EXO Gym: A Simulation Environment for Low-Bandwidth Training

Seth Howes [1] **Matt Beton** [1] **Mohamed Baioumy** [1] **Alex Cheema** [1] **Matt Reed** [2]

Abstract

Traditional algorithms for training multi-billion parameter models require clusters of GPUs connected via proprietary high-bandwidth networking equipment. Modern low-bandwidth training algorithms such as DiLoCo and SPARTA promise to remove this bandwidth constraint. However, running these experiments currently demands multi-node hardware and complex orchestration. We introduce EXO Gym, an open-source library that emulates up to M virtual workers on N physical accelerators, letting researchers prototype and benchmark distributed-training strategies from a single workstation. Communication behaviour is encapsulated in modular `Strategy` classes, so new optimizers, sparsity schedules or compression schemes can be expressed in a few lines of code and evaluated with full telemetry (loss, wall-clock, GPU utilization, bytes transferred). In experiments, EXO Gym reproduces published DiLoCo scaling on language models, extends the algorithm to convolutional networks, and enables a rapid sweep over SPARTA sparsity rates that would cost weeks on cloud resources. By collapsing the infrastructure barrier, EXO Gym puts exploratory distributed training within reach of small teams and paves the way for broader, faster progress in open-source AI.

1. Introduction

Open-source software forms the bedrock of modern computing. Because anyone can inspect and improve the code, defects get fixed fast and new features appear quickly. That process often yields systems that are more robust and secure than proprietary alternatives, with the Linux kernel and the Apache HTTP server acting as prime examples. For the same reason, firms such as Meta and Google have opened key internal projects - PyTorch (Paszke et al., 2019) and Kubernetes (Authors, 2014) - to the public, turning private prototypes into global infrastructure.

Artificial intelligence research has not followed the same path. At the time of writing, the top 5 strongest models on GPQA Diamond and SWE Bench (complex benchmarks for graduate level science understanding and software engineering tasks respectively) are all closed source (Vellum AI, 2025). Their creators are large companies that can afford to train at vast scale. Parameter counts, data volume, and inference-time compute all grow together. Increasing scale is almost certainly a necessary condition in continuing to create more and more intelligent systems (Kaplan et al., 2020). This is because general methods, trained using large amounts of compute, inevitably outperform hand-crafted methods (Sutton, 2019). The infrastructure required to meet this scale of computation demands high-end GPUs (almost exclusively from NVIDIA) and specialized networking gear (NVLink and InfiniBand). Building a frontier-class cluster now costs billions of dollars. Smaller groups without access to these resources struggle to reproduce state-of-the-art, or advance the frontier of AI research.

Open-source AI has fundamental differences compared to open-source software. In open-source software, the required infrastructure is often limited to having a functioning computer. However, in AI training any model remotely close to the frontier requires a large amount of highly connected GPUs. Even when training code is open-source, the compute budget needed to run it shuts out independent researchers. Good ideas die because their authors cannot test them. Due to this large barrier to entry, open-source AI is dominated by a handful of small labs who usually release their models as open-weight. Meta's Llama (Touvron et al., 2023) and Alibaba's Qwen (Bai et al., 2023) are recent examples of this.

The aggregate raw compute required to train ever-larger models already exists, but it is scattered across hundreds of millions of edge devices instead of being concentrated in data-centre GPU clusters. Consumer hardware is becoming astonishingly powerful. For example, Apple's 2025 Mac Studio with the M3 Ultra chip can be configured with a 32-core CPU, an 80-core GPU and up to 512 GB of unified memory-specifications that rival a small supercomputer just

[1]EXO Labs, London, United Kingdom [2]Stanford University, Palo Alto, USA. Correspondence to: Matt Beton <matt@exolabs.net>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

a few years ago. Yet these devices rarely share a high-bandwidth, low-latency interconnect; most connect only through the public Internet. The typical fixed-broadband connection in the United States delivers about 200 Mb/s (0.025 GB/s), whereas NVIDIA's current NVLink 5.0 fabric offers up to 900 GB/s of peer-to-peer bandwidth per GPU - a gap of roughly 35,000×. This bandwidth gulf makes existing distributed-training schemes such as Distributed Data Parallel (Li et al., 2020) impractical on edge devices, because they rely on sharing multi-gigabyte gradient tensors between accelerators every training step.

Recent work shows that high bandwidth is not always essential. Algorithms such as DiLoCo (Douillard et al., 2023), SPARTA (Beton et al., 2025), Streaming DiLoCo (Douillard et al., 2025), and DeMo (Peng et al., 2024) cut the communication load by orders of magnitude. DiLoCo has been shown to scale succesfully to 10-billion-parameter language models (Jaghouar et al., 2024). Yet many open questions remain: can these methods be used to train architectures other than language models, such as convolutional neural nets, diffusion models, or protein language models? If so, which optimizer perform best in these cases? Can DiLoCo be combined with different compression methods? Are there more effective ways to perform the averaging step? Experimentation in this domain is scarce because setting up multi-node tests is tedious and expensive. A "Gym-for-distributed-training" - a simulator and launcher that lets you prototype low-bandwidth algorithms - could replicate the impact OpenAI Gym (Brockman et al., 2016) had on RL and catalyse the future breakthroughs (in much the same way Gym was instrumental in the creation of proximal policy optimization (Schulman et al., 2017)).

Our new library, EXO Gym, makes it easy to test low-bandwidth training algorithms without needing to provision expensive and complex infrastructure. It lets users emulate M virtual nodes on N real devices, where $M \geq N$. For example, 4 nodes could be emulated on a single Mac Studio, or 24 nodes could be emulated using 8 Nvidia RTX 4090 GPUs. Researchers can prototype low-bandwidth ideas without hand-crafting cluster scripts or renting cloud machines for weeks. By lowering cost and complexity, EXO Gym puts distributed training within reach of small teams and, we hope, accelerates the next wave of open-source AI innovation.

The source code is available at `https://github.com/exo-explore/gym`.

2. Low-Bandwidth Training

Over the past year, a growing body of low-communication algorithms shows that you can train large models without requiring specialized high-bandwidth interconnects between all accelerators in a cluster. Prior to this, training over a low-bandwidth network using established methods for training multi-billion parameter models was impractical, as demonstrated in the motivating example in Table 1.

DiLoCo (Douillard et al., 2023) revisits the classical federated averaging idea but equips it with a large inner loop (AdamW (Kingma & Ba, 2017)) and a momentum-based outer loop (Nesterov SGD (Nesterov, 1983)). The DiLoCo paper demonstrates an 8-worker training run on the C4 (Dodge et al., 2021) corpus achieving equal performance to data-parallel training, whilst performing $500\times$ lower communication.

Following on from this, Prime Intellect introduced OpenDiLoCo, an open-source implementation of DiLoCo for training LLMs over the internet. In real-world tests that stretched across two continents and three cloud providers, this ensured 90–95 % GPU utilization and cleanly scaled to billion-parameter models, proving that DiLoCo's communication pattern survives hostile internet latencies.

SPARTA (Beton et al., 2025) showed than you can extend the synchronization interval even further by each worker randomly averaging a small proportion (e.g. 0.05%) of its weights with its peers. Because SPARTA communicates parameters instead of gradients, the exchange can be performed totally asynchronously. Combined with DiLoCo it acts as a regularizer, enabling a 100× longer DiLoCo interval (H = 10,000) with no wall-clock penalty and even allowing higher learning-rates.

The follow-on paper Streaming DiLoCo (Douillard et al., 2025) removes the last high-bandwidth bursts by (i) synchronizing one fragment of the model at a time, (ii) overlapping that transfer with continued compute, and (iii) quantizing the outer gradients to 4-bit. Together these tweaks preserve task performance while driving inter-worker bandwidth down by two orders of magnitude relative to data-parallel baselines.

DeMo attacks the same problem from an optimizer standpoint. By letting the slow components of the momentum vector drift locally and only synchronizing a compact "fast" Fourier/DCT signature each step, it slashes traffic several orders of magnitude yet still matches or beats AdamW on large-scale LLM pre-training - all without assuming any particular topology.

3. EXO Gym

EXO Gym is a lightweight, open-source simulator for conducting research into low-bandwidth training. It allows for spawning multiple *virtual* workers on a single physical device, letting you prototype and benchmark new communication schemes without scaling out physically. A single `Trainer` object augments any `torch.nn.Module` and

Table 1. 7B model training time and viability for different networking configurations on a training corpus equivalent to 1000 books.

Scenario	Training Time	Viability
Single GPU	1,000 minutes (16.7 hours)	Baseline – predictable sequential processing
2 GPUs (Data Center)	505 minutes (8.4 hours)	Excellent – nearly perfect 2× speed-up due to fast network
2 GPUs (Internet)	10,500 minutes (7.3 days)	Impractical – slow network makes synchronization extremely costly
DiLoCo (H=500)	520 minutes (8.7 hours)	Game-changing - matches data center performance over slow networks

```

train_data, val_data = ...
model = ...

trainer = LocalTrainer(model, train_data, val_data)

from exogym.strategy.diloco import DiLoCoStrategy
strategy = DiLoCoStrategy(
    optim_spec='adamw'
    H=200,
)

trainer.fit(
    num_epochs=1,
    max_steps=5000,
    strategy=strategy,
    num_nodes=4,
    device='mps',
)
```

Listing 1. EXO Gym makes running distributed training experiments simple.

`torch.utils.Dataset` with a `train` method that consumes a `Strategy` class. Communication behavior is fully configurable through these swappable `Strategy` classes, which specify the optimizer, synchronization rule, cadence, and any sparsity or compression applied to gradient exchanges.

The code in Listing 1 demonstrates how `Trainer` is used to perform a distributed training simulation on a regular torch model and dataset, and Listing 2 demonstrates how the SPARTA strategy is implemented in EXO Gym.

EXO Gym reduces the iteration speed and cost for researchers investigating novel low-bandwidth training algorithms. This includes:

- **Algorithm designers seeking rapid iteration.** Swapping in a new synchronization rule, optimizer, or compression schedule takes only a few lines of code, enabling fast A/B testing across model families.
- **Experimentalists exploring configuration space.** Because virtual nodes are cheap to spawn, users can sweep over parameters—such as the SPARTA sparsity rate p or DiLoCo's inner-loop length H - identifying optimal configurations that would be prohibitively costly to evaluate on real hardware.

4. Example Use-Cases

In this section we explore some of the early experiments we have run using EXO Gym:

4.1. SPARTA on Multiple Nodes

We have used EXO Gym to evaluate Sparse Parameter Averaging (SPARTA). This method consists of exchanging a small fraction of model parameters among workers at each timestep, with the full algorithm shown in Algorithm 1. Figure 1 shows how we can use EXO Gym to effortlessly simulate the effect on training performance of varying the proportion of parameters exchanged p on 4 nodes, without needing a physical multi-accelerator setup.

Algorithm 1 Sparse Parameter Averaging (SPARTA)

Require: Data shards $\{D_1, \ldots, D_k\}$, frequency $H \in \mathbb{R}$, sparsity rate $p \in \mathbb{R}$
for step $t = 1, \ldots, T$ **do**
  Sample indices $\mathcal{P}^{(t)}$ according to sampling strategy
  for worker $k = 1, \ldots, K$ **in parallel do**
    $\theta_k^{(t)} \leftarrow \texttt{AdamW}(\theta_k^{(t-1)}, D_k^{(t)})$
    $\texttt{AllReduce}(\tilde{\theta}_j^{(t)} \leftarrow \theta_{k,j}^{(t)}, j \in \mathcal{P}^{(t)})$
    Update $\theta_k^{(t)}$ with $\tilde{\theta}^{(t)}$ at the sampled indices
  end for
end for $\frac{1}{K}\sum_{k=1}^{K} \theta_k^{(T)}$

4.2. Training a CNN Using DiLoCo

EXO Gym also supports running training experiments with DiLoCo. While the original DiLoCo study focused exclusively on language models, we employed EXO Gym to benchmark its performance on convolutional neural networks (CNNs). Figure 2 shows that on a single node, reaching an evaluation loss below 2.5 requires approximately 6,000 steps; with two nodes, we achieve the same loss in under 4,000 steps.

5. Future Directions

Future work will explore:

Pipeline parallel. Beyond data-parallel training, we will implement pipeline and tensor parallel strategies to mirror

```python

class SPARTAGradient(Strategy):
    def step(self):
        self.optim.step()

        if self.config.num_nodes > 1:
            with torch.no_grad():
                for name, param in self.model.named_parameters():
                    indices_mask = self.index_selector.get_indices(param, self.iteration)

                    broadcast(indices_mask, src=0)
                    sparse_data = param.data[indices_mask]
                    all_reduce(sparse_data, op=dist.ReduceOp.SUM)
                    sparse_data /= dist.get_world_size()

                    param.masked_scatter_(indices_mask, sparse_data)

        self.iteration += 1
        super().step()
```

Listing 2. Defining a Strategy class in EXO Gym.

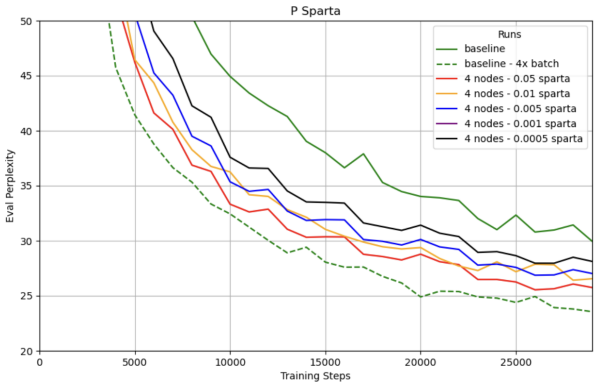

Figure 1. Training a language model using SPARTA on 4 nodes.

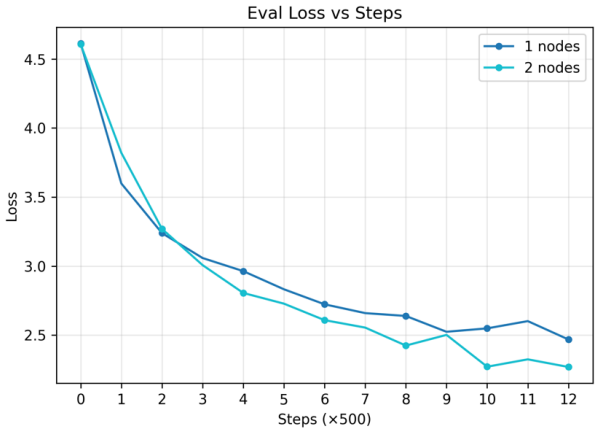

Figure 2. Training a CNN using DiLoCo simulated across two nodes.

the parallelism modes used in large-scale, state-of-the-art training runs.

Communication modeling. We aim to introduce configurable network simulators that capture variable latencies and bandwidths - e.g., high-bandwidth islands of accelerators linked by lower-bandwidth interconnects - to evaluate algorithm resilience under realistic network conditions.

Fault simulation. To better emulate deployments on heterogeneous, consumer-grade devices, we will simulate common failure modes (dropped packets, unexpected node outages, dynamic performance drops), enabling robust evaluation of distributed algorithms in the face of real-world faults.

6. Conclusion

Recent advancements in low-bandwidth training methods are breaking the constraint of needing massive clusters of expensive accelerators to train frontier models. However, the development of these novel algorithms is restricted to a select few labs with the requisite infrastructure required for empirical testing. EXO Gym is a simulation library that allows people to test novel low-bandwidth training algorithms. Internally, we have validated this tool by using it as the basis for all of our experimental research, such as Beton et al. (2025). We hope that by making this resource open source, researchers will be able to design and test novel algorithms without being restricted by lack of access to compute.

Impact Statement

"This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here."

References

Authors, K. Kubernetes: Production-grade container orchestration. https://github.com/kubernetes/kubernetes, 2014. Accessed: 2025-05-26.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

Beton, M., Howes, S., Cheema, A., and Baioumy, M. Improving the efficiency of distributed training using sparse parameter averaging. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*, 2025. URL https://openreview.net/forum?id=stFPf3gzq1.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

Dodge, J., Sap, M., Marasović, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M., and Gardner, M. Documenting large webtext corpora: A case study on the colossal clean crawled corpus, 2021. URL https://arxiv.org/abs/2104.08758.

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, 2023.

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, 2025. URL https://arxiv.org/abs/2501.18512.

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

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, 2020. URL https://arxiv.org/abs/2001.08361.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., and Chintala, S. Pytorch distributed: Experiences on accelerating data parallel training, 2020. URL https://arxiv.org/abs/2006.15704.

Nesterov, Y. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, $27(2)$: $372 - -376, 1983$.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Peng, B., Quesnelle, J., and Kingma, D. P. Demo: Decoupled momentum optimization, 2024. URL https://arxiv.org/abs/2411.19870.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Sutton, R. S. The bitter lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html, 2019. Accessed: 2025-05-26.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Vellum AI. Llm leaderboard. https://web.archive.org/web/20250526105950/https://www.vellum.ai/llm-leaderboard, 2025. Accessed: 2025-05-26.