We evaluate its ML effect through validation perplexity, while downstream task performance remains future work. The system setting is deliberately controlled. Our evaluation assumes

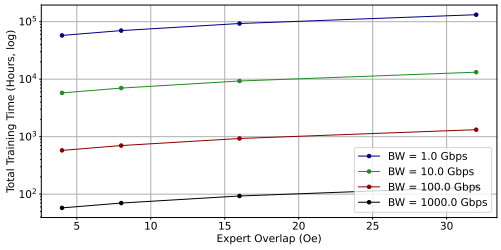

Figure 10: Analytical/modeled total training time versus expert overlap (O_e). Modeled time for a model ($d_{model} = 2048$, 64 layers) trained for 60B tokens on 32 H100s using DiLoCo (random). Reduced overlap accelerates training, especially under bandwidth constraints.

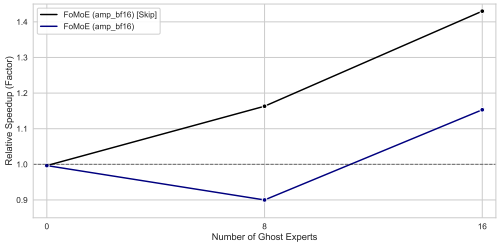

Figure 11: Measured speedup from token skipping. Relative speedup versus the number of ghost experts for a 150M MoE model with $d_{\text{model}} = 512$ and expert size 128 on 8 NVIDIA B200s. Skipping tokens achieves near-linear scaling, thereby validating the theoretical savings despite the lack of a custom kernel.

homogeneous workers, static traffic shaping, synchronous rounds, and representative WAN bandwidth/latency. It does not yet cover time-varying jitter, asymmetric links, persistent stragglers, hierarchical WAN topologies, or heterogeneous GPU clusters. Since the present implementation is synchronous, a slow site can still delay the corresponding synchronization round. Finally, the baseline and optimization scope are narrower than the full design space. We focus on DDP and full-replica low-communication methods such as DiLoCo/Photon-style training; a direct WAN-deployed EP baseline would be a useful stress test, although standard EP assumes high-bandwidth activation exchange and is therefore not the bottleneck targeted by FoMoE. We also use standard low-frequency distributed optimization, adapted to partial expert replication, in order to isolate the payload-size effect. Future work should co-design optimizers, placement schedules, and rules for migrating optimizer state specifically for partial expert replication.

8 Conclusion

This paper presents FoMoE, a framework that exploits MoE sparsity at the synchronization boundary in cross-datacenter training. Existing low-communication methods primarily reduce *how often* sites synchronize; FoMoE targets the complementary bottleneck of *how much* model state crosses the WAN at each round. By partitioning expert layers, FoMoE reduces communication overhead by up to $1.42\times$ over efficient baselines and $45.44\times$ over DDP in the studied regimes, while the "skip-token" mechanism delivers measured throughput speedups of up to $1.4\times$. The trained proxy regimes exhibit stable routing and convergence behavior, and the system model predicts that the same payload and memory reductions become increasingly important as sparse experts dominate larger configurations. Given these results, we argue that MoE sparsity should not stop at the datacenter boundary: when WAN bandwidth is the scarce resource, reducing the synchronization payload can be an effective communication-reduction mechanism alongside reducing synchronization frequency.

ACKNOWLEDGMENTS

This research was supported by the following entities: The Royal Academy of Engineering via DANTE (a RAEng Chair); the European Research Council, specifically the REDIAL project; SPRIND under the composite learning challenge; Google through a Google Academic Research Award; in addition to both IMEC and the Ministry of Education of Romania (through the Credit and Scholarship Agency).

REFERENCES

Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *CCS*, pp. 308–318. ACM, 2016.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: communication-efficient SGD via gradient quantization and encoding. In *NIPS*, pp. 1709–1720, 2017.

Athlur, S., Saran, N., Sivathanu, M., Ramjee, R., and Kwatra, N. Varuna: scalable, low-cost training of massive deep learning models. In *EuroSys*, pp. 472–487. ACM, 2022.

Ben Allal, L., Lozhkov, A., Penedo, G., Wolf, T., and von Werra, L. SmolLM-Corpus, July 2024. URL https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus.

Bonawitz, K. A., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, pp. 1175–1191. ACM, 2017.

Cagnasso, P., Belilovsky, E., and Oyallon, E. Unifying local communications and local updates for LLM pretraining. *CoRR*, abs/2606.11081, 2026. doi: 10.48550/arXiv.2606.11081. URL https://arxiv.org/abs/2606.11081.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost, 2016. URL https://arxiv.org/abs/1604.06174.

Cheng, Z. and Glasgow, M. Convergence of distributed adaptive optimization with local updates. In *ICLR*. OpenReview.net, 2025.

Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025. doi: 10.48550/arXiv.2507.06261. URL https://arxiv.org/abs/2507.06261.

Csordás, R., Irie, K., and Schmidhuber, J. Approximating two-layer feedforward networks for efficient transformers. In *EMNLP (Findings)*, pp. 674–692. Association for Computational Linguistics, 2023.

DeepSeek-AI. Deepseek-v3 technical report. *CoRR*, abs/2412.19437, 2024. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI, Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., Deng, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Yang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Chen, J., Yuan, J., Qiu, J., Song, J., Dong, K., Gao, K., Guan, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Pan, R., Xu, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Zheng, S., Wang, T., Pei, T., Yuan, T., Sun, T., Xiao, W. L., Zeng, W., An, W., Liu, W., Liang, W., Gao, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Chen, X., Nie, X., Sun, X., Wang, Z., and et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434, 2024.

Dey, N., Zhang, B. C., Noci, L., Li, M. B., Bordelon, B., Bergsma, S., Pehlevan, C., Hanin, B., and Hestness, J. Don't be lazy: Completep enables compute-efficient deep transformers. *CoRR*, abs/2505.01618, 2025.

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

Douillard, A., Feng, Q., Rusu, A. A., Kuncoro, A., Donchev, Y., Chhaparia, R., Gog, I., Ranzato, M., Shen, J., and Szlam, A. Dipaco: Distributed path composition. *CoRR*, abs/2403.10616, 2024.

Douillard, A., Donchev, Y., Rush, K., Kale, S., Charles, Z., Garrett, Z., Teston, G., Lacey, D., McIlroy, R., Shen, J., Ramé, A., Szlam, A., Ranzato, M., and Barham, P. Streaming diloco with overlapping communication: Towards a distributed free lunch. *CoRR*, abs/2501.18512, 2025.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., Zoph, B., Fedus, L., Bosma, M. P., Zhou, Z., Wang, T., Wang, Y. E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K. S., Duke, T., Dixon, L., Zhang, K., Le, Q. V., Wu, Y., Chen, Z., and Cui, C. Glam: Efficient scaling of language models with mixture-of-experts. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 2022.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39, 2022.

Gale, T., Narayanan, D., Young, C., and Zaharia, M. Megablocks: Efficient sparse training with mixture-of-experts. In *MLSys*. mlsys.org, 2023.

Guo, B., Mei, Y., Xiao, D., and Wu, W. PFL-MoE: Personalized federated learning based on mixture of experts. In *Web and Big Data*, volume 12858 of *Lecture Notes in Computer Science*, pp. 480–486. Springer, 2021. doi: 10.1007/978-3-030-85896-4_37. URL https://doi.org/10.1007/978-3-030-85896-4_37.

Guthrie, S. Inside the world's most powerful ai datacenter, Sept 2025. URL https://blogs.microsoft.com/blog/2025/09/18/inside-the-worlds-most-powerful-ai-datacenter/.

He, J., Zhai, J., Antunes, T., Wang, H., Luo, F., Shi, S., and Li, Q. Fastermoe: modeling and optimizing training of large-scale dynamic pre-trained models. In *PPoPP*, pp. 120–134. ACM, 2022.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M. X., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., and Chen, Z. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *NeurIPS*, pp. 103–112, 2019.

Hwang, C., Cui, W., Xiong, Y., Yang, Z., Liu, Z., Hu, H., Wang, Z., Salas, R., Jose, J., Ram, P., Chau, J., Cheng, P., Yang, F., Yang, M., and Xiong, Y. Tutel: Adaptive mixture-of-experts at scale. In *MLSys*. mlsys.org, 2023.

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 *ICLR*. OpenReview.net, 2025a.

Iacob, A., Sani, L., Safaryan, M., Giampouras, P., Horváth, S., Jovanovic, A., Kurmanji, M., Aleksandrov, P., Shen, W. F., Qiu, X., and Lane, N. D. DES-LOC: desynced low communication adaptive optimizers for training foundation models. *CoRR*, abs/2505.22549, 2025b.

Jaghouar, S., Ong, J. M., and Hagemann, J. Opendiloco: An open-source framework for globally distributed low-communication training. *CoRR*, abs/2407.07852, 2024.

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

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

Khaled, A., Kale, S., Douillard, A., Jin, C., Fergus, R., and Zaheer, M. Understanding outer optimizers in local SGD: learning rates, momentum, and acceleration. *CoRR*, abs/2509.10439, 2025.

Kim, J., Lee, B., Park, C., Oh, Y., Kim, B., Yoo, T., Shin, S., Han, D., Shin, J., and Yoo, K. M. Peri-ln: Revisiting normalization layer in the transformer architecture. In *ICML*. OpenReview.net, 2025.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. In *ICLR*. OpenReview.net, 2021.

Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. BASE layers: Simplifying training of large, sparse models. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6265–6274. PMLR, 2021.

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020a.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *MLSys*. mlsys.org, 2020b.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR (Poster)*. OpenReview.net, 2019.

Lu, X., Liu, Q., Xu, Y., Zhou, A., Huang, S., Zhang, B., Yan, J., and Li, H. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6159–6172. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.334. URL https://aclanthology.org/2024.acl-long.334/.

Ludziejewski, J., Krajewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygózdz, T., Sankowski, P., Cygan, M., and Jaszczur, S. Scaling laws for fine-grained mixture of experts. In *ICML*. OpenReview.net, 2024.

Malasnicki, J., Ciebiera, K., Borun, M., Pióro, M., Ludziejewski, J., Stefaniak, M., Krutul, M., Jaszczur, S., Cygan, M., Adamczewski, K., and Krajewski, J. μ-parametrization for mixture of experts. *CoRR*, abs/2508.09752, 2025.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017.

Mei, H., Cai, D., Zhou, A., Wang, S., and Xu, M. Fedmoe: Personalized federated learning via heterogeneous mixture of experts. *arXiv preprint arXiv:2408.11304*, 2024.

Micikevicius, P., Narang, S., Alben, J., Diamos, G. F., Elsen, E., García, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. Mixed precision training. In *ICLR (Poster)*. OpenReview.net, 2018.

Narayanan, D., Harlap, A., Phanishayee, A., Seshadri, V., Devanur, N. R., Ganger, G. R., Gibbons, P. B., and Zaharia, M. Pipedream: generalized pipeline parallelism for DNN training. In *SOSP*, pp. 1–15. ACM, 2019.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *CoRR*, abs/2508.10925, 2025.

Palak, Reddy, T. R., Kataria, B., Gandhi, R., Tandon, K., Bhattacherjee, D., and Padmanabhan, V. N. Improving training time and GPU utilization in geo-distributed language model training. *CoRR*, abs/2411.14458, 2024. doi: 10.48550/arXiv.2411.14458. URL https://arxiv.org/abs/2411.14458.

Patarasuk, P. and Yuan, X. Bandwidth optimal all-reduce algorithms for clusters of workstations. *J. Parallel Distributed Comput.*, 69(2):117–124, 2009.

Qi, J., Zhu, W., Li, L., Wu, M., Wu, Y., He, W., Gao, X., Zeng, J., and Heinrich, M. Dilocox: A low-communication large-scale training framework for decentralized cluster. *CoRR*, abs/2506.21263, 2025.

Rabenseifner, R. Optimization of collective reduction operations. In *International Conference on Computational Science*, volume 3036 of *Lecture Notes in Computer Science*, pp. 1–9. Springer, 2004.

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. *Journal of Machine Learning Research*, 21 (140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: memory optimizations toward training trillion parameter models. In *SC*, pp. 20. IEEE/ACM, 2020.

Rajbhandari, S., Li, C., Yao, Z., Zhang, M., Aminabadi, R. Y., Awan, A. A., Rasley, J., and He, Y. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18332–18346. PMLR, 2022.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *ICLR*. OpenReview.net, 2021.

Reisser, M., Louizos, C., Gavves, E., and Welling, M. Federated mixture of experts. *CoRR*, abs/2107.06724, 2021.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Pinto, A. S., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. In *NeurIPS*, pp. 8583–8595, 2021.

Roller, S., Sukhbaatar, S., Szlam, A., and Weston, J. Hash layers for large sparse models. In *NeurIPS*, pp. 17555–17566, 2021.

Ryabinin, M. and Gusev, A. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. *Advances in Neural Information Processing Systems*, 33:3659–3672, 2020. doi: 10.48550/arXiv.2002.04013. URL https://arxiv.org/abs/2002.04013.

Ryabinin, M., Dettmers, T., Diskin, M., and Borzunov, A. SWARM parallelism: Training large models can

be surprisingly communication-efficient. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29416–29440. PMLR, 2023. URL https://proceedings.mlr.press/v202/ryabinin23a.html.

Sani, L., Iacob, A., Cao, Z., Lee, R., Marino, B., Gao, Y., Zhao, W., Cai, D., Li, Z., Qiu, X., and Lane, N. D. Photon: Federated LLM pre-training. In *Eighth Conference on Machine Learning and Systems*, 2025. URL https://openreview.net/forum?id=AQgYcfg5EI.

Sarfi, A., Thérien, B., Lidin, J., and Belilovsky, E. Communication efficient LLM pre-training with SparseLoCo. *CoRR*, abs/2508.15706, 2025. doi: 10.48550/arXiv.2508.15706. URL https://arxiv.org/abs/2508.15706.

Sevilla, J. and Troynikov, A. Could decentralized training solve ai's power problem?, 2025. URL https://epoch.ai/blog/could-decentralized-training-solve-ais-power-problem. Accessed: 2025-12-11.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR (Poster)*. OpenReview.net, 2017.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053, 2019.

Stich, S. U. Local SGD converges fast and communicates little. In *ICLR (Poster)*. OpenReview.net, 2019.

Su, J., Ahmed, M. H. M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Team, K., Bai, Y., Bao, Y., Chen, G., Chen, J., Chen, N., Chen, R., Chen, Y., Chen, Y., Chen, Y., Chen, Z., Cui, J., Ding, H., Dong, M., Du, A., Du, C., Du, D., Du, Y., Fan, Y., Feng, Y., Fu, K., Gao, B., Gao, H., Gao, P., Gao, T., Gu, X., Guan, L., Guo, H., Guo, J., Hu, H., Hao, X., He, T., He, W., He, W., Hong, C., Hu, Y., Hu, Z., Huang, W., Huang, Z., Huang, Z., Jiang, T., Jiang, Z., Jin, X., Kang, Y., Lai, G., Li, C., Li, F., Li, H., Li, M., Li, W., Li, Y., Li, Y., Li, Z., Li, Z., Lin, H., Lin, X., Lin, Z., Liu, C., Liu, C., Liu, H., Liu, J., Liu, J., Liu, L., Liu, S., Liu, T. Y., Liu, T., Liu, W., Liu, Y., Liu, Y., Liu, Y., Liu, Y., Liu, Z., Lu, E., Lu, L., Ma, S., Ma, X., Ma, Y., Mao, S., Mei, J., Men, X., Miao, Y., Pan, S., Peng, Y., Qin, R., Qu, B., Shang, Z., Shi, L., Shi, S., Song, F., Su, J., Su, Z., Sun, X., Sung, F., Tang, H., Tao, J., Teng, Q., Wang, C., Wang, D., Wang, F., Wang, H., Wang, J., Wang, J., Wang, J., Wang, S., Wang, S., Wang, Y., Wang, Y., Wang, Y., Wang, Y., Wang, Y., Wang, Z., Wang, Z., Wang, Z., Wei, C., Wei, Q., Wu, W., Wu, X., Wu, Y., Xiao, C., Xie, X., Xiong, W., Xu, B., Xu, J., Xu, J., Xu, L. H., Xu, L., Xu, S., Xu, W., Xu, X., Xu, Y., Xu, Z., Yan, J., Yan, Y., Yang, X., Yang, Y., Yang, Z., Yang, Z., Yang, Z., Yao, H., Yao, X., Ye, W., Ye, Z., Yin, B., Yu, L., Yuan, E., Yuan, H., Yuan, M., Zhan, H., Zhang, D., Zhang, H., Zhang, W., Zhang, X., Zhang, Y., Zhang, Y., Zhang, Y., Zhang, Y., Zhang, Y., Zhang, Y., Zhang, Z., Zhao, H., Zhao, Y., Zheng, H., Zheng, S., Zhou, J., Zhou, X., Zhou, Z., Zhu, Z., Zhuang, W., and Zu, X. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Tran, V.-T., Khiem, L. H., and Pham, Q.-V. Revisiting sparse mixture of experts for resource-adaptive federated fine-tuning foundation models. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*, 2025. URL https://openreview.net/forum?id=IwNOUYgtuz.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

Xie, L., Luan, T., Cai, W., Yan, G., Chen, Z., Xi, N., Fang, Y., Shen, Q., Wu, Z., and Yuan, J. dflmoe: Decentralized federated learning via mixture of experts for medical data analysis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10203–10213, 2025.

Yang, G. and Hu, E. J. Tensor programs IV: feature learning in infinite-width neural networks. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11727–11737. PMLR, 2021.

Zec, E. L., Mogren, O., Martinsson, J., Sütfeld, L. R., and Gillblad, D. Specialized federated learning using a mixture of experts. *CoRR*, abs/2010.02056, 2020. doi: 10.48550/arXiv.2010.02056. URL https://arxiv.org/abs/2010.02056.

Zhai, M., He, J., Ma, Z., Zong, Z., Zhang, R., and Zhai, J. Smartmoe: Efficiently training sparsely-activated models through combining offline and online parallelization. In *USENIX ATC*, pp. 961–975. USENIX Association, 2023.

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V. Y., Dai, A. M., Chen, Z., Le, Q. V., and Laudon, J. Mixture-of-experts with expert choice routing. In *NeurIPS*, 2022.

APPENDIX

A EXTENDED RELATED WORKS

A.1 MoE architectures and scaling

The sparsely-gated MoE layer (Shazeer et al., 2017) established the paradigm of conditional computation, utilizing trainable gating to activate a sparse subset of expert FFNs per token. In this work, we adopt the σ-MoE formulation (Csordás et al., 2023), which replaces the competitive softmax gating with independent sigmoid activations to better approximate dense feed-forward networks.

Formally, given an input token vector $\mathbf{x} \in \mathbb{R}^d$, the layer output $\mathbf{y}$ is the weighted sum of a sparse subset of expert networks $\{E_i\}_{i=1}^N$:

$$\mathbf{y} = \sum_{i \in \mathcal{T}(\mathbf{x})} s_i(\mathbf{x}) E_i(\mathbf{x}) \,, \tag{13}$$

where $s_i(\mathbf{x}) = \sigma(\mathbf{w}_{g,i}^\top \mathbf{x})$ is the gating score for expert i, computed using the logistic sigmoid function $\sigma(\cdot)$ and a trainable gating vector $\mathbf{w}_{g,i}$. The set of active indices $\mathcal{T}(\mathbf{x})$ is determined by the Top-K operator applied to the sigmoid scores:

$$\mathcal{T}(\mathbf{x}) = \text{TopK}\left(\{\sigma(\mathbf{w}_{g,j}^\top \mathbf{x})\}_{j=1}^N, k\right) \,. \tag{14}$$

Unlike standard Softmax gating, which introduces competition by normalizing scores to sum to one, the σ-MoE allows independent expert activations, mitigating instability.

This computation is *conditional* because the experts $E_j(\mathbf{x})$ for indices $j \notin \mathcal{T}(\mathbf{x})$ are never executed. This mechanism decouples model capacity from compute cost: while the total parameter count scales linearly with the total number of experts N, the computational cost (FLOPs) per token scales only with the active set size k (where $k \ll N$), outside of the router network and sigmoid operations. This enables massive parameter scaling with near-constant training/inference costs.

Subsequent systems scaled this approach: GShard introduced Top-2 gating with automatic sharding (Lepikhin et al., 2021), while *Switch Transformers* demonstrated trillion-parameter training via Top-1 routing (Fedus et al., 2022). GLaM further validated the compute efficiency of MoE architectures against dense baselines (Du et al., 2022). Modern iterations, such as DeepSeek-V2, integrate MoE with optimized attention and KV-cache designs (DeepSeek-AI et al., 2024). Theoretical advancements include fine-grained scaling laws (Ludziejewski et al., 2024), contextualized by foundational dense scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022). Beyond NLP, scalability has been confirmed in vision domains (Riquelme et al., 2021).

A.2 Routing and load balancing

Load balancing is critical for maximizing MoE throughput and model quality. Approaches include BASE, which formulates assignment as a balanced linear optimization (Lewis et al., 2021); Expert-Choice, which inverts routing to allow experts to select tokens, ensuring perfect load balance (Zhou et al., 2022); and Hash Layers, which replace learned gating with fixed hashing functions (Roller et al., 2021). In practice, training stacks employ temperature schedules, capacity factors, and auxiliary losses to prevent expert collapse and minimize token dropping (Fedus et al., 2022; Lepikhin et al., 2021). Standard diagnostics—including routing entropy, load coefficients of variation, and overflow rates—are essential for monitoring these stability dynamics.

A.3 Systems support for MoE execution

Specialized frameworks optimize MoE execution through all-to-all communication, kernel fusion, and parallelization strategies. DeepSpeed-MoE and Tutel enable efficient expert parallelism and hierarchical communication (Rajbhandari et al., 2022; Hwang et al., 2023), while MegaBlocks introduces dropless block-sparse kernels to mitigate padding inefficiencies (Gale et al., 2023). FasterMoE and SmartMoE further optimize end-to-end training via dynamic hybrid parallelization schemes (He et al., 2022; Zhai et al., 2023). Crucially, these systems are designed for high-bandwidth intra-DC fabrics; our work adapts these principles to the bandwidth-constrained inter-DC regime.

A.4 Low-bandwidth distributed and federated pre-training

To accommodate low-bandwidth environments, federated and LocalSGD methods reduce synchronization frequency (McMahan et al., 2017; Stich, 2019), with extensions like FedProx and FedOPT addressing data heterogeneity (Li et al., 2020b; Reddi et al., 2021).

DiLoCo applies these principles to LM pre-training, utilizing large local steps and outer momentum to train across weakly connected device islands (Douillard et al., 2023), a capability operationalized by OpenDiLoCo (Jaghouar et al., 2024). Photon further demonstrates federated pre-training up to 7B parameters with perplexity parity and significantly reduced communication (Sani et al., 2025). Additional optimization strategies include decoupling embeddings via DEPT (Iacob et al., 2025a), desynchronizing optimizer states via DES-LOC (Iacob et al., 2025b), and co-designing modular paths via DiPaCo (Douillard et al., 2024). While these systems motivate our design, they rely on full replicas.

A.5 Expert placement and topology awareness

While intra-datacenter routing is well-studied, expert placement across heterogeneous wide-area networks (WAN) remains underexplored. Existing systems model placement primarily for intra-cluster efficiency (Hwang et al., 2023; He et al., 2022; Gale et al., 2023). Our work investigates whether selective expert replication can minimize WAN traffic without degrading quality. We evaluate *fixed*, *random*, and *affinity-based* placement policies, analyzing their impact on stability (routing entropy), specialization, and load balance (Lewis et al., 2021; Zhou et al., 2022; Fedus et al., 2022).

A.6 Federated MoE

Early research in Federated MoE focused on personalization, combining global generalists with local specialists (Zec et al., 2020; Guo et al., 2021), or employing client-specific gating (FedMix) to enhance generalization (Reisser et al., 2021). Subsequent systems extended this to Transformers: FedMoE assigns personalized sub-MoEs to clients (Mei et al., 2024), while A3SMoE optimizes expert utilization across heterogeneous devices (Tran et al., 2025). Decentralized approaches like dFLMoE, Learning@home/DMoE, and SWARM explore expert fusion, volunteer computing, and heterogeneous weak-link model parallelism (Xie et al., 2025; Ryabinin & Gusev, 2020; Ryabinin et al., 2023). However, these studies primarily address personalization, volunteer-device execution, inference constraints, or model-parallel scheduling; they do not address the payload size limitations inherent to communication-efficient federated pre-training.

A.7 Optimization, privacy, and governance

Our approach relies on established analyses of adaptive federated optimizers (FedAdam, FedYogi) and LocalSGD (Reddi et al., 2021; Stich, 2019). While privacy mechanisms like secure aggregation (Bonawitz et al., 2017; Abadi et al., 2016; Kairouz et al., 2021) can be integrated, they introduce trade-offs in quality and complexity. Additionally, routing and placement decisions can be logged to ensure auditability and fairness in non-IID settings.

A.8 Synthesis

Prior work establishes that: (i) MoE architectures achieve dense-model quality with reduced compute (Fedus et al., 2022; Du et al., 2022; DeepSeek-AI et al., 2024); and (ii) low-communication training can approach centralized baselines under WAN constraints (Douillard et al., 2023; Sani et al., 2025). However, current cross-silo systems remain limited by the requirement to transmit full dense models. Our work bridges this gap, introducing a federated MoE framework that targets *bytes per sync* through optimized expert placement, thereby shifting the studied trade-offs among perplexity, communication, and wall-clock time.

B Model Architecture and Configuration

To address the stability challenges of training large-scale models over bandwidth-constrained federated networks, FoMoE adopts a specialized Transformer architecture. We depart from monolithic baselines by integrating **Sigma-MoE** (σ-MoE) routing with a **Peripheral LayerNorm** (Peri-LN) strategy. These choices maintain gradient flow and prevent router collapse in the *Partial Replica* regime ($O_e < M$), where expert availability fluctuates due to random placement. The full architecture is shown in Fig. 12.

B.1 Peripheral LayerNorm (Peri-LN)

Standard Pre-LayerNorm (Pre-LN) allows unbounded growth in the residual stream, while Post-LayerNorm (Post-LN) impedes gradient propagation, causing instability in low-frequency synchronization settings.

We employ **Peripheral LayerNorm** (Kim et al., 2025), which applies normalization exclusively to the residual *branches*—both immediately preceding (Pre-LN) and following (Post-LN) the computational blocks—while preserving a linear, un-normalized main data path.

Formally, for a layer input $\mathbf{x}_l$ and computational block $\mathcal{F}$ (e.g., MHA or MoE):

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathrm{LN}(\mathcal{F}(\mathrm{LN}(\mathbf{x}_l))) \tag{15}$$

This configuration ensures that inputs to the routing mechanism maintain unit variance, preventing dot-product logits from saturating the gating function, while maintaining a clear gradient highway from loss to input. We additionally apply a **Post-Embedding LayerNorm** to the initial token embeddings. As demonstrated in (Kim et al., 2025), this modification helps control the variance entering the first Transformer block, which has been shown to improve training stability.

B.2 Attention Mechanism

We utilize Multi-Head Attention (MHA) enhanced with **Rotary Positional Embeddings (RoPE)** (Su et al., 2024). Positional information is injected into the Query ($\mathbf{Q}$) and Key ($\mathbf{K}$) vectors within the normalized branch (post-Peri-LN), ensuring relative positional information is preserved without dampening by residual stream statistics.

B.3 Federated Mixture-of-Experts (σ-MoE)

We replace the Feed-Forward Network (FFN) with a sparse Mixture-of-Experts layer based on the **σ-MoE** formulation.

B.3.1 Expert Configuration and Fine-Grained Scaling

To balance routing flexibility with memory fragmentation, we adopt a **Fine-Grained MoE** strategy (Ludziejewski et al., 2024). We decompose the standard dense FFN ($d_{ff} = 4d_{model}$) into N_e experts with granularity $G = 8$ and expansion $E = 1$, leading to $N_e = G \cdot E$ experts. To maintain an iso-parameter count with the dense baseline, the expert dimension scales as:

$$d_{expert} = \frac{4d_{model}}{N_e} \tag{16}$$

After each expert projection from d_{model} to d_{expert}, the SiLU nonlinearity is applied. For a $d_{model} = 512$ configuration, we employ $N_e = 16$ experts of dimension $d_{expert} = 128$. This granular partitioning allows the router to approximate larger functional manifolds by combining specialized sub-experts, mitigating the impact of missing experts in partial replication.

B.3.2 Sigmoid Routing

Unlike competitive Softmax routers (e.g., Switch Transformer), FoMoE employs **Sigmoid Routing** to enforce independent expert selection. This independence is critical for the *Partial Replica* setting ($O_e < M$): the unavailability of a specific expert on a worker must not skew the probability distribution of the remaining available experts. The gating probability for expert i is given by:

$$p_i(\mathbf{x}) = \sigma(\mathbf{x}^\top \mathbf{w}_{r,i}) \tag{17}$$

After projecting scores from d_{model} to N_e, the router selects the top-k experts (typically $k = 2$) based on these independent scores, with inputs normalized via the FFN's Peri-LN to ensure stable routing entropy.

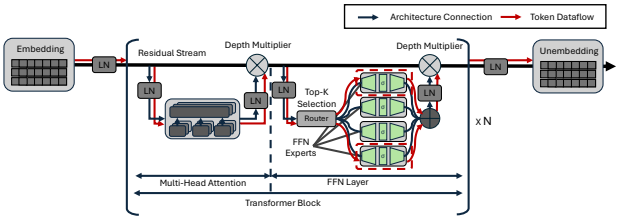

Figure 12: FoMoE Layer Architecture. The diagram illustrates the Peripheral LayerNorm (Peri-LN) strategy, where normalization is applied exclusively to the branches feeding the Attention and MoE blocks (Pre-LN) and immediately following them (Post-LN). This design maintains a linear residual stream to preserve gradient magnitude. Additionally, the architecture includes a depth multiplier applied to the output of each computational block, as well as layer normalization on the output of the embedding layer and the input of the unembedding layer.

C Hyperparameter Transferability and Implementation

To avoid prohibitive hyperparameter retuning at scale, we leverage established transfer techniques. *Maximal Update Parametrization* (μP) enables the transfer of learning rates and schedules across model widths (Yang & Hu, 2021), while CompleteP extends this invariance to model depth via residual rescaling, preventing instability in deep Transformers (Dey et al., 2025). We note, however, that scaling the expert count E requires specific adjustments to preserve feature learning efficiency (Malasnicki et al., 2025).

Transfer protocols. We transfer the following hyperparameters from baseline configurations: global learning rate schedules (scaled via μP/CompleteP), AdamW parameters ($\beta_1, \beta_2, \epsilon$), warmup duration, and router temperature. Conversely, regularization parameters (dropout, weight decay) and system-specific controls (overlap factor $\mathcal{O}_e$, reshuffling cadence T, local steps K) are treated as non-transferable and tuned individually at each scale. Consistent with (Malasnicki et al., 2025), we find that while μP effectively handles width scaling, significantly increasing E necessitates extending the learning rate warmup. Tuning was performed over a compute-optimal token budget (Hoffmann et al., 2022) using a linear warmup of 25% without a subsequent cooldown phase. For FoMoE, we evaluated learning rates $\eta \in \{0.001, 0.002, 0.004, 0.01, 0.02, 0.05\}$. Centralized baselines (dense and MoE) were tuned over $\eta \in \{0.01, 0.02, 0.03, 0.04, 0.07\}$. Across all architectures and aggregation methods, we observe that $\eta \approx 0.01$ yields the optimal learning rate, as shown in Fig. 13.

We use CompleteP to transfer the global learning rate schedule, AdamW settings ($\beta_1, \beta_2, \epsilon$), warmup duration, and router temperature across model scales for both our federated and centralized baseline experiments. This was repeated for different regularization factors and system-specific knobs, such as the overlapping factor $\mathcal{O}_e$, the shuffling cadence T, and the local steps K, as these do not scale with the CompleteP framework. In the sweeps, we found that the optimal learning rate across all our experiment types was $\eta \approx 0.01$.

Implementation. We implemented FoMoE using PyTorch's collective communication primitives. Standard frameworks such as DistributedDataParallel or FSDP are unsuitable for our setting, as they assume static, full-model replication and high-bandwidth intra-cluster interconnects. To support dynamic expert placement in WAN environments, we developed a custom coordination layer where the server explicitly manages the broadcast and aggregation of variable expert subsets at each synchronization round.

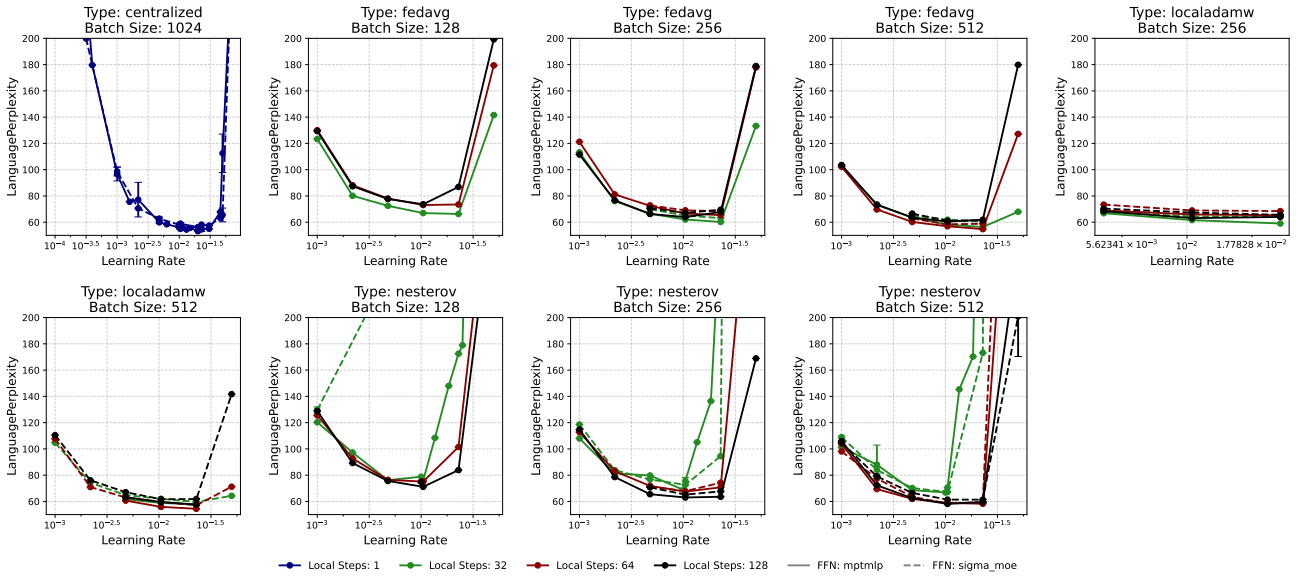

Figure 13: Learning rate tuning results for centralized (local step size of one) and federated baselines across varying local steps, architectures (dense vs. MoE), and aggregation methods. Across all configurations, a base learning rate of $\eta \approx 0.01$ consistently minimizes perplexity. We adopt this optimal rate for subsequent model scaling experiments.

D Theoretical Foundations and Cost Modeling

This appendix serves as the formal theoretical underpinning for the FoMoE system. We provide rigorous derivations for the cost models presented in Sec. 4, analyze the system's operational regimes, and formalize the trade-offs that govern the "Sweet Spot" for distributed MoE training.

D.1 System Definitions and Parameters

We model a synchronous training setup with M homogeneous workers connected by a heterogeneous network topology. The system is defined by the global parameters listed in Tab. 2.

Table 2: Global System and Model Parameters

Symbol	Definition	Context
M	Total number of workers (sites)	Infrastructure
K	Synchronization interval (local steps)	Optimization
B_{eff}	Effective WAN bandwidth	Infrastructure
$\mathcal{O}_e$	Expert Overlap Factor ($\in [1, M]$)	FoMoE Configuration
N_{le}	Number of Local Experts per worker	Derived: $\frac{\mathcal{O}_e}{M} N_e$
γ	Reshuffling frequency ($1/T_{\text{shuffle}}$)	Random Placement

D.2 Operational Regimes and The Distributed Paradigm Shift

To contextualize the FoMoE architecture, we mathematically distinguish it from prior paradigms based on the constraints imposed on the synchronization interval K and the expert overlap $\mathcal{O}_e$.

Regime 1: Distributed Data Parallelism (DDP). Defined by $\mathcal{O}_e = M$ (Full Replication) and $K = 1$. The communication payload is maximal ($P = 0, R \approx 2$). Training is strictly bandwidth-bound on WANs:

$$t_{\text{comm}}(\text{DDP}) \approx \frac{2S_{\text{model}}}{B_{\text{eff}}} \gg t_{\text{comp}} \tag{18}$$

Utilization approaches zero as M increases across low-bandwidth links.

Regime 2: Federated / Low-Frequency (e.g., DiLoCo). Defined by $\mathcal{O}_e = M$ and $K \gg 1$. Communication is amortized by K, but the payload remains the full model size S_{model}. The memory constraint is identical to DDP, limiting the maximum model size to the capacity of a single worker.

Regime 3: The FoMoE Regime (Partial Replication). Defined by $\mathcal{O}_e < M$ (typically $\mathcal{O}_e \ll M$) and $K > 1$. This regime introduces a "Communication-Sparsity" coupling. By partitioning experts, we reduce the steady-state memory and the synchronization payload linearly with $\mathcal{O}_e$. This breaks the "Full-Replica" barrier, allowing model size to scale linearly with M while t_{comm} decreases.

D.3 Extended Computational Cost Modeling

We refine the FLOPs derivation to explicitly account for both Router and FFN efficiency gains.

Router Efficiency. Standard gating networks project token embeddings of dimension d_{model} to N_e logits. In

FoMoE, tokens are only routed to locally resident experts. The router projection size is reduced from N_e to N_{le}.

$$F_{\text{gate}}(\mathcal{O}_e) = \frac{N_{\text{le}}}{N_e} F_{\text{gate}}^{\text{full}} = \frac{\mathcal{O}_e}{M} F_{\text{gate}}^{\text{full}} \tag{19}$$

This yields a linear reduction in router FLOPs proportional to the partition fraction.

FFN Efficiency and Skip-Token Scaling. As derived in the Main Text, the FFN cost is scaled by σ_{st}, accounting for the "Skip-Token" mechanism when $N_{\text{le}} < k$.

$$\sigma_{\text{st}} = \min\left(1, \frac{N_{\text{le}}}{N_e} \cdot \frac{N_e}{k}\right) \tag{20}$$

Combining these, the total FLOPs per token is:

$$F_{\text{token}} \approx F_{\text{dense}} + \sum_{\ell=1}^{L} \left(\underbrace{\sigma_{\text{st}} \frac{\min(k, N_{\text{le}})}{k} F_{\text{FFN}}}_{\text{FFN Reduction}} + \underbrace{\frac{\mathcal{O}_e}{M} F_{\text{gate}}}_{\text{Router Reduction}} \right) \tag{21}$$

D.4 Formal Derivation of Communication Factors

We formally derive the probabilistic broadcast factor P and the aggregation factor R used in the Main Text.

The Aggregation Factor (R). For a specific expert replicated on $\mathcal{O}_e$ workers, Ring-AllReduce requires $2(\mathcal{O}_e - 1)$ transfers of size $S_{\text{expert}}/\mathcal{O}_e$. Total data volume per expert: $\frac{2(\mathcal{O}_e-1)}{\mathcal{O}_e} S_{\text{expert}}$. This confirms the ring factor $R(\mathcal{O}_e) = \frac{2(\mathcal{O}_e-1)}{\mathcal{O}_e}$ used in Eq. (5). Note that for disjoint experts ($\mathcal{O}_e = 1$), $R = 0$.

The Broadcast Factor (P). Under random placement with reshuffling frequency γ, we model the set difference between expert assignments S_t and S_{t+1}. Assuming uniform random assignment from N_e experts:

- Probability of "Hit" (Expert $e \in S_{t+1}$ is in S_t): $p_{\text{hit}} = \frac{|S_t|}{N_e} = \frac{\mathcal{O}_e}{M}$.
- Probability of "Miss" (Broadcast required): $p_{\text{miss}} = 1 - \frac{\mathcal{O}_e}{M}$.

The expected broadcast volume is proportional to the miss probability scaled by the reshuffling frequency γ:

$$P(\mathcal{O}_e, M) = \gamma \left(1 - \frac{\mathcal{O}_e}{M}\right) \tag{22}$$

This derivation highlights the convexity of the cost function: P maximizes when $\mathcal{O}_e$ is small (disjoint), which is exactly where R is minimized. This necessitates $\gamma \ll 1$ (infrequent reshuffling) to make the disjoint regime viable.

D.5 Optimization Surface and "Sweet Spot" Analysis

We analyze the optimization surface of the Master Equation (12) to identify the efficient "Sweet Spot" for training. We classify the operational space into four regions based on the variables $\mathcal{O}_e$ and K.

1. **Region 1: Comm-Bound (High $\mathcal{O}_e$, Low K).** Approximating DDP. t_{comm} dominates due to the full ring factor $R \approx 2$. Throughput is throttled by B_{eff}.
2. **Region 2: Memory-Constrained (High $\mathcal{O}_e$, High K).** While communication is amortized by K, the high $\mathcal{O}_e$ prevents memory savings. Local batch size B_l is constrained, preventing the "dilution" of communication costs (see Memory-Time Coupling in Main Text).
3. **Region 3: The FoMoE Sweet Spot (Low $\mathcal{O}_e$, High K).** Minimizing $\mathcal{O}_e$ drives $R \to 0$ and maximizes permissible B_l. This maximizes the Communication-to-Compute Ratio (CCR), enabling high throughput on low-bandwidth WANs. This is the optimal operating point.
4. **Region 4: Placement Collapse (High γ, Low $\mathcal{O}_e$).** If reshuffling is too frequent ($\gamma \approx 1$), the Broadcast factor P dominates t_{comm}, negating the benefits of disjoint aggregation. The system must operate with $\gamma \ll 1/t_{\text{comm}}$.

D.6 Model Falsifiability

The theoretical models proposed here make specific, falsifiable predictions that can be validated empirically.

Prediction 1: Linearity of Broadcast Cost. We predict that for a fixed M, the broadcast volume during reshuffling will follow the curve $y = C \cdot (1 - \frac{\mathcal{O}_e}{M})$.

Prediction 2: Memory-Throughput Inverse Scaling. We predict that the maximum throughput (Tokens/Sec) scales super-linearly with $1/\mathcal{O}_e$ due to the compounding effects of reduced payload size and increased local batch size B_l.

E ADDITIONAL RESULTS

Table 3: Evaluation for FoMoE with LocalAdamW and DiLoCo across different configurations of workers M and synchronization frequencies K.

Table 4: LocalAdamW

	K		
M	32	64	128
2	105.03 ± 14.67	104.70 ± 14.80	106.91 ± 14.99
4	103.91 ± 14.43	104.47 ± 14.74	111.89 ± 15.57
8	106.30 ± 14.92	112.99 ± 15.71	159.71 ± 22.41

Table 5: DiLoCo

	k		
M	32	64	128
2	118.01 ± 16.06	117.94 ± 15.85	507.21 ± 74.84
4	137.27 ± 18.35	124.17 ± 16.43	420.49 ± 65.81
8	145.64 ± 19.69	133.37 ± 17.77	647.11 ± 94.47

Table 6: Evaluation for FoMoE with LocalAdamW and DiLoCo across different configurations of workers M, synchronization frequencies K, overlapping factors O_e and placement strategies.

				O_e			
Strategy	K	Placement Strategy	M	1	2	4	8
LocalAdamW	32	Fixed	2	111.45 ± 15.52	105.03 ± 14.67	-	-
			4	121.42 ± 16.55	112.00 ± 15.45	103.91 ± 14.43	-
			8	135.74 ± 18.57	122.49 ± 17.16	114.69 ± 16.08	106.91 ± 14.99
		Random	2	116.43 ± 16.06	105.03 ± 14.67	-	-
			4	124.47 ± 17.03	113.65 ± 15.81	103.91 ± 14.43	-
			8	140.37 ± 19.07	124.56 ± 17.32	115.48 ± 15.91	106.30 ± 14.92
	64	Fixed	2	112.38 ± 15.53	104.70 ± 14.80	-	-
			4	130.79 ± 17.67	119.53 ± 16.34	104.47 ± 14.74	-
			8	141.48 ± 19.31	132.39 ± 18.30	121.36 ± 17.02	112.99 ± 15.71
		Random	2	119.73 ± 16.79	104.70 ± 14.80	-	-
			4	129.74 ± 17.72	114.88 ± 15.88	104.47 ± 14.74	-
			8	145.92 ± 19.68	131.02 ± 18.31	121.04 ± 16.54	112.99 ± 15.71
	128	Fixed	2	120.71 ± 16.53	106.91 ± 14.99	-	-
			4	143.72 ± 19.07	123.50 ± 16.91	111.89 ± 15.57	-
			8	181.07 ± 23.70	156.84 ± 21.39	138.88 ± 19.34	159.71 ± 22.41
		Random	2	128.38 ± 17.88	106.91 ± 14.99	-	-
			4	148.62 ± 19.81	125.36 ± 17.15	111.89 ± 15.57	-
			8	195.09 ± 26.19	146.16 ± 19.65	131.86 ± 18.16	159.71 ± 22.41
DiLoCo	32	Fixed	2	131.69 ± 17.99	118.01 ± 16.06	-	-
			4	142.94 ± 18.82	130.58 ± 17.58	137.27 ± 18.35	-
			8	168.39 ± 21.84	149.95 ± 20.03	145.17 ± 19.42	145.64 ± 19.69
		Random	2	142.38 ± 18.67	118.01 ± 16.06	-	-
			4	157.49 ± 20.78	136.01 ± 18.03	137.27 ± 18.35	-
			8	170.65 ± 22.94	180.43 ± 24.33	160.62 ± 21.63	145.64 ± 19.69
	64	Fixed	2	144.15 ± 18.66	117.94 ± 15.85	-	-
			4	154.73 ± 20.29	142.25 ± 18.41	124.17 ± 16.43	-
			8	187.66 ± 23.94	158.91 ± 20.59	149.30 ± 19.85	133.37 ± 17.77
		Random	2	152.42 ± 19.27	117.94 ± 15.85	-	-
			4	155.17 ± 19.95	136.46 ± 17.82	124.17 ± 16.43	-
			8	182.99 ± 23.34	158.05 ± 20.29	144.43 ± 19.23	133.37 ± 17.77
	128	Fixed	2	289.86 ± 41.29	507.21 ± 74.84	-	-
			4	342.25 ± 47.39	383.65 ± 57.42	420.49 ± 65.81	-
			8	531.75 ± 75.05	384.71 ± 54.90	748.52 ± 105.02	647.11 ± 94.47
		Random	2	403.32 ± 59.70	507.21 ± 74.84	-	-
			4	282.21 ± 40.20	430.88 ± 63.02	420.49 ± 65.81	-
			8	533.30 ± 71.11	314.18 ± 44.39	424.59 ± 65.96	647.11 ± 94.47

F Reproducibility Details

This appendix expands the compact experimental description in Sec. 5. The entries below report settings that are needed to reproduce the experiments.

Table 7: Reproducibility summary for the experimental setting. Values are grouped by the setting they reproduce.

Item	Reproducibility detail
Model configurations	Five scales are used: Small Proxy (54M), Medium (150M), Large (2B), XL (13B), and XXL (100B). The larger configurations scale width and depth from the 54M proxy while keeping $d_{head} = 64$, expansion factor 4, expert hidden size 128, and active expert ratio 0.25.
Tokenization and context	Text is tokenized with `openai/gpt-oss-120b`; the tokenizer vocabulary size is 200,019. All reported training configurations use sequence length 2048.
Training corpus	The pre-training mixture uses `HuggingFaceTB/smollm-corpus` subsets `fineweb-edu-dedup` and `python-edu` in a 90/10 stream proportion.
Evaluation data	Evaluation uses the C4 `en` validation split. The reported standard deviations are computed across runs with different seeds.
Batch and budget	Global batch size is 1024 sequences. For $M \in \{2, 4, 8\}$ workers, local batch size is $1024/M$. The proxy training budget targets 20 tokens per parameter; after synchronization-boundary rounding, this corresponds to 640 optimizer batches, 160 warmup batches, and approximately 1.34B realized tokens for the 54M proxy.
Optimization	Local training uses `AdamW` with weight decay 0, betas $(0.9, 0.95)$, and $\epsilon = 10^{-8}$. The proxy learning-rate sweep covers $\{0.01, 0.01144714, 0.01310371, 0.015\}$ before selecting transferred rates for the main sweeps. `DiLoCo`/Nesterov uses server learning rate 0.7 and momentum 0.9; `LocalAdamW` synchronizes first and second optimizer moments.
Sweep grid	The main grid uses $K \in \{32, 64, 128\}$ local steps, $M \in \{2, 4, 8\}$ workers, seeds $\{42, 123, 456\}$, and full participation. Full-replica `MoE` baselines use $\mathcal{O}_e = M$; fixed and random placement sweeps use the valid lower powers of two for each M; skip-token sweeps additionally include full overlap and set `skip_tokens=True`.
Hardware reporting	Experiments were run on NVIDIA GPU cluster nodes.
WAN parameters	The system analysis uses controlled `WAN` parameters of 1.00 Gbps and 10.00 Gbps with 50.00 ms latency, no time-varying jitter, no asymmetric bandwidth, and no persistent stragglers.

G TRAINING ALGORITHM

Algorithm 2: FoMoE- Simplified View

Require: **Model Partition and Groups**
$\theta = \{\theta_{\text{dense}}, \theta_{\text{sparse}}\}$ — params partitioned into dense and sparse components
$\mathcal{W}_{\text{all}} = \{1, \dots, M\}$ — set of all workers
Require: **Hyper-parameters**
θ_0— initial params
$\beta_1, \beta_2 \in [0, 1)$ — decay rates AdamW momenta
`LocalOptimization` — update model parameters locally with AdamW and skip tokens
`OuterOpt` — update aggregated params using an outer optimizer (e.g., Adam)
Ensure: $\theta^T,\ u^{T-1},\ v^{T-1}$
for each worker $m \in \mathcal{W}_{\text{all}}$: initialize $\theta^0, \{(u_m^{-1}, v_m^{-1})\}_{m=1}^M$ — initialize random initial model and all local momenta.
for $t = 0, \dots, T-1$ **do**
  T synchronization rounds
  1. Global Calculation (happens once)
  $\mathcal{S} \leftarrow$ `PlacementStrategy`$(\theta^t_{\text{sparse}}, O_e, \text{Placement})$ — generate allocation set $\mathcal{S} = \{s_0, \dots, s_{M-1}\}$
  2. Vectorized / Simultaneous Assignment
  $\theta^t_{m,\text{sparse}} \leftarrow s_m, \quad \forall s_m \in \mathcal{S}$ — simultaneous distribution of sparse subsets
  $\theta^t_{m,\text{dense}} \leftarrow \theta^t_{\text{dense}}, \quad \forall m$ — broadcast shared dense parameters
  3. Model Compilation
  $\theta^t_m \leftarrow \{\theta^t_{m,\text{dense}}, \theta^t_{m,\text{sparse}}\}$ — broadcast specialized models to workers
  for all workers $m = 0, \dots, M-1$ **in parallel do**
    for $k = 0, \dots K-1$ **do**
      K local training steps
      `LocalOptimization`(θ^t_m) — AdamW optimization with skip tokens implementation
    end for
    $\Delta^{t+1} \leftarrow$`Ring All-Reduce`$(\{\theta^t_m\}_{m=1}^M)$ — aggregate dense parameters globally and sparse parameters within expert replica groups
    $\theta^{t+1} \leftarrow$`OuterOpt`(Δ^{t+1}) — perform outer optimization procedure
  end for
end for

The execution protocol proceeds in three phases per synchronization round (T). First, the `PlacementStrategy` generates a global allocation set $\mathcal{S}$, mapping expert subsets to workers according to the overlap factor $\mathcal{O}_e$; this step supports both static partitioning and dynamic reshuffling. Second, the system executes a vectorized assignment where dense parameters are broadcast globally, while sparse experts are routed exclusively to their assigned hosts. Finally, workers perform K steps of `LocalOptimization` in parallel. Crucially, this phase utilizes a 'skip-token' mechanism to bypass computations for non-resident experts, ensuring the computational savings of partial replication are realized. The round concludes with sparse aggregation, where gradients are synchronized only among the $\mathcal{O}_e$ replicas via `Ring All-Reduce` before the global state is updated.