AUTOLOCO: COMMUNICATION EFFICIENT DISTRIBUTED LLM TRAINING VIA ADAPTIVE SYNCHRONIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The pre-training of Large Language Models (LLMs) is increasingly conducted across multiple data centers. As training scales to a larger number of accelerators, the fraction of time spent on computation decreases, while the fraction spent on communication increases. Therefore, frequent synchronization becomes a growing bottleneck. Local update methods reduce this cost by allowing workers to perform several optimizer steps between synchronizations. Most local update methods set the number of local optimizer steps between synchronizations before training and keep this interval fixed throughout the run. However, the best interval can change during the entire train process. If the interval and optimizer are adapted to the current training state, the communication frequency is reduced while maintaining the training performance. In this work, we introduce AutoLoCo, an adaptive training framework to reduce communication in LLM training. It adapts the local interval using scalar training statistics and corrects each outer update. Our method is motivated by two observations: 1) the appropriate local interval varies across training stages, and 2) changing the number of inner steps per interval creates a mismatch with an unchanged outer optimizer, requiring a correction to the outer update. We optimize this mismatch by correction of the outer optimizer for the momentum and the learning rate using the accumulated inner learning rate. Our experiments under communication constraints demonstrate that AutoLoCo reduces communication frequency by 27% relative to DiLoCo while maintaining training performance.

1 INTRODUCTION

Developing high-capability Large Language Models (LLMs) requires substantial data and computation, making training efficiency a central concern (Dubey et al., 2024). LLMs' training increasingly relies on distributed accelerators, while useful compute may be spread across datacenters, institutions, and regions. These resources cannot always communicate as one tightly integrated cluster. Each synchronization incurs blocking latency and transfers updates for the entire model over links with limited bandwidth. Communication becomes a central constraint when training spans geographically distributed resources (Douillard et al., 2023; Jaghouar et al., 2024).

Existing methods reduce either the amount or the frequency of communication. DiLoCo follows the second approach. Each worker performs H local AdamW steps before a Nesterov outer optimizer updates the shared model from averaged pseudo-gradients (Douillard et al., 2023). By synchronizing only periodically, DiLoCo enables training across loosely connected compute resources without requiring frequent model-wide communication. However, standard DiLoCo fixes the communication interval throughout training. A short interval can incur unnecessary synchronization, when a long interval can allow excessive disagreement between local updates. Adaptive Periodic Averaging, STL-SGD, and the Quadratic Synchronization Rule adapt synchronization periods using worker disagreement or learning-rate schedules (Jiang & Agrawal, 2020; Shen et al., 2021; Gu et al., 2024). Related work also highlights interactions among the local interval, outer momentum, and outer learning rate (Kallusky et al., 2025). The remaining design question is how to coordinate online interval selection with outer updates adapted to each interval in DiLoCo.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The main challenge is to reduce synchronization while preserving training quality. As the inner learning rate decays and individual updates become smaller, later training stages tolerate longer local intervals. But changing communication interval also changes the local trajectory, accumulated inner learning rate, pseudo-gradient scale, and amount of local optimization represented by one outer update (Douillard et al., 2023; Kallusky et al., 2025). A fixed outer optimizer then become mismatched when communication interval varies. We introduce AutoLoCo, which couples online interval selection with outer optimization adapted to each interval. The controller uses local-drift energy and aggregation coherence from completed intervals to select a token-aligned horizon, which is mapped to local steps. The Outer Optimizer Correction adjusts outer momentum and learning rate by comparing the accumulated inner learning rates of the selected and base intervals. Both intervals are evaluated from the same point in the learning rate schedule. AutoLoCo reduces synchronization rounds by 27% compared with DiLoCo while achieving a lower mean training loss. The only additional communication consists of scalar summaries and decision metadata.

Our contributions are summarized as follows: **(1)** We introduce an online controller that adapts DiLoCo's local interval from interval decision. It operates in token-aligned units and maps each selected token budget to local steps, which significantly reduces communication frequency and overhead. **(2)** We introduce an outer optimizer correction for changing local horizons. It adjusts outer momentum and learning rate according to the accumulated inner learning rate represented by each interval. **(3)** We evaluate AutoLoCo across multiple model scales and communication settings, and open-source the experimental code and configurations to promote the advancement of LLMs.

2 Related Work

Reducing communication payload. Some distributed training works focus on reducing the number of bytes exchanged during each synchronization step through techniques such as quantization, sparsification with error feedback, and structured compression. QSGD and 1-bit Adam reduce the precision of communicated updates; Deep Gradient Compression and error feedback transmit selected coordinates while retaining omitted information; PowerSGD and FetchSGD replace dense updates with low-rank or sketched representations (Alistarh et al., 2017; Tang et al., 2021; Lin et al., 2018; Karimireddy et al., 2019; Vogels et al., 2019; Rothchild et al., 2020). Qsparse-local-SGD combines local computation with quantization and sparsification, reducing both communication frequency and payload size (Basu et al., 2019). Recent methods for foundation models adapt these ideas to pretraining and fine tuning over networks with limited bandwidth. CocktailSGD combines several compression operators, MuLoCo studies the compressibility of pseudo-gradients under different inner optimizers, and Streaming DiLoCo communicates parameter subsets while overlapping transfer with local training (Wang et al., 2023; Thérien et al., 2026; Douillard et al., 2025). These methods reduce payload size or exposed communication time.

Reducing Communication frequency. Local-update methods are performed as multiple optimizer steps between global averages. FedAvg established this pattern in federated learning, and Local SGD provided convergence guarantees for periodic averaging (McMahan et al., 2017; Stich, 2019). Cooperative SGD and unified Local SGD analyses further connect local work, communication topology, data heterogeneity, and convergence (Wang & Joshi, 2021; Gorbunov et al., 2021). Post-local SGD uses a stagewise variant: training begins synchronously and switches to local updates at a predefined point (Lin et al., 2020). DiLoCo brings this structure to language model pretraining through local AdamW, averaged pseudo-gradients, and a Nesterov outer optimizer; OpenDiLoCo demonstrates decentralized execution across geographically distributed workers (Douillard et al., 2023; Jaghouar et al., 2024). HALoS extends low-frequency training to hierarchical asynchronous aggregation over slow networks, while Scaling Laws for DiLoCo studies examine model size, worker count, token budget, and optimizer settings (Kim et al., 2025; Charles et al., 2025). AutoLoCo, on the other hand, maintains the benefit of infrequent communication, and use dynamic communication interval selected after each interval from current training behavior instead of a fixed communication clock.

Inner-outer optimizer design. In local-update training, the inner optimizer determines the trajectory followed between synchronizations, and the outer optimizer determines how the resulting pseudo-gradient updates the shared model. SlowMo adds momentum on a slower time scale after several local steps, and analyses how outer learning rate, momentum, and acceleration affect Local SGD (Wang et al., 2020b; Khaled et al., 2025). SNOO examines Nesterov momentum applied to

pseudo-gradients (Kallusky et al., 2025). Federated optimization provides related server-side mechanisms. FedOpt applies adaptive optimizers to aggregated client changes, Mime uses shared optimizer statistics to preserve centralized optimizer behavior, FedExP adapts the server step size from update geometry, and FADAS responds to asynchronous staleness (Reddi et al., 2021; Karimireddy et al., 2021; Jhunjhunwala et al., 2023; Wang et al., 2024b). FedNova further shows that updates can be normalized according to the effective local progress performed before aggregation (Wang et al., 2020a). These studies show that the outer optimizer reflect the optimization process summarized by each aggregated update. AutoLoCo follows this principle in DiLoCo. We use accumulated inner learning rate to represent the optimization performed within each interval, then uses this quantity to adjust the outer momentum and learning rate when the synchronization interval changes.

Adaptive synchronization intervals. Methods that adapt synchronization intervals during training generally determine the next interval by following a predefined schedule or updating the interval from online feedback collected during the current run. Adaptive synchronization analyses and communication-efficient SGD derive conditions under which the interval can grow over training (Haddadpour et al., 2019; Spiridonoff et al., 2021). STL-SGD increases the interval when the learning rate enters a new stage, and the Quadratic Synchronization Rule makes this relationship explicit through $H_t \propto 1/\eta_t^2$ (Shen et al., 2021; Gu et al., 2024). These schedule-based policies provide simple and predictable interval trajectories aligned with the optimizer schedule. Feedback-based methods instead adapt synchronization to observations from the current run. AdaComm uses an error–runtime model, dynamic model averaging communicates when local models deviate from a reference, and Adaptive Periodic Averaging adjusts the next interval from variance across workers (Wang & Joshi, 2019; Kamp et al., 2018; Jiang & Agrawal, 2020). These studies show that synchronization frequency respond to optimizer progress. AutoLoCo extends the next interval selection algorithm and how a DiLoCo outer optimizer should interpret pseudo-gradients produced by intervals that represent different amounts of local optimization.

AutoLoCo extends interval selection method by treating synchronization timing and outer optimization as a coupled design problem. 1) We make the synchronization interval responsive to the current training state. This allows the communication frequency to adapt as the model's tolerance for longer local intervals changes over the course of training. 2) We interpret intervals of different lengths according to the amount of local optimization they represent. Longer intervals make each pseudo-gradient represent more local optimization and change its scale, which make each outer step span more inner optimization and change the effective time unit of the outer momentum and learning rate. Therefore, we combine adaptive synchronization with outer optimization adapted to each interval, which is a complementary method to compression, quantification, and asynchronous communication architectures.

3 METHOD

In this section, we introduce our method AutoLoCo, which is refined on the DiLoCo inner-outer training loop. DiLoCo performs a fixed number of local steps before averaging pseudo-gradients and applying a Nesterov outer update. We preserves this loop but replaces the fixed horizon with two adaptive components. The horizon component selects a token-aligned horizon and maps it to executable local steps, while the outer correction normalizes the pseudo-gradient and adjusts the outer momentum and learning rate. These components determine the number of local steps before each synchronization and align the outer optimizer with the amount of local optimization represented by the resulting pseudo-gradient.

3.1 ADAPTIVE INTERVAL SELECTION

Interval Evaluation. At the end of interval t, we choose the token-equivalent horizon H_{t+1}^{tok} for the next interval. The controller has the task of keeping the accepted horizon, evaluating a larger candidate, or using a smaller horizon. Let N be the number of workers, $\Delta_t^{(i)}$ the pseudo-gradient from worker i, and $\bar{\Delta}_t = N^{-1}\sum_{i=1}^{N}\Delta_t^{(i)}$ their average. We use

$$D_t = \frac{1}{N}\sum\nolimits_{i=1}^{N}\left\|\Delta_t^{(i)}\right\|_2^2, \qquad C_t = \frac{N\left\|\bar{\Delta}_t\right\|_2^2}{D_t+\epsilon}, \tag{1}$$

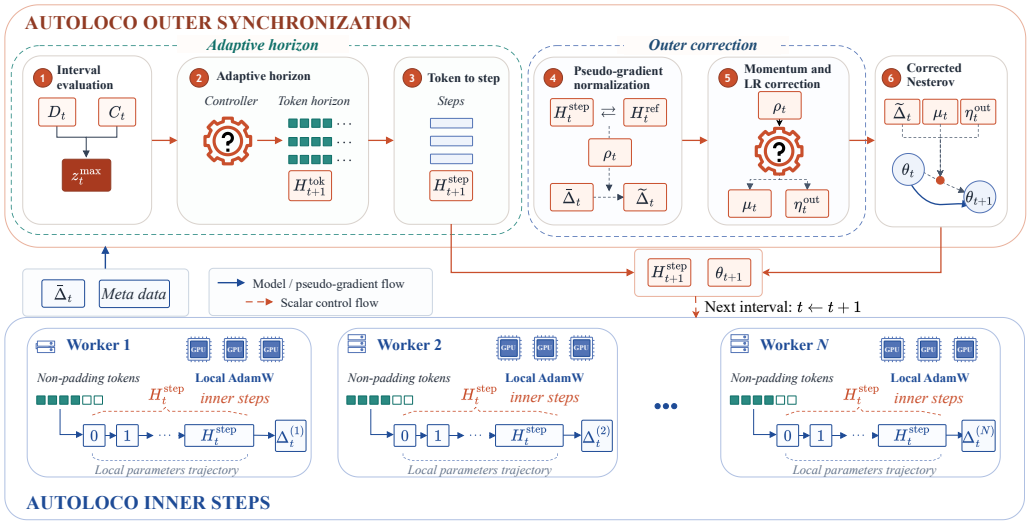


Figure 1: AutoLoCo adapts the local horizon and corrects the outer optimizer in both loops.

where $\epsilon > 0$ is a numerical constant. D_t measures the mean squared displacement from the synchronized model to local endpoints, while C_t measures how much remains aligned after averaging. They separate large but coherent local motion from movement that cancels across workers. Appendix C.1 derives the relation between D_t, C_t, and disagreement among local endpoints. We compare both quantities with an accepted-horizon reference profile estimated from the current run. The profile stores D_t^{ref}, C_t^{ref}, and Λ_t^{ref}, where $\Lambda_t^{\mathrm{ref}} = \sum_{h=1}^{H_t^{\mathrm{step}}} \eta_{t,h}$ is evaluated on the accepted-horizon interval used to update the profile. When inner learning rates vary, equal numbers of local steps correspond to different amounts of optimization. STL-SGD and the Quadratic Synchronization Rule address this by coupling the communication period to the learning-rate schedule, and FedNova normalizes updates according to effective local progress (Shen et al., 2021; Gu et al., 2024; Wang et al., 2020a). Following this principle, we use accumulated inner learning rate to normalize the drift in each interval before comparing it with the accepted-horizon profile. We write

$$\xi_t = \frac{\sum_{h=1}^{H_t^{\mathrm{step}}} \eta_{t,h}}{\Lambda_t^{\mathrm{ref}} + \epsilon}, \tag{2}$$

where H_t^{step} is the executed step horizon, h indexes its local steps, and $\eta_{t,h}$ is the inner learning rate at step h. We then write the two departures from the profile directly as

$$r_{D,t} = \log\left(\frac{\sqrt{D_t}}{\sqrt{D_t^{\mathrm{ref}}}\,\xi_t + \epsilon} + \epsilon\right), \quad r_{C,t} = -\log\left(\frac{C_t}{C_t^{\mathrm{ref}} + \epsilon} + \epsilon\right). \tag{3}$$

Here, $r_{D,t}$ measures excess drift after accounting for accumulated inner optimization. $r_{C,t}$ measures the loss of aggregation coherence relative to the accepted-horizon profile. Let $c_{D,t}$ and $c_{C,t}$ be the median residuals in the accepted-horizon histories, $s_{D,t}$ and $s_{C,t}$ be their median-absolute-deviation scales. Appendix A.1 explains how these histories are collected and updated. The controller uses

$$z_t^{\max} = \max\left\{\frac{[r_{D,t} - c_{D,t}]_+}{s_{D,t}}, \frac{[r_{C,t} - c_{C,t}]_+}{s_{C,t}}\right\}, \tag{4}$$

where $[x]_+ = \max(x, 0)$. A small $z_t^{\max}$ supports a larger candidate, reference-consistent behavior keeps the accepted horizon, persistent moderate deviation reduces it, and severe deviation reduces it immediately. We evaluate a candidate before adoption and re-estimate the reference profile after adoption or reduction. The controller combines this assessment with its current phase and horizon bounds to select H_{t+1}^{tok}. Appendix A.2 provides the assessment thresholds, confirmation procedure, and horizon acceptance or rejection rules.

Token to Step Mapping. We control local work in non-padding tokens rather than raw steps because padding, packing, and minibatch composition can change the training volume of one optimizer step. Here, $n_{t,h}$ is the global non-padding token count at local step h, and M_t is the sum of these counts over interval t. H_{base} is the recipe token horizon, and M_{base} is the median token mass of current-run bootstrap intervals executed at H_{base}. After each interval, we compute $\bar{n}_t = M_t / H_t^{\text{step}}$ and update $\hat{n}_t = \beta \hat{n}_{t-1} + (1-\beta)\bar{n}_t$, where $\hat{n}_t$ is the exponential moving average of tokens per executable step and β is its coefficient. Before rounding, the mapped step horizon is

$$H_t^{\text{step}} \approx \frac{M_{\text{base}} H_t^{\text{tok}}}{H_{\text{base}}(\hat{n}_{t-1} + \epsilon)}. \tag{5}$$

We round this value to the nearest admissible multiple of the horizon granularity q_H, using the procedure in Appendix A.3. Applying the same mapping with $H_t^{\text{tok}} = H_{\text{base}}$ gives H_t^{ref}, the executable horizon for one recipe token interval at the current token density.

3.2 Outer Optimizer Correction for Variable Local Horizons

Variable horizons change the meaning of an outer update. A longer interval alter the scale of the averaged pseudo-gradient and make one momentum update span more inner optimization. With a fixed outer optimizer, it applies the same update rule to every synchronization. Prior work on slow outer momentum, DiLoCo Nesterov updates, normalization under unequal local progress, and adaptive server optimization shares the principle: the outer update should align with the local computation that produced it (Wang et al., 2020b; Kallusky et al., 2025; Wang et al., 2020a; Reddi et al., 2021). We apply this principle across successive DiLoCo intervals. Our outer correction compares the selected interval with one base token interval at the same position in the inner learning-rate schedule, then uses the resulting ratio to normalize the pseudo-gradient, adjust momentum retention, and bound the outer-step scale. A detailed description of this mismatch appears in Appendix B.1.

Interval ratio and pseudo-gradient normalization. For interval t, H_t^{step} is the selected step horizon, and H_t^{ref} is the horizon corresponding to one base token interval at the current token density. Let h index local steps, let $\eta_{t,h}$ be the inner learning rate at step h, and let $\epsilon > 0$ be a numerical constant. We preview both horizons from the same scheduler position and define

$$\rho_t = \max\left(\frac{\sum_{h=1}^{H_t^{\text{step}}} \eta_{t,h}}{\sum_{h=1}^{H_t^{\text{ref}}} \eta_{t,h} + \epsilon}, 1\right). \tag{6}$$

The interval ratio ρ_t measures the accumulated inner learning rate of the selected interval in base-interval units. After workers complete the interval and average their pseudo-gradients, we set

$$\widetilde{\Delta}_t = \frac{\bar{\Delta}_t}{\rho_t}, \tag{7}$$

where $\bar{\Delta}_t$ is the averaged pseudo-gradient and $\widetilde{\Delta}_t$ is the normalized outer-optimizer input. At $\rho_t = 1$, the pseudo-gradient remains unchanged. For $\rho_t > 1$, the normalization expresses it on the scale of one base interval. Appendix B.2 details the scheduler-aligned comparison, and Appendix B.3 describes how the normalization affects the DiLoCo update. Appendix C.2 provides a local displacement interpretation of this ratio and bounds the approximation error. For experiment settings, C4 pretraining does not use normalization, while 8B fine-tuning uses normalization. Section 4.3 discusses this difference and presents the corresponding ablation results.

Momentum and outer-step adjustment. Normalization corrects the optimizer input, but a longer interval also changes how long the incoming momentum state persists in units of inner optimization. Let μ_{base} denote the base outer momentum. We first set $\mu_t = \mu_{\text{base}}^{\rho_t}$, then constrain the interval-adjusted momentum μ_t to the configured range $[\mu_{\min}, \mu_{\max}]$. We next set $\kappa_t = \min(\rho_t, \kappa_{\max})$, where κ_t is the bounded outer-step scale and $\kappa_{\max}$ is its upper limit. Given the base outer learning rate $\eta_{\text{base}}^{\text{out}}$, we use

$$\eta_t^{\text{out}} = \eta_{\text{base}}^{\text{out}} \kappa_t \frac{1-\mu_t}{1-\mu_{\text{base}}}, \tag{8}$$

where η_t^{out} is the outer learning rate applied to interval t. The same ρ_t controls pseudo-gradient normalization, momentum retention, and bounded scaling of the outer step. The derivation is given in Appendix C.3, and the reference behavior is described in Appendix B.4. The experiment settings are reported in Table 12.

3.3 AUTOLOCO: ADAPTIVE GRADIENT SYNCHRONIZATION MEETS OUTER OPTIMIZER CORRECTION

AutoLoCo couples synchronization timing with outer optimization through a feedback loop that operates at the interval level. At the start of interval t, the token mapper converts the token-equivalent horizon H_t^{tok}, selected from the previous completed interval, into H_t^{step}, the executable local-step horizon, and H_t^{ref}, the executable horizon corresponding to one base token interval at the current token density. These horizons determine ρ_t and the interval-adjusted outer optimizer settings before local execution. Each worker starts from θ_t, follows a local trajectory of length H_t^{step}, and contributes a pseudo-gradient. After synchronization, we normalize the averaged pseudo-gradient and apply the corrected outer update. Once this update produces θ_{t+1}, we compute D_t, C_t, and the completed token statistics. These quantities update the accepted-horizon profile and token mapper, and determine H_{t+1}^{tok}. The horizon component then sets the local trajectory. The outer correction determines how the resulting pseudo-gradient updates the synchronized model.

The two components meet in the corrected Nesterov outer update used by DiLoCo. Let v_{t-1} be the outer momentum state entering interval t, v_t its updated value, and d_t the Nesterov update direction. We compute

$$v_t = \mu_t v_{t-1} + \widetilde{\Delta}_t, \qquad d_t = \widetilde{\Delta}_t + \mu_t v_t, \qquad \theta_{t+1} = \theta_t - \eta_t^{\mathrm{out}} d_t. \tag{9}$$

Here, θ_t and θ_{t+1} are the synchronized parameters before and after the outer update. The normalized pseudo-gradient $\widetilde{\Delta}_t$ provides the optimizer input, μ_t controls momentum retention, and η_t^{out} scales the parameter update.

Algorithm 1 in Appendix B.5 traces one complete AutoLoCo interval. Appendices D.1 and D.2 give the inherited DiLoCo update and the conditions for recovering DiLoCo, respectively.

4 EXPERIMENTS

We evaluate AutoLoCo with LLaMA style configuration by 215M total parameters (Touvron et al., 2023) and Llama-3.1-8B (Dubey et al., 2024). We pre-train on C4 (Raffel et al., 2020) for 50,000 optimizer steps and match this budget across methods, following the controlled comparisons of Jaghouar et al. (2024). Our main setup uses 8 workers with batch size of 128 per worker, giving a global batch of 1,024. With a sequence length of 1,024, each step processes 1.05M input token positions. To test AutoLoCo on a larger model, we full-parameter fine-tune Llama-3.1-8B for 1 epoch on UltraChat-200k (Ding et al., 2023) with 8 and 16 workers. For the experiment of 16 workers, we set up actual distributed workers across different hosts. Both settings use a global batch of 32 and a maximum sequence length of 2,048. In the 16 workers setup, the effective throughput measured per-worker ranges from 1.26 to 1.46 Gbit/s. Appendix E reports the implementation settings, data processing and evaluation protocol. Appendix G provides implementation details and additional experimental results on combining AutoLoCo with low-precision communication.

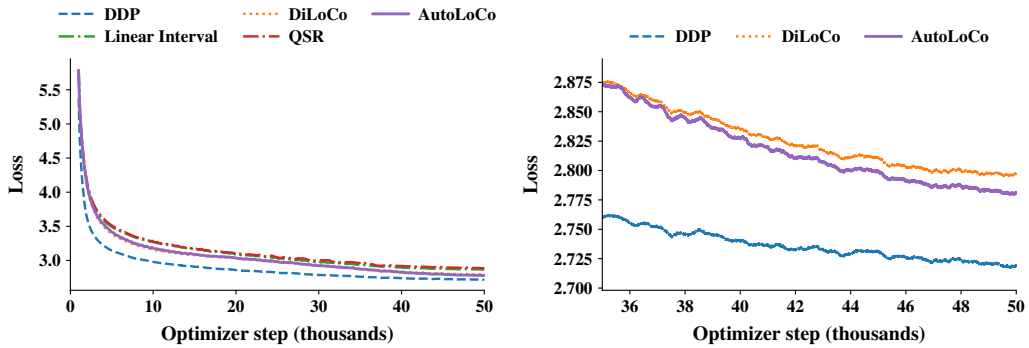
Table 1: Quality and communication cost at step 50,000. Training loss averages eight workers over the final 1,000 steps; validation evaluates the final synchronized checkpoint. Payload is measured in full-model BF16 tensors per worker. † counts gradient synchronizations; other counts are outer synchronizations. ‡ Given the differences in training settings, we retain their proxies for selecting communication intervals and adapt the methods to LLM training. Bold and underline indicate the best and second-best results, respectively, in the loss and synchronization count columns.

Method	Training loss ↓	Number of syncs ↓	Payload per sync	Validation loss ↓
DDP	**2.7227**	50,000†	1×	**2.7233**
DiLoCo	2.8004	100	1×	2.8020
Linear Interval‡	2.8706	100	1×	2.8694
QSR‡	2.8914	**41**	1×	2.8793
AutoLoCo	<u>2.7843</u>	<u>73</u>	1×	<u>2.7842</u>

4.1 Pretrain Quality and Communication Cost

We compared with two strong baselines, data-parallel training (DDP) (Li et al., 2020) and DiLoCo. In the main setup, DiLoCo follows the configuration reported in prior work. AutoLoCo retains most settings and adjusts the parameters of its adaptive components. AutoLoCo uses 73 synchronizations compared with 100 for DiLoCo, while training loss improves from 2.8004 to 2.7843 and validation loss from 2.8020 to 2.7842 (Table 1). With the same payload per synchronization, this saves 27% of the cumulative logical volume. Appendix E.1 gives the protocol, and Appendix F.2 examines the effect of the outer settings under identical intervals.

For pre-training, we also compared the Quadratic Synchronization Rule (QSR) (Gu et al., 2024) and the linearly increasing communication-interval schedule of Spiridonoff et al. (2021), which we call Linear Interval. Our local AdamW implementations of Linear Interval and QSR follow the prior works, use direct model averaging. AutoLoCo finishes with better validation loss than both. DDP remains the quality reference, reaching validation loss 2.7233 with gradient synchronization at every step. The training loss advantage over DiLoCo emerges late (Figure 2).



(a) Full trajectory (b) Final portion

Figure 2: Training loss over the full budget and during the final portion of training.

The threshold comparison reflects this change: AutoLoCo reaches 2.90 slightly later, but 2.85 and 2.80 earlier (Table 2 and Figure 4). The results in the table also show that AutoLoCo achieves a more competitive rate of loss reduction in the later stages of training than DiLoCo and the other two methods. The analysis in Appendix F.1 shows the same late advantage. The communication savings are clearer than the change in total runtime. Observed interface I/O falls by 26.5%, and accumulated synchronization boundary time by 12.9%, from 40.74 to 35.50 minutes. Our diagnostic and control accounting totals 0.304% of training time (Table 13). This includes preparation work and auxiliary diagnostics alongside the controller. Control endpoint accounting totals 2.89 MiB for the run.

Table 2: Cumulative recorded step time (hours) to first reach each threshold of the 8 worker mean loss, using a 1,000-step trailing average. A dash indicates that the threshold is not reached within 50,000 steps. The best time in each column is shown in bold.

Loss threshold	3.50	3.20	3.00	2.90	2.85	2.80
Method	Cumulative recorded step time (h) ↓					
DiLoCo	1.31	2.83	7.48	10.60	12.44	15.26
Linear Interval	1.68	4.34	9.13	12.87	–	–
QSR	1.69	4.30	9.40	14.27	–	–
AutoLoCo	1.39	3.03	7.52	10.61	**12.09**	**14.22**

4.2 Fine-Tuning Quality and Communication Cost

We fine-tune Llama-3.1-8B on UltraChat for 1 epoch with a global batch of 32. Both 8 workers and 16 workers experiments share this setting. DiLoCo follows the experimental configuration reported

in prior work. Our AutoLoCo recipe uses similar hyper parameter compared with main setup. The only difference is that we apply pseudo-gradient normalization in the 8B experiments. Section 4.3 discusses this difference and presents the corresponding ablation results. Appendices E.1 and E.3 give the training and evaluation protocol.

Table 3: UltraChat training and communication with eight workers, a global batch of 32 for 1 epoch. Training loss covers the final 1,000 steps. Logical payload represents the volume of model data exchanged at the algorithmic level. Actual traffic over physical links depends on the distributed framework and the collective communication algorithm.

Method	DDP	DiLoCo	AutoLoCo
Train loss ↓	0.8736	0.9102	**0.8759**
Syncs ↓	6,490	65	49
Logical payload (GB)			
Per sync	–	16.061	16.061
Total ↓	–	1,043.934	**786.966**
UltraChat test			
NLL ↓	0.8745	0.9112	**0.8766**
PPL ↓	2.3976	2.4874	**2.4027**

Table 4: UltraChat training and communication with 16 workers, a global batch of 32 for 1 epoch. Both methods communicate BF16 pseudo-gradients. Logical payload represents the volume of model data exchanged at the algorithmic level. Actual traffic over physical links depends on the distributed framework and the collective communication algorithm. Metric definitions follow Table 3.

Method	DiLoCo	AutoLoCo
Train loss ↓	0.9003	**0.8761**
Syncs ↓	65	**49**
Wall time (h) ↓	7.2312	**6.4958**
Logical payload (GB)		
Per sync	16.061	16.061
Total ↓	1,043.934	**786.966**
UltraChat test		
NLL ↓	0.9016	**0.8772**
PPL ↓	2.4635	**2.4041**

Table 5: Downstream evaluation after UltraChat SFT with eight workers. Underline marks the next highest distinct score. Evaluation settings are given in Appendix E.3.

Metric	DDP	DiLoCo	AutoLoCo
IFEval Strict ↑	37.15	32.72	**32.90**
IFBench Loose ↑	17.33	**16.00**	17.33
MMLU-Pro Accuracy ↑	36.54	31.21	**36.25**
BBH CoT EM ↑	61.79	56.03	**61.10**
HumanEval+ ↑	25.61	15.24	**22.56**

Table 6: Downstream evaluation after UltraChat SFT with 16 workers. Scores are percentages; higher is better. Bold marks the best observed score in each row. Evaluation settings match Table 5.

Metric	DiLoCo	AutoLoCo
IFEval Strict ↑	34.01	**34.38**
IFBench Loose ↑	17.33	**17.67**
MMLU-Pro Accuracy ↑	33.10	**36.23**
BBH CoT EM ↑	57.93	**62.71**
HumanEval+ ↑	17.07	**23.17**

With eight workers, AutoLoCo reduces synchronizations from 65 to 49 and logical payload by 24.6% relative to DiLoCo. Test NLL improves from 0.9112 to 0.8766, leaving a gap of 0.0021 to DDP (Table 3). For 16 workers. AutoLoCo reaches test NLL 0.8772 compared with DiLoCo's 0.9016, while retaining 49 synchronizations and the 24.6% payload reduction (Table 4).

With eight workers, AutoLoCo improves on DiLoCo by about five percentage points on MMLU-Pro and BBH, and by 7.32 points on HumanEval+ (Table 5). The gap to DDP is below 0.7 points on MMLU-Pro and BBH. IFEval improves by just 0.18 points over DiLoCo and remains 4.25 points behind DDP. With 16 workers, AutoLoCo scores above DiLoCo on all five tasks (Table 6). The largest gain is again in code generation: HumanEval+ improves by 6.10 percentage points, followed by BBH at 4.78 points. Both instruction-following scores improve by less than half a point.

4.3 Ablation

We examine how adaptive synchronization and outer correction contribute to AutoLoCo on C4 with 4 workers. All runs share the initialization, training data, inner AdamW recipe, and 50,000 step budget, with a global batch of 512. Appendix E.4 gives the component settings.

Table 7: Component ablation on C4 with 4 workers and 50,000 optimizer steps. Adaptive intervals include horizon selection and token mapping. Bold marks the best training loss.

Variant	Adaptive intervals	Outer correction		Training loss ↓	Syncs ↓	Training PPL
		Momentum	Learning rate			
DiLoCo	–	–	–	2.8251	100	16.8618
Adaptive intervals	✓	–	–	2.8310	73	16.9618
Adaptive + momentum	✓	✓	–	2.8484	73	17.2603
AutoLoCo	✓	✓	✓	**2.8116**	**73**	**16.6362**

Adaptive intervals alone save communication but finish with a training loss of 2.8310, slightly above DiLoCo's 2.8251. Adjusting momentum alone increases the loss to 2.8484. During the $H = 750$ phase, the median outer update magnitude is 36.45% smaller than with adaptive intervals alone, consistent with a more conservative outer response. Adding the outer-step adjustment reverses this loss increase. AutoLoCo reaches 2.8116, an improvement of 0.0368 over the variant with momentum adjustment alone and 0.0135 over DiLoCo. These results support our design of adjusting momentum retention and outer-step scaling together as local intervals change.

Table 8: Normalization in C4 pretraining (4 workers)

Metric	Normalization	
	Without	With
Training loss	2.811582	2.826429
Syncs	73	73
Training time (h)	23.1001	23.1300

Table 9: Normalization in fine-tuning(16 workers).

Metric	Normalization	
	Without	With
Training loss	0.876554	0.876079
Syncs	49	49
Test NLL	0.877551	0.877184
Test PPL	2.405003	2.404119

Pseudo-gradient normalization. Normalization has different effects across the two training settings. For fixed coefficients and a constant input, normalization changes the Nesterov steady-state response coefficient to the unnormalized averaged pseudo-gradient from $\eta_t^{\text{out}}/(1-\mu_t)$ to $\eta_t^{\text{out}}/[\rho_t(1-\mu_t)]$. Under our outer correction, this cancels the additional amplification by ρ_t when the upper bound on outer-step scaling is inactive. With identical horizon trajectories and outer coefficients, normalization reduces the median outer-update magnitude by 47.50% during the $H = 750$ phase in C4, but increases final training loss. In the reported SFT comparison, normalization slightly lowers test NLL. This observation is consistent with a more conservative outer update, but does not by itself establish better retention of pretrained capabilities. These results motivate different normalization settings for the two evaluated training recipes. Therefore, whether to apply pseudo-gradient normalization should depend on the task.

5 Conclusion

We introduced AutoLoCo, which adapts synchronization intervals and corrects the outer optimizer for variable local horizons. Our pretraining and fine-tuning experiments show that AutoLoCo reduces communication frequency while preserving training performance. Relative to DiLoCo, the reductions are 27% in pretraining and 24.6% in fine-tuning. Beyond the precision settings evaluated here, future work could explore lower precision communication with error feedback to compensate for quantization error. Another direction is to extend the adaptive range to longer communication intervals and study their effects on training performance and stability.

Reproducibility Statement

To support reproducibility, we release the AutoLoCo experiment configurations, and methods for reproducing our method and baselines. We also report the model, optimizer, controller, and communication settings used in our experiments.

AI use statement

We used generative AI tools assist with literature searches, improve language clarity and readability, and to assist with edition of software code. We reviewed the AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnović. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/6c340f25839e6acdc73414517203f5f0-Abstract.html.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d202ed5bcfa858c15a9f383c3e386ab2-Abstract.html.

Zachary Charles, Gabriel Teston, Lucio Dery, John Rush, Nova Fallen, Zachary Garrett, Arthur D. Szlam, and Arthur Douillard. Communication-efficient language model training scales reliably and robustly: Scaling laws for DiLoCo. In *Advances in Neural Information Processing Systems*, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/99acb4c087266e80b547aed79247266b-Abstract-Conference.html.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023. URL https://arxiv.org/abs/2305.14233.

Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. DiLoCo: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023. URL https://arxiv.org/abs/2311.08105.

Arthur Douillard, Yanislav Donchev, Keith Rush, Satyen Kale, Zachary Charles, Zachary Garrett, Gabriel Teston, Dave Lacey, Ross McIlroy, Jiajun Shen, Alexandre Ramé, Arthur Szlam, Marc'Aurelio Ranzato, and Paul Barham. Streaming DiLoCo with overlapping communication: Towards a distributed free lunch. *arXiv preprint arXiv:2501.18512*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783v1.

Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. Local SGD: Unified theory and new efficient methods. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pp. 3556–3564. PMLR, 2021. URL https://proceedings.mlr.press/v130/gorbunov21a.html.

Xinran Gu, Kaifeng Lyu, Sanjeev Arora, Jingzhao Zhang, and Longbo Huang. A quadratic synchronization rule for distributed deep learning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/128911cc894d57bcae78074a9551c132-Abstract-Conference.html.

Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Local SGD with periodic averaging: Tighter analysis and adaptive synchronization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/c17028c9b6e0c5deaad29665d582284a-Abstract.html.

Sami Jaghouar, Jack Min Ong, and Johannes Hagemann. OpenDiLoCo: An open-source framework for globally distributed low-communication training. *arXiv preprint arXiv:2407.07852*, 2024. URL https://arxiv.org/abs/2407.07852.

Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. FedExP: Speeding up federated averaging via extrapolation. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=IPrzNbddXV.

Peng Jiang and Gagan Agrawal. Adaptive periodic averaging: A practical approach to reducing communication in distributed learning. *arXiv preprint arXiv:2007.06134*, 2020. URL https://arxiv.org/abs/2007.06134.

Dominik Kallusky, Vinay Rao, Vishal Nandavanam, and Hao-Jun Michael Shi. SNOO: Step-K nesterov outer optimizer—the surprising effectiveness of nesterov momentum applied to pseudo-gradients. *arXiv preprint arXiv:2510.15830*, 2025. URL https://arxiv.org/abs/2510.15830.

Michael Kamp, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel. Efficient decentralized deep learning by dynamic model averaging. In *Machine Learning and Knowledge Discovery in Databases*, pp. 393–409. Springer, 2018. doi: 10.1007/978-3-030-10925-7_24. URL https://doi.org/10.1007/978-3-030-10925-7_24.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U. Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3252–3261. PMLR, 2019. URL https://proceedings.mlr.press/v97/karimireddy19a.html.

Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Breaking the centralized barrier for cross-device federated learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/f0e6be4ce76ccfa73c5a540d992d0756-Abstract.html.

Ahmed Khaled, Satyen Kale, Arthur Douillard, Chi Jin, Rob Fergus, and Manzil Zaheer. Understanding outer optimizers in local SGD: Learning rates, momentum, and acceleration. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2VX79YLT9s.

Geon-Woo Kim, Junbo Li, Shashidhar Gandham, Omar Baldonado, Adithya Gangidi, Pavan Balaji, Zhangyang Wang, and Aditya Akella. HALoS: Hierarchical asynchronous local SGD over slow networks for geo-distributed large language model training. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 30548–30566. PMLR, 2025. URL https://proceedings.mlr.press/v267/kim25y.html.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. PyTorch distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12):3005–3018, 2020. doi: 10.14778/3415478.3415530. URL https://arxiv.org/abs/2006.15704.

Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local SGD. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1eyO1BFPr.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SkhQHMW0W.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2507.02833.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LkFG3lB13U5.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph E. Gonzalez, and Raman Arora. FetchSGD: Communication-efficient federated learning with sketching. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8253–8265. PMLR, 2020. URL https://proceedings.mlr.press/v119/rothchild20a.html.

Shuheng Shen, Yifei Cheng, Jingchang Liu, and Linli Xu. STL-SGD: Speeding up local SGD with stagewise communication period. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9576–9584, 2021. doi: 10.1609/aaai.v35i11.17153. URL https://ojs.aaai.org/index.php/AAAI/article/view/17153.

Artin Spiridonoff, Alex Olshevsky, and Yannis Paschalidis. Communication-efficient SGD: From local SGD to one-shot averaging. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/cc06a6150b92e17dd3076a0f0f9d2af4-Abstract.html.

Sebastian U. Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1g2JnRcFX.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit Adam: Communication efficient large-scale training with Adam's convergence speed. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10118–10129. PMLR, 2021. URL https://proceedings.mlr.press/v139/tang21a.html.

Benjamin Thérien, Xiaolong Huang, Irina Rish, and Eugene Belilovsky. MuLoCo: Muon is a practical inner optimizer for DiLoCo. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/64378.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. URL https://arxiv.org/abs/2302.13971.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/d9fbed9da256e344c1fa46bb46c34c5f-Abstract.html.

Jianyu Wang and Gauri Joshi. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD. In *Proceedings of Machine Learning and Systems*, volume 1, 2019. URL https://proceedings.mlsys.org/paper_files/paper/2019/hash/4a0151b47bd93c5de2a0b57831981a0d-Abstract.html.

Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of local-update SGD algorithms. *Journal of Machine Learning Research*, 22(213):1–50, 2021. URL https://www.jmlr.org/papers/v22/20-147.html.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/564127c03caab942e503ee6f810f54fd-Abstract.html.

Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. SlowMo: Improving communication-efficient distributed SGD with slow momentum. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=SkxJ8REYPH.

Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Ré, and Ce Zhang. CocktailSGD: Fine-tuning foundation models over 500 mbps networks. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 36058–36076. PMLR, 2023. URL https://proceedings.mlr.press/v202/wang23t.html.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. URL https://papers.nips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

Yujia Wang, Shiqiang Wang, Songtao Lu, and Jinghui Chen. FADAS: Towards federated adaptive asynchronous optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 51701–51733. PMLR, 2024b. URL https://proceedings.mlr.press/v235/wang24bv.html.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

A Implementation details for adaptive horizon selection

A.1 Current Run Reference

We estimate the accepted-horizon reference profile from the current run. We keep the recipe horizon throughout the 1,000 step inner-learning-rate warm-up, which spans the first two 500 step intervals. Beginning with the interval that completes warm-up, we collect two finite intervals at the initial horizon and set the reference statistics to the geometric means of their observations. We then keep the

same horizon for three additional intervals to estimate the normal variation of $r_{D,t}$ and $r_{C,t}$. For each residual family, the robust center is the median of at most 16 stored residuals. We first assess each completed interval and then add its residuals to the history only when the interval is increase supporting or reference consistent. After calibration, only increase-supporting or reference-consistent intervals executed at the accepted horizon update the residual history. The two diagnostic references use a log-space exponential moving average, while Λ_t^{ref} follows the latest accepted interval because the inner learning-rate schedule changes over training.

A.2 Interval Evaluation

Monitoring the accepted horizon. As in Table 10, a reference-consistent interval keeps the accepted horizon. A larger horizon is considered only after the post-reduction hold-off ends and at least one interval at the accepted horizon is consistent with the reference profile. A moderate deviation keeps the current horizon, but two consecutive moderate deviations trigger a horizon reduction. A severe deviation reduces the horizon immediately. A non-finite required statistic uses the same reduction path and may invoke the minimum-horizon fallback.

Table 10: Interval assessments for the active D_t and C_t classifier in the pre-training experiments.

Interval assessment	**Range of $z_t^{\max}$**
Increase supported	$z_t^{\max} < 1.5$
Reference consistent	$1.5 \le z_t^{\max} < 2.5$
Moderate deviation	$2.5 \le z_t^{\max} < 4.0$
Severe deviation	$z_t^{\max} \ge 4.0$

Selecting a longer interval. When an increase is supported, we search the admissible horizons above the accepted value. Each larger horizon defines a longer interval and is mapped to steps before evaluation. We preview the accumulated inner learning rate that the interval would cover at the current scheduler position and compare that exposure with the accepted horizon. We choose the longest admissible interval whose previewed exposure is no more than 1.15 times that of the accepted horizon.

A previously rejected interval leaves its token-equivalent horizon as a rejected-horizon bound. While that bound is active, the next interval uses the nearest admissible horizon to the geometric midpoint between the accepted horizon and the rejected bound. The controller also stores the accumulated inner learning rate observed when the bound was created. If a later preview at the rejected bound falls to at most 0.90 of that stored value, we retire the bound because learning-rate decay has changed the amount of local optimization represented by the same horizon.

Evaluating the longer interval. We evaluate the first interval executed at a larger candidate horizon. If it supports an increase, we adopt the candidate immediately. If it is reference consistent, we retain the same candidate token horizon for one additional interval and adopt it if that interval supports an increase or is reference consistent. A moderate or severe deviation in either interval rejects the candidate and restores the previously accepted horizon. After adoption or return, we re-estimate the reference profile at the horizon used next before resuming monitoring. Figure 3 summarizes these transitions.

A.3 Horizon Updates

Token mapping calibration. The token mapper is calibrated independently of the diagnostic profile. During the first three recipe-horizon intervals, we record the exact global non-padding token count and set M_{base} to their median. We initialize the tokens-per-step estimate from the first completed interval and then update $\hat{n}_t$ after every interval with $\beta = 0.9$. The newly selected H_{t+1}^{tok} is mapped only after this update, so the next interval uses the latest observed token density. No historical trace or external fixed-horizon run supplies M_{base}.

Before M_{base} and $\hat{n}_t$ are available, we temporarily execute the same numerical value for H_t^{tok} and H_t^{step}. After calibration, we evaluate the ratio in Equation (5) and choose the nearest admissi-

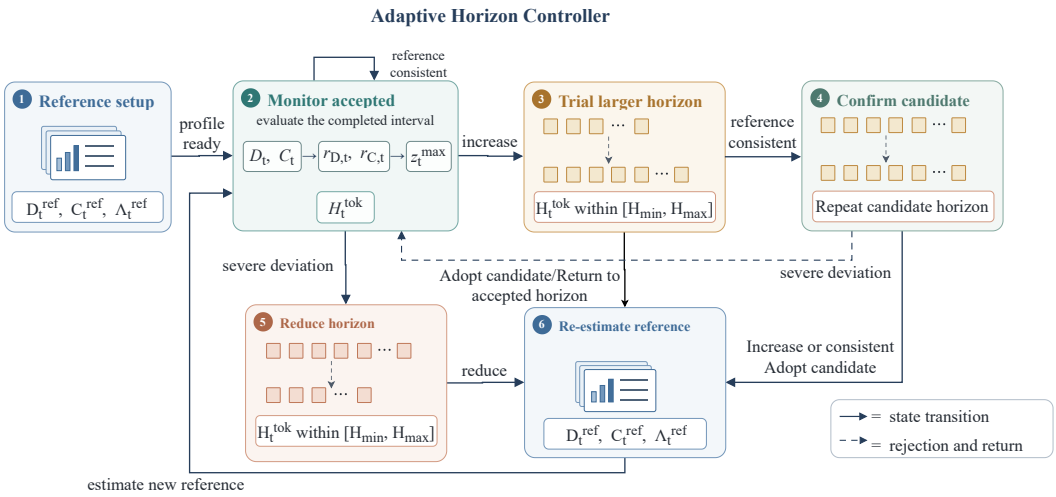


Figure 3: Adaptive horizon selection in AutoLoCo. A longer candidate is tested only after the reference and monitoring requirements are met. A first candidate interval that supports an increase leads to immediate adoption, whereas a reference-consistent result requires one additional interval. Dashed intermodule arrows denote candidate rejection and return to the previously accepted horizon. The reference profile is re-estimated after adoption, return, or reduction. Numerical safeguards are omitted.

ble multiple of q_H. The interval $[H_{\min}, H_{\max}]$ constrains the token-equivalent controller domain. Applying the same mapping at $H_t^{\text{tok}} = H_{\text{base}}$ gives H_t^{ref}, which the outer correction uses as the current-density version of one recipe token interval.

Reducing the horizon. Under normal training dynamics, H remains stable or gradually increases; the reduction rule is used to handle abnormal or persistently adverse observations. A confirmed moderate deviation reduces the current token-equivalent horizon by a factor of 1.20, while a severe deviation reduces it by a factor of 1.50. We then select the nearest admissible multiple of q_H that is strictly smaller than the current horizon and lies within $[H_{\min}, H_{\max}]$. When a larger candidate is rejected, the controller normally returns to the previously accepted horizon rather than applying an additional multiplicative reduction. Repeated severe or invalid observations may trigger the $H_{\min}$ fallback. After a reduction, one interval at the new horizon initializes its reference profile, three additional intervals estimate reference variability, and one monitoring interval must complete before the controller can propose another increase.

B Implementation Details for the Outer Optimizer Correction

B.1 Why Variable Horizons Change the Outer Update

DiLoCo processes its outer optimizer once per communication interval. Under a fixed local horizon, every outer update follows the same configured number of inner steps. Once the horizon varies, successive updates no longer share this common outer clock. A longer interval accumulate more inner learning rate, change the scale of the averaged pseudo-gradient, and allow the incoming momentum state to persist across more inner optimization.

Prior work motivates treating the update after local computation as a distinct optimization layer. SlowMo and SNOO study momentum on the outer optimization clock, while FedNova and adaptive federated optimization account for the local computation represented by an aggregated update (Wang et al., 2020b; Kallusky et al., 2025; Wang et al., 2020a; Reddi et al., 2021). AutoLoCo addresses the corresponding problem across successive DiLoCo intervals. Workers use one shared horizon within each interval, but that horizon may change between intervals. We therefore compare the

selected interval with one base token interval at the same scheduler position and use this comparison throughout the outer correction.

B.2 Scheduler-Aligned Interval Comparison

Let s_t denote the global inner-step index at the beginning of interval t. The token mapper provides H_t^{step}, the executable horizon selected for this interval, and H_t^{ref}, the executable horizon corresponding to one base token interval at the current token density. Starting from s_t, their accumulated inner learning rates are

$$\Lambda_t = \sum_{h=1}^{H_t^{\text{step}}} \eta_{t,h}, \qquad \Lambda_t^{\text{base}} = \sum_{h=1}^{H_t^{\text{ref}}} \eta_{t,h}. \tag{10}$$

Here, Λ_t is the accumulated inner learning rate of interval t, and Λ_t^{base} is the base-interval comparator evaluated from the same scheduler position. The superscript base refers to the underlying DiLoCo recipe. It is distinct from Λ_t^{ref}, which is the empirical accumulated inner-learning-rate statistic stored in the accepted-horizon reference profile.

For the detailed calculation, we write

$$\rho_t^{\text{unc}} = \frac{\Lambda_t}{\Lambda_t^{\text{base}} + \epsilon}, \qquad \rho_t = \max\left(\rho_t^{\text{unc}}, 1\right). \tag{11}$$

The value ρ_t^{unc} is the interval ratio before the lower bound is applied. The outer correction uses ρ_t. An interval whose accumulated inner learning rate does not exceed the base comparator therefore uses the reference value $\rho_t = 1$.

Both scheduled sums become available once H_t^{step} and H_t^{ref} have been determined. Our implementation evaluates exact scheduler previews of these sums before interval t begins and stores the resulting correction context for that interval. After local execution, the completed Λ_t is recorded for the adaptive-horizon reference profile. The completed measurement and the preview cover the same scheduled steps for a full, non-truncated interval, but they serve different parts of AutoLoCo. The preview configures the outer correction, and the completed measurement supports the next horizon decision.

B.3 Pseudo-Gradient Normalization

Workers first complete the selected local horizon and average their pseudo-gradients into $\bar{\Delta}_t$. We then apply Equation (7) to this averaged quantity, immediately before the outer optimizer step. Every worker therefore uses the same ρ_t and the same normalized pseudo-gradient $\widetilde{\Delta}_t$.

The mathematical definition divides by ρ_t. The implementation uses $\rho_t + \epsilon$ in the denominator for numerical protection. At $\rho_t = 1$, normalization is the identity. As ρ_t increases, the outer-optimizer input is reduced in proportion to the additional accumulated inner learning rate represented by the selected interval. Pseudo-gradient normalization is enabled independently of the momentum and outer-learning-rate adjustments.

B.4 Momentum Retention and Bounded Outer-Step Scaling

Under the base recipe, one outer update retains a fraction μ_{base} of the incoming momentum state. If interval t represents ρ_t base intervals, repeated application of this retention gives $\mu_{\text{base}}^{\rho_t}$. For integer ρ_t, the expression inside the clipping operation equals the retention obtained from repeated base-momentum updates. The exponent extends the same relation to non-integer interval ratios. This is described in detail in Appendix C.3. The bounds $\mu_{\min}$ and $\mu_{\max}$ constrain the amount of momentum retained by one outer update.

The interval ratio also determines the outer-step scale:

$$\kappa_t = \min\left(\rho_t, \kappa_{\max}\right). \tag{12}$$

We choose the outer learning rate so that its momentum-adjusted scale is κ_t times the base value:

$$\frac{\eta_t^{\text{out}}}{1-\mu_t} = \kappa_t \frac{\eta_{\text{base}}^{\text{out}}}{1-\mu_{\text{base}}}. \tag{13}$$

Solving this relation gives

$$\eta_t^{\text{out}} = \eta_{\text{base}}^{\text{out}}\, \kappa_t\, \frac{1-\mu_t}{1-\mu_{\text{base}}}. \tag{14}$$

The momentum exponent changes retention over the interval, while κ_t limits the corresponding increase in the momentum-adjusted outer-step scale.

At $\rho_t = 1$, the interval ratio recovers the base case. Provided that $\mu_{\text{base}} \in [\mu_{\min}, \mu_{\max}]$, we obtain

$$\widetilde{\Delta}_t = \bar{\Delta}_t, \qquad \mu_t = \mu_{\text{base}}, \qquad \kappa_t = 1, \qquad \eta_t^{\text{out}} = \eta_{\text{base}}^{\text{out}}.$$

For $\rho_t > 1$, the correction lowers momentum retention and increases the momentum-adjusted outer-step scale only up to $\kappa_{\max}$. We update the momentum coefficient without rescaling the stored momentum state.

B.5 The AutoLoCo Algorithm

Algorithm 1 AutoLoCo outer interval

Require: synchronized parameters θ_t, incoming momentum state v_{t-1}, selected token-equivalent horizon H_t^{tok}, accepted-horizon reference profile and controller state, $M_{\text{base}}, \hat{n}_{t-1}, H_{\text{base}}, q_H, \beta, \epsilon$, and inner learning rates $\eta_{t,h}, \eta_{\text{base}}^{\text{out}}, \mu_{\text{base}}, \mu_{\min}, \mu_{\max}, \kappa_{\max}$

Horizon and outer-correction setup

Compute H_t^{tok}, H_t^{step}, H_t^{ref}, ρ_t, μ_t, κ_t, η_t^{out}

Local inner loops

for $i \leftarrow 1$ to N **in parallel do**

  $\theta_{t,0}^{(i)} \leftarrow \theta_t$

  for $h \leftarrow 1$ to H_t^{step} **do**

    $\theta_{t,h}^{(i)} \leftarrow \text{InnerOpt}\left(\theta_{t,h-1}^{(i)}; \mathcal{B}_{t,h}^{(i)}, \eta_{t,h}\right)$

  end for

  $\Delta_t^{(i)} \leftarrow \theta_t - \theta_{t,H_t^{\text{step}}}^{(i)}$

end for

Synchronization and corrected outer update

$\bar{\Delta}_t \leftarrow \dfrac{1}{N}\sum_{i=1}^{N} \Delta_t^{(i)}$ ▷ synchronize pseudo-gradients

$\widetilde{\Delta}_t \leftarrow \dfrac{\bar{\Delta}_t}{\rho_t}$

$v_t \leftarrow \mu_t v_{t-1} + \widetilde{\Delta}_t$

$d_t \leftarrow \widetilde{\Delta}_t + \mu_t v_t$

$\theta_{t+1} \leftarrow \theta_t - \eta_t^{\text{out}} d_t$

Interval evaluation and next horizon

$D_t \leftarrow \dfrac{1}{N}\sum_{i=1}^{N} \left\| \Delta_t^{(i)} \right\|_2^2, \qquad C_t \leftarrow \dfrac{N \left\| \bar{\Delta}_t \right\|_2^2}{D_t + \epsilon}$

$\Lambda_t \leftarrow \sum_{h=1}^{H_t^{\text{step}}} \eta_{t,h}, \qquad M_t \leftarrow \sum_{h=1}^{H_t^{\text{step}}} n_{t,h}$

$\bar{n}_t \leftarrow \dfrac{M_t}{H_t^{\text{step}}}, \qquad \hat{n}_t \leftarrow \beta \hat{n}_{t-1} + (1-\beta)\bar{n}_t$

Assess D_t, C_t, and Λ_t. Determine the current training status.

Update the reference profile and select H_{t+1}^{tok}

return θ_{t+1}, v_t, and H_{t+1}^{tok}

C Theoretical Analysis of AutoLoCo

C.1 Interpretation of Local-Drift Energy and Aggregation Coherence

During H_t^{step} local updates without communication, workers develop different parameter deviations from their shared starting point, a phenomenon known as worker drift. We want to characterize the relationship between this drift and the synchronization interval. To solve this problem, we relate the local-drift energy D_t and aggregation coherence C_t to disagreement among the local endpoints of a completed interval. Suppose all N workers start interval t from θ_t and complete H_t^{step} local updates. Their pseudo-gradients and exact arithmetic mean are

$$\Delta_t^{(i)} = \theta_t - \theta_{t,H_t^{\text{step}}}^{(i)}, \qquad \bar{\Delta}_t = \frac{1}{N}\sum_{i=1}^{N} \Delta_t^{(i)}.$$

For the numerical constant $\epsilon > 0$, the diagnostics are

$$D_t = \frac{1}{N}\sum_{i=1}^{N} \|\Delta_t^{(i)}\|_2^2, \qquad C_t = \frac{N\|\bar{\Delta}_t\|_2^2}{D_t + \epsilon}.$$

The derivation applies to any inner optimizer and allows H_t^{step} to vary across intervals.

The pseudo-gradient sign convention gives the arithmetic mean of the local endpoints as

$$\begin{aligned} \frac{1}{N}\sum_{i=1}^{N} \theta_{t,H_t^{\text{step}}}^{(i)} &= \frac{1}{N}\sum_{i=1}^{N} \big(\theta_t - \Delta_t^{(i)}\big) \\ &= \theta_t - \bar{\Delta}_t. \end{aligned} \tag{15}$$

This arithmetic mean need not equal the synchronized parameters θ_{t+1} produced by the Nesterov outer update. Centering each local endpoint around this mean gives

$$\begin{aligned} \theta_{t,H_t^{\text{step}}}^{(i)} - \big(\theta_t - \bar{\Delta}_t\big) &= \big(\theta_t - \Delta_t^{(i)}\big) - \big(\theta_t - \bar{\Delta}_t\big) \\ &= \bar{\Delta}_t - \Delta_t^{(i)}. \end{aligned} \tag{16}$$

Taking squared Euclidean norms and averaging therefore reduces endpoint disagreement to the mean squared deviation of the pseudo-gradients. Expanding this quantity yields

$$\begin{aligned} &\frac{1}{N}\sum_{i=1}^{N} \|\Delta_t^{(i)} - \bar{\Delta}_t\|_2^2 \\ &\quad= \frac{1}{N}\sum_{i=1}^{N} \Big(\|\Delta_t^{(i)}\|_2^2 - 2\big\langle \Delta_t^{(i)}, \bar{\Delta}_t \big\rangle + \|\bar{\Delta}_t\|_2^2\Big) \\ &\quad= D_t - 2\left\langle \frac{1}{N}\sum_{i=1}^{N} \Delta_t^{(i)}, \bar{\Delta}_t \right\rangle + \|\bar{\Delta}_t\|_2^2 \\ &\quad= D_t - \|\bar{\Delta}_t\|_2^2. \end{aligned} \tag{17}$$

Since $D_t + \epsilon > 0$, the definition of C_t gives

$$\|\bar{\Delta}_t\|_2^2 = \frac{C_t(D_t + \epsilon)}{N}. \tag{18}$$

Substituting into Equation (17) expresses the mean squared distance of the local endpoints from their arithmetic mean in terms of D_t and C_t:

$$\begin{aligned} &\frac{1}{N}\sum_{i=1}^{N} \Big\|\theta_{t,H_t^{\text{step}}}^{(i)} - \big(\theta_t - \bar{\Delta}_t\big)\Big\|_2^2 \\ &\quad= \frac{1}{N}\sum_{i=1}^{N} \|\Delta_t^{(i)} - \bar{\Delta}_t\|_2^2 \\ &\quad= D_t - \frac{C_t(D_t + \epsilon)}{N}. \end{aligned} \tag{19}$$

This shows that D_t and C_t jointly provide an exact characterization of the disagreement among worker endpoints accumulated during a local synchronization interval. The endpoint disagreement is $D_t - C_t(D_t + \epsilon)/N$.

The squared norm is convex, so

$$\|\bar{\Delta}_t\|_2^2 \leq \frac{1}{N}\sum_{i=1}^{N}\left\|\Delta_t^{(i)}\right\|_2^2 = D_t. \tag{20}$$

Consequently, for finite pseudo-gradients,

$$\begin{aligned} 0 \leq C_t &\leq \frac{ND_t}{D_t + \epsilon} < N, \\ 0 \leq D_t &- \frac{C_t(D_t + \epsilon)}{N} \leq D_t. \end{aligned} \tag{21}$$

So, C_t is a bounded measure of coherence, while the endpoint disagreement determined by D_t and C_t is nonnegative and cannot exceed the total drift energy D_t.

If $D_t = 0$, then $\Delta_t^{(i)} = 0$ for every worker, so $C_t = 0$ and all local endpoints coincide with θ_t. More generally, if all worker pseudo-gradients are identical, the local endpoints coincide and

$$C_t = \frac{ND_t}{D_t + \epsilon}, \tag{22}$$

attaining the upper bound for the given D_t.

The Interpretation is, D_t and C_t distinguish large but coherent local motion from motion with substantial disagreement across workers. Local-drift energy D_t measures the mean squared displacement of the local endpoints from the synchronized parameters θ_t. Aggregation coherence C_t quantifies the squared norm retained after averaging pseudo-gradients, relative to $D_t + \epsilon$ and scaled by N. For fixed N and D_t, a larger C_t therefore implies smaller endpoint disagreement.

C.2 Learning-Rate Mass and Scheduler-Aligned Interval Ratios

Changing the number of local inner steps changes the local computation represented by each outer update and can alter the scale of the averaged pseudo-gradient. Fixed outer coefficients do not explicitly account for these changes. AutoLoCo therefore uses outer correction to adjust momentum retention and the outer learning rate for the selected interval. This correction requires a measure of the interval's scale relative to the base recipe. We measure this scale using the ratio of accumulated inner learning rates, or learning-rate masses. We then examine when this ratio also describes the relative scale of local displacements.

The token mapper provides H_t^{step} and H_t^{ref}, the selected executable horizon and the executable horizon corresponding to one base token interval at the current token density. Both horizons start at the same position in the inner learning-rate schedule. Their learning-rate masses are

$$\Lambda_t = \sum_{h=1}^{H_t^{\mathrm{step}}} \eta_{t,h}, \qquad \Lambda_t^{\mathrm{base}} = \sum_{h=1}^{H_t^{\mathrm{ref}}} \eta_{t,h}. \tag{23}$$

The comparator $\Lambda_t^{\mathrm{base}}$ is evaluated at the current scheduler position. It is not the historical Λ_t^{ref} stored in the accepted-horizon reference profile. This choice measures the effect of the selected horizon relative to the current base interval, rather than comparing different training stages. For the displacement analysis below, assume that the inner learning rates over the compared ranges are positive. The protected ratio defined later also accommodates nonnegative learning rates.

Learning-rate mass and local displacement. Using the pseudo-gradient sign convention, we telescope the local updates to obtain

$$\begin{aligned}\Delta_t^{(i)} &= \theta_t - \theta_{t,H_t^{\mathrm{step}}}^{(i)} \\ &= \sum_{h=1}^{H_t^{\mathrm{step}}} \left(\theta_{t,h-1}^{(i)} - \theta_{t,h}^{(i)}\right) \\ &= \sum_{h=1}^{H_t^{\mathrm{step}}} \eta_{t,h} \frac{\theta_{t,h-1}^{(i)} - \theta_{t,h}^{(i)}}{\eta_{t,h}}.\end{aligned} \tag{24}$$

The quotient in the last line is the effective parameter displacement per unit inner learning rate. It includes the effects of the inner optimizer.

Dividing Equation (24) by $\Lambda_t > 0$ yields

$$\begin{aligned}\frac{\Delta_t^{(i)}}{\Lambda_t} &= \sum_{h=1}^{H_t^{\mathrm{step}}} \frac{\eta_{t,h}}{\Lambda_t} \frac{\theta_{t,h-1}^{(i)} - \theta_{t,h}^{(i)}}{\eta_{t,h}}, \\ \sum_{h=1}^{H_t^{\mathrm{step}}} \frac{\eta_{t,h}}{\Lambda_t} &= 1.\end{aligned} \tag{25}$$

Here, Λ_t is the total learning-rate weight, and $\Delta_t^{(i)}/\Lambda_t$ is a weighted average of the effective local updates. Then, interpreting local displacement as proportional to learning-rate mass additionally requires the effective updates to remain sufficiently close over the compared steps.

Comparison with the base interval. Consider $H_t^{\mathrm{step}} > H_t^{\mathrm{ref}}$. The point $\theta_{t,H_t^{\mathrm{ref}}}^{(i)}$ is the base-horizon prefix endpoint of the same local trajectory. It therefore shares the initial parameters, local optimizer state, and minibatch prefix with the selected interval, and no intermediate outer update is inserted. We have

$$\begin{aligned}\Lambda_t - \Lambda_t^{\mathrm{base}} &= \sum_{h=H_t^{\mathrm{ref}}+1}^{H_t^{\mathrm{step}}} \eta_{t,h}, \\ \Delta_t^{(i)} - \left(\theta_t - \theta_{t,H_t^{\mathrm{ref}}}^{(i)}\right) &= \sum_{h=H_t^{\mathrm{ref}}+1}^{H_t^{\mathrm{step}}} \left(\theta_{t,h-1}^{(i)} - \theta_{t,h}^{(i)}\right).\end{aligned} \tag{26}$$

Subtracting the base-prefix displacement scaled by $\Lambda_t/\Lambda_t^{\mathrm{base}}$ gives

$$\begin{aligned}&\Delta_t^{(i)} - \frac{\Lambda_t}{\Lambda_t^{\mathrm{base}}}\left(\theta_t - \theta_{t,H_t^{\mathrm{ref}}}^{(i)}\right) \\ &\quad= \sum_{h=H_t^{\mathrm{ref}}+1}^{H_t^{\mathrm{step}}} \left[\theta_{t,h-1}^{(i)} - \theta_{t,h}^{(i)} - \frac{\eta_{t,h}}{\Lambda_t^{\mathrm{base}}}\left(\theta_t - \theta_{t,H_t^{\mathrm{ref}}}^{(i)}\right)\right] \\ &\quad= \sum_{h=H_t^{\mathrm{ref}}+1}^{H_t^{\mathrm{step}}} \eta_{t,h} \left[\frac{\theta_{t,h-1}^{(i)} - \theta_{t,h}^{(i)}}{\eta_{t,h}} - \frac{\theta_t - \theta_{t,H_t^{\mathrm{ref}}}^{(i)}}{\Lambda_t^{\mathrm{base}}}\right].\end{aligned} \tag{27}$$

By the triangle inequality and positivity of the learning rates,

$$\begin{aligned}&\left\|\Delta_t^{(i)} - \frac{\Lambda_t}{\Lambda_t^{\mathrm{base}}}\left(\theta_t - \theta_{t,H_t^{\mathrm{ref}}}^{(i)}\right)\right\|_2 \\ &\quad\le \sum_{h=H_t^{\mathrm{ref}}+1}^{H_t^{\mathrm{step}}} \eta_{t,h} \left\|\frac{\theta_{t,h-1}^{(i)} - \theta_{t,h}^{(i)}}{\eta_{t,h}} - \frac{\theta_t - \theta_{t,H_t^{\mathrm{ref}}}^{(i)}}{\Lambda_t^{\mathrm{base}}}\right\|_2 \\ &\quad\le \left(\Lambda_t - \Lambda_t^{\mathrm{base}}\right) \max_{H_t^{\mathrm{ref}} < h \le H_t^{\mathrm{step}}} \left\|\frac{\theta_{t,h-1}^{(i)} - \theta_{t,h}^{(i)}}{\eta_{t,h}} - \frac{\theta_t - \theta_{t,H_t^{\mathrm{ref}}}^{(i)}}{\Lambda_t^{\mathrm{base}}}\right\|_2.\end{aligned} \tag{28}$$

Equation (28) bounds the error in representing the selected-interval displacement as the base-prefix displacement scaled by $\Lambda_t/\Lambda_t^{\text{base}}$. The bound is the additional learning-rate mass multiplied by the largest deviation of the appended effective updates from the learning-rate-weighted mean of the base prefix. When this product is small, the selected interval's displacement is close to this scaled base-prefix displacement. The relation is exact when every appended effective update equals the base-prefix mean. So, the LR-mass ratio has a local-displacement interpretation under the stated condition on effective-update variation.

For the outer correction, AutoLoCo uses the lower-bounded ratio $\rho_t = \max\left(\frac{\Lambda_t}{\Lambda_t^{\text{base}}+\epsilon}, 1\right)$. We next use ρ_t to coordinate momentum retention and bounded outer-step scaling relative to the base recipe.

C.3 Momentum Retention and Bounded Outer-Step Scaling

The interval ratio relates the selected interval to the base recipe. We use this ratio to adjust momentum retention and scale the resulting outer direction. In the version without pseudo-gradient normalization, the Nesterov outer update is

$$\begin{aligned} v_t &= \mu_t v_{t-1} + \bar{\Delta}_t, \\ d_t &= \bar{\Delta}_t + \mu_t v_t, \\ u_t &= \theta_t - \theta_{t+1} = \eta_t^{\text{out}} d_t. \end{aligned} \tag{29}$$

A fixed momentum coefficient discounts the incoming state once per synchronization, regardless of the local computation between synchronizations. To motivate the retention rule before clipping, we hold the base unit of learning-rate mass fixed and consider only the contribution of the incoming state. One base unit retains a fraction μ_{base}, where $0 < \mu_{\text{base}} < 1$. We require retention to depend only on the accumulated exposure and to compose multiplicatively when that exposure is partitioned. For a positive, continuous retention rule, this makes log retention additive and therefore linear in exposure. Calibration at one base unit gives log retention $\rho_t \log \mu_{\text{base}}$ at ρ_t units. The retained fraction is therefore

$$\exp(\rho_t \log \mu_{\text{base}}) = \mu_{\text{base}}^{\rho_t}. \tag{30}$$

This gives the exponential term in the interval-adjusted momentum rule.

Changing momentum also changes the response to repeated pseudo-gradient inputs. To characterize this effect, fix μ_t, η_t^{out}, and $\bar{\Delta}_t$ at their interval-t values and repeatedly apply the same outer recurrence, with $0 \le \mu_t < 1$. This is a fixed-input response analysis, not an assumption that training inputs remain constant. Subtracting $\bar{\Delta}_t/(1-\mu_t)$ from the momentum recurrence gives

$$\begin{aligned} v_t - \frac{\bar{\Delta}_t}{1-\mu_t} &= \mu_t v_{t-1} + \bar{\Delta}_t - \frac{\bar{\Delta}_t}{1-\mu_t} \\ &= \mu_t \left(v_{t-1} - \frac{\bar{\Delta}_t}{1-\mu_t} \right). \end{aligned} \tag{31}$$

Each repetition contracts the distance to $\bar{\Delta}_t/(1-\mu_t)$ by the factor μ_t, so the momentum converges to this fixed point. At the fixed point, the direction and update become

$$\begin{aligned} d_t &= \bar{\Delta}_t + \mu_t \frac{\bar{\Delta}_t}{1-\mu_t} = \frac{\bar{\Delta}_t}{1-\mu_t}, \\ u_t &= \eta_t^{\text{out}} d_t = \frac{\eta_t^{\text{out}}}{1-\mu_t} \bar{\Delta}_t. \end{aligned} \tag{32}$$

The outer learning rate and momentum jointly determine the fixed-input response coefficient $\eta_t^{\text{out}}/(1-\mu_t)$. To set this coefficient to κ_t times its base value, we require

$$\frac{\eta_t^{\text{out}}}{1-\mu_t} = \kappa_t \frac{\eta_{\text{base}}^{\text{out}}}{1-\mu_{\text{base}}}. \tag{33}$$

Solving for the outer learning rate yields

$$\eta_t^{\text{out}} = \eta_{\text{base}}^{\text{out}} \kappa_t \frac{1-\mu_t}{1-\mu_{\text{base}}}. \tag{34}$$

The factor $(1-\mu_t)/(1-\mu_{\text{base}})$ compensates for the effect of momentum on the fixed-input response, while κ_t specifies the desired relative scale. The choice $\kappa_t = \min(\rho_t, \kappa_{\max})$ is a bounded-scaling policy.

D Relation Between AutoLoCo and DiLoCo

D.1 DiLoCo Preliminaries

AutoLoCo preserves DiLoCo's inner and outer training loop, so we begin with one outer interval. At the start of interval t, all N workers share the parameters θ_t. Worker i updates its local copy for H_t^{step} steps and reaches a local endpoint. Equation (35) follows worker i from the shared model to this endpoint and defines the resulting pseudo-gradient:

$$\begin{aligned} \theta_{t,0}^{(i)} &= \theta_t, \\ \theta_{t,h}^{(i)} &= \mathrm{InnerOpt}\left(\theta_{t,h-1}^{(i)}; \mathcal{B}_{t,h}^{(i)}, \eta_{t,h}\right), \qquad h = 1, \dots, H_t^{\text{step}}, \\ \Delta_t^{(i)} &= \theta_t - \theta_{t,H_t^{\text{step}}}^{(i)}. \end{aligned} \tag{35}$$

Here, h indexes the local optimizer steps. At step h, worker i processes minibatch $\mathcal{B}_{t,h}^{(i)}$ with inner learning rate $\eta_{t,h}$. The horizon H_t^{step} is the number of local updates performed before the next synchronization. We use AdamW as $\mathrm{InnerOpt}$, following DiLoCo.

After the local trajectories end, the workers synchronize and average their pseudo-gradients. The outer optimizer uses this average to produce the next synchronized model:

$$\bar{\Delta}_t = \frac{1}{N}\sum_{i=1}^{N} \Delta_t^{(i)}, \qquad \theta_{t+1} = \mathrm{OuterOpt}\left(\theta_t, \bar{\Delta}_t\right). \tag{36}$$

Equation (36) omits the outer momentum state and scalar optimizer settings. Section 3.2 specifies the scalar optimizer settings, and Equation (9) gives the explicit update with its momentum state. DiLoCo combines many local AdamW steps with a Nesterov outer update. DiLoCo sets $H_t^{\text{step}} = H_{\text{fix}}$ in every interval and keeps the outer learning rate and momentum constant. AutoLoCo retains Equations (35) and (36), but allows the executable horizon and scalar outer optimizer settings to vary across intervals.

D.2 DiLoCo as a Special Case

AutoLoCo recovers fixed-horizon DiLoCo when the selected horizon and outer optimizer remain at their base settings. Suppose the controller always selects H_{base}, and suppose the mapper operates at the reference token density. It then returns

$$H_t^{\text{tok}} = H_{\text{base}}, \qquad H_t^{\text{step}} = H_t^{\text{ref}} = H_{\text{base}}, \qquad \forall t. \tag{37}$$

The selected interval and the base comparator now cover the same part of the inner learning-rate schedule. Their accumulated inner learning rates are therefore equal:

$$\Lambda_t = \Lambda_t^{\text{base}}, \qquad \rho_t = 1. \tag{38}$$

Assume that $\mu_{\text{base}} \in [\mu_{\min}, \mu_{\max}]$. Substituting $\rho_t = 1$ into the outer correction gives

$$\widetilde{\Delta}_t = \bar{\Delta}_t, \qquad \mu_t = \mu_{\text{base}}, \qquad \kappa_t = 1, \qquad \eta_t^{\text{out}} = \eta_{\text{base}}^{\text{out}}. \tag{39}$$

With the same incoming momentum state, the Nesterov update becomes

$$\begin{aligned} v_t &= \mu_{\text{base}} v_{t-1} + \bar{\Delta}_t, \\ d_t &= \bar{\Delta}_t + \mu_{\text{base}} v_t, \\ \theta_{t+1} &= \theta_t - \eta_{\text{base}}^{\text{out}} d_t. \end{aligned} \tag{40}$$

This is the DiLoCo outer update under the pseudo-gradient sign convention in Equation (35). Thus, AutoLoCo reduces exactly to DiLoCo in the base setting.

E EXPERIMENTAL SETTINGS

E.1 TRAINING AND EVALUATION PROTOCOL

We simulate 8 independent workers for the main C4 pre-training and SFT experiments. Each local-training worker maintains its own model replica and AdamW state. We also evaluate SFT with 16 workers communicating over a physical network. Table 11 summarizes the training settings. Within each comparison, methods share the model initialization, data, and training budget.

Table 11: Training settings. Batch sizes count sequences per optimizer step. For SFT, the paired values correspond to eight and sixteen workers. The C4 ablations use four workers and a global batch of 512, with the remaining training settings unchanged.

Setting	C4 pre-training	UltraChat SFT
Model	LLaMA (215M total)	Llama-3.1-8B Base
Initialization	Shared random initialization	Shared pretrained checkpoint
Training budget	50,000 steps	One epoch after filtering
Workers	4 / 8	8 / 16
Effective batch per worker	128	4 / 2
Global batch	512 / 1,024	32
Sequence length	1,024	At most 2,048
Inner optimizer	AdamW	AdamW
Peak inner learning rate	4×10^{-4}	5×10^{-6}
AdamW coefficients	$(0.9, 0.95)$	$(0.9, 0.999)$
Weight decay	0.1	0
Gradient-norm clipping	1.0	1.0
Learning-rate schedule	Cosine	Cosine
Warm-up	1,000 steps	3% of steps
Compute precision	BF16	BF16
Random seed	42	42

We shard the English training split across workers and truncate or pad each document to 1,024 tokens without packing. The main budget therefore covers 52.43B input token positions, including padding; the 4 worker ablations cover 26.21B.

For SFT, we update all parameters of Llama-3.1-8B on UltraChat-200k. We filter invalid conversations and examples without assistant targets after truncation, then train for one pass through the retained train data. All SFT comparisons consume the same shuffled global batches of 32, with the final incomplete batch dropped. Conversations are dynamically padded without packing, and only assistant target tokens contribute to the loss. The token mapper counts all non-padding input tokens, including the conversational context.

We evaluate the final synchronized model using token-averaged negative log-likelihood. Our C4 evaluator is configured to use 4,096 validation documents. For SFT, we use a subset of 1,000 valid test conversations and average loss over assistant targets.

E.2 BASELINES AND AUTOLOCO CONFIGURATION

DDP synchronizes gradients at every optimizer update. DiLoCo uses local AdamW and a Nesterov outer optimizer, with $H_{\text{fix}} = 500$ for C4 and $H_{\text{fix}} = 100$ for SFT. DiLoCo follows the configuration reported in prior work.

For QSR (Gu et al., 2024), we follow the inner learning rate division at the start of an interval, square the ratio, and round down, with a minimum of 500 steps. Warm-up intervals remain fixed at 500 steps. Linear-Interval (Spiridonoff et al., 2021) uses intervals of 10, 20, ..., 990 steps, giving 100 scheduled synchronizations within the 50,000 step budget. This matches the same number of synchronization compared with DiLoCo. Both baselines use local AdamW with direct model averaging and retain worker-local optimizer states.

Table 12: AutoLoCo configuration. Shared settings apply to both tasks.

Setting	C4	UltraChat
Base token horizon H_{base}	500	100
Token-horizon bounds $[H_{\min}, H_{\max}]$	$[250, 750]$	$[50, 150]$
Horizon quantum q_H	50	10
Base outer LR $\eta_{\text{base}}^{\text{out}}$	0.7	0.7
Base outer momentum μ_{base}	0.7 / 0.9	0.7
Shared setting	**Value**	
Token-density EMA coefficient β	0.9	
Mapper / reference bootstrap intervals	3 / 2	
Candidate / moderate / severe thresholds	1.5 / 2.5 / 4.0	
Candidate LR-exposure multiplier	1.15	
Moderate / severe reduction divisors	1.20 / 1.50	
Momentum bounds $[\mu_{\min}, \mu_{\max}]$	$[0.45, 0.90]$	
Maximum outer-step scale $\kappa_{\max}$	1.6	

E.3 UltraChat Fine-Tuning and Evaluation Configuration

Training settings. All runs in Section 4.2 use Llama-3.1-8B, a global batch of 32, and 6,490 optimizer steps. The 8 workers comparison uses 8 DDP replicas or 8 independent local workers. The larger comparison uses 16 local workers. DiLoCo synchronizes every 100 local steps and uses outer momentum 0.9. AutoLoCo includes pseudo-gradient normalization, adaptive momentum and learning-rate correction.

Downstream evaluation. IFEval (Zhou et al., 2023) and IFBench (Pyatkin et al., 2025) report prompt-level strict and loose accuracy, respectively. MMLU-Pro (Wang et al., 2024a) uses 5-shot generation and reports accuracy. BBH (Suzgun et al., 2023) uses 3-shot chain-of-thought prompting and reports micro-averaged exact match of the generated answers. HumanEval+ (Liu et al., 2023) reports plus pass@1 across 164 tasks, with one greedy generation per task and both the base and additional tests required to pass. All downstream scores are expressed as percentages, and differences in the main text are percentage points.

E.4 Ablation Configuration

The experiment in Section 4.3 uses 4 workers on C4, a global batch of 512, and 50,000 optimizer steps. All configurations share the initialization, training data, and inner AdamW recipe, with seed 42. DiLoCo synchronizes every $H = 500$ local steps. For a controlled comparison, we set DiLoCo's outer momentum to 0.7, the same value used by AutoLoCo μ_{base}, instead of the 0.9 adopted in the earlier experiments. The adaptive-interval variant enables horizon selection and token mapping while retaining the base outer optimizer. The intervals + momentum variant additionally adjusts momentum retention and keeps the outer learning rate at its base value. Full AutoLoCo also enables the bounded outer-step adjustment.

F Additional Results of Pre-training

F.1 Training Trajectories and Thresholds

At 25%, 50%, 75%, and 100% of the budget, AutoLoCo's loss minus DiLoCo's is $+0.0187$, $+0.0026$, -0.0060, and -0.0161, respectively. Each value uses the 1,000 steps ending at that progress point. Table 2 reports the first crossings of training-loss thresholds under this aggregation. AutoLoCo reaches 2.85 at step 37,336, compared with 38,217 for DiLoCo.

Table 13: Control accounting over 73 AutoLoCo rounds. Each block uses the maximum worker duration per round. Scalar bytes are included in endpoint accounting.

Block	Cost per round
Preparation + local drift statistics	1,126.07 ms
Aggregated gradient norm for C_t	111.81 ms
Outer preparation	100.70 ms
Post-optimizer scalar/control span	1,090.98 ms
Total	2,429.55 ms
Scalar contributions (worker payload)	256 B
Control accounting (all endpoints)	40.56 KiB

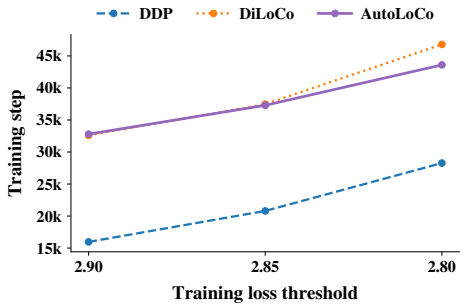


Figure 4: Training steps at loss thresholds of 2.90, 2.85, and 2.80. It shows the number of training steps required to reach each loss value.

F.2 Executed Intervals and Outer Settings

AutoLoCo adopts five candidate horizons and executes the following sequence, written as executable step horizon times number of intervals: $500\times7, 550\times5, 600\times5, 650\times5, 700\times5, 750\times45, 250\times1$. The 73 intervals sum to 50,000 steps. The last 250 steps are the remaining training budget. These runs exercise horizon growth and operation at the upper bound. Linear Interval executes horizons of 10, 20, . . ., 990 steps, followed by a 500-step remainder. QSR executes 41 intervals, with median horizon 603.

G Additional Experiments of Low-precision communication

Communicating with lower precision and reducing the frequency of synchronization are complementary ways to reduce communication overhead. We combine AutoLoCo and INT8 in the 16-worker UltraChat experiment from Section 4.2.

G.1 Low-precision communication Configuration

Codec and aggregation. Each tensor part is divided into contiguous blocks of up to 4,096 elements. For a nonzero block, the FP32 scale is its maximum absolute value divided by 127. Elements are divided by this scale, rounded to the nearest integer with ties to even, and clipped to $[-127, 127]$. A zero block uses scale 1. Scale bounds prevent underflow to zero and overflow during reconstruction. Decoding multiplies the signed INT8 values by the block scale in FP32. The codec uses no error feedback. Worker contributions are decoded and averaged in FP32; the mean is then encoded once for distribution. The worker responsible for reducing a tensor part applies the same encoding and decoding to its own contribution and receives the same decoded mean as the other workers.

Training and controller. The INT8 runs retain the communication and evaluation recipe in Appendices E.1 and E.3. DiLoCo uses 100 local steps per interval. AutoLoCo uses pseudo-gradient normalization, adaptive momentum and learning-rate correction. Model computation remains in BF16, while master parameters, AdamW states, the outer reference, and Nesterov momentum remain in FP32. The controller measures D_t from local pseudo-gradients before quantization and computes C_t using the decoded aggregate. AutoLoCo uses the same horizon controller as in the BF16 runs.

G.2 Compatibility with Low-precision communication

With INT8 communication, AutoLoCo reduces total logical payload by 24.6% relative to DiLoCo while improving test NLL from 0.9016 to 0.8772 (Table 14). For both methods, test NLL differs from the corresponding BF16 result by less than 0.0001 (Table 4). Combining AutoLoCo with INT8 reduces total logical payload by 62.3% relative to DiLoCo with BF16 communication.

Table 14: INT8 pseudo-gradient communication with 16 workers on UltraChat. Optimizer settings follow Appendices E.1–E.3. INT8 applies to pseudo-gradient communication.

			Logical payload (GB)		UltraChat test	
Method	Train loss ↓	Syncs ↓	Per sync	Total	NLL ↓	PPL ↓
DiLoCo (INT8)	0.9004	65	8.038	522.477	0.9016	2.4636
AutoLoCo (INT8)	0.8761	49	8.038	393.867	0.8772	2.4040