LAYER-WISE ADAPTIVE GRADIENT COMPRESSION WITH GENERALIZED ERROR ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reducing gradient communication is critical for large-scale data-parallel training. While layer-wise adaptive compression schemes can outperform uniform layer-wise ones, they typically rely on L_2 distance in gradient space as the error measure with a fixed measurement schedule. We propose Generalized Gradient Compression (GenComp), which generalizes layer-wise adaptive compression along three axes: 1) the space in which compression error is measured, on gradients or on latent activations obtained by simulating gradient updates and forward passes; 2) the error measure; 3) the frequency of measurement. We show that gradient- and activation-based error measurements, in tandem with error measures addressing information loss and different frequencies of measurement, induce distinct layer-wise compression dynamics, and can improve task performance and compression ratio. Across sparsification, quantization, and low-rank compression schemes applied to different model architectures, GenComp achieves significantly higher compression ratios than recent layer-wise adaptive schemes while retaining similar performance, which leads to reduced end-to-end training time for larger models.

1 INTRODUCTION

Scaling deep neural networks has led to major advances across a wide range of applications, but also to rapidly growing computational and communication costs. In large-scale distributed and data-parallel training, models are replicated across devices, each processing a shard of data and synchronizing gradients via collective operations such as AllReduce (Wang et al., 2023). While simple and widely adopted, this synchronization introduces significant communication and synchronization overhead, which becomes a major bottleneck for wall-clock time at scale (Ben-Nun & Hoefler, 2019). Gradient compression schemes mitigate this bottleneck by reducing the number of communicated bits between devices, using techniques such as quantization, sparsification, and low-rank approximation (Alistarh et al., 2017; Vogels et al., 2019; Shi et al., 2019b). However, most schemes are used with global compression configurations or with the same configuration uniformly across layers (Alimohammadi et al., 2024). This ignores the substantial heterogeneity in layer sizes and sensitivity to approximation error (Achille et al., 2019; Dupuis et al., 2020), which can be exploited to achieve more compression while maintaining model performance.

Recent advances in layer-wise adaptive gradient compression dynamically adapt compression parameters per layer across training (Alimohammadi et al., 2024; Nguyen et al., 2025; Essoullami et al., 2026). They typically optimize the trade-off between communication cost and compression error by selecting an optimal combination of layer-wise compression configurations. While the communication cost per iteration can be estimated accurately by counting transmitted elements, it is not as clear how to measure the compression error. Methods such as L-GreCo (Alimohammadi et al., 2024) and Quantized Optimistic Dual Averaging (QODA) (Nguyen et al., 2025) measure layer-wise compression error as the squared L_2 norm between compressed and uncompressed gradients. This heuristic error measure captures the magnitude of the gradient perturbation, but not its direction or distribution discrepancy. It also does not address the information loss in the layer's activations due to compression.

We study how the choice of compression error measurement affects layer-wise adaptive compression along three axes: the space in which error is measured, the error measure, and how often it is measured. First, in addition to measuring error on gradients, we introduce online error measurement

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.

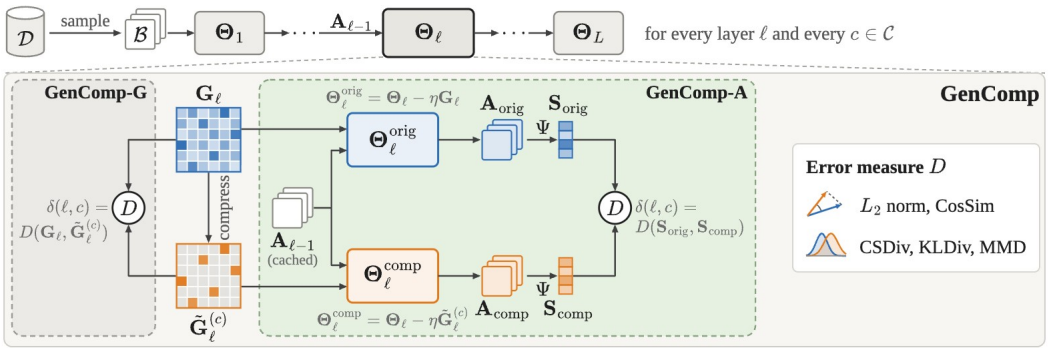


Figure 1: Overview of GenComp and its activation- and gradient-based variants. For layer ℓ and compression configuration c, GenComp estimates the compression error $\delta(\ell, c)$ either directly on the uncompressed and compressed gradients (GenComp-G), or by simulating parameter updates with both gradients, passing the cached input $\mathbf{A}_{\ell-1}$ through each updated copy of the layer, and measuring the discrepancy between the resulting activations (GenComp-A). The notation is introduced in Section 4.

on latent activations by simulating gradient updates and forward passes at different compression levels. Activations retain per-sample structure and expose distributional discrepancies in latent features that are averaged out in minibatch gradients. Second, beyond the L_2 norm, we consider information loss error measures, applied both in the gradient and activation space. Third, we vary how often errors are measured, from cyclic, frequent measurements down to a single measurement early in training. Building on compression parameter selection with dynamic programming, we compare these combinations in terms of their resulting compression profiles, compression ratio, task performance, and training time.

Our contributions are the following: **1)** We propose GenComp, a framework for layer-wise adaptive compression that generalizes existing schemes in the space where compression error is measured, on gradients or on latent activations, the error measure, and the frequency of measurement. **2)** We demonstrate that compression errors can be measured once early in training, rather than cyclically throughout, while still achieving full accuracy recovery and high compression ratios at a reduced computational cost, depending on the choice of measure and measurement space. **3)** We experimentally evaluate GenComp in data-parallel training, and achieve improvements over existing baselines in task performance, compression ratios and training time. **4)** We theoretically motivate layer-wise activation-based distributional discrepancy measures as an upper bound on model output distribution discrepancies, and establish convergence guarantees for layer-wise contracting compression operators such as those used in GenComp.

2 Related Work

Layer-wise adaptive gradient compression. Gradient compression schemes are typically applied uniformly across all layers of the neural network. This neglects the inherent heterogeneity of deep learning models, where layers differ vastly in parameter count and sensitivity to noise. Shi et al. (2020) identified this limitation and proposed LAGS-SGD, which selects different sparsification thresholds for different layers. Accordion (Agarwal et al., 2021) adapts compression parameters to critical learning regimes in which compression-induced gradient errors can have irrecoverable detrimental impact on model training (Achille et al., 2019). Alimohammadi et al. (2024) introduced L-GreCo, which applies to quantization, sparsification and low-rank approximation schemes, and can be combined with other methods. It formulates the compression parameter selection as a convex optimization problem; however, it relies on the L_2 norm of the gradient error as its optimization measure, which may not preserve the functional behavior of the layer. Nguyen et al. (2025) proposed QODA, which incorporates the statistical heterogeneity across layers to dynamically adapt quantization levels. Essoullami et al. (2026) proposed LEGACY, an inexpensive but non-adaptive layer-wise compression parameter scheduler based on layer size and training step heuristics. [1]

[1] A complete list of related work with detailed descriptions is provided in Appendix A.

3 PROBLEM SETUP AND LAYER-WISE ADAPTIVE GRADIENT COMPRESSION

Let $[n]$ denote the set $\{1, \ldots, n\}$ with $n \in \mathbb{N}$. We consider a distributed and synchronous data-parallel learning setting with $P \in \mathbb{N}$ distinct computation nodes. At each iterate x_t, each node $p \in [P]$ computes a stochastic gradient $G^p(x_t)$ of a deep neural network on a local batch of data. These gradients are then compressed by a compression operator $S : \mathbb{R}^d \to \mathbb{R}^d$, and averaged to perform one stochastic gradient update across nodes. For the rest of this work, except for Section 5 and Appendix B, we drop the index p and iteration x_t for notational simplicity. For a deep neural network with L layers, building on layer-wise and adaptive compression frameworks (Agarwal et al., 2021; Alimohammadi et al., 2024; Nguyen et al., 2025; Essoullami et al., 2026) each layer ℓ along with a compression configuration $c \in \mathcal{C}$ incurs some communication cost $\omega(\ell, c)$, *i.e., compressed gradient size*; and compression error $\delta(\ell, c)$, *i.e., information loss with respect to the uncompressed layer activation*. Our goal is to optimize per-layer compression schemes to minimize communication cost under a bound on cumulative compression error across layers. Given a global error budget $\mathcal{E}_{\max} > 0$, we address the following constrained optimization problem:

$$\min_{\mathbf{c}} \sum_{\ell=1}^{L} \omega(\ell, c_\ell) \quad \text{s.t.} \quad \sum_{\ell=1}^{L} \delta(\ell, c_\ell) \leq \mathcal{E}_{\max}. \tag{1}$$

The error budget is anchored to a reference, uniformly applied compression configuration c_{ref} that is known to preserve model accuracy (Alistarh et al., 2017; Lin et al., 2018; Renggli et al., 2019; Vogels et al., 2019; Markov et al., 2022), setting $\mathcal{E}_{\max} = \sum_{\ell=1}^{L} \delta(\ell, c_{\text{ref}})$ to guarantee that any heterogeneous per-layer scheme does not exceed a validated error level. By discretizing the errors, an optimal solution can be found using dynamic programming (Alimohammadi et al., 2024). Further details are provided in Appendix D.

4 GENERALIZED GRADIENT COMPRESSION

In the following, we present two variants of the GenComp framework: GenComp-A and GenComp-G, which measure layer-wise error on activations and gradients, respectively. While solving Equation (1) guarantees an optimal solution for *selecting* $\mathbf{c}^*$ given an error constraint, it remains agnostic to how the error values are calculated. To adapt layer-wise configurations to the learning dynamics of uncompressed updates, GenComp-A quantifies information loss on latent representations within the model. It simulates a parameter update via SGD on each layer, and uses cached latent features to simulate a forward pass through each updated layer. GenComp-A measures *compression error as a discrepancy between the resulting layer activations with and without compression*. In contrast, GenComp-G estimates information loss by *directly measuring the discrepancy between compressed and uncompressed gradients*. For both GenComp variants, we introduce new error measures to capture *geometric and distributional discrepancies*.

GenComp-A: activation-based compression error measurement. Let $\mathcal{B} \subset \mathcal{D}$ denote a minibatch sampled from the dataset. To minimize computational overhead, GenComp-A leverages activations from the forward pass of the final training step prior to compression error estimation. Let $\mathbf{A}_{\ell-1}$ denote the output activations of layer $\ell - 1$ (and thus the input to layer ℓ) during a forward pass of minibatch $\mathcal{B}$. GenComp-A uses cached activations $\mathbf{A}_{\ell-1}$ as input to simulate a forward pass through layer ℓ. For layer ℓ with parameters $\mathbf{\Theta}_\ell$ and learning rate η, let $\mathbf{G}_\ell$ denote the accumulated uncompressed gradient and $\tilde{\mathbf{G}}_\ell^{(c)}$ denote its compressed-decompressed variant. We simulate the two distinct updates:

$$\mathbf{\Theta}_\ell^{\text{orig}} = \mathbf{\Theta}_\ell - \eta \mathbf{G}_\ell, \quad \mathbf{\Theta}_\ell^{\text{comp}} = \mathbf{\Theta}_\ell - \eta \tilde{\mathbf{G}}_\ell^{(c)}.$$

We then compute the latent features of layer ℓ by applying the updated parameters on cached inputs:

$$\mathbf{A}_{\text{orig}} = f_{\mathbf{\Theta}_\ell^{\text{orig}}}(\mathbf{A}_{\ell-1}), \quad \mathbf{A}_{\text{comp}} = f_{\mathbf{\Theta}_\ell^{\text{comp}}}(\mathbf{A}_{\ell-1}).$$

GenComp-A's compression error $\delta(\ell, c)$ efficiently estimates the discrepancy between $\mathbf{A}_{\text{orig}}$ and $\mathbf{A}_{\text{comp}}$.

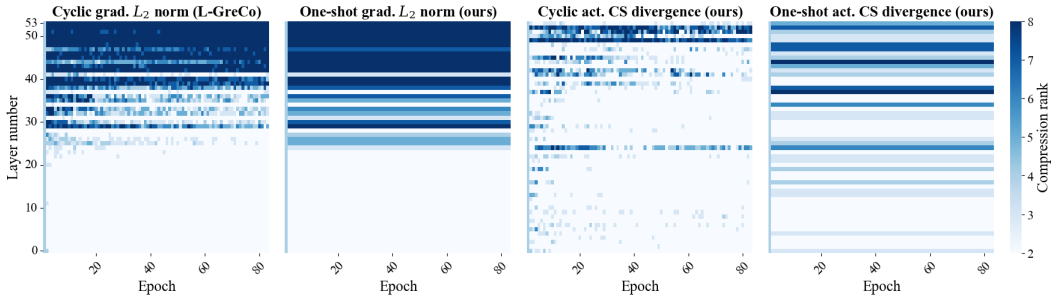


Figure 2: Evolution of compression profiles from training ResNet50 on ImageNet with low-rank gradient compression. We demonstrate visually that different choices of activation vs. gradient measurement, error measure, and cyclic vs. one-shot estimation produce significantly different profiles.

Error measures and error normalization. In addition to the L_2 norm, GenComp works with a general nonparametric and parametric discrepancy measure between tensors. Let $D(\cdot, \cdot)$ denote a generic discrepancy measure between two activation tensors. Suppose layer ℓ is convolutional and outputs $\mathbf{A}_{\text{orig}}$ and $\mathbf{A}_{\text{comp}}$ are high-dimensional feature maps. We apply a downsampling operator Ψ, specifically, the average pooling operator, to fully downsample the feature maps along the height and width dimensions. This reduces each sample to a vector of activation scores in the channel dimension, the scores being averaged over the height and width dimensions: $\mathbf{S}_{\text{orig}} = \Psi(\mathbf{A}_{\text{orig}})$, $\mathbf{S}_{\text{comp}} = \Psi(\mathbf{A}_{\text{comp}})$. We do this to 1) reduce dimensionality of the activations and speed up compression error computations during training, and to 2) flatten activations to per-sample vectors. Such a mapping allows us to interpret the activation scores $\mathbf{S}_{\text{orig}}$ and $\mathbf{S}_{\text{comp}}$ as samples drawn from two distributions of per-channel activations. $D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}})$ measures *how the distribution of averaged responses across channels in one layer changes due to gradient compression with a given configuration compared to that of the communication-heavy uncompressed baseline*. For other types of layers, we flatten the activations along all except the batch dimension to obtain per-sample activation scores: $\mathbf{S}_{\text{orig}} = \text{Flatten}(\mathbf{A}_{\text{orig}})$, $\mathbf{S}_{\text{comp}} = \text{Flatten}(\mathbf{A}_{\text{comp}})$.

We consider a number of popular measures with various appealing properties to measure the amount of information loss for output prediction (see Figure 2). With GenComp-G, we evaluate the same measures applied directly on gradients; given the original and compressed-decompressed gradients $\mathbf{G}$ and $\tilde{\mathbf{G}}$, we then compute $D(\mathbf{G}, \tilde{\mathbf{G}})$ instead of $D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}})$. We evaluate the following measures to compare the outputs of our simulated SGD updates and forward passes: L_2 norm, cosine similarity, Cauchy-Schwarz (CS) divergence (Yu et al., 2024), maximum mean discrepancy (MMD) (Gretton et al., 2012), and Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951). Their full definitions are given in Appendix D.3. For layer ℓ and configuration c, the activation-based and gradient-based GenComp variants are given by $\delta(\ell, c) = D(\mathbf{S}^{\ell}_{\text{orig}}, \mathbf{S}^{\ell,c}_{\text{comp}})$ and $\delta(\ell, c) = D(\mathbf{G}_{\ell}, \tilde{\mathbf{G}}^{(c)}_{\ell})$, respectively.

One-shot parameter selection schedule. Current layer-wise compression schemes select an optimal compression profile $\mathbf{c}^*$ at scheduled training steps to dynamically adapt compression to how layer-wise compression sensitivity evolves across training (Alimohammadi et al., 2024; Essoullami et al., 2026). Performing the simulated SGD step and forward pass in GenComp for every candidate configuration $c \in \mathcal{C}$ introduces additional computational overhead compared to measuring compression error on gradients directly, making such per-epoch profiling more computationally expensive. We analyze how the compression profile evolves across training and observe that when using gradient L_2 norm as error measure, the configurations selected are highly stationary throughout training (see Figure 2). Motivated by this observation, we argue that infrequent profile updates, ranging from lowering the update frequency to even a single, *one-shot* profile selection, can be sufficient in practice. We show experimentally that performing a one-shot compression profile selection after an early warmup phase, and reusing the resulting compression profile for the remainder of training, can attain compression characteristics comparable to, and in some cases better than, those obtained by cyclic profile updates (Section 6). This sparse scheduling substantially reduces the total computational cost of GenComp, which is especially important for GenComp-A which requires a simulated SGD step and forward pass. It is also important when using more computationally demanding error measures,

like KDE-based measures. At the same time, we find that some of our proposed measures, e.g., CS divergence on activation scores, induce compression profiles that change more dynamically over the course of training (Figure 2). For these highly dynamic measures, one-shot profile selection can be less advantageous, as it sacrifices the benefit of maintaining a compression profile that closely tracks the model's learning dynamics. In such settings, a cyclic selection schedule may be preferable, though it may still be sufficient to schedule selections much less frequently than once every epoch.

Algorithmic overview. Each step of calculating the activation-based compression error $\delta(\ell, c)$ to optimize a compression profile is detailed in Algorithm 1. GenComp-G is obtained by computing $\mathbf{E}_{\ell,c} \leftarrow D(\mathbf{G}_\ell, \tilde{\mathbf{G}}_\ell^{(c)})$ instead of activation scores. We assume we have chosen a discrepancy measure D, and that we have already computed a cost matrix $\mathbf{W} \in \mathbb{R}^{\{1,\dots,L\}\times\mathcal{C}}$ containing the communication cost $\omega(\ell_i, c_j)$ of each layer ℓ_i when compressed with configuration c_j. We apply dynamic programming to find an optimal configuration profile $\mathbf{c}^*$. Note that $\mathbf{S}_{\text{orig}}$ is only computed once per layer, regardless of the number of compression choices $|\mathcal{C}|$. This algorithm can be run cyclically across training, or in a few-shot or one-shot manner as described earlier.

GenComp can be viewed as a generalization of layer-wise adaptive baselines including L-GreCo: it is recovered as the special case where (i) the error measure D is the L_2 norm, (ii) compression error is measured on gradients $\mathbf{G}_\ell$ rather than on activations, and (iii) the compression profile is updated once every epoch. Moreover, non-adaptive, uniform layer-wise compression schemes, are also special cases of GenComp. Consider the class of methods that fix a single compression configuration c and apply it uniformly across all layers across training (e.g., uniform per-layer quantization, uniform per-layer top-k sparsification, and uniform per-layer low-rank approximation). Within our framework, such schemes apply a uniform configuration $\mathbf{c}^* = (c, \dots, c)$ across layers with a single compression profile $|\mathcal{C}| = 1$.

Layer-wise error propagation. In the standard setting described above, GenComp-A measures $\delta(\ell, c)$ for each layer in isolation: the cached input $\mathbf{A}_{\ell-1}$ stems from a forward pass of the uncompressed model, so the estimate does not reflect that layers $1, \dots, \ell-1$ are updated with compressed gradients. GenComp-A offers a variant that estimates this propagated error. Instead of caching per-layer activations, we cache the minibatch $\mathcal{B}$ and simulate updating every layer $\ell' < \ell$ with its gradient compressed uniformly at the reference configuration, $\mathbf{\Theta}_{\ell'} - \eta\tilde{\mathbf{G}}_{\ell'}^{(c_{\text{ref}})}$ for every $\ell' < \ell$. A single forward pass through this simulated model yields the input $\tilde{\mathbf{A}}_{\ell-1}$ to layer ℓ where all previous layers are updated with uniformly compressed gradients, and $\delta(\ell, c)$ is computed as before with $\tilde{\mathbf{A}}_{\ell-1}$ in place of $\mathbf{A}_{\ell-1}$. Measuring the exact layer-wise propagation error for each possible configuration c would require simulating a forward pass for every combination of per-layer choices. The computational cost would then scale exponentially with L. Assuming uniformly compressed layers up to layer $\ell-1$ instead yields a cheap proxy, costing only one additional forward pass per profile selection, that captures the effect of all preceding layers being compressed closer to the effect of the final profile $\mathbf{c}^*$. Since c_{ref} is assumed to preserve model accuracy (Section 3), the simulated error propagation represents a plausible, non-degrading compression error estimation of the earlier layers.

5 Theoretical motivation and guarantees

Layer-wise distributional activation errors bound output distribution discrepancy. Our motivation for using distributional measures stems from knowledge distillation, in which the objective is typically to match the output distributions of student and teacher models (Hinton et al., 2015). By measuring layer-wise discrepancies in GenComp-A, we approximate the discrepancy on output distributions imposed by gradient perturbations due to compression. In Appendix C, we show that for a given profile $\mathbf{c}$, the MMD between models updated with compressed vs. uncompressed gradients is upper-bounded by the layer-wise MMD between activations, with CS divergence controlling the angular component (Theorem C.1). These are the exact activation-based discrepancies GenComp-A measures. GenComp-A thus provides an empirical estimate of the upper bound on output distributional perturbation caused by gradient compression given $\mathbf{c}$. Furthermore, the total variation distance of the marginal predicted class probabilities between models updated with compressed vs. uncompressed gradients is upper-bounded by the MMD between model outputs updated as described above, and thus also by the layer-wise discrepancies measured in GenComp-A (Corollary C.3).

Convergence guarantees for layer-wise gradient compression. We provide convergence guarantees for layer-wise contracting compression operators used with GenComp, including layer-wise sparsification operators (Shi et al., 2020; Alimohammadi et al., 2024) and quantization operators (Alistarh et al., 2017; Nguyen et al., 2025) under appropriate scaling, as well as exact low-rank approximation operators (Vogels et al., 2019), in a distributed data-parallel setting as described in Section 3. In this section, we provide the key theoretical result and defer detailed theory and proofs to Appendix B.

Definition 1 (Contracting compression operator)**.** A possibly randomized operator $S : \mathbb{R}^d \to \mathbb{R}^d$ is a *contracting compression operator* if there exists a $c \geq 1$ such that $\mathbb{E}\|v - S(v)\|^2 \leq (1 - 1/c)\,\|v\|^2$ for all $v \in \mathbb{R}^d$.

Definition 2 (Layer-wise contracting compression operator)**.** A possibly randomized operator $S : \mathbb{R}^d \to \mathbb{R}^d$ is a *layer-wise contracting compression operator* if 1) there exists a partition of the parameter space dimension $\{1, \ldots, d\}$ into disjoint subsets $I_1 \dot\cup \ldots \dot\cup I_L$, such that for each layer ℓ, there is an operator $S^{(\ell)} : \mathbb{R}^{I_l} \to \mathbb{R}^{I_l}$ with $S^{(\ell)}(x_{I_\ell}) = S(x)_{I_\ell}$ for all $x \in \mathbb{R}^d$, and 2) $S^{(\ell)}$ is a contracting compression operator with constant $c^{(\ell)} \geq 1$.

In Proposition B.3 and Proposition B.4, we establish that *layer-wise top-k sparsification, layer-wise random-k sparsification, and, under appropriate scaling, layer-wise quantization are layer-wise contracting compression operators*. It can also be established that exact low-rank projection is a contracting compression operator; see Vogels et al. (2019, Remark 1). We establish convergence results for *convex and non-convex optimization with layer-wise contracting compression operators* in Appendix B. To ensure convergence while using biased compression operators, each computation node stores the accumulated differences between the compressed and uncompressed gradients in an error feedback term m^p (Karimireddy et al., 2019). It is also necessary to make some analytical assumptions on the compressed gradients, to ensure that the mean of the transmitted compressed gradients is not too dissimilar to the compressed mean of their uncompressed counterparts. Such assumptions are key in proving that a general compression scheme converges, as they rule out the unlikely cases where compression errors at different computation nodes amplify compared to the compressed mean (Alistarh et al., 2018). There are two possible choices of assumptions that lead to our results (Assumption B.5 and Assumption B.6); we provide one *assumption choice* here for the non-convex optimization, which *generalizes that of Alistarh et al. (2018).*

Assumption 3. There exists some $\xi > 0$ such that for the iterates (x_t) of the algorithm it holds for each layer $\ell \in [L]$ that

$$\mathbb{E}\Big\|S^{(\ell)}\Big(\frac{1}{P}\sum_{p=1}^{P}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t))\Big) - \frac{1}{P}\sum_{p=1}^{P} S^{(\ell)}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t))\Big\| \leq \xi\|\alpha_t G^{(\ell)}(x_t)\|,$$

with the expectation taken with respect to the randomness in $S^{(\ell)}$.

Theorem 4 (Non-convex and smooth optimization). *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a β-smooth objective function with a global minimum at x^*. Suppose the layer-wise gradients of the iterates x_t of an error feedback compensated SGD in a distributed setting with a compression profile $\mathbf{c}^*$ as an output of GenComp uniformly satisfy $\mathbb{E}\|G^{(\ell)}(x_t)\|^2 \leq M^{(\ell)}$ for all $\ell \in [L]$. Let the learning rate $(\alpha_t)_{t\in\mathbb{N}_0}$ satisfy $\alpha_t^2/\alpha_{t+1}^2 \leq 4/(4 - 1/c_{\max})$ where $c_{\max} := \max_{\ell\in[L]} c^{(\ell)}$ denotes the maximal compression rate across layers. Under Assumption 3 or Assumption B.6, the iterates satisfy*

$$\frac{\sum_{t=0}^{T-1}\alpha_t\mathbb{E}\|\nabla f(x_t)\|^2}{\sum_{t=0}^{T-1}\alpha_t} \leq \frac{4(f(x_0) - f(x^*))}{3\sum_{t=0}^{T-1}\alpha_t} + \frac{2\beta\sum_{t=0}^{T-1}\alpha_t^2\sum_{\ell=1}^{L}M^{(\ell)}}{3\sum_{t=0}^{T-1}\alpha_t} + \frac{\tilde{B}\beta^2\sum_{t=0}^{T-1}\alpha_t^3\sum_{\ell=1}^{L}c^{(\ell)2}M^{(\ell)}}{3\sum_{t=0}^{T-1}\alpha_t},$$

where $\tilde{B} = 16(6\xi^2 + 7)$ under Assumption 3 and $\tilde{B} = 32$ under Assumption B.6.

Theorem 4 proves that any compression profile $\mathbf{c}^*$ that is an output of Algorithm 1 yields iterates x_t which converge with the rates given in Corollary B.10. The rates on these bounds *improve on the previously published result of Shi et al. (2020, Corollary 2) in that the third term in our bounds depends on the square of the maximal compression ratio instead of the cube*. Our result for convex objectives (Theorem B.7) is also *under weaker assumptions than Stich et al. (2018)*, whose results

were obtained under the additional assumption of L-smoothness. Furthermore, our results *generalize to the layer-wise structure* that S is applied on. We use that different layers can have different values of $c^{(\ell)}$ to obtain tighter bounds depending on the value of $M^{(\ell)}$. We provide the convergence theorem for convex problems in Theorem B.7, as well as upper bound rates for both theorems in Corollary B.8 and Corollary B.10, and proofs in Appendix B.2 and Appendix B.3.

6 Experimental evaluation

Setup and metrics. We evaluate GenComp in distributed data-parallel training with layer-wise adaptive low-rank approximation, sparsification, and unbiased quantization. We train on several combinations of models and datasets: ResNet18 (He et al., 2016) on CIFAR-100 (Krizhevsky, 2009) (60,000 samples, 100 classes) for 200 epochs, ResNet50 on ImageNet (Deng et al., 2009) (1.2M samples, 1,000 classes) for 90 epochs, and Transformer-XL on Wikitext-103 (Merity et al., 2017) (103M tokens) for 15,000 iterations. Unless explicitly stated otherwise, all runs are performed in a distributed setting with 4 nodes of 4 GPUs each. We provide details of the hardware and hyperparameter configurations used in Appendix E.3. We report the following metrics (all results are given as mean $\pm$ standard deviation over 3 runs, unless stated otherwise): *Task performance:* top-1 accuracy for all image datasets, top-5 accuracy for ImageNet (reported in %), and perplexity for Wikitext-103. *Compression ratio:* cumulative uncompressed over compressed gradient size ($\frac{\#\text{Uncompressed elements}}{\#\text{Compressed elements}}$) which measures communication savings. *Total time:* end-to-end wall-clock training time measured in seconds. Δ*Time:* change in training time relative to the gradient L_2 norm baseline. In the following, we use the standard non-error propagation setup of GenComp, and provide results for the error propagation counterpart in Appendix E.1.

Compression schemes, baselines and parameter ranges. We evaluate GenComp with low-rank approximation, sparsification, and unbiased quantization. As concrete instances, we use PowerSGD (Vogels et al., 2019) for low-rank approximation, top-k sparsification (Stich et al., 2018) for sparsification, and QSGD (Alistarh et al., 2017) for unbiased quantization. Baselines use uniform layer-wise compression and cyclic error estimation with the gradient L_2 norm. For both baselines, we use the default compression parameters of Alimohammadi et al. (2024): rank 4 and 32 on the image datasets and Wikitext-103, respectively, for low-rank approximation, top 1% on image datasets and 10% on Wikitext-103 for top-k sparsification, and 4 bits on all datasets for unbiased quantization. After a fixed warm-up, cyclic error measurements are performed once per epoch for ResNet models, and once every 5,000 steps for Transformer-XL; one-shot variants estimate errors once after the same warm-up time. We include results without compression in Appendix E.1. For low-rank approximation and unbiased quantization, we use the parameter ranges of Alimohammadi et al. (2024). For layer-wise top-k sparsification, we restrict to a smaller, evenly spaced subset of values, listed in Appendix E.3.

Layer-wise adaptive low-rank approximation. We present results from using GenComp with PowerSGD low-rank approximation when training Transformer-XL on Wikitext-103 in Table 1. In Appendix E.1, we also show results from low-rank approximation experiments training ResNet18 on CIFAR-100, where we add Accordion and LEGACY results for comparison (Table 4), and ResNet50 on ImageNet (Table 5), as well as results for GenComp-A's error propagation-aware variant (Table 7), and Transformer-XL/Wikitext-103 results in 8-node experiments (Table 3). From Table 1, we observe that most GenComp variants increase the *compression ratio* beyond both baselines, while *perplexity* generally remains similar or within 0.5 points of them. The largest gains come from KL divergence and from activation-based CS divergence in the one-shot setting, which roughly double the compression ratio of L-GreCo, whereas activation-based CosSim and MMD are more conservative, selecting compression ratios between the two baselines while matching the best perplexity. One-shot variants perform on par with their cyclic counterparts in both compression and perplexity. The higher compression ratios generally translate into measurable reductions in *training time*: for Transformer-XL, every GenComp variant trains faster than both baselines, and the most aggressive ones reduce total training time by more than 10% relative to the gradient L_2 norm baseline, giving significantly increased timing gains compared to ResNet models. We attribute this to Transformer-XL being a considerably larger model (192M parameters) than the ResNet models (ResNet18 and ResNet50 have 12M and 26M parameters, respectively), making the amount of uncompressed gradient elements to

Table 1: Results for GenComp with low-rank approximation training Transformer-XL on Wikitext-103. The best results are highlighted in bold, and the second best results are underlined.

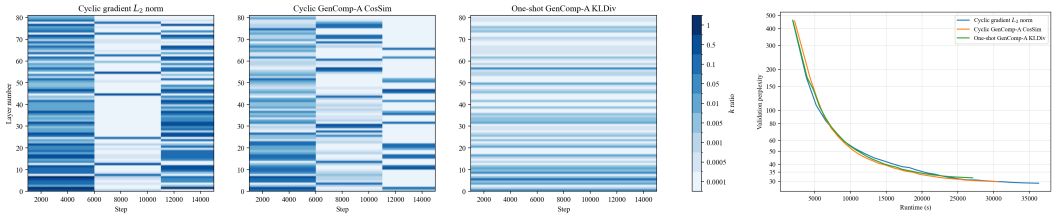
	Selection	Error measure	Perplexity ↓	Comp. ratio ↑	Total time (s) ↓	ΔTime
Baseline	—	Uniform	28.12 ± 0.16	14.1 ± 0.0	34510 ± 298	2.5% ↑
	Cyclic	L_2 Norm	$\mathbf{28.02 \pm 0.04}$	23.3 ± 0.1	33660 ± 1141	—
GenComp-A	Cyclic	L_2 Norm	28.31 ± 0.14	33.1 ± 0.1	32997 ± 200	2.0% ↓
		CosSim	$\underline{28.03 \pm 0.19}$	16.2 ± 0.1	32543 ± 75	3.3% ↓
		CSDiv	28.55 ± 0.06	20.9 ± 0.1	32069 ± 234	4.7% ↓
		KLDiv	28.53 ± 0.02	37.7 ± 0.3	30693 ± 153	8.8% ↓
		MMD	28.22 ± 0.18	16.0 ± 0.2	32835 ± 223	2.5% ↓
	One-shot	L_2 Norm	28.27 ± 0.20	32.1 ± 0.1	32107 ± 121	4.6% ↓
		CosSim	$\mathbf{28.02 \pm 0.14}$	15.8 ± 0.2	33033 ± 414	1.9% ↓
		CSDiv	28.81 ± 0.18	49.0 ± 4.0	$\underline{30092 \pm 456}$	10.6% ↓
		KLDiv	28.38 ± 0.24	38.9 ± 1.5	32497 ± 2221	3.5% ↓
		MMD	28.05 ± 0.21	16.5 ± 0.3	33002 ± 2015	2.0% ↓
GenComp-G	Cyclic	CosSim	28.30 ± 0.10	27.6 ± 0.1	32724 ± 364	2.8% ↓
		CSDiv	28.37 ± 0.09	26.5 ± 1.0	32223 ± 779	4.3% ↓
		KLDiv	28.81 ± 0.19	$\underline{51.2 \pm 0.4}$	$\mathbf{29987 \pm 250}$	10.9% ↓
		MMD	28.26 ± 0.20	27.9 ± 0.0	32751 ± 718	2.7% ↓
	One-shot	L_2 Norm	28.24 ± 0.11	24.6 ± 0.0	33294 ± 400	1.1% ↓
		CosSim	28.33 ± 0.14	27.4 ± 0.1	32138 ± 828	4.5% ↓
		CSDiv	28.22 ± 0.10	25.0 ± 2.0	33511 ± 340	0.4% ↓
		KLDiv	28.82 ± 0.04	$\mathbf{52.2 \pm 0.0}$	30279 ± 293	10.0% ↓
		MMD	28.26 ± 0.07	27.9 ± 0.1	32777 ± 440	2.6% ↓

communicate much larger. As a consequence, reducing communication volume here shortens training more substantially. We also observe similar timing gains in our 8 nodes $\times$ 2 GPUs Transformer-XL/Wikitext-103 experiment (Table 3), suggesting that the improvement carries over to settings with higher inter-node communication, even for the same number of GPUs. We note that at higher GPU counts with a fixed global batch size, communication takes up a larger share of each step compared to computation, and one would expect even higher proportional timing gains with GenComp. Our top-k results on Transformer-XL point to the same improvement from the other direction: given a constant communication bottleneck, the heavier compression achieved by GenComp leads to time savings of up to 39% over the less aggressively compressing gradient L_2 norm baseline.

Layer-wise adaptive sparsification and unbiased quantization. We also use GenComp with layer-wise adaptive sparsification and unbiased quantization; we present Transformer-XL/Wikitext-103 results for top-k sparsification in Table 2, and ResNet18/CIFAR-100 results for both top-k sparsification and unbiased quantization in Table 6, with experiment details in Appendix E.1. For top-k sparsification with Transformer-XL, in an otherwise equal setting as the Transformer-XL low-rank approximation results listed above, GenComp improves the *compression ratio* of the gradient L_2 norm baseline by factors of 2.4$\times$ to 18$\times$ while cutting *training time* by 19–39%. As shown in Figure 3b, cyclic GenComp-A with CosSim strikes a balance that beats the gradient L_2 norm on both convergence and total training time: it converges ahead of the baseline for most of its training at 2.5$\times$ its compression ratio and thus spending almost 1.5 hours less time. One-shot GenComp-A with KL divergence reaches a compression ratio of 195.4, while also staying ahead of the baseline through large parts of training and finishing in roughly 2.5 hours less training time. Other methods achieve even higher compression ratios, though at a cost of higher perplexity. Interestingly, the one-shot variant of the gradient L_2 achieves the best perplexity, but was also the only method slower than its baseline counterpart, which demonstrates the trade-off between convergence and throughput. However, for a higher number of GPUs and assuming a fixed global batch size, or given a larger model, we extrapolate that the same compression ratios would yield improved timing gains,

Table 2: Results for GenComp with top-k sparsification training Transformer-XL on Wikitext-103. The best results are highlighted in bold, and the second best results are underlined.

	Selection	Error measure	Perplexity ↓	Comp. ratio ↑	Total time (s) ↓	ΔTime
Baseline	—	Uniform	<u>28.19 ± 0.12</u>	5.0 ± 0.0	91567 ± 4992	152.2% ↑
	Cyclic	Gradient L_2 Norm	29.02 ± 0.00	24.6 ± 0.1	36311 ± 605	—
GenComp-A	Cyclic	CosSim	30.23 ± 0.44	59.7 ± 6.2	29541 ± 1017	18.6% ↓
		CSDiv	42.59 ± 2.07	100.2 ± 10.0	27063 ± 785	25.5% ↓
		KLDiv	34.10 ± 0.60	66.5 ± 1.6	28494 ± 1065	21.5% ↓
	One-shot	CosSim	29.41 ± 0.15	62.2 ± 3.0	28491 ± 163	21.5% ↓
		CSDiv	47.95 ± 6.45	**465.6 ± 3.2**	**22283 ± 5672**	38.6% ↓
		KLDiv	32.27 ± 0.44	195.4 ± 38.2	26655 ± 649	26.6% ↓
GenComp-G	Cyclic	KLDiv	45.26 ± 0.61	465.0 ± 0.2	<u>24558 ± 765</u>	32.4% ↓
	One-shot	L_2 Norm	**28.02 ± 0.03**	16.8 ± 0.3	44638 ± 488	22.9% ↑
		KLDiv	47.53 ± 6.03	<u>465.2 ± 0.5</u>	25218 ± 1200	30.5% ↓



(a) Selected compression profiles.

(b) Validation perplexity.

Figure 3: Top-k sparsification with Transformer-XL on Wikitext-103. **(a)** Per-layer k ratio chosen for each of the 81 layers at every training step for the gradient L_2 baseline, and GenComp-A with cyclic CosSim and one-shot KLDiv. **(b)** Validation perplexity against wall-clock runtime.

though with similar convergence behavior, in which case faster throughput becomes dominant in the convergence-throughput trade-off.

7 Limitations, Future Work and Conclusion

Compression profile selection schemes related to GenComp measure the error imposed on each layer individually (Alimohammadi et al., 2024), or do not measure it at all (Essoullami et al., 2026). In either case, such schemes do not account explicitly for the likely case that compression in an early layer may induce an error that compounds to subsequent layers, especially under heavy compression. While the error propagation-aware variant of GenComp-A alleviates this by compounding the effect of simulated uniform compression in earlier layers than the one being measured, it still does not capture the exact propagated error for all possible compression profiles in those layers. Future work in compression error estimation should aim at more precisely capturing the compounding effect of layer-wise compression for specific per-layer configurations.

We presented GenComp, a layer-wise adaptive compression framework that generalizes how compression error is measured during profile selection: on gradients or on latent activations, with different error measures, and at different frequencies. Empirically, across multiple models, datasets, and three families of compressors (layer-wise adaptive low-rank approximation, layer-wise adaptive sparsification, layer-wise adaptive unbiased quantization), GenComp achieves significantly higher compression rates than the baselines while retaining similar task performance on low-rank approximation and sparsification experiments, which leads to improved end-to-end training time for larger models. Moreover, we show that for suitable combinations of error measure and measurement space, *one-shot* profile selection can match or surpass cyclic selection in compression ratio and task performance, eliminating overhead of cyclic measurements.

AI USE STATEMENT

In this work, we used generative AI tools for assisting in developing and writing the claims and proof in Appendix C. We have not used generative AI tools for developing conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, and interpreting results. Generating synthetic datasets, assisting with translation, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable tasks to this work. Additionally, we used generative AI tools for creating scientific figures or images, creating and editing software code, drafting and editing some parts of the paper, summarizing and analyzing existing literature, brainstorming, identifying relevant literature, and proposing titles for the paper. We have reviewed all AI-assisted work, including verification and testing of LLM-generated code for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide an overarching description of our experimental evaluation, with implementation details including models and datasets, distributed training configuration, measured metrics, used compression schemes and baselines, and parameter ranges, to facilitate reproducibility in Section 6. Additionally, Appendix E.3 further details hardware and hyperparameter configurations, and experiment-specific configurations and other details. The code for reproducing all experiments will be made available upon acceptance.

REFERENCES

Lusine Abrahamyan, Yiming Chen, Giannis Bekoulis, and Nikos Deligiannis. Learned Gradient Compression for Distributed Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems (T-NN)*, 33(12):7330–7344, 2021.

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical Learning Periods in Deep Networks. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=BkeStsCcKQ.

Saurabh Agarwal, Hongyi Wang, Kangwook Lee, Shivaram Venkataraman, and Dimitris Papailiopoulos. Adaptive Gradient Communication via Critical Learning Regime Identification. In A. Smola, A. Dimakis, and I. Stoica (eds.), *Conference on Machine Learning and Systems (MLSys)*, volume 3, pp. 55–80, 2021. URL https://proceedings.mlsys.org/paper_files/paper/2021/file/acd593d2db87a799a8d3da5a860c028e-Paper.pdf.

Alham Aji and Kenneth Heafield. Sparse Communication for Distributed Gradient Descent. In *EMNLP 2017: Conference on Empirical Methods in Natural Language Processing*, pp. 440–445. Association for Computational Linguistics (ACL), 2017.

Mohammadreza Alimohammadi, Ilia Markov, Elias Frantar, and Dan Alistarh. L-GreCo: Layerwise-adaptive Gradient Compression For Efficient Data-parallel Deep Learning. In *Conference on Machine Learning and Systems (MLSys)*, 2024.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6c340f25839e6acdc73414517203f5f0-Paper.pdf.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cedric Renggli. The Convergence of Sparsified Gradient Methods. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/314450613369e0ee72d0da7f6fee773c-Paper.pdf.

Leighton Pate Barnes, Huseyin A Inan, Berivan Isik, and Ayfer Özgür. rTop-k: A Statistical Estimation Approach to Distributed SGD. *IEEE Journal on Selected Areas in Information Theory*, 1(3):897–907, 2020.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with Quantization, Sparsification and Local Computations. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Tal Ben-Nun and Torsten Hoefler. Demystifying Parallel and Distributed Deep Learning: An In-depth Concurrency Analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed Optimisation for Non-Convex Problems. In *International Conference on Machine Learning (ICML)*, pp. 560–569. PMLR, 2018.

Aleksandr Beznosikov, Samuel Horváth, Peter Richtárik, and Mher Safaryan. On Biased Compression for Distributed Learning. *Journal of Machine Learning Research (JMLR)*, 24:1–50, 2023.

Tien-Dung Cao, Tram Truong-Huu, Hien Tran, and Khanh Tran. A federated deep learning framework for privacy preservation and communication efficiency. *Journal of Systems Architecture*, 124: 102413, 2022.

Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. AdaComp: Adaptive Residual Gradient Compression for Data-Parallel Distributed Training. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Chia-Yu Chen, Jiamin Ni, Songtao Lu, Xiaodong Cui, Pin-Yu Chen, Xiao Sun, Naigang Wang, Swagath Venkataramani, Vijayalakshmi Viji Srinivasan, Wei Zhang, et al. ScaleCom: Scalable Sparsified Gradient Compression for Communication-Efficient Distributed Training. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:13551–13563, 2020a.

Mengqiang Chen, Zijie Yan, Jiangtao Ren, and Weigang Wu. Standard Deviation Based Adaptive Gradient Compression For Distributed Deep Learning. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 529–538. IEEE, 2020b.

Guoxin Cui, Jun Xu, Wei Zeng, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. MQGrad: Reinforcement Learning of Gradient Quantization in Parameter Server. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, pp. 83–90, 2018.

Laizhong Cui, Xiaoxin Su, Yipeng Zhou, and Yi Pan. Slashing Communication Traffic in Federated Learning by Transmitting Clustered Model Updates. *IEEE Journal on Selected Areas in Communications*, 39(8):2572–2589, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. doi: 10.1109/CVPR.2009.5206848.

Xiaoge Deng, Tao Sun, Feng Liu, and Dongsheng Li. SignGD with error feedback meets lazily aggregated technique: Communication-efficient algorithms for distributed learning. *Tsinghua Science and Technology*, 27(1):174–185, 2021.

Tim Dettmers. 8-Bit Approximations for Parallelism in Deep Learning. In *International Conference on Learning Representations (ICLR)*, 2016. URL `http://arxiv.org/abs/1511.04561`.

Etienne Dupuis, David Novo, Ian O'Connor, and Alberto Bosio. Sensitivity Analysis and Compression Opportunities in DNNs Using Weight Sharing. In *2020 23rd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pp. 1–6, 2020. doi: 10.1109/DDECS50862.2020.9095658.

Aritra Dutta, El Houcine Bergou, Ahmed M Abdelmoniem, Chen-Yu Ho, Atal Narayan Sahu, Marco Canini, and Panos Kalnis. On the Discrepancy between the Theoretical Analysis and Practical Implementations of Compressed Communication for Distributed Deep Learning. In *AAAI Conference on Artificial Intelligence*, volume 34, pp. 3817–3824, 2020.

Negar Foroutan Eghlidi and Martin Jaggi. Sparse Communication for Training Deep Networks. *arXiv preprint arXiv:2009.09271*, 2020.

Mostapha Essoullami, El houcine Bergou, and Aritra Dutta. LEGACY: A Lightweight Dynamic Gradient Compression Strategy for Distributed Deep Learning. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=DsBYEO3B8O.

Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel M Roy, and Ali Ramezani-Kebrya. Adaptive Gradient Quantization for Data-Parallel SGD. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:3174–3185, 2020.

Jiarui Fang, Haohuan Fu, Guangwen Yang, and Cho-Jui Hsieh. RedSync: Reducing synchronization bandwidth for distributed deep learning training system. *Journal of Parallel and Distributed Computing*, 133:30–39, 2019.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The Early Phase of Neural Network Training. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=Hkl1iRNFwS.

Eduard Gorbunov, Konstantin P Burlachenko, Zhize Li, and Peter Richtárik. MARINA: Faster Non-Convex Distributed Learning with Compression. In *International Conference on Machine Learning (ICML)*, pp. 3788–3798. PMLR, 2021.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research (JMLR)*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

Jinrong Guo, Wantao Liu, Wang Wang, Jizhong Han, Ruixuan Li, Yijun Lu, and Songlin Hu. Accelerating Distributed Deep Learning By Adaptive Gradient Quantization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1603–1607. IEEE, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015. URL https://arxiv.org/abs/1503.02531.

Samuel Horvóth, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. Natural Compression for Distributed Deep Learning. In *Mathematical and Scientific Machine Learning*, pp. 129–141. PMLR, 2022.

Kaifan Hu, Chengkun Wu, and En Zhu. HGC: Hybrid Gradient Compression in Distributed Deep Learning. In *International Conference on Artificial Intelligence and Security*, pp. 15–27. Springer, 2021.

Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, et al. Communication-efficient Distributed SGD with Sketching. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Jiawei Jiang, Fangcheng Fu, Tong Yang, and Bin Cui. SketchML: Accelerating Distributed Machine Learning with Data Sketches. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 1269–1284, 2018.

Peng Jiang and Gagan Agrawal. A Linear Speedup Analysis of Distributed Deep Learning with Sparse and Quantized Communication. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In *International Conference on Machine Learning (ICML)*, pp. 3252–3261. PMLR, 2019.

Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication. In *International Conference on Machine Learning (ICML)*, pp. 3478–3487. PMLR, 2019.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*, 2009.

Di Kuang, Mengqiang Chen, Danyang Xiao, and Weigang Wu. Entropy-Based Gradient Compression for Distributed Deep Learning. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 231–238. IEEE, 2019.

S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. ISSN 00034851. URL http://www.jstor.org/stable/2236703.

Yuhang Li, Xin Dong, and Wei Wang. Additive Powers-of-Two Quantization: An Efficient Non-uniform Discretization for Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2020a.

Zhize Li and Peter Richtárik. CANITA: Faster Rates for Distributed Convex Optimization with Communication Compression. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 13770–13781, 2021.

Zhize Li, Dmitry Kovalev, Xun Qian, and Peter Richtarik. Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization. In *International Conference on Machine Learning (ICML)*, pp. 5895–5904. PMLR, 2020b.

Hyeontaek Lim, David G Andersen, and Michael Kaminsky. 3LC: Lightweight and Effective Traffic Compression for Distributed Machine Learning. In A. Talwalkar, V. Smith, and M. Zaharia (eds.), *Conference on Machine Learning and Systems (MLSys)*, volume 1, pp. 53–64, 2019. URL https://proceedings.mlsys.org/paper_files/paper/2019/file/03004620ea802b9118dd44d69f07af56-Paper.pdf.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *International Conference on Learning Representations (ICLR)*, 2018.

Bo Liu, Wenbin Jiang, Shaofeng Zhao, Hai Jin, and Bingsheng He. GradSA: Gradient Sparsification and Accumulation for Communication-Efficient Distributed Deep Learning. In *International Conference on Green, Pervasive, and Cloud Computing*, pp. 77–91. Springer, 2020.

Qu Lu, Wantao Liu, Jizhong Han, and Jinrong Guo. Multi-stage Gradient Compression: Overcoming the Communication Bottleneck in Distributed Deep Learning. In *International Conference on Neural Information Processing*, pp. 107–119. Springer, 2018.

Peng Luo, F Richard Yu, Jianyong Chen, Jianqiang Li, and Victor CM Leung. A Novel Adaptive Gradient Compression Scheme: Reducing the Communication Overhead for Distributed Deep Learning in the Internet of Things. *IEEE Internet of Things Journal*, 8(14):11476–11486, 2021.

Ahmed M Abdelmoniem, Ahmed Elzanaty, Mohamed-Slim Alouini, and Marco Canini. An Efficient Statistical-based Gradient Compression Technique for Distributed Training Systems. *Conference on Machine Learning and Systems (MLSys)*, 3:297–322, 2021.

Amirhossein Malekijoo, Mohammad Javad Fadaeieslam, Hanieh Malekijou, Morteza Homayounfar, Farshid Alizadeh-Shabdiz, and Reza Rawassizadeh. FEDZIP: A Compression Framework for Communication-Efficient Federated Learning. *arXiv preprint arXiv:2102.01593*, 2021.

Yuzhu Mao, Zihao Zhao, Guangfeng Yan, Yang Liu, Tian Lan, Linqi Song, and Wenbo Ding. Communication-Efficient Federated Learning with Adaptive Quantization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022.

Ilia Markov, Hamidreza Ramezanikebrya, and Dan Alistarh. CGX: Adaptive System Support for Communication-Efficient Deep Learning. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference*, Middleware '22, pp. 241–254, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393409. doi: 10.1145/3528535.3565248. URL https://doi.org/10.1145/3528535.3565248.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed Precision Training. In *International Conference on Learning Representations (ICLR)*, 2018.

Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *Optimization Methods and Software*, 40(5):1181–1196, 2025.

Anh Duc Nguyen, Ilia Markov, Frank Zhengqing Wu, Ali Ramezani-Kebrya, Kimon Antonakopoulos, Dan Alistarh, and Volkan Cevher. Layer-wise Quantization for Quantized Optimistic Dual Averaging. In *International Conference on Machine Learning (ICML)*, 2025.

NVIDIA. NVIDIA Deep Learning Examples for Tensor Cores. https://github.com/NVIDIA/DeepLearningExamples, 2020.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization. In *International Conference on Machine Learning (ICML)*, ICML'12, pp. 1571–1578, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Ali Ramezani-Kebrya, Fartash Faghri, Ilya Markov, Vitalii Aksenov, Dan Alistarh, and Daniel M Roy. NUQSGD: Provably Communication-efficient Data-parallel SGD via Nonuniform Quantization. *Journal of Machine Learning Research (JMLR)*, 22(114):1–43, 2021.

Cedric Renggli, Saleh Ashkboos, Mehdi Aghagolzadeh, Dan Alistarh, and Torsten Hoefler. SparCML: High-Performance Sparse Communication for Machine Learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362290. doi: 10.1145/3295500.3356222. URL https://doi.org/10.1145/3295500.3356222.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. FetchSGD: Communication-Efficient Federated Learning with Sketching. In *International Conference on Machine Learning (ICML)*, pp. 8253–8265. PMLR, 2020.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Sparse Binary Compression: Towards Distributed Deep Learning with minimal Communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-Bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs. In *Interspeech*, volume 2014, pp. 1058–1062. Singapore, 2014.

Shaohuai Shi, Qiang Wang, Kaiyong Zhao, Zhenheng Tang, Yuxin Wang, Xiang Huang, and Xiaowen Chu. A Distributed Synchronous SGD Algorithm with Global Top-k Sparsification for Low Bandwidth Networks. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2238–2247. IEEE, 2019a.

Shaohuai Shi, Qiang Wang, Kaiyong Zhao, Zhenheng Tang, Yuxin Wang, Xiang Huang, and Xiaowen Chu. A Distributed Synchronous SGD Algorithm with Global Top-k Sparsification for Low Bandwidth Networks. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2238–2247, 2019b. doi: 10.1109/ICDCS.2019.00220.

Shaohuai Shi, Zhenheng Tang, Qiang Wang, Kaiyong Zhao, and Xiaowen Chu. Layer-wise Adaptive Gradient Sparsification for Distributed Deep Learning with Convergence Guarantees. In Giuseppe {De Giacomo}, Alejandro Catala, Bistra Dilkina, Michela Milano, Senen Barro, Alberto Bugarin, and Jerome Lang (eds.), *ECAI 2020 - 24th European Conference on Artificial Intelligence, including 10th Conference on Prestigious Applications of Artificial Intelligence, PAIS 2020 - Proceedings*, Frontiers in Artificial Intelligence and Applications, pp. 1467–1474. IOS Press BV, 2020. doi: 10.3233/FAIA200253.

Shaohuai Shi, Xianhao Zhou, Shutao Song, Xingyao Wang, Zilin Zhu, Xue Huang, Xinan Jiang, Feihu Zhou, Zhenyu Guo, Liqiang Xie, et al. Towards Scalable Distributed Training of Deep Learning on Public Cloud Clusters. *Conference on Machine Learning and Systems (MLSys)*, 3: 401–412, 2021.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with Memory. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Nikko Strom. Scalable Distributed DNN Training Using Commodity GPU Cloud Computing. In *Interspeech 2015*, pp. 1488–1492, 2015. doi: 10.21437/Interspeech.2015-354.

Jun Sun, Tianyi Chen, Georgios Giannakis, and Zaiyue Yang. Communication-Efficient Distributed Learning via Lazily Aggregated Quantized Gradients. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. Communication Compression for Decentralized Training. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

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 *International Conference on Machine Learning (ICML)*, pp. 10118–10129. PMLR, 2021.

Yusuke Tsuzuku, Hiroto Imachi, and Takuya Akiba. Variance-based Gradient Compression for Efficient Distributed Deep Learning. *arXiv preprint arXiv:1802.06058*, 2018.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/d9fbed9da256e344c1fa46bb46c34c5f-Paper.pdf.

Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. ATOMO: Communication-efficient Learning via Atomic Sparsification. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Zeqin Wang, Ming Wen, Yuedong Xu, Yipeng Zhou, Jessie Hui Wang, and Liang Zhang. Communication compression techniques in distributed deep learning: A survey. *Journal of Systems Architecture*, 142:102927, 2023. ISSN 1383-7621. doi: https://doi.org/10.1016/j.sysarc.2023.102927. URL https://www.sciencedirect.com/science/article/pii/S1383762123001066.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient Sparsification for Communication-Efficient Distributed Optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error Compensated Quantized SGD and its Applications to Large-scale Distributed Optimization. In *International Conference on Machine Learning (ICML)*, pp. 5325–5333. PMLR, 2018.

Hang Xu, Kelly Kostopoulou, Aritra Dutta, Xin Li, Alexandros Ntoulas, and Panos Kalnis. DeepReduce: A Sparse-tensor Communication Framework for Federated Deep Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:21150–21163, 2021.

Enda Yu, Dezun Dong, Yemao Xu, Shuo Ouyang, and Xiangke Liao. CD-SGD: Distributed Stochastic Gradient Descent with Compression and Delay Compensation. In *Proceedings of the 50th International Conference on Parallel Processing*, pp. 1–10, 2021.

Mingchao Yu, Zhifeng Lin, Krishna Narra, Songze Li, Youjie Li, Nam Sung Kim, Alexander Schwing, Murali Annavaram, and Salman Avestimehr. GradiVeQ: Vector Quantization for Bandwidth-Efficient Gradient Aggregation in Distributed CNN Training. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Shujian Yu, Xi Yu, Sigurd Løkse, Robert Jenssen, and Jose C Principe. Cauchy-Schwarz Divergence Information Bottleneck for Regression. In *International Conference on Learning Representations (ICLR)*, 2024.

Xin Zhang, Jia Liu, Zhengyuan Zhu, and Elizabeth S Bentley. Communication-Efficient Network-Distributed Optimization with Differential-Coded Compressors. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 317–326. IEEE, 2020.

A Related work

A.1 Gradient compression schemes

To alleviate the communication bottleneck in distributed data-parallel training, a wide variety of gradient compression techniques have been proposed Wang et al. (2023). Communication compression techniques are grouped into quantization, sparsification, low-rank matrix decomposition, hybrid compression and context specific schemes.

Quantization. Communication quantization reduces the accuracy of each element that is communicated and therefore reduces communication overhead. Quantization techniques include truncated expressions like FP16 Micikevicius et al. (2018), 8-bit approximations Dettmers (2016), 1-Bit SGD Seide et al. (2014), SignSGD Bernstein et al. (2018), SignNUM Bernstein et al. (2018), EF-SignSGD Karimireddy et al. (2019), LE-SignSGD Deng et al. (2021) and 1-Bit Adam Tang et al. (2021). In the family of uniform quantization schemes, Alistarh et al. (2017) introduced QSGD, a family of quantization schemes that balance communication bandwidth and convergence rate, giving a theoretically grounded trade-off between the two. Other uniform quantization techniques include ECQ-SGD Wu et al. (2018), TernGrad Wen et al. (2017), DIANA Mishchenko et al. (2025), ADIANA Li et al. (2020b), CANITA Li & Richtárik (2021), MARINA Gorbunov et al. (2021), 3LC Lim et al. (2019), FedPC Cao et al. (2022), CD-SGD Yu et al. (2021), PQASGD Jiang & Agrawal (2018), LAQ Sun et al. (2019), ATOMO Wang et al. (2018), AdaQS Guo et al. (2020), MQGRad Cui et al. (2018) and AQG Mao et al. (2022). As the normalized gradient distribution is usually not uniform, non-uniform quantization schemes like NC Horvóth et al. (2022), NUQSGD Ramezani-Kebrya et al. (2021), APoT Li et al. (2020a), ALQ Faghri et al. (2020) and MUSCS Cui et al. (2021) have been developed.

Sparsification. Sparsification schemes reduce communication by transmitting only a subset of gradient coordinates. Probabilistic sparsification schemes like random-K Stich et al. (2018), Block-Random-K Eghlidi & Jaggi (2020), GradSparse Wangni et al. (2018) and Probcomp-LPAC Luo et al. (2021) select each element according to its probability. Magnitude sparsification in contrast selects gradients based on their absolute value. Magnitude based sparsification schemes include rTop-k Barnes et al. (2020), gTop-k Shi et al. (2019a), Scalecom Chen et al. (2020a), LGC Abrahamyan et al. (2021), Threshold-v Dutta et al. (2020), DGC Lin et al. (2018), GradDrop Aji & Heafield (2017), RedSync Fang et al. (2019), MSTop-k Shi et al. (2021), EGC Kuang et al. (2019), GradSA Liu et al. (2020), SDAGC Chen et al. (2020b), SIDCo M Abdelmoniem et al. (2021), Sketched-SGD Ivkin et al. (2019) and FetchSGD Rothchild et al. (2020).

Low-rank approximation. Low-rank approximation schemes, such as PowerSGD Vogels et al. (2019) and GradiVeQ Yu et al. (2018), project the gradient matrix into a lower-dimensional subspace. PowerSGD assumes the gradient matrix is low-rank and decomposes it into two smaller matrices using power iteration, significantly reducing the number of parameters transmitted.

Hybrid compression schemes. To minimize both communication overhead and maintain performance, approaches like Sparse one-bit quantization Strom (2015), Qsparse-local-SGD Basu et al. (2019), SBC Sattler et al. (2019), HGC Hu et al. (2021), FEDZIP Malekijoo et al. (2021), Variance-based Compression Tsuzuku et al. (2018) and AdaComp Chen et al. (2018) combine sparsification and quantization schemes. Other hybrid compression schemes include MGC Lu et al. (2018) which uses dynamic combination with training, DeepReduce Xu et al. (2021) where sparse gradients are approximated and SketchML Jiang et al. (2018) with sketch-based quantization on no-zero gradients.

Context-specific compression. In addition to the previously mentioned compression schemes, there exist schemes tailored towards specific environments such as communication architecture, resource environment and application (see e.g., Tang et al. (2018), Koloskova et al. (2019), Zhang et al. (2020)).

Crucially, many lossy compression techniques rely on *error feedback* Karimireddy et al. (2019), where the compression error is accumulated locally and added to the gradient in the next step to ensure convergence.

A.2 LAYER-WISE ADAPTIVE COMPRESSION

While the aforementioned methods are effective, they are typically applied uniformly across all layers of the neural network. This neglects the inherent heterogeneity of deep learning models, where layers differ vastly in parameter count and sensitivity to noise. Shi et al. (2020) identified this limitation and proposed Layer-wise Adaptive Gradient Sparsification (LAGS-SGD), which selects different sparsification thresholds for different layers. Accordion (Agarwal et al., 2021) identifies critical learning regimes in which compression-induced gradient errors can have irrecoverable detrimental impact on model training (Achille et al., 2019; Frankle et al., 2020), and sets layer-wise compression parameters to adaptively avoid too much compression while in such critical regimes. Alimohammadi et al. (2024) introduced L-GreCo, a rigorous framework for layer-wise adaptive compression applicable to both quantization, sparsification and low-rank approximation approaches. Unlike heuristic approaches, L-GreCo formulates the selection of compression parameters (e.g., bit-width or sparsity level) as a convex optimization problem. It uses a dynamic programming algorithm to automatically determine the optimal per-layer compression settings that maximize the compression ratio subject to a global error constraint. They also show that L-GreCo can be combined with Accordion by using Accordion's selected compression parameters as a basis for the global error constraint. However, L-GreCo relies on the L_2 norm of the gradient error ($\|\mathbf{G}_\ell - \tilde{\mathbf{G}}_\ell^{(c)}\|_2^2$) as its optimization measure. While this minimizes numerical deviation, it may not preserve the functional behavior of the layer after updating it with compressed gradients. Nguyen et al. (2025) proposed Quantized Optimistic Dual Averaging (QODA), a layer-wise quantization framework which incorporates the statistical heterogeneity across layers to dynamically adapt the quantization level per layer over the course of training, which they incorporate with L-GreCo in their experiments. Most recently, Essoullami et al. (2026) proposed LEGACY, a lightweight framework that performs rule-based scheduling of layer-wise compression parameters during training, based on two heuristic observations: the model tends to be more sensitive to compression earlier in training, and larger, over-parameterized layers tend to be better candidates for aggressive compression. However, to precisely decide a combination of per-layer compression levels, this approach requires beforehand knowledge of at which points during training one expects compression sensitivity to change, as well as how layer size correlates to compression sensitivity (it is not given that these perfectly correlate). This is highly individual for different model architectures and layers, and requires extensive hyperparameter tuning, while several of the aforementioned methods detect this automatically.

B PROOF OF CONVERGENCE OF LAYER-WISE GRADIENT COMPRESSION SCHEMES

B.1 PRELIMINARIES

In the following, the notation $[i]$ for any $i \in \mathbb{N}$ abbreviates the set $\{1, \dots, i\}$.

We consider a distributed data-parallel learning setting, with $P \in \mathbb{N}$ separate computation nodes. At each iterate x_t, each node $p \in [P]$ computes a stochastic gradient $G^p(x_t)$. These gradients are then compressed by the compression operator $S : \mathbb{R}^d \to \mathbb{R}^d$, and averaged to perform the step in stochastic gradient descent. To ensure convergence while using biased compression operators, each computation node stores the accumulated differences between the compressed and uncompressed gradients in an error feedback term m^p. More precisely, given an objective function $f : \mathbb{R}^d \to \mathbb{R}$, and a starting point $x_0 \in \mathbb{R}^d$, and initial error feedback $m_0^{p,(\ell)} = 0$, the LAGS-SGD performs the following steps.

$$g_t^p = m_t^p + \alpha_t G^p(x_t)$$
$$x_{t+1} = x_t - \frac{1}{P}\sum_{p=1}^{P} S(g_t^p)$$
$$m_{t+1}^p = m_t^p + g_t^p - S(g_t^p)$$

Definition B.1 (Contracting compression operator)**.** A possibly randomized operator $S : \mathbb{R}^d \to \mathbb{R}^d$ is a *contracting compression operator* if there exists a $c \geq 1$ if for all $v \in \mathbb{R}^d$

$$\mathbb{E}\|v - S(v)\|^2 \leq \left(1 - \frac{1}{c}\right)\|v\|^2.$$

Definition B.2 (Layer-wise contracting compression operator)**.** A possibly randomized operator $S : \mathbb{R}^d \to \mathbb{R}^d$ is a *layer-wise contracting compression operator* if

- there exists a partition of the parameter space dimension $\{1, \dots, d\} = I_1 \dot\cup \dots \dot\cup I_L$, such that for each layer ℓ, there is an operator $S^{(\ell)} : \mathbb{R}^{I_l} \to \mathbb{R}^{I_l}$ with $S^{(\ell)}(x_{I_\ell}) = S(x)_{I_\ell}$ for all $x \in \mathbb{R}^d$,
- and $S^{(\ell)}$ is a contracting compression operator with constant $c^{(\ell)} \geq 1$.

Let $c_{\max} := \max_{\ell \in [L]} c^{(\ell)}$ denote the maximal compression rate among the layers.

We now establish that both layer-wise top-k sparsification, layer-wise Rand-k sparsification, and, under appropriate scaling, layer-wise quantization are layer-wise contracting compression operators.

Proposition B.3. *Let S be the layer-wise top-k sparsification operator, or the layer-wise Rand-k sparsification operator that keeps $k^{(\ell)} > 0$ elements from the $d^{(\ell)}$-dimensional layer ℓ for all layers $\ell \in [L]$. Defining the compression constant $c^{(\ell)} := d^{(\ell)}/k^{(\ell)}$, S is a layer-wise contracting compression operator with compression levels $(c^{(\ell)})_{\ell \in [L]}$.*

Proof. S is compressing layer-wise as it is constructed from sparsification operators for each individual layer. A proof of the contraction property of each operator $S^{(\ell)}$ with constant $c^{(\ell)}$ can be found in (Stich et al., 2018, Lemma A.1). □

Next, we discuss the quantization operator for a single layer. As introduced by Alistarh et al. (2017), the quantization operator S with quantization level $s \in \mathbb{N}$ compresses a vector $x \in \mathbb{R}^d$ by setting

$$S(x)_i = \|x\| \cdot \operatorname{sgn}(x_i) \cdot \xi_i(x, s),$$

where $\xi_i(x_i, s)$ is a random variable defined as follows. For $x \neq 0$, we find an integer $0 \leq k \leq c$ such that $|x_i|/\|x\| \in [k/s, k+1)/s]$. With that, we define

$$\xi(x,s) := \begin{cases} \frac{k}{s} & \text{with probability } 1 - p(\frac{|x_i|}{\|x\|}, c) \\ \frac{k+1}{s} & \text{with probability } p(\frac{|x_i|}{\|x\|}, c) \end{cases}$$

where $p(a, s) := as - k$. This operation rounds each normalized absolute coordinate at random to the next higher or lower quantization step, with a probability that ensures that the expectation remains the original absolute coordinate.

Alistarh et al. (2017, Lemma 3.1) prove that S is unbiased, i.e. $\mathbb{E}(S(x)) = x$, and for the variance it holds that

$$\mathbb{E}\|S(x) - x\|^2 \leq \varepsilon\|x\|^2.$$

where

$$\varepsilon := \min\left(\frac{d}{s^2}, \frac{\sqrt{d}}{s}\right).$$

A layer-wise quantization operator S is constructed by assigning each layer ℓ a quantization operator $S^{(\ell)}$. As the next proposition shows, after appropriate scaling of each layer-individual operator, the quantization operator becomes a layer-wise contracting compression operator, consisting of L operators $S^{(\ell)}$.

Proposition B.4. *Let S be the operator that applies quantization individually to each layer, using quantization levels $s^{(\ell)}$. We define*

$$\varepsilon^{(\ell)} := \min\left(\frac{d^{(\ell)}}{s^{(\ell)2}}, \frac{\sqrt{d^{(\ell)}}}{s^{(\ell)}}\right).$$

Then, the operator $\tilde{S}$ that results from scaling each layer operator $S^{(\ell)}$ by $1/(\varepsilon^{(\ell)} + 1)$ is a layer-wise contracting compression operator with compression levels $c^{(\ell)} = \varepsilon^{(\ell)} + 1$.

Proof. The operator acts layer-wise by its definition. Let $S^{(\ell)}$ be the operator for layer ℓ. We have

$$\mathbb{E}\|S^{(\ell)}(x)\|^2 \leq (\varepsilon^{(\ell)} + 1)\|x\|^2$$

Applying Theorem 7 from (Beznosikov et al., 2023) with $\lambda = 1/(\varepsilon^{(\ell)} + 1)$ to $S^{(\ell)}$ proves that $\tilde{S}^{(\ell)} := \lambda S^{(\ell)}$ is a contracting compression operator with constant $c^{(\ell)} = \varepsilon^{(\ell)} + 1$, as

$$\frac{1}{\lambda(2 - \lambda(\varepsilon^{(\ell)} + 1))} = \frac{1}{\lambda} = \varepsilon^{(\ell)} + 1.$$

□

The fact that the compression takes place at each node separately complicates the theoretical analysis of the algorithm. The key difficulty in the proof is that the compression operator is applied locally instead of to the full stochastic gradient. It is therefore necessary to make some analytical assumptions on the compressed gradients, to ensure that the arithmetic mean of the transmitted compressed vectors is not too dissimilar to the compressed mean of their uncompressed counterparts. Such assumptions are key in proving that a general compression scheme converges. The first assumption is generalized from (Alistarh et al., 2018).

Assumption B.5. There exists some $\xi > 0$ such that for the iterates (x_t) of the algorithm it holds for each layer $\ell \in [L]$ that

$$\mathbb{E}\left\|S^{(\ell)}\left(\frac{1}{P}\sum_{p=1}^{P}(m_t^p + \alpha_t G^p(x_t))\right) - \frac{1}{P}\sum_{p=1}^{P} S^{(\ell)}(m_t^p + \alpha_t G^p(x_t))\right\| \leq \xi\|\alpha_t G^{(\ell)}(x_t)\|,$$

with the expectation taken with respect to the randomness in S.

Alternatively, we prove the theorems under the following assumption, which is based on Shi et al. (2020).

Assumption B.6. For the sequence of iterates (x_t) that are computed by the algorithm and for each layer $l \in [L]$, it holds that

$$\begin{aligned}&\mathbb{E}\left\|\frac{1}{P}\sum_{p=1}^{P}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) - \frac{1}{P}\sum_{p=1}^{P} S^{(\ell)}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t))\right\| \\ \leq&\mathbb{E}\left\|\frac{1}{P}\sum_{p=1}^{P}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) - S^{(\ell)}\left(\frac{1}{P}\sum_{p=1}^{P}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t))\right)\right\|.\end{aligned}$$

Having established the preliminaries, we state the main theorem.

Theorem B.7. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a μ-strongly convex objective function with minimizer x^*, where the layer-wise gradients of the iterates x_t uniformly satisfy $\mathbb{E}\|G^{(\ell)}(x_t)\|^2 \leq M^{(\ell)}$ for all $\ell \in [L]$. Let the learning rates be given by $\alpha_t = 8/(\mu(t+a))$ for some $a \geq 8c_{\max}$. Let $(\delta_t)_{t=0}^{T-1}$ be defined as $\delta_t = (t+a)^2/S_T$ where $S_T = \sum_{t=0}^{T-1}(t+a)^2$. Let the average iterate be defined as $\bar{x}_T = \sum_{t=0}^{T-1} \delta_t x_t$. If Assumption B.5 or Assumption B.6 holds, the iterates satisfy*

$$\mathbb{E}f(\bar{x}_T) - f(x^*) \leq \frac{a^3 \sum_{\ell=1}^{L} M^{(\ell)}}{4\mu S_T} + \frac{4T(T+2a)}{\mu S_T}\left(\frac{B}{2}\sum_{\ell=1}^{L} c^{(\ell)} M^{(\ell)} + \frac{1}{2}\sum_{\ell=1}^{L} M^{(\ell)}\right) + \frac{CT}{\mu S_T}\left(\sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)}\right),$$

where $B = 9$ and $C = 256$ if Assumption B.6 holds, and $B = 4(6\xi^2+7)+1$. and $C = 128(6\xi^2+7)$ if Assumption B.5 holds.

An immediate consequence of Theorem B.7 is the following corollary.

Corollary B.8. *Let the assumptions of Theorem B.7 be met with $a = 8c_{\max}$. Then, it holds that*

$$\begin{aligned}\mathbb{E}f(\bar{x}_T) - f(x^*) \leq\ & \mathcal{O}\left(\frac{c_{\max}\sum_{\ell=1}^{L} M^{(\ell)}}{\mu T}\right) + \mathcal{O}\left(\frac{\sum_{\ell=1}^{L} c^{(\ell)} M^{(\ell)}}{\mu T}\right) \\ & + \mathcal{O}\left(\frac{c_{\max}\sum_{\ell=1}^{L} c^{(\ell)} M^{(\ell)}}{\mu T^2}\right) + \mathcal{O}\left(\frac{\sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)}}{\mu T^2}\right) \\ & + \mathcal{O}\left(\frac{c_{\max}^3\sum_{\ell=1}^{L} M^{(\ell)}}{\mu T^3}\right).\end{aligned}$$

As shown by Stich et al. (2018), this follows from the fact that $S_T \geq T^3/3$.

Theorem B.9. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a β-smooth objective function with a global minimum at x^* where the layer-wise gradients of the iterates x_t uniformly satisfy $\mathbb{E}\|G^{(\ell)}(x_t)\|^2 \leq M^{(\ell)}$ for all $\ell \in [L]$. Let the learning rate $(\alpha_t)_{t\in\mathbb{N}_0}$ satisfy $\alpha_t^2/\alpha_{t+1}^2 \leq 4/(4-1/c_{\max})$. If Assumption B.5 or Assumption B.6 holds, the iterates satisfy*

$$\frac{\sum_{t=0}^{T-1}\alpha_t \mathbb{E}\|\nabla f(x_t)\|^2}{\sum_{t=0}^{T-1}\alpha_t} \leq \frac{4(f(x_0)-f(x^*))}{3\sum_{t=0}^{T-1}\alpha_t} + \frac{2\beta\sum_{t=0}^{T-1}\alpha_t^2\sum_{\ell=1}^{L} M^{(\ell)}}{3\sum_{t=0}^{T-1}\alpha_t} + \frac{\tilde{B}\beta^2\sum_{t=0}^{T-1}\alpha_t^3\sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)}}{3\sum_{t=0}^{T-1}\alpha_t},$$

where $\tilde{B} = 32$ if Assumption B.6 holds, and $\tilde{B} = 16(6\xi^2+7)$ if Assumption B.5 holds.

Using this theorem, we get the following rates on the upper bounds.

Corollary B.10. *Let the assumptions of Theorem B.9 be met, and the learning rates be chosen as $\alpha_t = \theta/\sqrt{T}$ for some $\theta > 0$. Then, we have that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(x_t)\|^2 \leq \mathcal{O}\left(\frac{1}{\theta\sqrt{T}}\right) + \mathcal{O}\left(\frac{\beta\theta\sum_{\ell=1}^{L} M^{(\ell)}}{\sqrt{T}}\right) + \mathcal{O}\left(\frac{32\beta^2\theta^2\sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)}}{T}\right).$$

This result is unsurprising in light of the analysis in (Karimireddy et al., 2019, Theorem 2), which established the same rate for SGD variants with error feedback.

B.2 Proof of Theorem B.7

The proof of Theorem B.7 holds along the lines of (Stich et al., 2018, Theorem 2.4). Different from (Stich et al., 2018), in this paper, we do not use the maximal compression parameter $c_{\max}$ uniformly

over all layers, but derive an upper bound that entails all layer-individual compression parameters $c^{(\ell)}$.

A crucial part of the proof is to define a sequence of iterates $(\tilde{x}_t)_{t\in\mathbb{N}}$ with

$$\begin{aligned}\tilde{x}_0 &:= x_0,\\ \tilde{x}_{t+1} &:= \tilde{x}_t - \alpha_t G(x_t),\end{aligned}$$

where $G(\cdot) := 1/P \sum_{p=1}^{P} G^p(\cdot)$ is the full stochastic gradient.

Lemma B.11. *Assuming that for $l \in [k]$ and for all choices of $v^{(\ell)} \in \mathbb{R}^{d_l}$, it holds that*

$$\mathbb{E}\left\|\sum_{p=1}^{P} v^{p,(\ell)} - \sum_{p=1}^{P} S^{(\ell)}(v^{p,(\ell)})\right\|^2 \leq \mathbb{E}\left\|\sum_{p=1}^{P} v^{p,(\ell)} - S^{(\ell)}\left(\sum_{p=1}^{P} v^{p,(\ell)}\right)\right\|^2,$$

where the expectation is taken with respect to the randomness in S and $S^{(\ell)}$, we obtain

$$\mathbb{E}\left\|\sum_{p=1}^{P} v^{p,(\ell)} - \sum_{p=1}^{P} S^{(\ell)}(v^{p,(\ell)})\right\|^2 \leq \left(1 - \frac{1}{c^{(\ell)}}\right)\left\|\sum_{p=1}^{P} v^{p,(\ell)}\right\|^2,$$

Proof. This statement follows directly from the contracting property of the compression operator $S^{(\ell)}$. □

We prove the convergence first under Assumption B.6. For the next two lemmas in the proof, we need a technical proposition.

Proposition B.12. *Let $(s_t)_{t=0}^{\infty}$ and $(\alpha_t)_{t=0}^{\infty}$ be sequences of nonnegative real numbers which satisfy*

$$\begin{aligned}s_0 &= 0,\\ s_{t+1} &\leq \rho s_t + K\alpha_t^2 \qquad \forall t \in \mathbb{N}_0,\\ \alpha_t^2 &\leq \frac{2}{1+\rho}\alpha_{t+1}^2 \qquad \forall t \in \mathbb{N}_0,\end{aligned}$$

for constants $K > 0$ and $0 < \rho < 1$. Then, it holds for all $t \in \mathbb{N}_0$

$$s_t \leq \frac{2K}{1-\rho}\alpha_t^2. \tag{2}$$

Proof. We prove the claim by induction. For $t = 0$ the claim holds as $s_0 = 0$ and the right hand side of equation 2 is nonnegative.

For the inductive step, we observe that

$$\begin{aligned}s_{t+1} &\leq \rho s_t + K\alpha_t^2\\ &\leq \rho\frac{2K}{1-\rho}\alpha_t^2 + K\alpha_t^2\\ &\leq \frac{1+\rho}{1-\rho}K\alpha_t^2\\ &\leq \frac{1+\rho}{1-\rho}\frac{2}{1+\rho}K\alpha_{t+1}^2\\ &= \frac{2K}{1-\rho}\alpha_{t+1}^2.\end{aligned}$$

□

Lemma B.13. *If Assumption B.6 holds and we have that $\mathbb{E}\|G^{(\ell)}(x_t)\|^2 \leq M^{(\ell)}$, we get*

$$\mathbb{E}\|\tilde{x}_{t+1}^{(\ell)} - x_{t+1}^{(\ell)}\|^2 \leq \left(1 - \frac{1}{2c^{(\ell)}}\right)\mathbb{E}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 + 2c^{(\ell)}M^{(\ell)}\alpha_t^2.$$

If furthermore the learning rate satisfies $\alpha_t > 0$ for all t and $\alpha_t^2/\alpha_{t+1}^2 \leq 4/(4 - 1/c^{\max})$, we get that

$$\mathbb{E}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 \leq 8c^{(\ell)2}M^{(\ell)}\alpha_t^2.$$

Proof. By the definition of the algorithm, we get

$$m_{t+1}^{p,(\ell)} = m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t) - S^{(\ell)}\left(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)\right).$$

Iterating yields

$$m_{t+1}^{p,(\ell)} = \sum_{j=0}^{t} \alpha_j G^{p,(\ell)}(x_j) - S^{(\ell)}\left(m_j^{p,(\ell)} + \alpha_j G^{p,(\ell)}(x_j)\right).$$

By averaging over the different machines, we get

$$\begin{aligned}
\frac{1}{P}\sum_{p=1}^{P} m_{t+1}^{p,(\ell)} &= \sum_{j=0}^{t} \frac{1}{P}\sum_{p=1}^{P} \left\{\alpha_j G^{p,(\ell)}(x_j) - S^{(\ell)}\left(m_j^{p,(\ell)} + \alpha_j G^{p,(\ell)}(x_j)\right)\right\} \\
&= x_0^{(\ell)} - \sum_{j=0}^{t} \frac{1}{P}\sum_{p=1}^{P} S^{(\ell)}\left(m_j^{p,(\ell)} + \alpha_j G^{p,(\ell)}(x_j)\right) \\
&\quad - \left\{x_0^{(\ell)} - \sum_{j=0}^{t} \frac{1}{P}\sum_{p=1}^{P} \alpha_j G^{p,(\ell)}(x_j)\right\} \\
&= x_{t+1}^{(\ell)} - \tilde{x}_{t+1}^{(\ell)}.
\end{aligned}$$

With that, we derive an upper bound on the expected distances between the iterates x_t and the perturbed iterates $\tilde{x}_t$ individually for each layer.

$$\begin{aligned}
\mathbb{E}\|x_{t+1}^{(\ell)} - \tilde{x}_{t+1}^{(\ell)}\|^2 &= \mathbb{E}\left\|\frac{1}{P}\sum_{p=1}^{P} m_{t+1}^{p,(\ell)}\right\|^2 \\
&= \mathbb{E}\left\|\frac{1}{P}\sum_{p=1}^{P}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) - \frac{1}{P}\sum_{p=1}^{P} S^{(\ell)}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t))\right\|^2 \\
&\leq \left(1 - \frac{1}{c^{(\ell)}}\right)\left\|\frac{1}{P}\sum_{p=1}^{P}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t))\right\| \\
&= \left(1 - \frac{1}{c^{(\ell)}}\right)\left\|x_t^{(\ell)} - \tilde{x}_t^{(\ell)} + \alpha_t G^{(\ell)}(x_t))\right\|
\end{aligned}$$

For all $\gamma > 0$ it holds that $\|a+b\|^2 \leq (1+\gamma)\|a\|^2 + (1+\gamma^{-1})\|b\|^2$. Choosing $\gamma = 1/(2c^{(\ell)})$, we deduce

$$\begin{aligned}
\mathbb{E}\|x_{t+1}^{(\ell)} - \tilde{x}_{t+1}^{(\ell)}\|^2 &\leq \left(1 - \frac{1}{c^{(\ell)}}\right)\left\{\left(1 + \frac{1}{2c^{(\ell)}}\right)\|x_t^{(\ell)} - \tilde{x}_t^{(\ell)}\|^2 + (1 + 2c^{(\ell)})\alpha_t^2\|G^{(\ell)}(x_t)\|^2\right\} \\
&\leq \left(1 + \frac{1}{2c^{(\ell)}}\right)\|x_t^{(\ell)} - \tilde{x}_t^{(\ell)}\|^2 + 2c^{(\ell)}\alpha_t^2\|G^{(\ell)}(x_t)\|^2.
\end{aligned}$$

By taking expectations and using the tower property of expectation, we arrive at the first claim of the lemma.

For the second part, we use $\rho := 1 - 1/(2c^{(\ell)})$ and $K = 2c^{(\ell)}M^{(\ell)}$ to see that the sequences $s_t := \mathbb{E}\|x_t^{(\ell)} - \tilde{x}_t^{(\ell)}\|^2$ and α_t satisfy the assumptions of Proposition B.12 due to the first claim of this lemma, and the fact that

$$\frac{2}{1+\rho} = \frac{2}{1 + 1 - 1/(2c^{(\ell)})} = \frac{4}{4 - 1/c^{(\ell)}},$$

which shows that

$$\frac{\alpha_t^2}{\alpha_{t+1}^2} \leq \frac{4}{4 - 1/c^{\max}} \leq \frac{4}{4 - 1/c^{(\ell)}} = \frac{2}{1-\rho}.$$

Thus, applying Proposition B.12 yields

$$\mathbb{E}\|x_t^{(\ell)} - \tilde{x}_t^{(\ell)}\|^2 \leq \frac{2 \cdot 2c^{(\ell)} M^{(\ell)}}{1 - (1 - 1/(2c^{(\ell)}))} \alpha_t^2 = 8c^{(\ell)2} M^{(\ell)} \alpha_t^2,$$

which completes the proof of the lemma. □

Also, using Assumption B.5, we prove a similar bound on the difference between the iterates.

Lemma B.14. *If Assumption B.5 holds and we have that $\mathbb{E}\|G^{(\ell)}(x_t)\|^2 \leq M^{(\ell)}$, we get*

$$\mathbb{E}\|\tilde{x}_{t+1}^{(\ell)} - x_{t+1}^{(\ell)}\|^2 \leq \left(1 - \frac{1}{2c^{(\ell)}}\right) \mathbb{E}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 + (6\xi^2 + 7)c^{(\ell)} M^{(\ell)} \alpha_t^2.$$

If furthermore the learning rate satisfies $\alpha_t > 0$ for all t and $\alpha_t^2/\alpha_{t+1}^2 \leq 4/(4 - 1/c^{\max})$, we get that

$$\mathbb{E}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 \leq 4(6\xi^2 + 7)c^{(\ell)2} M^{(\ell)} \alpha_t^2.$$

Proof. For any $\gamma_1 > 0$, we have

$$\begin{aligned}
\|x_{t+1}^{(\ell)} - \tilde{x}_{t+1}^{(\ell)}\|^2 &= \mathbb{E}_S \left\| \frac{1}{P} \sum_{p=1}^{P} (m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) - \frac{1}{P} \sum_{p=1}^{P} S^{(\ell)}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \right\|^2 \\
&= \mathbb{E}_S \Bigg\| \frac{1}{P} \sum_{p=1}^{P} (m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \\
&\quad - S^{(\ell)} \left(\frac{1}{P} \sum_{p=1}^{P} (m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \right) \\
&\quad + S^{(\ell)} \left(\frac{1}{P} \sum_{p=1}^{P} (m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \right) \\
&\quad - \frac{1}{P} \sum_{p=1}^{P} S^{(\ell)}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \Bigg\|^2 \\
&\leq (1 + \gamma_1) \mathbb{E}_S \Bigg\| \frac{1}{P} \sum_{p=1}^{P} (m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \\
&\quad - S^{(\ell)} \left(\frac{1}{P} \sum_{p=1}^{P} (m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \right) \Bigg\|^2 \\
&\quad + (1 + \gamma_1^{-1}) \mathbb{E}_S \left\| S^{(\ell)} \left(\frac{1}{P} \sum_{p=1}^{P} (m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \right) \right. \\
&\quad \left. - \frac{1}{P} \sum_{p=1}^{P} S^{(\ell)}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t)) \right\|^2 .
\end{aligned} \tag{3}$$

Using the compression property of $S^{(\ell)}$ and the assumption, we derive for all $\gamma_2 > 0$

$$\begin{aligned}\|x_{t+1}^{(\ell)} - \tilde{x}_{t+1}^{(\ell)}\|^2 &\leq (1+\gamma_1)\left(1 - \frac{1}{c^{(\ell)}}\right)\left\|\frac{1}{P}\sum_{p=1}^{P}(m_t^{p,(\ell)} + \alpha_t G^{p,(\ell)}(x_t))\right\|^2 \\ &\quad + (1+\gamma_1^{-1})\xi^2\alpha_t^2\|G^{(\ell)}(x_t)\|^2 \\ &\leq (1+\gamma_1)\left(1 - \frac{1}{c^{(\ell)}}\right)\left\|x_t^{(\ell)} - \tilde{x}_t^{(\ell)} + \alpha_t G^{p,(\ell)}(x_t))\right\|^2 \\ &\quad + (1+\gamma_1^{-1})\xi^2\alpha_t^2\|G^{(\ell)}(x_t)\|^2 \\ &\leq (1+\gamma_1)(1+\gamma_2)\left(1 - \frac{1}{c^{(\ell)}}\right)\|x_t^{(\ell)} - \tilde{x}_t^{(\ell)}\|^2 \\ &\quad + \alpha_t^2\left\{(1+\gamma_1^{-1})\xi^2 + (1+\gamma_1)\left(1 - \frac{1}{c^{(\ell)}}\right)(1+\gamma_2^{-1})\right\} \\ &\quad \|G^{(\ell)}(x_t)\|^2.\end{aligned}$$

Choosing $\gamma_1 = \gamma_2 = 1/(5c^{(\ell)})$ allows us, combined with $c^{(\ell)} > 1$, to deduce that

$$\begin{aligned}(1+\gamma_1)(1+\gamma_2)\left(1 - \frac{1}{c^{(\ell)}}\right) &\leq 1 + \frac{1}{2c^{(\ell)}}, \\ (1+\gamma_1^{-1}) &\leq 6c^{(\ell)}, \\ (1+\gamma_1)\left(1 - \frac{1}{c^{(\ell)}}\right)(1+\gamma_2^{-1}) &\leq 7c^{(\ell)}.\end{aligned}$$

By taking expectations, and using the bounds above, the first part of the proof is completed.

For the second part, choosing $\rho = 1 - 1/(2c^{(\ell)})$ and $K = (6\xi^2 + 7)$ allows the same argumentation as in the proof of Lemma B.13, yielding

$$\mathbb{E}\|x_t^{(\ell)} - \tilde{x}_t^{(\ell)}\|^2 \leq \frac{2\cdot(6\xi^2+7)c^{(\ell)}M^{(\ell)}}{1-(1-1/(2c^{(\ell)}))}\alpha_t^2 = 4(6\xi^2+7)c^{(\ell)2}M^{(\ell)}\alpha_t^2,$$

□

Proving the theorem follows the same structure, independent of the assumption we choose. Therefore, we continue to prove the theorem under Assumption B.6.

Lemma B.15. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be μ-strongly convex, let the gradients satisfy $\mathbb{E}\|G^{(\ell)}(x_t)\|^2 \leq M^{(\ell)}$ for all $t \in \mathbb{N}_0$. Then, for the iterates it holds*

$$\begin{aligned}\mathbb{E}f(x_t) - f(x^*) = \frac{1}{2\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right)\mathbb{E}\|\tilde{x}_t - x^*\|^2 - \frac{1}{2\alpha_t}\mathbb{E}\|\tilde{x}_{t+1} - x^*\|^2 \\ + \sum_{\ell=1}^{L}\frac{1}{2}\left(\frac{1}{\alpha_t c^{(\ell)}} + \mu\right)\mathbb{E}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 + \frac{\alpha_t}{2}(1+c^{(\ell)})M^{(\ell)}.\end{aligned}$$

Proof. By utilizing the recursion for $\tilde{x}_t$ we get

$$\begin{aligned}\|\tilde{x}_{t+1}^{(\ell)} - x^{*,(\ell)}\|^2 &= \|\tilde{x}_t^{(\ell)} - \alpha_t G^{(\ell)}(x_t) - x^{*,(\ell)}\|^2 \\ &= \|\tilde{x}_t^{(\ell)} - x^{*,(\ell)}\|^2 - 2\alpha_t\langle\tilde{x}_t^{(\ell)} - x^{*,(\ell)}, G^{(\ell)}(x_t)\rangle + \alpha_t^2\|G^{(\ell)}(x_t)\|^2.\end{aligned}$$

Rearranging this, we get

$$\langle\tilde{x}_t^{(\ell)} - x^*, G^{(\ell)}(x_t)\rangle = \frac{1}{2\alpha_t}\left(\|\tilde{x}_t^{(\ell)} - x^{*,(\ell)}\|^2 - \|\tilde{x}_{t+1}^{(\ell)} - x^{*,(\ell)}\|^2\right) + \frac{\alpha_t}{2}\|G^{(\ell)}(x_t)\|^2.$$

Furthermore, we get by applying Young's inequality that

$$\langle x_t^{(\ell)} - \tilde{x}_t^{(\ell)}, G^{(\ell)}(x_t)\rangle \leq \frac{1}{2}\left(\gamma\|x_t - \tilde{x}_t\|^2 + \gamma^{-1}\|G^{(\ell)}(x_t)\|^2\right).$$

Combining these, we deduce by choosing $\gamma = 1/(c^{(\ell)}\alpha_t)$

$$\begin{aligned}\langle x_t^{(\ell)} - x^{*,(\ell)}, G^{(\ell)}(x_t)\rangle &= \langle x_t^{(\ell)} - \tilde{x}^{(\ell)}, G^{(\ell)}(x_t)\rangle + \langle \tilde{x}^{(\ell)} - x^{*,(\ell)}, G^{(\ell)}(x_t)\rangle \\ &= \frac{1}{2\alpha_t}\left(\|\tilde{x}_t^{(\ell)} - x^{*,(\ell)}\|^2 - \|\tilde{x}_{t+1}^{(\ell)} - x^{*,(\ell)}\|^2\right) + \frac{1}{2c^{(\ell)}\alpha_t}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 \\ &\quad + \frac{1}{2}(1 + c^{(\ell)})\alpha_t\|G^{(\ell)}(x_t)\|^2.\end{aligned} \tag{4}$$

We now turn our attention to the objective f and use its strong convexity. From that, we get

$$f(x_t) - f(x^*) \leq \langle x_t - x^*, \nabla f(x_t)\rangle - \frac{\mu}{2}\|x_t - x^*\|^2.$$

From the triangle inequality, we get

$$\|\tilde{x}_t - x^*\|^2 \leq 2\|\tilde{x}_t - x_t\|^2 + 2\|x_t - x^*\|^2,$$

which implies that

$$-\|x_t - x^*\|^2 \leq \|\tilde{x}_t - x_t\|^2 - \frac{1}{2}\|\tilde{x}_t - x^*\|^2.$$

Applying this to equation equation B.2 shows that

$$\begin{aligned}f(x_t) - f(x^*) &\leq \langle x_t - x^*, \nabla f(x_t)\rangle - \frac{\mu}{2}\|\tilde{x}_t - x_t\|^2 - \frac{\mu}{4}\|\tilde{x}_t - x^*\|^2 \\ &= \sum_{\ell=1}^{L}\langle x_t^{(\ell)} - x^{*,(\ell)}, \nabla^{(\ell)} f(x_t)\rangle + \frac{\mu}{2}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 - \frac{\mu}{4}\|\tilde{x}_t^{(\ell)} - x^{*,(\ell)}\|^2.\end{aligned} \tag{5}$$

Now, using the fact that $\mathbb{E}G^{(\ell)}(x_t) = \nabla^{(\ell)} f(x_t)$, the combination of Equations equation 5 and equation 4 yields

$$\begin{aligned}\mathbb{E}f(x_t) - f(x^*) &\leq \sum_{\ell=1}^{L}\frac{1}{2\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right)\mathbb{E}\|\tilde{x}_t^{(\ell)} - x^{*,(\ell)}\|^2 - \frac{1}{2\alpha_t}\mathbb{E}\|\tilde{x}_t^{(\ell)} - x^{*,(\ell)}\|^2 \\ &\quad + \frac{1}{2}\left(\frac{1}{\alpha_t c^{(\ell)}} + \mu\right)\mathbb{E}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 + \frac{\alpha_t}{2}(1 + c^{(\ell)})M^{(\ell)} \\ &= \frac{1}{2\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right)\mathbb{E}\|\tilde{x}_t - x^*\|^2 - \frac{1}{2\alpha_t}\mathbb{E}\|\tilde{x}_{t+1} - x^*\|^2 \\ &\quad + \sum_{\ell=1}^{L}\frac{1}{2}\left(\frac{1}{\alpha_t c^{(\ell)}} + \mu\right)\mathbb{E}\|\tilde{x}_t^{(\ell)} - x_t^{(\ell)}\|^2 + \frac{\alpha_t}{2}(1 + c^{(\ell)})M^{(\ell)}.\end{aligned}$$

□

Next, we apply the bound derived in Lemma B.13.

Corollary B.16. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be μ-strongly convex, let the gradients satisfy $\mathbb{E}\|G^{(\ell)}(x_t)\|^2 \leq M^{(\ell)}$ for all $t \in \mathbb{N}_0$. Let the learning rate be given by $\alpha_t = 8/(\mu(t + a))$ with $a \geq 8c_{\max}$. Then, for the iterates, it holds*

$$\begin{aligned}\mathbb{E}f(x_t) - f(x^*) &\leq \frac{1}{2\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right)\mathbb{E}\|\tilde{x}_t - x^*\|^2 - \frac{1}{2\alpha_t}\mathbb{E}\|\tilde{x}_{t+1} - x^*\|^2 \\ &\quad \frac{5}{2}\alpha_t\sum_{\ell=1}^{L}c^{(\ell)}M^{(\ell)} + \frac{1}{2}\alpha_t\sum_{\ell=1}^{L}M^{(\ell)} + 2\mu\alpha_t^2\sum_{\ell=1}^{L}c^{(l)2}M^{(\ell)}.\end{aligned}$$

Proof. We first show the applicability of Lemma B.13 by proving that

$$\frac{\alpha_t^2}{\alpha_{t+1}^2} \leq \frac{4}{4 - 1/c_{\max}}. \tag{6}$$

By the definition of (α_t), this is equivalent to

$$\frac{(t+a+1)^2}{(t+a)^2} \leq \frac{4}{4-1/c_{\max}}.$$

By multiplying wit $(t+a)^2$ and rearranging, this is equivalent to

$$\left(1 - \frac{4}{4-1/c_{\max}}\right)(t+a)^2 + 2(t+a)^+ t \leq 0.$$

By solving this quadratic inequality, we get that it holds if

$$t \geq -a - 1 + 4c_{\max} + 2\sqrt{4c_{\max}^2 - c_{\max}}.$$

As we have $t \geq 0$ and the right side is smaller than 0 for $a \geq 8c_{\max}$, this equation holds, proving Equation equation 6.

We therefore apply Lemma B.13 to Lemma B.15 to get that

$$\begin{aligned}
\mathbb{E}f(x_t) - f(x^*) &\leq \frac{1}{2\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right)\mathbb{E}\|\tilde{x}_t - x^*\|^2 - \frac{1}{2\alpha_t}\mathbb{E}\|\tilde{x}_{t+1} - x^*\|^2 \\
&\quad + \sum_{\ell=1}^{L}\frac{1}{2}\left(\frac{1}{\alpha_t c^{(\ell)}} + \mu\right)8\alpha_t^2 c^{(l)2} M^{(\ell)} + \frac{\alpha_t}{2}(1 + c^{(\ell)})M^{(\ell)} \\
&= \frac{1}{2\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right)\mathbb{E}\|\tilde{x}_t - x^*\|^2 - \frac{1}{2\alpha_t}\mathbb{E}\|\tilde{x}_{t+1} - x^*\|^2 \\
&\quad + \frac{9}{2}\alpha_t \sum_{\ell=1}^{L} c^{(\ell)} M^{(\ell)} + \frac{1}{2}\alpha_t \sum_{\ell=1}^{L} M^{(\ell)} + 4\mu\alpha_t^4 \sum_{\ell=1}^{L} c^{(l)2} M^{(\ell)}.
\end{aligned}$$

□

It remains to obtain a bound on $\mathbb{E}\|x_0 - x^*\|^2$, which is derived from (Rakhlin et al., 2012).

Lemma B.17. *Let f be μ-strongly convex, and $\mathbb{E}\|G^{(\ell)}(x_0)\|^2 \leq M^{(\ell)}\ \forall \ell \in [L]$. Then we have*

$$\mathbb{E}\|x_0 - x^*\|^2 \leq \frac{4\sum_{\ell=1}^{L} M^{(\ell)}}{\mu^2}.$$

Proof. It holds that

$$\mathbb{E}\|G(x_0)\|^2 = \mathbb{E}\sum_{\ell=1}^{L}\|G^{(\ell)}(x_0)\|^2 \leq \sum_{\ell=1}^{L} M^{(\ell)}.$$

Applying Lemma 2 from (Rakhlin et al., 2012) proves the claim. □

With that, we prove Theorem B.7.

Proof of Theorem B.7. Let $e_t := \mathbb{E}(f(x_t) - f(x^*)$ and $T \in \mathbb{N}$. By multiplying with a sequence $(\delta_t)_{t=0}^{T-1}$ and summing, we get

$$\begin{aligned}
\sum_{t=0}^{T-1} \delta_t e_t \leq \frac{\delta_0}{2\alpha_0}\left(1 - \frac{\mu\alpha_0}{2}\right)\mathbb{E}\|\tilde{x}_0 - x^*\|^2 + \frac{1}{2}\sum_{t=1}^{T-1}\left(\frac{\delta_t}{\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right) - \frac{\delta_{t-1}}{\alpha_{t-1}}\right) \\
\mathbb{E}\|\tilde{x}_t - x^*\|^2 - \frac{\delta_{T-1}}{2\alpha_{T-1}}\mathbb{E}\|\tilde{x}_T - x^*\|^2 \sum_{t=0}^{T-1}\delta_t \frac{9}{2}\alpha_t \sum_{\ell=1}^{L} c^{(\ell)} M^{(\ell)} \\
+ \sum_{t=0}^{T-1}\frac{1}{2}\alpha_t\delta_t \sum_{\ell=1}^{L} M^{(\ell)} + \sum_{t=0}^{T-1} 4\mu\alpha_t^2\delta_t \sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)} \\
\leq \frac{\delta_0}{2\alpha_0}\left(1 - \frac{\mu\alpha_0}{2}\right)\mathbb{E}\|\tilde{x}_0 - x^*\|^2 + \frac{1}{2}\sum_{t=1}^{T-1}\left(\frac{\delta_t}{\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right) - \frac{\delta_{t-1}}{\alpha_{t-1}}\right)\mathbb{E}\|\tilde{x}_t - x^*\|^2 \\
- \frac{\delta_{T-1}}{2\alpha_{T-1}}\mathbb{E}\|\tilde{x}_T - x^*\|^2 \left(\frac{9}{2}\sum_{\ell=1}^{L} c^{(\ell)} M^{(\ell)}\right)\sum_{t=0}^{T-1}\delta_t\alpha_t + \left(\frac{1}{2}\sum_{\ell=1}^{L} M^{(\ell)}\right) \\
+ \sum_{t=0}^{T-1}\delta_t\alpha_t + \left(4\mu\sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)}\right)\sum_{t=0}^{T-1}\alpha_t^2\delta_t.
\end{aligned}$$

Similarly to (Stich et al., 2018), we employ the sequence $\delta_t = (t+a)^2/S_T$, with $S_T = \sum_{t=0}^{T-1}(t+a)^2$. This implies that

$$\frac{\delta_t}{\alpha_t} = \frac{\mu(t+a)^3}{8S_T}.$$

Next, we bound each term on the right side. For the first term, we observe that

$$\frac{\delta_0}{2\alpha_0}\left(1 - \frac{\mu\alpha_0}{2}\right)\mathbb{E}\|\tilde{x}_0 - x^*\|^2 \leq \frac{\mu a^3}{16 S_T}\frac{4\sum_{\ell=1}^{L} M^{(\ell)}}{\mu^2} \leq \frac{a^3\sum_{\ell=1}^{L} M^{(\ell)}}{4\mu S_T}.$$

After showing that the second term is nonpositive, we remove it. To this end, for all $t \in [T-1]$

$$\begin{aligned}
\frac{\delta_t}{\alpha_t}\left(1 - \frac{\mu\alpha_t}{2}\right) - \frac{\delta_{t-1}}{\alpha_{t-1}} &= \frac{\mu}{8S_T}\left((t+a)^3 - (t+a-1)^3 - \frac{4(t+a)^3}{t+a}\right) \\
&= \frac{\mu}{8S_T}(-(t+a)^2 - 3(t+a) + 1) \\
&\leq \frac{\mu}{8S_T}(-a^2 + 1) \leq 0.
\end{aligned}$$

For the remaining terms, bounds are derived by

$$\sum_{t=0}^{T-1}\delta_t\alpha_t = \frac{8}{\mu S_T}\sum_{t=0}^{T-1}(t+a) = \frac{8}{\mu S_T}\frac{1}{2}T(T+2a-1) \leq \frac{4T(T+2a)}{\mu S_T}$$

and

$$\sum_{t=0}^{T-1}\delta_t\alpha_t^2 = \frac{64}{\mu^2 S_T}\sum_{t=0}^{T-1} 1 = \frac{64T}{\mu^2 S_T}.$$

Putting together these bounds, we get an upper bound of

$$\begin{aligned}
\sum_{t=0}^{T-1}\delta_t e_t \leq \frac{a^3\sum_{\ell=1}^{L} M^{(\ell)}}{4\mu S_T} + \frac{4T(T+2a)}{\mu S_T}\left(\frac{9}{2}\sum_{\ell=1}^{L} c^{(\ell)} M^{(\ell)} + \frac{1}{2}\sum_{\ell=1}^{L} M^{(\ell)}\right) \\
+ \frac{256T}{\mu S_T}\left(\sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)}\right).
\end{aligned}$$

Finally, the theorem follows from applying the convexity of f for $\bar{x}_T := \sum_{t=0}^{T-1} \delta_t x_t$ as

$$\begin{aligned}\mathbb{E}f(\bar{x}_T) - f(x^*) &= \mathbb{E}\left(f(\sum_{t=0}^{T-1}\delta_t x_t) - f(\sum_{t=0}^{T-1}\delta_t x^*)\right)\\ &\leq \mathbb{E}\sum_{t=0}^{T-1}(\delta_t f(x_t) - \delta_t f(x^*))\\ &= \sum_{t=0}^{T-1}\delta_t e_t.\end{aligned}$$

□

B.3 Proof of Theorem B.9

The proof follows the proofs provided in (Alistarh et al., 2018) and (Shi et al., 2020), but provides a tighter upper bound by Lemma B.13.

Lemma B.18. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be β-smooth and the layer-wise stochastic gradients satisfy $\mathbb{E}\|G^{(\ell)}(x_t)\|^2 \leq M^{(\ell)}$ for all $t \in \mathbb{N}_0$. Then, the iterates satisfy*

$$\alpha_t\mathbb{E}\|\nabla f(x_t)\|^2 \leq \frac{4}{3}(\mathbb{E}f(\tilde{x}_{t+1}) - \mathbb{E}f(\tilde{x}_t)) + \frac{4}{3}\beta^2\alpha_t\mathbb{E}\|x_t - \tilde{x}_t\|^2 + \frac{2}{3}\beta\alpha_t^2\sum_{\ell=1}^{L}M^{(\ell)}. \tag{7}$$

Proof. From the β-smoothness, we get

$$f(\tilde{x}_{t+1}) - f(\tilde{x}_t) \leq \langle\tilde{x}_{t+1} - \tilde{x}_t, \nabla f(\tilde{x}_t)\rangle + \frac{L}{2}\|\tilde{x}_{t+1} - \tilde{x}_t\|^2 = -\alpha_t\langle G(x_t), \nabla f(\tilde{x}_t)\rangle + \frac{L\alpha_t^2}{2}\|G(x_t)\|^2.$$

Taking expectation with respect to the randomness in the stochastic gradients and using its unbiasedness proves that

$$\mathbb{E}_t f(\tilde{x}_{t+1} - f(\tilde{x}_t) \leq -\alpha_t\langle\nabla f(x_t), \nabla f(\tilde{x}_t)\rangle + \frac{\beta\alpha_t^2}{2}\mathbb{E}_t\|G(x_t)\|^2.$$

Using that $-2\langle\nabla f(x_t), \nabla f(\tilde{x}_t)\rangle = \|\nabla f(x_t) - \nabla f(\tilde{x}_t)\|^2 - \|\nabla f(x_t)\|^2 - \|\nabla f(\tilde{x}_t)\|^2$, we obtain

$$\mathbb{E}_t f(\tilde{x}_{t+1}) - f(\tilde{x}_t) \leq \frac{\alpha_t}{2}\|\nabla f(x_t) - \nabla f(\tilde{x}_t)\|^2 - \frac{\alpha_t}{2}\|\nabla f(x_t)\|^2 - \frac{\alpha_t}{2}\|\nabla f(\tilde{x}_t)\|^2 + \frac{\beta\alpha_t^2}{2}\mathbb{E}_t\|G(x_t)\|^2.$$

By rearranging and applying expectation, we arrive at

$$\begin{aligned}&\alpha_t\left(\mathbb{E}\|\nabla f(x_t)\|^2 + \mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 + \mathbb{E}\|\nabla f(x_t) - \nabla f(\tilde{x}_t)\|^2\right)\\ &\leq 2\left(\mathbb{E}_t f(\tilde{x}_{t+1}) - f(\tilde{x}_t)\right) + 2\alpha_t\mathbb{E}\|\nabla f(x_t) - \nabla f(\tilde{x}_t)\|^2 + \beta\alpha_t^2\sum_{\ell=1}^{L}M^{(\ell)}.\end{aligned}$$

Noting that

$$\|\nabla f(x_t)\|^2 \leq 2\|\nabla f(\tilde{x}_t) - \nabla f(x_t)\|^2 + 2\|\nabla f(\tilde{x})\|^2$$

and the β-smoothness of f, we get

$$\begin{aligned}\frac{3\alpha_t}{2}\mathbb{E}\|\nabla f(\tilde{x})\|^2 &\leq 2(\mathbb{E}f(\tilde{x}_t) - \mathbb{E}f(\tilde{x}_{t+1})) + 2\alpha_t\mathbb{E}\|\nabla f(x_t) - \nabla f(\tilde{x}_t)\|^2 + \beta\alpha_t^2\sum_{\ell=1}^{L}M^{(\ell)}\\ &\leq 2(\mathbb{E}f(\tilde{x}_t) - \mathbb{E}f(\tilde{x}_{t+1})) + 2\alpha_t\mathbb{E}\|x_t - \tilde{x}_t\|^2 + \beta\alpha_t^2\sum_{\ell=1}^{L}M^{(\ell)}.\end{aligned}$$

Multiplying this equation with $2/3$ completes the proof. □

Proof of Theorem B.9. We apply Lemma B.13 to Equation 7 and obtain

$$\alpha_t \mathbb{E}\|\nabla f(x_t)\|^2 \leq \frac{4}{3}(\mathbb{E}f(\tilde{x}_t) - \mathbb{E}f(\tilde{x}_{t+1})) + \frac{32}{3}\beta^2\alpha_t^3 \sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)} + \frac{2}{3}\beta\alpha_t^2 \sum_{\ell=1}^{L} M^{(\ell)}.$$

Summing these inequalities from $t = 0$ to $t = T - 1$ yields

$$\sum_{t=0}^{T-1} \alpha_t \mathbb{E}\|\nabla f(x_t)\|^2 \leq \frac{4}{3}(\mathbb{E}f(\tilde{x}_0) - \mathbb{E}f(\tilde{x}_T)) + \frac{2}{3}\beta \sum_{t=0}^{T-1} \alpha_t^2 \sum_{\ell=1}^{L} M^{(\ell)} + \frac{32}{3}\beta^2 \sum_{t=0}^{T-1} \alpha_t^3 \sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)}.$$

Since $\tilde{x}_0 = x_0$ and $f(x^*) \leq f(\tilde{x}_T)$, we get

$$\begin{aligned}\frac{\sum_{t=0}^{T-1} \alpha_t \mathbb{E}\|\nabla f(x_t)\|^2}{\sum_{t=0}^{T-1} \alpha_t} \leq \frac{4(f(x_0) - f(x^*))}{3\sum_{t=0}^{T-1} \alpha_t} + \frac{2\beta \sum_{t=0}^{T-1} \alpha_t^2 \sum_{\ell=1}^{L} M^{(\ell)}}{3\sum_{t=0}^{T-1} \alpha_t} \\ + \frac{32\beta^2 \sum_{t=0}^{T-1} \alpha_t^3 \sum_{\ell=1}^{L} c^{(\ell)2} M^{(\ell)}}{3\sum_{t=0}^{T-1} \alpha_t}.\end{aligned}$$

□

C LAYER-WISE DISCREPANCY BOUNDS

C.1 MAXIMUM MEAN DISCREPANCY

For a compression profile $\mathbf{c} \in \mathcal{C}^L$, let $\widetilde{f}^{\mathbf{c}}$ be the model updated with compressed gradients with profile $\mathbf{c}$, and let f be the uncompressed counterpart. Analogously, define f_l and $\widetilde{f}_l^{\mathbf{c}}$ as the component functions at the l-th layer, where $\widetilde{f}_l^{\mathbf{c}}$ is updated corresponding to c_l. Define the hybrid models

$$F^{\mathbf{c},(l)} := \widetilde{f}_L^{\mathbf{c}} \circ \cdots \circ \widetilde{f}_{l+1}^{\mathbf{c}} \circ f_l \circ \cdots \circ f_1, \qquad l = 0, \ldots, L,$$

where

$$F^{\mathbf{c},(0)} = \widetilde{f}^{\mathbf{c}}, \qquad F^{\mathbf{c},(L)} = f.$$

Let $T^{\mathbf{c},(l)}$ denote the corresponding output random variable, i.e.

$$T^{\mathbf{c},(l)} \mid X \sim F^{\mathbf{c},(l)}(X).$$

For each layer l, define

$$\mathbf{S}_{\mathrm{orig},l} = f_l \circ \cdots \circ f_1(X) \in \mathbb{R}^{d_l},$$

and

$$\mathbf{S}^{\mathbf{c}}_{\mathrm{comp},l} = \widetilde{f}_l^{\mathbf{c}}\left(f_{l-1} \circ \cdots \circ f_1(X)\right) \in \mathbb{R}^{d_l}.$$

Let κ_l denote the RBF kernel used to compute the layer-wise MMD, with reproducing kernel Hilbert space (RKHS) $\mathcal{H}_l$, and let $\kappa_{\mathcal{Y}}$ be a kernel on the output space with RKHS $\mathcal{H}_{\mathcal{Y}}$.

For every l, let

$$\widetilde{H}_l^{\mathbf{c}} = \widetilde{f}_L^{\mathbf{c}} \circ \cdots \circ \widetilde{f}_{l+1}^{\mathbf{c}}$$

denote the compressed suffix following layer l, and define the pullback operator $\Phi_l^{\mathbf{c}}$ by

$$(\Phi_l^{\mathbf{c}} g)(\mathbf{s}) := \mathbb{E}[g(T) \mid T \sim \widetilde{H}_l^{\mathbf{c}}(\mathbf{s})].$$

for $s \in \mathbb{R}^{d_l}$ and $g \in \mathcal{H}_{\mathcal{Y}}$.

Theorem C.1 (Propagation of layer-wise MMD to the output)**.** *For a $c \in \mathcal{C}$ and $l \in [L]$, let $\Phi_l^{\mathbf{c}}$ denote the pullback operator defined above. If there exists $C_l < \infty$ such that*

$$\Phi_l^{\mathbf{c}} g \in \mathcal{H}_l, \quad \|\Phi_l^{\mathbf{c}} g\|_{\mathcal{H}_l} \le C_l \|g\|_{\mathcal{H}_{\mathcal{Y}}}$$

for every $g \in \mathcal{H}_{\mathcal{Y}}$, $c \in \mathcal{C}$ and $l \in [L]$, then

$$\boxed{\mathrm{MMD}_{\kappa_{\mathcal{Y}}}\left(T, \widetilde{T}^{\mathbf{c},(0)}\right) \le \sum_{l=1}^{L} C_l \mathrm{MMD}_{\kappa_l}\left(\mathbf{S}_{\mathrm{orig},l}, \mathbf{S}^{\mathbf{c}}_{\mathrm{comp},l}\right).}$$

Proof. By definition of the RKHS MMD, we have

$$\begin{aligned}
\mathrm{MMD}_{\kappa_{\mathcal{Y}}}\left(T, \widetilde{T}^{\mathbf{c},(0)}\right) &= \sup_{g \in \mathcal{H}_{\mathcal{Y}} : \|g\|_{\mathcal{H}_{\mathcal{Y}}} \le 1} \left| \mathbb{E}[g(T)] - \mathbb{E}[g(\widetilde{T}^{\mathbf{c},(0)})] \right| \\
&= \sup_{g \in \mathcal{H}_{\mathcal{Y}} : \|g\|_{\mathcal{H}_{\mathcal{Y}}} \le 1} \left| \sum_{l=1}^{L} \left(\mathbb{E}[g(T^{\mathbf{c},(l)})] - \mathbb{E}[g(T^{\mathbf{c},(l-1)})] \right) \right| \\
&\le \sum_{l=1}^{L} \sup_{g \in \mathcal{H}_{\mathcal{Y}} : \|g\|_{\mathcal{H}_{\mathcal{Y}}} \le 1} \left| \mathbb{E}[g(T^{\mathbf{c},(l)})] - \mathbb{E}[g(T^{\mathbf{c},(l-1)})] \right|. \\
&\le \sum_{l=1}^{L} \sup_{g \in \mathcal{H}_{\mathcal{Y}} : \|g\|_{\mathcal{H}_{\mathcal{Y}}} \le 1} \left| \mathbb{E}[(\Phi_l^{\mathbf{c}} g)(\mathbf{S}^{\mathbf{c}}_{\mathrm{comp},l})] - \mathbb{E}[(\Phi_l^{\mathbf{c}} g)(\mathbf{S}_{\mathrm{orig},l})] \right|. \\
&\le \sum_{l=1}^{L} C_l \sup_{h_l \in \mathcal{H}_l : \|h_l\|_{\mathcal{H}_l} \le 1} \left| \mathbb{E}[h_l(\mathbf{S}^{\mathbf{c}}_{\mathrm{comp},l})] - \mathbb{E}[h_l(\mathbf{S}_{\mathrm{orig},l})] \right|. \\
&= \sum_{l=1}^{L} C_l \mathrm{MMD}_{\kappa_l}\left(\mathbf{S}_{\mathrm{orig},l}, \mathbf{S}^{\mathbf{c}}_{\mathrm{comp},l}\right).
\end{aligned}$$

□

Remark C.2 (On the suffix RKHS assumption). The assumption that the pullback of every output test function through the compressed suffix belongs exactly to the activation RKHS,

$$\Phi_l^{\mathbf{c}} g \in \mathcal{H}_l,$$

is in general not automatic. However, this assumption can be relaxed to approximate RKHS representability. Let

$$\nu_l^{\mathbf{c}} := \frac{1}{2}(P_l + Q_l^c),$$

where $P_l, Q_l^{\mathbf{c}}$ denote the marginal distributions of $S_{\mathrm{orig},l}$, $S_{\mathrm{comp},l}^{\mathbf{c}}$, respectively. Suppose that, for every g in the output RKHS with $\|g\|_{\mathcal{H}_\mathcal{Y}} \leq 1$ and every $\varepsilon > 0$, there exists $h_{g,\varepsilon} \in \mathcal{H}_l$ such that

$$\|\Phi_l^{\mathbf{c}} g - h_{g,\varepsilon}\|_{L_2(\nu_l^{\mathbf{c}})} \leq \varepsilon$$

and

$$\|h_{g,\varepsilon}\|_{\mathcal{H}_l} \leq C_l(\varepsilon).$$

Then the corresponding one-layer propagation bound becomes

$$\mathrm{MMD}_{k_\mathcal{Y}}\left(T^{\mathbf{c},(l)}, T^{\mathbf{c},(l-1)}\right) \leq C_l(\varepsilon)\mathrm{MMD}_{\kappa_l}\left(\mathbf{S}_{\mathrm{orig},l}, \mathbf{S}_{\mathrm{comp},l}^{\mathbf{c}}\right) + 2\varepsilon.$$

Consequently, the layer-wise telescoping argument yields

$$\mathrm{MMD}_{\kappa_\mathcal{Y}}\left(T, \widetilde{T}^{\mathbf{c},(0)}\right) \leq \sum_{l=1}^{L}\left[C_l(\varepsilon_l)\mathrm{MMD}_{\kappa_l}\left(\mathbf{S}_{\mathrm{orig},l}, \mathbf{S}_{\mathrm{comp},l}^{\mathbf{c}}\right) + 2\varepsilon_l\right].$$

Assuming that $\Phi_l^c g \in L_2(\nu_l^c)$, such approximations exist for Gaussian RBF kernels, as they are dense in $L_2(\nu_l^c)$ for finite Borel measures. However, the RKHS norm bound $C_l^c(\varepsilon)$ may increase as $\varepsilon \downarrow 0$. The membership assumption used in the theorem can therefore be viewed as an idealized condition under which the approximation error vanishes and the constants $C_l^c(\varepsilon)$ remain uniformly bounded.

Corollary C.3. *In the setting from Theorem C.1, the output space MMD controls the total variation distance of the marginal predicted class probabilities between f and $\widetilde{f}^{\mathbf{c}}$:*

$$\boxed{\mathrm{TV}(\mathbb{E}[f(X)], \mathbb{E}[\widetilde{f}^{\mathbf{c}}(X)]) \leq \mathrm{MMD}_{\kappa_\mathcal{Y}}\left(T, \widetilde{T}^{\mathbf{c},(0)}\right).}$$

Proof. Setting $\mathcal{H}_\mathcal{Y} := \mathbb{R}^{|\mathcal{Y}|}$ and $g_j := \mathbf{1}\{y = j\}$ for $j \in [|\mathcal{Y}|]$, then

$$\left|P\left(T = j\right) - P\left(\widetilde{T}^{\mathbf{c},(0)} = j\right)\right| = \left|\mathbb{E}[g_j(T)] - \mathbb{E}[g_j(\widetilde{T}^{\mathbf{c},(0)})]\right| \leq \mathrm{MMD}_{\kappa_\mathcal{Y}}\left(T, \widetilde{T}^{\mathbf{c},(0)}\right)$$

for all $j \in [|\mathcal{Y}|]$. Thus,

$$\begin{aligned}
\mathrm{TV}(\mathbb{E}[f(X)], \mathbb{E}[\widetilde{f}^{\mathbf{c}}(X)]) &= \max_j \left|\mathbb{E}\left[f(X)_j - \widetilde{f}^{\mathbf{c}}(X)_j\right]\right| \\
&= \max_j \left|P\left(T = j\right) - P\left(\widetilde{T}^{\mathbf{c},(0)} = j\right)\right| \\
&\leq \mathrm{MMD}_{\kappa_\mathcal{Y}}\left(T, \widetilde{T}^{\mathbf{c},(0)}\right)
\end{aligned}$$

□

C.2 Cauchy-Schwarz divergence

We refer to Remark 2 of Yu et al. (2024), which says that the CS-divergence measures the cosine similarity between the mean embeddings of the marginal distributions of S_{comp} and S_{orig}:

$$\mu_{\mathrm{orig}} := \mathbb{E}_{S\sim S_{\mathrm{orig}}}[\kappa(S, \cdot)], \quad \mu_{\mathrm{comp}} := \mathbb{E}_{S\sim S_{\mathrm{comp}}}[\kappa(S, \cdot)].$$

The squared MMD between the activations can be expressed as

$$\begin{aligned}\mathrm{MMD}^2(S_{\mathrm{comp}}, S_{\mathrm{orig}}) &= \|\mu_{\mathrm{orig}}\|_{\mathcal{H}}^2 + \|\mu_{\mathrm{comp}}\|_{\mathcal{H}}^2 - 2\langle \mu_{\mathrm{orig}}, \mu_{\mathrm{comp}} \rangle_{\mathcal{H}} \\ &= \|\mu_{\mathrm{orig}}\|_{\mathcal{H}}^2 + \|\mu_{\mathrm{comp}}\|_{\mathcal{H}}^2 \\ &\quad - 2\|\mu_{\mathrm{comp}}\|_{\mathcal{H}} \|\mu_{\mathrm{orig}}\|_{\mathcal{H}} \exp(-D_{\mathrm{CS}}(S_{\mathrm{comp}}, S_{\mathrm{orig}})/2) \\ &= (\|\mu_{\mathrm{orig}}\|_{\mathcal{H}} - \|\mu_{\mathrm{comp}}\|_{\mathcal{H}})^2 \\ &\quad + 2\|\mu_{\mathrm{comp}}\|_{\mathcal{H}} \|\mu_{\mathrm{orig}}\|_{\mathcal{H}} \left(1 - \exp(-D_{\mathrm{CS}}(S_{\mathrm{comp}}, S_{\mathrm{orig}})/2)\right).\end{aligned}$$

Thus, controlling the CS divergence between activations implies controlling the angular difference of the mean embeddings. If compression approximately preserves the norm of the mean embedding, then a small CS-divergence significantly bounds the layer-wise MMD. By inserting the above result into the layer-wise sum in Theorem C.1, we obtain a partial upper bound on the model output MMD where CS divergence controls the angular component of the above summands.

D METHOD DETAILS

D.1 PRELIMINARIES

Our methods fit into a layer-wise adaptive compression framework which measures compression error on gradients to decide layer-wise compression levels, like L-GreCo (Alimohammadi et al., 2024). The framework finds an optimal selection of compression levels to minimize communication overhead during distributed training while keeping model performance acceptably high.

D.1.1 MINIMIZING COMMUNICATION COST UNDER ERROR BOUND

Let a neural network $\mathcal{M}$ with L layers $\ell \in \{1, \ldots, L\}$ be parameterized by $\theta = \{\theta_1, \ldots, \theta_L\}$. Let $\mathcal{C}$ be a finite set of available compression configurations for all layers. We define two useful mappings to set up compression optimization for each layer ℓ and choice $c \in \mathcal{C}$:

- A **communication cost function** $\omega(\ell, c) : \{1, \ldots, L\} \times \mathcal{C} \to \mathbb{R}^+$, representing the communication volume or transmission size after compression, i.e., the size of the gradient produced from compressing a layer's gradient at a given compression level.
- A **compression error measure** $\delta(\ell, c) : \{1, \ldots, L\} \times \mathcal{C} \to \mathbb{R}^+$, representing a chosen measure of compression error induced from compressing a layer's gradient at a given compression level.

Given a global error budget $\mathcal{E}_{\max}$ representing a maximum tolerable cumulative compression error, the objective is to select an optimal configuration profile $\mathbf{c}^* = (c_1, \ldots, c_L)$ that solves Equation (1). A critical component of this formulation is the selection of the global error budget $\mathcal{E}_{\max}$. To ensure that compression does not degrade the final model performance, Alimohammadi et al. (2024) set $\mathcal{E}_{\max}$ to be the cumulative error of a reference compression scheme. It leveraged existing literature to identify compression parameters $c_{\text{ref}} \in \mathcal{C}$ for specific compression schemes, i.e., quantization, sparsification, and matrix decomposition, that are empirically known to allow for full accuracy recovery across various models and datasets when c_{ref} is applied uniformly to each layer ℓ.

By calculating the total cumulative compression error $\mathcal{E}_{\max} = \sum_{\ell=1}^{L} \delta(\ell, c_{\text{ref}})$ produced by the reference configuration, L-GreCo establishes a reference upper bound. This allows the L-GreCo framework to explore heterogeneous compression levels across layers that may be more aggressive than the reference in some layers, provided the total error remains below the validated threshold $\mathcal{E}_{\max}$.

D.1.2 THE L-GRECO SOLUTION STRATEGY

Under specific L_2 gradient error measurement to approximate the optimization trajectory of an uncompressed baseline without any statistical learning guarantees for learned representations from compressed updates, Alimohammadi et al. (2024) have proposed L-GreCo that formulates a knapsack problem from Equation (1) by discretizing the compression errors followed by a dynamic programming. L-GreCo runs in cycles of T training steps: gradients are accumulated per layer, aggregated into a representative gradient matrix $\mathbf{G}_\ell$, and, for each layer–configuration pair, both the resulting transmission size $\omega(\ell, c)$ and error measure $\delta(\ell, c) = \|\mathbf{G}_\ell - \tilde{\mathbf{G}}_\ell^{(c)}\|_2^2$ are evaluated, where $\tilde{\mathbf{G}}_\ell^{(c)}$ is obtained by compressing and decompressing $\mathbf{G}_\ell$ under configuration c. An optimal compression profile $\mathbf{c}^*$ is then selected via dynamic programming for subsequent training steps within the error budget.

This framework operates in cycles of T training steps, where the measurement of compression errors and DP algorithm are performed at the end of each cycle. In the training steps leading up to this, gradients for each layer ℓ are accumulated. At the end of the cycle, the accumulated gradients are aggregated into a single representative gradient matrix $\mathbf{G}_\ell$ for each layer ℓ. Once $\mathbf{G}_\ell$ is obtained, the compression error measurements are calculated. For every layer $\ell \in \{1, \ldots, L\}$ and every available compression configuration $c_j \in \mathcal{C}$, the framework calculates the associated compression error $\delta(\ell, c_j)$ and the resulting transmission size $\omega(\ell, c_j)$, and stores the values. In the original L-GreCo implementation, δ is calculated as the squared L_2 norm of the difference between the

original and compressed-decompressed gradient:

$$\delta(\ell, c) = \|\mathbf{G}_\ell - \tilde{\mathbf{G}}_\ell^{(c)}\|_2^2, \tag{8}$$

where $\tilde{\mathbf{G}}_\ell$ is obtained by compressing $\mathbf{G}_\ell$ with configuration c and subsequently decompressing it. Finally, a DP algorithm is employed using the collected compression errors and transmission sizes to select the optimal configuration vector $\mathbf{c}^*$ by solving equation 1.

D.2 Algorithmic description of GenComp-A

An algorithmic description of GenComp-A is provided in Algorithm 1.

Algorithm 1 Activation-based compression profile selection

Require: Model, parametrized $\mathbf{\Theta}$, learning rate η, cached activations $\{\mathbf{A}_{\ell-1}\}_{\ell=1}^{L}$, accumulated and aggregated gradients $\{\mathbf{G}_\ell\}_{\ell=1}^{L}$, choices $\mathcal{C}$, error measure D, costs $\mathbf{W}$, budget $\mathcal{E}_{\max}$.
Ensure: Selected configuration profile $\mathbf{c}^*$.
  $\mathbf{E} \leftarrow \mathbf{0} \in \mathbb{R}^{\{1,\dots,L\}\times\mathcal{C}}$ {Empty compression error matrix}
  for each layer $\ell = 1, \dots, L$ **do**
    $\mathbf{\Theta}_\ell^{\text{orig}} \leftarrow \mathbf{\Theta}_\ell - \eta\mathbf{G}_\ell$; $\mathbf{S}_{\text{orig}} = \Psi(f_{\mathbf{\Theta}_\ell^{\text{orig}}}(\mathbf{A}_{\ell-1}))$
    for each $c \in \mathcal{C}$ **do**
      $\mathbf{\Theta}_\ell^{\text{comp}} \leftarrow \mathbf{\Theta}_\ell - \eta\tilde{\mathbf{G}}_\ell^{(c)}$; $\mathbf{S}_{\text{comp}} = \Psi(f_{\mathbf{\Theta}_\ell^{\text{comp}}}(\mathbf{A}_{\ell-1}))$
      $\mathbf{E}_{\ell,c} \leftarrow D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}})$
    end for
  end for
  $\mathbf{c}^* \leftarrow$ DynamicProgramming$(\mathbf{E}, \mathbf{W}, \mathcal{E}_{\max})$
  return $\mathbf{c}^*$

D.3 Error measures

We use the following measures with GenComp, defined here with activation scores for GenComp-A and used equivalently directly on gradients for GenComp-G:

- L_2 **norm:** $D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}}) := \|\mathbf{S}_{\text{orig}} - \mathbf{S}_{\text{comp}}\|_2$.
- **Cosine similarity:** We use the following calculation, which rewards strong angular alignment regardless of whether the vectors point in the same or opposite direction, and penalizes orthogonality. $D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}}) := 1 - \|\frac{\mathbf{S}_{\text{orig}}\cdot\mathbf{S}_{\text{comp}}}{\|\mathbf{S}_{\text{orig}}\|_2\cdot\|\mathbf{S}_{\text{comp}}\|_2}\|$.
- **Cauchy-Schwarz (CS) divergence:** We use kernel density estimation as proposed by Yu et al. (2024, Eq. 11) and estimate $D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}})$ on the minibatch of activation scores (κ is the RBF kernel function):

$$\begin{aligned} D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}}) := \log\Big(\frac{1}{m^2}\sum_{i,j=1}^{m}\kappa(\mathbf{S}_{\text{orig,i}}, \mathbf{S}_{\text{orig,j}})\Big) \\ + \log\Big(\frac{1}{n^2}\sum_{i,j=1}^{n}\kappa(\mathbf{S}_{\text{comp,i}}, \mathbf{S}_{\text{comp,j}})\Big) \\ - 2\log\Big(\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\kappa(\mathbf{S}_{\text{orig,i}}, \mathbf{S}_{\text{comp,j}})\Big). \end{aligned}$$

- **Maximum mean discrepancy (MMD):** We follow Gretton et al. (2012, Eq. 3) and estimate $D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}})$ on the minibatch of activation scores with kernel density estimation (κ is

the RBF kernel function):

$$D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}}) := \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} \kappa(\mathbf{S}_{\text{orig,i}}, \mathbf{S}_{\text{orig,j}}) + \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} \kappa(\mathbf{S}_{\text{comp,i}}, \mathbf{S}_{\text{comp,j}}) - \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \kappa(\mathbf{S}_{\text{orig,i}}, \mathbf{S}_{\text{comp,j}}).$$

- **Kullback-Leibler (KL) divergence:** In contrast to CS divergence and MMD, we treat the activations as categorical distributions over their feature dimensions. We apply softmax on $\mathbf{S}_{\text{orig}}$ and $\mathbf{S}_{\text{comp}}$ followed by a logarithm on $\mathbf{S}_{\text{orig}}$ before calculating KL divergence to avoid numerical issues during computation:

$$D(\mathbf{S}_{\text{orig}}, \mathbf{S}_{\text{comp}}) := \text{Softmax}(\mathbf{S}_{\text{comp}}) \cdot \log \left(\frac{\text{Softmax}(\mathbf{S}_{\text{comp}})}{\log \text{Softmax}(\mathbf{S}_{\text{orig}})} \right)$$

E ADDITIONAL RESULTS AND EXPERIMENT DETAILS

E.1 ADDITIONAL RESULTS

Table 3: Results for selected GenComp variants with low-rank approximation training Transformer-XL on Wikitext-103, on 8 nodes with 2 GPUs each. All results are reported from a single run and use cyclic selection. The best results are highlighted in bold, and the second best results are underlined.

	Error measure	Perplexity ↓	Comp. ratio ↑	Total time (s) ↓	ΔTime
Baseline	Gradient L_2 Norm	<u>28.12</u>	23.3	23663	—
GenComp-A	CosSim	**28.01**	16.2	23854	0.8% ↑
	KLDiv	28.42	<u>37.6</u>	**21739**	8.1% ↓
GenComp-G	KLDiv	28.47	**51.3**	<u>22326</u>	5.7% ↓

Table 4: Results for GenComp with low-rank approximation training ResNet18 on CIFAR-100. The best results are highlighted in bold. Results for Accordion (Agarwal et al., 2021) and LEGACY (Essoullami et al., 2026) are presented from Essoullami et al. (2026, Table 3) for comparison. We achieve increased compression ratio at similar accuracy recovery compared to the baselines, with lowered training time for one-shot methods.

	Selection	Comp. error measure	Top-1 acc. ↑	Comp. ratio ↑	Total time (s) ↓	ΔTime
Baseline	—	Uniform	75.55 ± 0.31	72.2 ± 0.0	2052 ± 6	6.2% ↓
	—	Accordion	75.11	—	—	—
	—	LEGACY	75.55	—	—	—
	Cyclic	L_2 Norm (L-GreCo)	75.36 ± 0.14	108.4 ± 0.7	2188 ± 9	—
GenComp-A	Cyclic	L_2 Norm	74.97 ± 0.32	115.9 ± 3.2	2240 ± 11	2.4% ↑
		CosSim	75.30 ± 0.21	119.6 ± 4.6	2363 ± 15	8.0% ↑
		CSDiv	75.11 ± 0.36	121.5 ± 0.8	2284 ± 5	4.4% ↑
		KLDiv	75.01 ± 0.19	124.8 ± 2.0	2234 ± 10	2.1% ↑
		MMD	75.31 ± 0.41	123.4 ± 3.9	2277 ± 11	4.1% ↑
	One-shot	L_2 Norm	75.23 ± 0.18	90.8 ± 4.2	2077 ± 19	5.1% ↓
		CosSim	75.46 ± 0.13	94.2 ± 4.9	2178 ± 18	0.5% ↓
		CSDiv	75.30 ± 0.18	98.6 ± 4.4	2069 ± 4	5.4% ↓
		KLDiv	75.12 ± 0.25	105.7 ± 0.5	2072 ± 14	5.3% ↓
		MMD	75.24 ± 0.43	84.5 ± 2.9	2075 ± 14	5.2% ↓
GenComp-G	Cyclic	CosSim	75.26 ± 0.24	105.2 ± 0.7	2313 ± 26	5.7% ↑
		CSDiv	75.10 ± 0.09	122.5 ± 2.4	2230 ± 7	1.9% ↑
		KLDiv	74.95 ± 0.25	134.0 ± 0.2	2169 ± 11	0.9% ↓
		MMD	75.60 ± 0.33	126.5 ± 4.4	2247 ± 14	2.7% ↑
	One-shot	L_2 Norm	75.34 ± 0.05	106.1 ± 1.7	2076 ± 12	5.1% ↓
		CosSim	**75.84 ± 0.38**	93.4 ± 1.0	2178 ± 19	0.5% ↓
		CSDiv	74.97 ± 0.24	**136.0 ± 0.0**	**2025 ± 20**	7.4% ↓
		KLDiv	75.16 ± 0.22	134.6 ± 1.4	2049 ± 10	6.4% ↓
		MMD	74.99 ± 0.32	112.5 ± 9.5	2074 ± 9	5.2% ↓

Layer-wise adaptive low-rank approximation with ResNet models. We apply GenComp for layer-wise adaptive low-rank approximation when training ResNet18 on CIFAR-100 (Table 4). Overall, most adaptive variants, especially those based on information-theoretic and kernel measures, substantially improve the *compression ratio* relative to the gradient L_2-norm and uniform compression baselines, while remaining within about 0.5% top-1 *accuracy* of these baselines and within about 1% of the no-compression model. One-shot variants generally achieve compression comparable to their cyclic counterparts, with some measures giving slightly higher and others slightly lower compression,

but without clear loss in accuracy. Because one-shot selection avoids repeated evaluations, it consistently reduces wall-clock training time relative to the cyclic gradient L_2 norm baseline, and in some cases even undercuts the uniform baseline. Even with higher compression ratios, we did not measure significantly lowered training times on ResNet18 experiments beyond time savings from one-shot error measurements. We attribute this to ResNet18 being a relatively small model (12M parameters), making the communication bottleneck so small that timing gains become marginal. Nevertheless, these results show that GenComp can substantially increase low-rank compression ratios while maintaining baseline-level accuracy, and that one-shot selection can achieve similar gains with limited runtime overhead.

We also use GenComp with low-rank approximation when training ResNet50 on ImageNet. Due to limited computational resources, we only evaluate a subset of measures, but the trends closely mirror those observed for ResNet18 on CIFAR-100: our cyclic variants increase *compression ratio* compared to both baselines while maintaining *accuracy* and *training time* close to them. One-shot variants generally match or slightly improve baseline accuracies at similar or slightly lower compression, and achieve among the lowest total training times reported. Though this is a larger-scale setting than with ResNet18/CIFAR-100, the model size is at a significantly smaller scale (26M parameters) than Transformer-XL (192M parameters), limiting the potential for as significant timing gains as we observe in Table 1.

Table 5: Results for GenComp with low-rank approximation for training ResNet50 on ImageNet.

	Selection	Measure	Top-1 acc. ↑	Top-5 acc. ↑	Ratio ↑	Time (s) ↓	ΔTime
Baseline	—	Uniform	75.16 ± 0.43	92.34 ± 0.27	66.54 ± 0.00	22290 ± 10	1.11% ↑
	Cyclic	L_2 Norm	75.30 ± 0.30	92.48 ± 0.12	96.66 ± 0.45	22045 ± 31	—
GenComp-A	Cyclic	CosSim	**75.64 ± 0.01**	92.71 ± 0.07	97.93 ± 0.46	22319 ± 129	1.24% ↑
		CSDiv	75.33 ± 0.03	92.35 ± 0.06	**109.28 ± 2.53**	22228 ± 58	0.83% ↑
		MMD	75.06 ± 0.16	92.33 ± 0.11	104.76 ± 1.26	22258 ± 46	0.97% ↑
	One-shot	CosSim	75.42 ± 0.08	92.51 ± 0.10	92.06 ± 6.74	22185 ± 89	0.64% ↑
		CSDiv	75.41 ± 0.43	92.55 ± 0.22	93.37 ± 0.93	22037 ± 22	0.04% ↓
GenComp-G	One-shot	L_2 Norm	75.60 ± 0.15	**92.73 ± 0.04**	94.25 ± 5.58	**22034 ± 3**	0.05% ↓

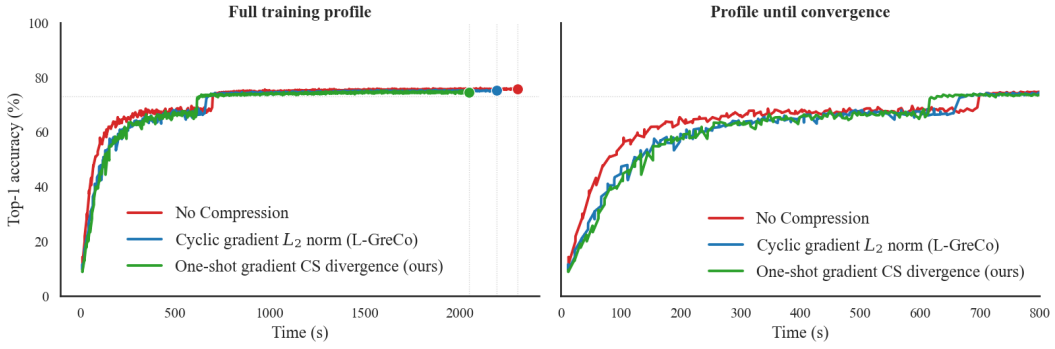


Figure 4: Validation accuracy over time during ResNet18/CIFAR-100 training with PowerSGD. We achieve faster wall-clock convergence with one-shot gradient CS divergence (ours) against no compression and the cyclic gradient L_2 norm baseline.

Layer-wise adaptive sparsification and unbiased quantization with ResNet18 on CIFAR-100. We use GenComp with layer-wise adaptive sparsification and unbiased quantization when training ResNet18 on CIFAR-100. To obtain the results in Table 6, we compress gradients but communicate reconstructed (full-precision) tensors, meaning the compression ratio does not affect timing gains, only accuracy recovery. We therefore omit timing results in Table 6 and focus solely on accuracy recovery and obtained compression ratio. With CS divergence and MMD on activations for sparsification, we use default kernel bandwidths (Appendix E.3). For KL divergence, the error measure values were

Table 6: Results for GenComp with sparsification and unbiased quantization to train ResNet18 on CIFAR-100. Almost every method more than doubles compression ratio with similar accuracy for sparsification. MMD achieves some increased compression with unbiased quantization at comparable accuracy to baselines.

	Selection	Measure	Top-k sparsification		Unbiased quantization	
			Top-1 Acc. ↑	Ratio ↑	Top-1 Acc. ↑	Ratio ↑
Baseline	—	Uniform	74.43 ± 0.34	47.97 ± 0.00	74.50 ± 0.24	7.88 ± 0.00
	Cyclic	L_2 Norm	74.16 ± 0.05	101.16 ± 3.71	$\mathbf{75.45 \pm 0.44}$	11.02 ± 0.03
GenComp-A	Cyclic	L_2 Norm	73.87 ± 0.59	204.93 ± 3.91	74.41 ± 0.44	12.80 ± 0.23
		CosSim	74.40 ± 0.07	187.23 ± 40.94	73.69 ± 0.27	14.56 ± 0.11
		CSDiv	73.74 ± 0.28	231.23 ± 74.09	72.62 ± 0.19	15.00 ± 0.02
		KLDiv	73.70 ± 0.29	254.15 ± 51.09	73.56 ± 0.22	14.66 ± 0.10
		MMD	74.03 ± 0.61	206.35 ± 9.78	72.35 ± 0.48	$\mathbf{15.04 \pm 0.04}$
	One-shot	L_2 Norm	74.42 ± 0.01	268.96 ± 15.24	74.74 ± 0.42	12.22 ± 0.04
		CosSim	74.22 ± 0.57	$\mathbf{337.17 \pm 10.48}$	73.99 ± 0.73	13.63 ± 1.15
		CSDiv	74.20 ± 0.32	302.32 ± 0.00	74.33 ± 0.20	13.80 ± 0.33
		KLDiv	74.41 ± 0.19	319.48 ± 14.69	74.49 ± 0.40	12.61 ± 0.67
		MMD	73.98 ± 0.40	305.82 ± 7.29	74.19 ± 0.16	13.94 ± 0.18
GenComp-G	Cyclic	CosSim	74.25 ± 0.42	202.31 ± 2.38	73.58 ± 0.11	14.09 ± 0.01
		CSDiv	74.49 ± 0.45	292.36 ± 7.96	72.40 ± 0.01	14.91 ± 0.20
		KLDiv	74.29 ± 0.35	175.21 ± 19.17	74.65 ± 0.16	10.23 ± 0.15
		MMD	74.19 ± 0.11	215.80 ± 15.32	75.20 ± 0.15	11.33 ± 0.03
	One-shot	L_2 Norm	74.49 ± 0.49	311.67 ± 9.38	74.99 ± 0.49	10.43 ± 0.16
		CosSim	74.31 ± 0.56	215.66 ± 0.35	73.75 ± 0.20	14.10 ± 0.01
		CSDiv	74.04 ± 0.19	306.54 ± 4.06	72.65 ± 0.33	14.90 ± 0.03
		KLDiv	$\mathbf{74.50 \pm 0.56}$	269.10 ± 3.13	75.06 ± 0.18	9.97 ± 0.04
		MMD	74.32 ± 0.20	252.93 ± 1.38	74.92 ± 0.23	11.46 ± 0.08

often too small and insensitive to compression configurations, so we rescale them to obtain more outspread error values. For sparsification with gradient-space KL divergence, we normalize $\mathbf{G}$ and $\tilde{\mathbf{G}}$ by $\max(\mathrm{std}(\mathbf{G}), \mathrm{std}(\tilde{\mathbf{G}}))$. For unbiased quantization, we apply the same normalization scheme for KL divergence in both gradient and activation space, and additionally scale up gradient norms for activation-space MMD, KL, and CS divergence. Timing results are omitted here, since we only simulate compression and do not evaluate timing gains from the compression.

For sparsification, most cyclic variants more than double the gradient L_2 norm baseline *compression ratio* and substantially outperform uniform sparsification, while keeping top-1 *accuracy* within about 0.5 percentage points of the uniform baseline. One-shot variants typically push compression even further, often well beyond the gradient L_2 norm baseline, while maintaining accuracy essentially on par with baselines. For unbiased quantization, the achievable compression range is smaller, but most adaptive variants still improve the *compression ratio* over uniform and gradient L_2 norm baselines, with accuracy that is generally close to the best baselines. Across sparsification and quantization, as with low-rank approximation, GenComp finds higher-compression configurations at near-baseline accuracy, demonstrating its effectiveness as a general layer-wise adaptive compression framework.

Error propagation-aware variant of GenComp. We provide results from running a selection of GenComp methods on ResNet18/CIFAR-100 with low-rank approximation, using the error-propagation aware version of GenComp. We do not observe consistent significant changes between all compounding error-aware and non-aware variants. Since this variant entails some more computational complexity and memory cost than the non-propagation-aware variant (from storing a copy of each layer during simulated uniform compression, and one more forward pass to obtain activations), we find it more beneficial to use the original GenComp variant, as we do in other experiments.

Table 7: Results for GenComp with error propagation-aware compression profile selection, training ResNet18 on CIFAR-100 with low-rank approximation, using the 4×4 GH200 GPU setup. All values are obtained from 3 runs except the baseline (1 run).

Method	Compounding error-aware	Val. acc. (%) ↑	Comp. rate ↑
Cyclic GenComp-G L_2 Norm (baseline)	×	74.13	108.12
Cyclic GenComp-A L_2 Norm	×	74.51 ± 0.41	116.25 ± 3.37
	✓	74.24 ± 0.25	114.51 ± 5.41
Cyclic GenComp-A CSDiv	×	73.85 ± 0.47	126.17 ± 3.71
	✓	74.28 ± 0.15	121.26 ± 1.77
One-shot GenComp-A L_2 Norm	×	**74.85 ± 0.14**	90.79 ± 4.23
	✓	73.98 ± 0.49	90.25 ± 4.85
One-shot GenComp-A CSDiv	×	74.17 ± 0.30	93.81 ± 4.81
	✓	74.26 ± 0.33	84.52 ± 1.79

Table 8: No-compression results for all models and datasets.

Model / Dataset	Top-1 Acc. ↑	Top-5 Acc. ↑	Perplexity ↓	Total Time (s) ↓
ResNet18 / CIFAR-100	76.03 ± 0.18	—	—	2291 ± 13
ResNet50 / ImageNet	76.26 ± 0.19	92.85 ± 0.21	—	21403 ± 21
Transformer-XL / Wikitext-103	—	—	27.32 ± 0.03	58234 ± 2634

E.2 Computation and memory overhead

Timing breakdown. We measure profile selection overhead as a fraction of total training wall-clock time, separating error-table construction from the dynamic programming solver. Tables 9 and 10 report both, contrasting once per epoch cyclic profile selection with one-shot selection.

The DP solver contributes well under 1% of total training time in every setting. Error-table construction is the dominant cost, and activation-based measures roughly double it relative to the gradient-L_2 baseline in the cyclic regime, reflecting the additional simulated SGD step and forward pass per candidate configuration. PowerSGD's error-table cost is consistently higher than that of top-k and QSGD because each candidate rank requires several power iterations and a full low-rank reconstruction, whereas top-k and QSGD evaluate each candidate with a single sort or quantize/dequantize. Costs are markedly higher on ResNet-18/CIFAR-100 than on ResNet-50/ImageNet, as the smaller per-step training cost on CIFAR-100 amplifies the fraction attributable to fixed-cost periodic work. These overheads remain in the same regime as those reported by Alimohammadi et al. (2024, Table 1).

One-shot profile selection eliminates this overhead almost entirely. Across all compressors and measures, error-table cost drops by roughly two orders of magnitude, bringing activation-based variants below the cost of cyclic gradient-L_2 profile selection despite the richer error measurement.

Table 9: Error-table construction time as a percentage of total training wall-clock time on ResNet-18/CIFAR-100. Reported as cycle / one-shot. Numbers are from a single representative run per configuration.

Compressor	Gradient L_2	Activation CSDiv	Activation CosSim
PowerSGD	4.81% / 0.02%	8.10% / 0.03%	6.80% / 0.03%
top-k	0.18% / <0.01%	1.15% / 0.01%	0.74% / 0.01%
QSGD	0.49% / <0.01%	2.78% / 0.02%	1.30% / 0.01%

Table 10: DP solver time as a percentage of total training wall-clock time, for PowerSGD profile selection. Reported as cycle / one-shot. Numbers are from a single representative run per configuration.

Setup	Gradient L_2	Activation CSDiv	Activation CosSim
ResNet-18/CIFAR-100	0.84% / <0.01%	0.77% / <0.01%	0.79% / <0.01%
ResNet-50/ImageNet	0.02% / <0.01%	0.02% / <0.01%	0.02% / <0.01%

Memory cost. The steady-state memory footprint of GenComp matches that of L-GreCo: both methods maintain an error feedback buffer approximately equal to the model parameter count for biased compressors, replicated across all workers. The remaining buffers discussed below are transient, allocated only around profile selection and released once the error table is built.

Across the steps leading up to a profile selection, L-GreCo accumulates gradients into a representative gradient matrix per layer, adding roughly one model-sized buffer. GenComp accumulates the same gradient matrix and additionally maintains a snapshot buffer of per-layer parameter clones, taken at the start of the accumulation window, used as the basis for the simulated updates. This brings the total to roughly two model-sized buffers, held only on the worker performing the adjustment.

The principal memory difference from L-GreCo is a cache of input activations captured during the same forward pass at which the parameter snapshot is taken, used as input to the simulated forward pass during error-table construction. This cache scales linearly with the profiling minibatch size and with the total activation footprint of the model. The profiling minibatch size, the number of samples used during the simulated forward pass, defaults to the full local training batch and can be reduced to a subset to control memory. Like the snapshot buffer, the cache is held only on the worker performing the adjustment.

E.3 Hyperparameters and configuration details

For both ResNet-18/CIFAR-100 and ResNet-50/ImageNet we follow the training recipes used by Alimohammadi et al. (2024, Section 5.1, Appendix B, Appendix C), with hyperparameters matching those in their published codebase. Their ResNet-18/CIFAR-100 setup builds on the PowerSGD reference codebase (Vogels et al., 2019), while their ResNet-50/ImageNet recipe is adapted from the NVIDIA Training Examples benchmark PyTorch reference (NVIDIA, 2020). We use the same data augmentation pipelines as Alimohammadi et al. (2024), detailed below. The full hyperparameter listings are given in Tables 11 and 12.

Table 11: Training hyperparameters for ResNet-18/CIFAR-100.

Parameter	Value
Number of workers	16
Optimizer	SGD with momentum
Global batch size	256
Momentum	0.9
Base LR	0.1
LR schedule	Step decay by $\times 0.2$ at epochs 60, 120, 160
Epochs	200
Weight decay	10^{-4}

For CIFAR-100 we apply standard data augmentation: a 32×32 random crop with 4-pixel zero padding, followed by a random horizontal flip. Each channel is normalized using the per-channel mean $(0.5071, 0.4865, 0.4409)$ and standard deviation $(0.2673, 0.2564, 0.2762)$ computed on the CIFAR-100 training set.

For ImageNet we use the standard ResNet-50 augmentation pipeline: a random resized crop to 224×224 with scale range $[0.08, 1.0]$, followed by a random horizontal flip. Each channel is normalized using the standard ImageNet means $(0.485, 0.456, 0.406)$ and standard deviations $(0.229, 0.224, 0.225)$.

Table 12: Training hyperparameters for ResNet-50/ImageNet.

Parameter	Value
Number of workers	16
Optimizer	SGD with momentum
Global batch size	4096
Momentum	0.875
LR warmup	Linear for 8 epochs, starting from 0.256
LR schedule	Cosine
Epochs	90
Weight decay	$1/32768$
Label smoothing	0.1

For ResNet18 on CIFAR-100, the warm-up time before gradients are accumulated for compression error estimation includes 10 iterations of uncompressed training, then uniform compression with the default compression configuration until the first adjustment. We chose this to be lower than the convention of following the LR warmup schedule (Alimohammadi et al., 2024) as we observed that both options converged similarly in this setup, meaning we could spend fewer iterations at no compression without sacrificing accuracy recovery. For ResNet50 on ImageNet, the compression warm-up is aligned with the linear LR warm-up of 8 epochs, following the convention of beginning compression once the optimizer has reached its base learning rate.

Transformer-XL on WikiText-103. For Transformer-XL on Wikitext-103 we likewise follow the training recipe of Alimohammadi et al. (2024), trained in FP32 with hyperparameters matching their published codebase with the exceptions of 16 workers instead of 8 and 15,000 total steps instead of 40,000, as listed in Table 13, and use the same pre-processing tools and tokenizers (NVIDIA, 2020). We report test perplexity after the last training step.

Table 13: Training hyperparameters for Transformer-XL/WikiText-103.

Parameter	Value
Number of workers	16
Optimizer	LAMB
Global batch size	256
LR warmup	Linear for 1000 steps
LR schedule	Cosine to 10^{-3}
Training steps	15,000
Weight decay	0

The compression warm-up is aligned with the LR warm-up of 1000 iterations, after which the first profile is selected once 10 gradients have been accumulated. Cyclic variants, including the gradient L_2 norm baseline, re-select the profile every 5000 iterations. For activation-based measures, the cached input to each layer is capped at 1024 tokens, and the KDE-based measures (CS divergence, MMD) subsample to at most 4096 rows before forming kernel matrices, using the same rows for the original and compressed activations.

In our experiments, once all compression errors are calculated but before the dynamic programming, we rescale the compression error matrix ($\mathbf{E}$ in Algorithm 1) with min-max normalization:

$$\mathbf{E}_{i,j}^{\text{norm}} = \frac{\mathbf{E}_{i,j} - \min(\mathbf{E})}{\max(\mathbf{E}) - \min(\mathbf{E})}$$

for each element $\mathbf{E}_{i,j}$ in $\mathbf{E}$. We do this for numerical stability, and for spreading out error estimates on a common scale. In our reported results, we apply normalization on our methods, and not for the gradient L_2 norm baselines as it is not part of the original L-GreCo algorithm (Alimohammadi et al., 2024). We also tested the gradient L_2 norm baseline on ResNet18 and no significant effect from applying the error table normalization (see Table 14).

Table 14: Effect of min-max normalization on GenComp with low-rank approximation (PowerSGD) training ResNet18 on CIFAR-100, with the 4×4 GH200 GPU setup.

	Val. acc. ↑	Comp. ratio ↑	Total time (s) ↓
Without min-max (baseline)	73.93	109.04	10043
With min-max (normalized)	74.38	107.94	10050

For convolutional layers, the simulated forward pass uses each layer's actual stride, padding, dilation, and grouping, recorded at hook registration time and reused via `F.conv2d`.

For low-rank approximation and unbiased quantization, we use the parameter ranges of Alimohammadi et al. (2024). For layer-wise top-k sparsification, we restrict to the following subset of parameter values as our parameter range in ResNet experiments: $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. For Transformer-XL, we expand the parameter range to $\mathcal{C} = \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$

For KDE-based measures (CS divergence, MMD), unless otherwise stated for specific experiments, we set the kernel bandwidth σ via the median heuristic (Gretton et al., 2012). Otherwise, we use default kernel bandwidths as follows. For CS divergence, we have set it to $\sigma = 10$. For default MMD, we use the following bandwidths with an RBF kernel: $\{10, 15, 20, 50\}$. For median pairwise distance-adjusted MMD, we use the following multipliers for bandwidths: $\{0.25, 0.5, 1.0, 2.0, 4.0\}$.

Hardware details. Experiments on CIFAR-100 and ImageNet have been run on 4 nodes of 4 Nvidia L4 GPUs each, using NCCL. Measured inter-GPU and inter-node bandwidth is 36 Gbps with `nccl-tests`, higher than the 13–16 Gbps (inter-GPU) and 10 Gbps (inter-node) reported by Alimohammadi et al. (2024), so communication is less of a bottleneck for models of this size, and timing gains from compression are expected to be smaller. For much larger models, where the gradient element count increases with the parameter count, communication dominates the step time and the gains grow accordingly. Their setup uses Nvidia RTX 3090 GPUs (35 FP32 TFLOPS) vs. 30 FP32 TFLOPS for L4, making computation relatively more expensive in this setup. Experiments with Transformer-XL on WikiText-103 have been run on 4 nodes with 4 Nvidia GH200 GPUs each, also using NCCL. Measured intra-node bandwidth is 1480 Gbps, while inter-node bandwidth is only 5.4 Gbps. In this setup, inter-node communication is more of a bottleneck and timing gains from compression are more pronounced. Their RTX 3090 GPUs are also slower than the GH200 (67 FP32 TFLOPS), making computation relatively cheaper in this setup.