TAH-Quant: Effective Activation Quantization in Pipeline Parallelism over Slow Network

Anonymous authors
Paper under double-blind review

Abstract

Decentralized training of large language models offers the opportunity to leverage computational resources across geographically distributed participants, but is often bottlenecked by network communication, particularly in pipeline-parallel settings. While pipeline parallelism partitions model layers across devices to handle large-scale models, it requires frequent communication of intermediate activations, which can be challenging when network bandwidth is limited. To address these issues, we propose TAH-Quant (**T**ile-wise **A**daptive **H**adamard **Quant**ization), a novel activation quantization framework for pipeline parallelism. TAH-Quant integrates fine-grained tile-wise quantization, entropy-guided tile-wise adaptive bit allocation for bit usage, and a Hadamard-based transformation with pivot swapping to effectively suppress outliers. We prove that pipeline parallel training equipped with TAH-Quant maintains a convergence rate of $\mathcal{O}(1/\sqrt{T})$, matching that of vanilla stochastic gradient descent. Extensive experiments demonstrate that TAH-Quant achieves an aggressive activation quantization ratio of 3–4 bits, providing up to 4.3× throughput speedup over uncompressed FP32 and up to 1.33× wall-clock speedup over AQ-SGD, while preserving training convergence, avoiding AQ-SGD's activation-cache overhead, and generalizing well across various training scenarios.

1 Introduction

Decentralized or open collaborative training of large language models (LLMs) has recently gained significant attention as it enables the pooling of computational resources across multiple geo-distributed participants, thus facilitating the training of models that exceed the capacity of any single resource contributor Ryabinin & Gusev (2020); Yuan et al. (2022); Gandhi et al. (2024). However, a major barrier to these approaches is network communication: unlike specialized clusters equipped with high-speed interconnects, decentralized settings typically rely on slower networks, severely constraining training efficiency Wang et al. (2022; 2023b). On the other hand, scaling LLM training in the state-of-the-art scale necessitates distributed parallel training — particularly pipeline parallelism Huang et al. (2019); Narayanan et al. (2019; 2021), which partitions model layers across multiple stages to support training large-scale models. Yet, pipeline parallelism inherently requires frequent transmission of activations and the corresponding gradients between adjacent pipeline stages. In this paper, we explore *how to effectively compress the communication volume to accommodate pipeline parallelism over a slow network.*

Enabling efficient activation compression for pipeline parallelism over slow network connections has significant implications for democratizing large-scale LLM training Yuan et al. (2022); Wang et al. (2022). Currently, the capability to train state-of-the-art models remains concentrated among institutions equipped with specialized high-performance computing resources. Effectively addressing network communication bottlenecks would substantially reduce barriers to participation, allowing a broader array of contributors, including universities, startups, and individuals, to collaboratively train or fine-tune LLMs Douillard et al. (2025).

On the other hand, a significant obstacle arises from the fact that naive activation compression (e.g., quantization Han et al. (2016); Hubara et al. (2017)) methods can negatively affect training convergence. Unlike gradient compression in data parallelism — where quantization errors typically behave as unbiased

noise compatible with optimization procedures, compressing intermediate activations directly influences the neural network's forward computation, consequently introducing bias into gradient estimates in the later backward propagation Evans & Aamodt (2021); Chakrabarti & Moseley (2019). Specifically, in pipeline parallel training, compression errors incurred during activation transmission propagate through nonlinear transformations and can distort gradient calculations during the backward pass Wang et al. (2022). Thus, aggressively reducing activation precision without careful management will result in performance degradation or even training divergence.

To restrict the error propagation introduced by activation quantization, prior efforts, such as AQ-SGD Wang et al. (2022), have attempted to address this issue by compressing the changes in activations between training epochs rather than the activations themselves, thereby providing theoretical convergence guarantees leveraging the help from the error compensation. Although effective in preserving model accuracy, AQ-SGD requires storing previous activations for the whole dataset to compute these changes, resulting in substantial memory overhead. Such an approach poses practical limitations, especially in resource-constrained environments for the large-volumes of training data where storage capacity and system complexity are critical considerations.

In this paper, we solve this problem with a new approach for effective activation quantization in pipeline parallelism. In particular, we make the following key contributions:

Contribution 1. We propose TAH-Quant, an activation quantization approach to alleviate communication bottlenecks in pipeline-parallel training of LLMs. Specifically, our method includes: (**i**) a fine-grained, tile-wise group quantization technique for localized precision control, effectively limiting quantization error; (**ii**) an entropy-guided, tile-wise adaptive bit allocation method that dynamically assigns precision based on activation distribution characteristics at a granularity that strictly refines token-level allocation, further optimizing the compression efficiency; and (**iii**) a Hadamard-based outlier suppression transform enhanced by a pivot element swap, which effectively mitigates quantization errors arising from extreme activation values. Collectively, these carefully designed techniques enable efficient, accurate low-bit quantization of activations, substantially improving the practicality of decentralized and collaborative LLM training.

Contribution 2. We analyze the effect of TAH-Quant's activation compression on pipeline-parallel optimization under standard stochastic assumptions, together with an empirically validated condition on the resulting compression error. Concretely, we prove that pipeline parallel training equipped with TAH-Quant converges at a rate of $\mathcal{O}(1/\sqrt{T})$, matching that of vanilla SGD.

Contribution 3. We conduct extensive experiments on various LLM training tasks, including fine-tuning (`GPT2-XL`), instruction tuning (`Qwen2.5-3B`), and from-scratch pretraining (`LLaMA-3.2-1B`). We show that TAH-Quant can aggressively quantize activations to 3–4 bits without sacrificing convergence relative to the state-of-the-art, *i.e.*, `AQ-SGD`, while introducing no additional activation-cache storage overhead. These results demonstrate that TAH-Quant is broadly applicable to different training tasks.

2 Preliminary and Related Work

Decentralized training of LLM. Decentralized training of LLMs has garnered significant attention as an interesting attempt to democratize access to large-scale LLM training development Ryabinin & Gusev (2020); Borzunov et al. (2022; 2023); Gandhi et al. (2024); Blagoev et al. (2025). Early efforts demonstrated the feasibility of collaborative training across geographically distributed participants with constrained resources under the scope of data parallelism Diskin et al. (2021); Borzunov et al. (2022), where various effective gradient compression methods have been explored Wang et al. (2023b). To further scale out the training computation, more advanced modes of parallel strategies have been integrated Yuan et al. (2022); Ryabinin et al. (2023); Lu et al. (2024); Strati et al. (2024), for example, Yuan et al. Yuan et al. (2022) addressed the challenges of training foundation models in heterogeneous environments by introducing a scheduling algorithm that optimally allocates computational tasks across decentralized GPUs; Ryabinin et al. Ryabinin et al. (2023) proposed SWARM Parallelism, where temporary randomized pipelines between nodes are adaptively rebalanced to handle dynamic efficient training of large Transformer models using preemptive instances with limited network bandwidth. Douillard et al. Douillard et al. (2025) improve DiLoCo with sequential synchronization, comm–compute overlap, and quantized exchange. Most recently, Ramasinghe et

al. Ramasinghe et al. (2025) propose *Subspace Networks*, which reduce model-parallel activation traffic by constraining projection weights to a shared low-rank subspace. This direction is complementary to TAH-Quant: Subspace Networks reduce communication through weight-parameterization constraints, whereas TAH-Quant is a drop-in activation-compression module that leaves the model parameterization unchanged.

Activation compression in training. Activation compression techniques have been studied to reduce memory and computational overhead in model training Liu et al. (2021); Bersatti et al. (2020); Georgiadis (2019); Fu et al. (2020); Liu et al. (2022); Chen et al.; Bian et al. (2024). Concretely, inherent sparsity in activation has been studied to minimize storage and computation in neural networks. For example, Zhang et al. Zhang et al. (2024) investigate the natural occurrence of sparse activations in pretrained Transformers and dynamically alternate between sparse and dense training phases to enhance pretraining efficiency Rhu et al. (2018); Jiang et al. (2022); Zhang et al. (2024); Li et al.. On the other hand, quantization-based methods Evans et al. (2020); Liu et al. (2021); Wang et al. (2023a) reduce the precision of activations to lower bit-widths, thereby decreasing memory usage. For example, Han et al. Han et al. (2016) presented Deep Compression, combining pruning, trained quantization, and Huffman coding. Hubara et al. Hubara et al. (2017) explore training neural networks with low-precision weights and activations. Chakrabarti et al. Chakrabarti & Moseley (2019) propose backpropagation with approximate activations for memory-efficient training. Chen et al. Chen et al. (2021a) introduced ActNN, employing 2-bit activation compressed training.

Quantization for LLM. Quantization has emerged as a key technique for serving LLMs efficiently by reducing the precision of weights Lin et al. (2024b); Frantar et al. (2022), activations Xiao et al. (2023), and KV-cache Liu et al. (2024c) for the process of generative inference, parameter-efficient fine-tuning Dettmers et al. (2023), and large-scale pretraining You et al. (2024); Liu et al. (2024a). For example, AWQ Lin et al. (2024b) quantizes the LLM weights by identifying a small subset of "salient" weight channels and scales them up before quantization, thereby preserving accuracy even at 4-bit weight precision. KIVI Liu et al. (2024c) proposes a tuning-free 2-bit quantization of the KV cache (with per-channel asymmetric scaling), dramatically reducing memory and enabling longer context lengths with negligible impact on generation quality. QLoRA Dettmers et al. (2023) demonstrated that a 4-bit quantized base model can be *fine-tuned* via low-rank adapters to reach the same performance as full `FP16` fine-tuning. LLM-QAT Liu et al. (2024a) introduces a data-free QAT scheme that allows 4-bit quantization of weights, activations, and even the KV cache while preserving performance for training. One essential problem in such quantization methods is how to effectively resolve the issues of outliers in the quantization group Lin et al. (2024a); Hu et al. (2025); You et al. (2024). An especially simple yet effective transformation for outlier suppression is the *Hadamard transform* Theodoridis & Koutroumbas (2009). Formally, the $N \times N$ Hadamard matrix $\mathbf{H}_N \in \{\pm 1\}^{N \times N}$ is defined such that $\mathbf{H}_N \mathbf{H}_N^T = N\mathbf{I}_N$ (so $\frac{1}{\sqrt{n}}\mathbf{H}_N$ is an orthonormal matrix). Multiplying a vector with dimension N by $\mathbf{H}_N$ will evenly redistribute the vector's components across N dimensions. Recent quantization studies leverage the *Hadamard transform* to suppress outliers and reduce quantization error, mainly for tensor-wise weight quantization in generative inference scenarios, such as QuaRot Ashkboos et al. (2024) and SpinQuant Liu et al. (2024b). In contrast, we apply lightweight Hadamard transforms for *activation communication* in pipeline-parallel training at a finer tile-wise granularity.

3 Activation Quantization

Pipeline-parallel training Huang et al. (2019); Narayanan et al. (2019; 2021) requires communication of intermediate activations between devices. This communication can become a key bottleneck on slow interconnects. To alleviate this by quantization-based compression, we leverage three carefully-designed mechanisms *in order to reduce the quantization error*, including: (**i**) fine-grained tile-wise group quantization for localized precision control (Section 3.1); (**ii**) an entropy-guided tile-wise adaptive bit-width allocation (Section 3.2); and (**iii**) a Hadamard-transform-based outlier suppression with a pivot element swap (Section 3.3). We also discuss how we integrate the proposed TAH-Quant quantization method in pipeline parallel training in Section 3.5. We enumerate the details below.

3.1 Fine-Grained Tile-Wise Group Quantization

First, we introduce a fine-grained, tile-wise group quantization scheme for localized precision control. Specifically, instead of quantizing the entire activation tensor with a single set of parameters, we partition it into small tiles and quantize each tile independently. For example, consider an activation tensor $\mathbf{a}$ of shape $B \times S \times C$, i.e., $\mathbf{a} \in \mathbb{R}^{B \times S \times C}$, where B, S, and C denote the batch size, sequence length, and number of channels (i.e., model dimension), respectively. We partition this tensor along the channel dimension into multiple tiles by grouping contiguous channels within each token. Each such tile (i.e., quantization group) can be noted as $\mathbf{a}_{i,j,t} \in \mathbb{R}^G$, where G is the quantization group size determined by $G = \frac{C}{N_t}$, N_t is the number of partitions of all the channels, and $i = 1, \ldots, B$, $j = 1, \ldots, S$, $t = 1, \ldots, N_t$ are the indices for each tile-wise quantization group. Note that each tile will form a separate quantization group with its own scale and zero-point. This approach ensures that each group is quantized using an optimal dynamic range, improving low-bit accuracy. By confining quantization error to small groups, we avoid the coarse tensor-wise scale being dominated by a few extreme values.

3.2 Tile-Wise Adaptive Bit Allocation

The fine-grained grouping addresses local range variation; however, even within a single token the activation values may have non-uniform importance: some contiguous channel windows are smooth while others contain sharp peaks. We therefore introduce an entropy-based, tile-wise adaptive precision allocation strategy that dynamically adjusts the quantization bit width at the granularity of small channel-window allocation tiles within each token rather than entire tokens.

Tile decomposition. Let $A \in \mathbb{Z}_{>0}$ denote the allocation-tile size, with $A \leq C$ and A chosen to align with the quantization tile boundaries of Section 3.1. Each token is split into $M = C/A$ non-overlapping channel-window allocation tiles. We reshape the activation tensor

$$\mathbf{a} \in \mathbb{R}^{B \times S \times C} \;\longmapsto\; \tilde{\mathbf{a}} \in \mathbb{R}^{B \times (SM) \times A},$$

so that each row $\tilde{\mathbf{a}}_{i,j'} \in \mathbb{R}^A$ represents one allocation tile, indexed by $j' = 1, \ldots, SM$, that combines a token index and a channel-window index. Setting $A = C$ (so $M = 1$) recovers the previously studied token-level allocation as a strict special case; choosing $A < C$ yields finer precision control. In our default configuration, $A = G$, so each allocation tile is also a quantization group.

Per-tile entropy. For every allocation tile $\tilde{\mathbf{a}}_{i,j'} \in \mathbb{R}^A$ we form its normalized magnitude distribution

$$p_k \;=\; \frac{|\tilde{a}_{i,j',k}|}{\|\tilde{\mathbf{a}}_{i,j'}\|_1 + \epsilon}, \quad k = 1, \ldots, A, \tag{1}$$

where ϵ is a small positive constant for numerical stability, and define

$$\mathcal{H}\big(\tilde{\mathbf{a}}_{i,j'}\big) \;=\; \sum_{k=1}^{A} p_k \log\big(p_k + \varsigma\big), \tag{2}$$

with ς a small positive constant avoiding zero arguments to the logarithm. As before, $\mathcal{H}$ is high when energy is spread evenly across the A channels and low when the tile contains a sharp outlier.

Top-p% tile selection. We rank the SM allocation tiles of every sample by their entropy and assign `INT4` (high precision) to the top-p% *highest-entropy* tiles and `INT3` (low precision) to the rest. The intuition is unchanged from the token-level argument: high-entropy tiles lack a single channel that the Hadamard transform of Section 3.3 can isolate, so they need extra precision; low-entropy tiles concentrate energy on a few channels and thus tolerate aggressive quantization *after* the outlier-suppression transform. The resulting 1-bit per-tile bit map is packed with the quantized payload as metadata.

Why tiles rather than whole tokens. Empirically we observe that, within a single token, the entropy of different channel-window tiles can differ by more than an order of magnitude — some tiles contain a strong outlier while others are perfectly smooth. Whole-token allocation forces both kinds of tiles to share

the same bit budget, wasting bits on smooth regions and starving spiky regions. Tile-wise allocation avoids this mismatch: a token may now be encoded as a heterogeneous mixture of `INT4` and `INT3` tiles. Since tile-wise allocation reduces to token-level allocation when $A = C$ (every allocation tile is a whole token), the assumption used by our convergence proof (Assumption 4.4) carries over verbatim — in fact the per-bit quantization error can only decrease.

3.3 Hadamard-Based Outlier Suppression Transform

Outliers in the activation values can severely degrade the accuracy of low-bit quantization even within a small quantization group. To mitigate quantization error caused by extreme outliers in activation groups, we propose an adaptive Hadamard transform strategy, which consists of three steps: (**i**) a *heuristic-based outlier detection* to decide if transform is needed, (**ii**) a *Hadamard transform with pivot element swap* to redistribute the outlier values in the quantization group, and (**iii**) an *asymmetric uniform quantization* of the values in the quantization group.

Outlier detection heuristic: Given any quantization group $\mathbf{a}_{i,j,t} = [a_1^{i,j,t}, a_2^{i,j,t}, \ldots, a_G^{i,j,t}] \in \mathbb{R}^G$, where $G = \frac{C}{N_t}$ is the quantization group size. For the rest parts in Section 3.3, we simplify the notation as $\mathbf{a}_{i,j,t} = \boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_G] \in \mathbb{R}^G$ to introduce the quantization method within each tile. In order to detect whether an outlier is present, we define the following heuristic:

$$r = \frac{|\alpha^{(1)}|}{|\alpha^{(2)}| + \varrho} \tag{3}$$

Where $\alpha^{(1)}$ and $\alpha^{(2)}$ represent the elements in $\boldsymbol{\alpha}$ with the largest and the second largest absolute values, ϱ is a small positive constant. If r exceeds a threshold τ (empirically, we set $\tau = 2.0$), we will deem $\mathbf{a}_{i,j,t}$ to contain an outlier and apply the Hadamard-based transform as we will introduce below; otherwise, we skip this transform for that tile. We choose τ via a sensitivity study: $\tau = 2.0$ consistently improves convergence compared to always applying ($\tau = 0$) or never applying ($\tau = \infty$) the transform; detailed results are provided in Appendix E.

Hadamard transform with pivot element swap: For a group identified to have an outlier, we perform a *pivot element swap* to align the pivot (the element with the largest absolute value) with the Hadamard matrix structure. Let $d = \arg\max_k |\alpha_k|$ denote the pivot element index (i.e., $\alpha_d = \alpha^{(1)}$). We define a permutation matrix $\mathbf{P}_d \in \mathbb{R}^{G \times G}$ that swaps the first and d-th coordinates, which yields a permuted vector by multiplying this permutation matrix:

$$[\alpha_d, \alpha_2, \ldots, \alpha_1, \ldots, \alpha_G] = [\alpha_1, \alpha_2, \ldots, \alpha_G]\mathbf{P}_d = \boldsymbol{\alpha}\mathbf{P}_d$$

Next, we multiply this transformed vector by a Hadamard matrix $\mathbf{H}_G \in \{\pm 1\}^{G \times G}$ to redistribute the values and resolve the issue of outliers:

$$\dot{\boldsymbol{\alpha}} = \boldsymbol{\alpha}\mathbf{P}_d \frac{1}{\sqrt{G}} \mathbf{H}_G \tag{4}$$

After applying this transform, the extreme value in the original $\boldsymbol{\alpha}$ will be redistributed across all components in the transformed vector $\dot{\boldsymbol{\alpha}}$. This transform greatly reduces the dynamic range of the group: the formerly pivot value is no longer isolated in a single position, yielding a more balanced tile for the activation vector. As a result, the quantization error can be reduced, since a tighter quantization scale can represent the values with higher precision. Notably, because $\mathbf{H}_G$ is orthogonal (i.e., $\mathbf{H}_G\mathbf{H}_G^T = G\mathbf{I}_G$), we can later invert the transform by applying $\mathbf{H}_G^T$ to the de-quantized values when recovery of the original domain is required.

Asymmetric uniform quantization: After the above two steps, the activation values should be uniformly distributed and centered if the outlier issue once existed. Thus, we can apply the a standard asymmetric quantizer (i.e., You et al. (2024)) — if the computed heuristic $r \leq \tau$, we apply this quantizer for the original vector $\boldsymbol{\alpha}$; otherwise, we apply this quantizer for the transformed vector $\dot{\boldsymbol{\alpha}}$.

3.4 Computational Overhead

Following notations in Sections 3.1 and 3.2, we analyze the computational overhead theoretically and empirically. Pivot swapping selects the top-2 entries per quantization tile and performs a conditional swap,

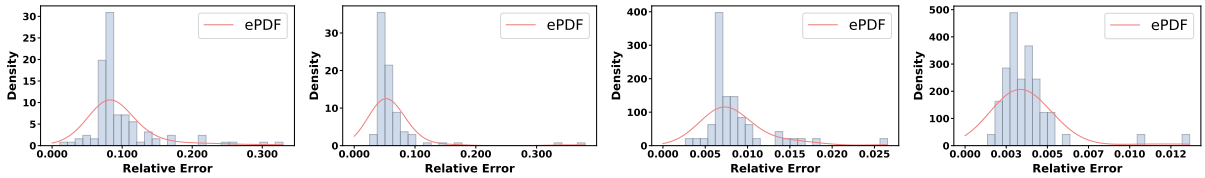
costing $\mathcal{O}(G)$ per tile and $\mathcal{O}(BSN_tG)$ overall (implemented via parallel tensor ops). Entropy computation reduces over A channels per allocation tile, giving $\mathcal{O}(BSC)$ in aggregate, while tile-wise bit allocation ranks $SM = S \cdot C/A$ entropies per sample, costing $\mathcal{O}(B\,SM\log(SM))$, which remains negligible relative to attention and FFN even for small allocation tiles. For outlier tiles, the Hadamard transform applies a $G \times G$ multiplication, costing $\mathcal{O}(G^2)$ per quantization tile and at most $\mathcal{O}(BSN_tG^2)$. Under the default 80% `INT4` + 20% `INT3` setting with $A = G = 64$, TAH-Quant transmits about 4.41 bits per activation element, including roughly 0.61 bits for scale, zero-point, and bit-map metadata. Heterogeneous precisions and metadata are packed into a contiguous `uint8` byte stream before communication, so the communication layer observes a fixed-shape buffer rather than variable-length tensors. In contrast, training step time is dominated by attention $\mathcal{O}(BSC(S+C))$ and FFN $\mathcal{O}(BSC^2)$. Empirically, profiling a 4-stage pipeline with micro-batch size 2 shows TAH-Quant adds only $\sim 1\%$ overhead with the default $A = G = 64$.

3.5 TAH-Quant in Pipeline Parallel Training

Given the carefully designed TAH-Quant quantization method, it is straightforward to integrate it into standard pipeline-parallel training. We illustrate this process in Algorithm 1. For clarity, we present a two-stage pipeline, which can be easily extended to an arbitrary number of stages. Following `AQ-SGD`, we use a simple fixed-point (naive) compressor for gradients in the backward pass. This choice is motivated by system efficiency rather than an algorithmic limitation. Backward propagation typically incurs substantially more computation than forward propagation, providing more opportunity for computation–communication overlap, so a higher-bit naive quantizer (e.g., 6–8 bits) is usually sufficient in practice. We emphasize that TAH-Quant also applies to backward gradients under ultra-low precision. An empirical comparison is provided in Appendix G.

Algorithm 1 TAH-Quant in a two-stage pipeline parallel training.

Initialize: sub-network $a(-)$ weights $\mathbf{x}^{(a)}$, sub-network $b(-)$ weights $\mathbf{x}^{(b)}$, optimizer ρ.
for t = 1, . . . , T **do**
  Randomly sample training batch ξ_t.
  // Forward propagation:
  Machine a sends the quantized output activations $Q_{\text{TAH-Quant}}\left(a(\xi_t, \mathbf{x}_t^{(a)})\right)$ to Machine b.
  Machine b dequantizes the received activation $Q_{\text{TAH-Quant}}\left(a(\xi_t, \mathbf{x}_t^{(a)})\right)$.
  // Backward propagation:
  Machine b sends the quantized gradients w.r.t the activations $Q_{\text{Naive}}\left(\nabla_a(F \circ b)|_{\xi_t}\right)$ back to Machine a.
  Machine a dequantizes the received gradient w.r.t the activations $Q_{\text{Naive}}\left(\nabla_a(F \circ b)|_{\xi_t}\right)$.
  // Parameter updates:
  Machine a updates its parameters with gradients $\hat{\mathbf{g}}^t\left(\mathbf{x}^{(a)}\right)$ using optimizer ρ.
  Machine b updates its parameters with gradients $\hat{\mathbf{g}}^t\left(\mathbf{x}^{(b)}\right)$ using optimizer ρ.
end for
Output: $\mathbf{x} = (\mathbf{x}_T^{(a)}, \mathbf{x}_T^{(b)})$



(a) step wise; tile size 64. (b) step wise; tile size 32. (c) full dataset; tile size 64. (d) full dataset; tile size 32.

Figure 1: Empirical justification of Assumption 4.4.

4 Theoretical Analysis

In this section, we present convergence guarantees for TAH-Quant, which aims to solve the following stochastic optimization problem in a pipeline-parallel fashion:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \quad \mathbb{E}_{\xi\in\mathcal{D}}[F(\mathbf{x};\xi)] \tag{5}$$

where $\mathbf{x}$ denotes the model weights distributed across different pipelines, and ξ represents random data drawn from the distribution $\mathcal{D}$. We denote $\nabla F(\mathbf{x};\xi)$ as the stochastic gradient and $\nabla f(\mathbf{x})$ as the full-batch gradient. Without loss of generality, we consider using momentum SGD as the optimizer ρ in Algorithm 1:

$$\mathbf{m}^t = (1-\beta_1)\mathbf{m}^{t-1} + \beta_1\hat{\mathbf{g}}^t, \tag{6}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta\mathbf{m}^t, \tag{7}$$

where $\beta_1 \in (0,1)$ is the momentum coefficient and η is the learning rate. The vector $\hat{\mathbf{g}}^t$ is a quantized estimate of the stochastic gradient $\nabla F(\mathbf{x}^t;\xi^t)$, obtained through Lines 3–7 of Algorithm 1. Specifically, it takes the form $\big(\hat{\mathbf{g}}^t(\mathbf{x}^{(a_1)}), \hat{\mathbf{g}}^t(\mathbf{x}^{(a_2)}), \ldots, \hat{\mathbf{g}}^t(\mathbf{x}^{(a_N)})\big)$, where $a_1, a_2, \ldots, a_N$ index the machines in the pipeline-parallel system. Our analysis can be extended to the Adam optimizer with a few more involved derivations.

4.1 Assumptions

Assumption 4.1 (Lower Boundedness)**.** The loss function $f : \mathbb{R}^d \to \mathbb{R}$ satisfies $\inf_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) > -\infty$.

Assumption 4.2 (L-Smoothness)**.** The loss function f is L-smooth, *i.e.*, it holds for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ that

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \le L\|\mathbf{x}-\mathbf{y}\|_2.$$

Assumption 4.3 (Stochastic Gradient)**.** We assume that for some $\sigma > 0$, the stochastic gradient oracle satisfies

$$\begin{aligned} \mathbb{E}\big[\nabla F(x^t;\xi^t)\big] &= \nabla f(x^t), \\ \mathbb{E}\big[\|\nabla F(x^t;\xi^t) - \nabla f(x^t)\|^2\big] &\le \sigma^2. \end{aligned} \tag{8}$$

Assumptions 4.1–4.3 are standard assumptions commonly used in stochastic optimization. The following assumption states that gradient quantization through TAH-Quant proposed in Algorithm 1 does not introduce significant distortion to the true stochastic gradient.

Assumption 4.4 (Quantization Error)**.** Let $\mathbf{g}^t$ denote the original stochastic gradient $\nabla F(\mathbf{x}^t, \xi^t)$, and $\hat{\mathbf{g}}^t$ denote the quantized stochastic gradient obtained through TAH-Quant. For some $\delta \in (0,1]$, it holds that

$$\|\hat{\mathbf{g}}^t - \mathbf{g}^t\|^2 \le (1-\delta)\|\mathbf{g}^t\|^2, \tag{9}$$

$$\|\mathbb{E}_{\xi^t\sim\mathcal{D}}[\hat{\mathbf{g}}^t] - \nabla f(\mathbf{x}^t)\|^2 \le (1-\delta)\|\nabla f(\mathbf{x}^t)\|^2, \tag{10}$$

The above assumption ensures that the quantized gradient $\hat{\mathbf{g}}$ remains close to the true gradient $\mathbf{g}$, with their closeness measured by the quantization coefficient δ. A larger δ (*i.e.*, $\delta \to 1$) indicates a smaller quantization error. When $\delta = 1$, we have $\hat{\mathbf{g}} = \mathbf{g}$, implying no quantization error.

Empirical justification of Assumption 4.4. We now empirically verify that TAH-Quant satisfies Assumption 4.4. To validate inequality (9), we conduct fine-tuning experiments on the Gemma2-2B model using the Math-7K dataset. At each training step, we compute the relative error $\|\hat{\mathbf{g}}^t - \mathbf{g}^t\|^2/\|\mathbf{g}^t\|^2$, as shown in Figures 1a and 1b. The results indicate that the relative errors remain below 0.4 across all steps, confirming the validity of (9) with $\delta = 0.6$. To validate inequality (10), we conduct experiments on the same model and dataset. At each step, we compute both the expected compressed gradient $\mathbb{E}_{\xi^t\sim\mathcal{D}}[\hat{\mathbf{g}}^t]$ and the full-batch gradient $\nabla f(\mathbf{x}^t)$, and then evaluate the relative error $\|\mathbb{E}_{\xi^t\sim\mathcal{D}}[\hat{\mathbf{g}}^t] - \nabla f(\mathbf{x}^t)\|^2/\|\nabla f(\mathbf{x}^t)\|^2$, as shown in Figures 1c and 1d. All relative errors are below 0.1, confirming the validity of (10) with $\delta = 0.9$. In both experiments, we use tile sizes of 64 and 32, with 80% `INT4` and 20% `INT3` quantization. These experiments demonstrate the effectiveness of TAH-Quant, which quantizes variables to smaller sizes without incurring significant errors.

4.2 Convergence Guarantees

Under the above assumptions, we are ready to provide convergence guarantees of our proposed TAH-Quant method.

Theorem 4.5. *Under Assumptions 4.1–4.4, if* $\delta \in (0,1)$, $\beta_1 \in \left(0, \frac{\delta}{24-12\delta}\right)$ *and* $\eta \leq \min\left\{\frac{1}{2L}, \frac{\beta_1}{L}\cdot\sqrt{\frac{\delta}{8}}\right\}$, *TAH-Quant with momentum SGD converges as*

$$\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}\Big[\big\|\nabla f(\mathbf{x}^t)\big\|_2^2\Big] \leq \frac{8\big(f(\mathbf{x}^0)-\inf_{\mathbf{x}} f(\mathbf{x})\big)}{\delta\,\eta\,(T+1)} + \frac{8\,\big\|\mathbf{m}^0-\nabla f(\mathbf{x}^0)\big\|_2^2}{\delta\,\beta_1\,(T+1)} + \frac{24\,\beta_1\,\sigma^2}{\delta}.$$

Corollary 4.6. *Under Assumptions 4.1–4.4, if we choose* $\beta_1 = \left(\frac{24}{\delta}+\sigma\sqrt{\frac{\delta^{1/2}(T+1)}{L\Delta}}\right)^{-1}$, $\eta = \left(2L+\frac{2^{3/2}L}{\delta^{1/2}\beta_1}\right)^{-1}$, *TAH-Quant with momentum SGD converges as*

$$\frac{\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2]}{T+1} = \mathcal{O}\left(\frac{L\Delta}{\delta^{5/2}(T+1)} + \sqrt{\frac{L\Delta\sigma^2}{\delta^{5/2}(T+1)}}\right),$$

where $\Delta := f(\mathbf{x}^0) - \inf_{\mathbf{x}} f(\mathbf{x}) + (\delta/L)\cdot\|\mathbf{m}^0-\nabla f(\mathbf{x}^0)\|_2^2$ *(Proofs are in Appendix B).*

Remark. Corollary 4.6 yields three key implications. First, it guarantees that the proposed TAH-Quant algorithm converges to a stationary solution of problem (5). Second, it shows that TAH-Quant achieves a convergence rate of $\mathcal{O}(1/\sqrt{T})$, matching that of vanilla momentum SGD without gradient quantization. This demonstrates that TAH-Quant effectively preserves the valuable gradient information during quantization. Third, the theorem indicates that the convergence rate is affected by the quantization error, quantified by the coefficient δ. This is consistent with our expectations. Since TAH-Quant maintains a relatively large δ (i.e., close to 1), the quantization error remains moderate and does not significantly slow convergence.

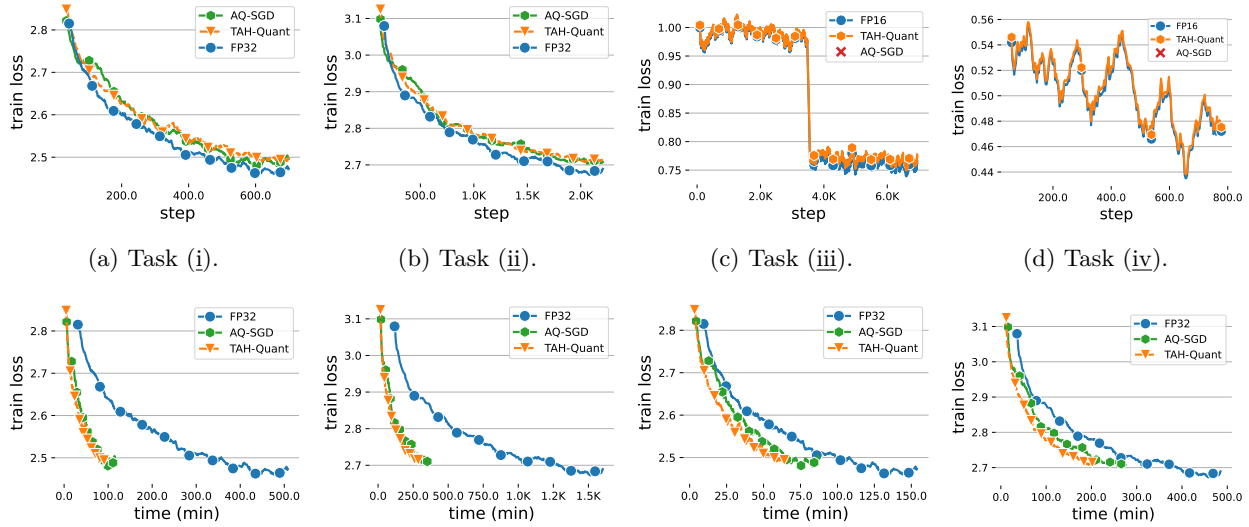
5 Evaluation

We demonstrate that TAH-Quant significantly accelerates LLM training over slow network connections. Specifically, we show that: (**i**) on representative benchmarks, TAH-Quant enables aggressive quantization of activations and gradients without compromising convergence performance or incurring notable additional system overhead (Section 5.2); and (**ii**) the effectiveness of our system design is validated through a series of ablation studies (Section 5.3).

5.1 Experimental Setup

Datasets and benchmarks. We evaluate the proposed method on five distinct training scenarios spanning language modeling and instruction-following tasks. Specifically, we fine-tune `GPT-2XL` (1.5B parameters) on (**i**) `WikiText-2`, a standard Wikipedia-based language modeling benchmark, and (**ii**) `ArXiv21`, a corpus of research paper abstracts from arXiv. To assess performance on instruction data, we fine-tune `Qwen2.5-3B` (3B parameters) on (**iii**) `Magicoder-Evol-Instruct-110K`, a dataset of 110k code-related instruction-response pairs, and (**iv**) `Open-Platyups`, a composite open-source instruction tuning dataset covering multiple domains. Finally, we launch from-scratch pretraining of `LLaMA-3.2-1B` (1.2B parameters) on (**v**) `SlimPajama-6B`, a deduplicated 6B-token subset of the SlimPajama corpus tokenized with the LLaMA-3.2 tokenizer. These setups cover both general and specialized tasks, as well as supervised instruction tuning and LLM pretraining. We additionally include a supplementary `LLaMA-8B` pretraining run on `Proof-Pile` in Appendix C.

Distributed cluster. The fine-tuning and instruction-tuning experiments (Tasks (**i**)–(**iv**)) are conducted on UCloud ucl using a distributed cluster of 8 instances, each equipped with an Nvidia RTX 3090 GPU.



(a) Task (i). (b) Task (ii). (c) Task (iii). (d) Task (iv).

(e) Task (i), 100Mbps. (f) Task (ii), 100Mbps. (g) Task (i), 500Mbps. (h) Task (ii), 500Mbps.

Figure 2: Top row: training-convergence comparison on the fine-tuning and instruction-tuning tasks (loss vs. steps). Bottom row: end-to-end training performance under different network bandwidths (loss vs. wall-clock time) on tasks (**i**) and (**ii**). See Section 5.1 for the detailed experimental setup of each task. Pretraining task (**v**) is summarized in Table 1.

Each model is partitioned into 8 or 4 pipeline stages (one stage per GPU) to execute pipeline parallelism. The cluster's default interconnect bandwidth is 10 Gbps. To emulate slow-network conditions, we throttle inter-instance communication using Linux traffic control (tc), artificially limiting the bandwidth to sub-1 Gbps during training. This setting is common in decentralized environments and is frequently used in prior LLM training evaluations Kim et al. (2025); Lim et al. (2024); Li et al. (2024); Erben et al. (2023); Wang et al. (2023b); Borzunov et al. (2023); similar constraints also arise in decentralized inference Mei et al. (2025).

Pretraining setup (Task (v)). Because the pretraining workload is substantially heavier than fine-tuning, we run Task (**v**) on a single 8-GPU server shared by both the `FP16` baseline and the `TAH-Quant` run, each occupying 4 GPUs and configured with pipeline depth PP=4. Sequence length is 4096, the global batch size is 256 (i.e. $\sim$1.05 M tokens / step), and the learning rate follows a cosine decay (peak 1.5×10^{-4}, min 1×10^{-5}).

Baselines. We compare our approach with two baseline communication strategies[1].

- `FP32/FP16`, which uses full-precision 32-bit (in Tasks (**i**) and (**ii**)) or 16-bit floating point (in Tasks (**iii**), (**iv**), and (**v**)) communication with no compression.
- `AQ-SGD`, the error-compensated low-bit activation quantization method with theoretical convergence guarantees Wang et al. (2022).

Default bit allocation. Unless otherwise specified, TAH-Quant uses 80% INT4 + 20% INT3 mixed-precision activation quantization, with quantization tile size $G = 64$ and allocation-tile size $A = 64$. This 4/3-bit setting is the lowest-bit configuration that remains consistently stable in training; more aggressive choices (e.g., INT2) often lead to non-convergence. The 80/20 split balances communication reduction and convergence stability (Appendix F).

5.2 End-to-End Performance Results

To systematically evaluate the proposed TAH-Quant quantization method, we conduct the experiment and report the corresponding results in terms of training convergence, downstream task performance, and end-to-end training time.

[1]Each baseline is integrated into the same pipeline parallel training setup for fair comparison.

Convergence. Figure 2 (top row) and Table 1 together summarize the convergence comparisons across tasks, demonstrating the efficacy and robustness of TAH-Quant. Specifically, on tasks (**i**) and (**ii**), where `AQ-SGD` is executable due to manageable dataset sizes and a multi-epoch training paradigm, TAH-Quant achieves comparable or slightly superior convergence compared to `AQ-SGD`. On larger-scale tasks (i.e., tasks (**iii**), (**iv**), and (**v**)), where `AQ-SGD` becomes infeasible due to prohibitive storage requirements (Task (**iii**)) or the single-epoch training constraint (Tasks (**iv**) and (**v**)), TAH-Quant still closely matches the standard `FP16` baseline. For from-scratch pretraining of `LLaMA-3.2-1B` on `SlimPajama-6B` (Table 1), TAH-Quant matches `FP16` not only in training loss but also in held-out validation perplexity throughout the entire ∼6 B-token training window, indicating that the proposed activation quantization does not degrade out-of-sample performance.[2]

Table 1: Pretraining `LLaMA-3.2-1B` from scratch on `SlimPajama-6B` (PP=4, sequence length 4096, global batch size 256). Train loss is the rolling mean over a 300-step window at the indicated token milestone; validation perplexity is measured at the closest evaluation checkpoint. Across the entire ∼6 B-token horizon, TAH-Quant is statistically indistinguishable from the uncompressed `FP16` baseline.

	Train loss (↓)						*Validation perplexity* (↓)					
Tokens	1 B	2 B	3 B	4 B	5 B	6 B	1 B	2 B	3 B	4 B	5 B	6 B
`FP16`	3.778	3.326	3.149	3.032	2.954	2.893	39.18	28.54	25.36	23.03	21.42	20.67
`TAH-Quant`	3.781	3.325	3.147	3.031	2.953	2.891	38.47	28.51	25.30	22.99	21.41	20.64

Table 2: `Qwen2.5-3B` SFT evaluation. Fine-tuning data are Open-Platyups for ARC/TQ/WG and Magicoder for HE; see Appendix A.

Model	AVG	Open-Platyups			Magicoder
		ARC	TQ	WG	HE
`Qwen`	51.13	47.35	48.85	68.67	39.63
`SFT-FP16`	59.08	50.00	50.49	69.38	66.46
`SFT-TAH`	59.32	49.91	49.61	70.00	67.68

Furthermore, Table 2 reports downstream evaluations for the SFTed `Qwen2.5-3B` model in Tasks (**iii**) and (**iv**), showing that fine-tuning with TAH-Quant preserves model quality and achieves nearly identical performance to `FP16` across multiple benchmarks. Overall, the results highlight that TAH-Quant enables aggressive activation quantization without extra memory overhead, making it broadly applicable and scalable in realistic training scenarios.

End-to-end training time. TAH-Quant is designed to preserve per-step convergence while reducing communication time, so its benefit appears primarily in wall-clock efficiency under bandwidth-limited networks rather than in fewer optimization steps. As illustrated in the bottom row of Figure 2, TAH-Quant reaches the same loss faster than both uncompressed communication and `AQ-SGD`, with up to 1.33× wall-clock speedup over `AQ-SGD`. We attribute the speedup over `AQ-SGD` to eliminating the activation-cache offloading used by its error-compensation mechanism. Table 3 reports the corresponding `GPT-2XL` training throughput under 1Gbps, 500Mbps, 300Mbps, and 100Mbps bandwidth; TAH-Quant achieves up to 4.3× throughput speedup over uncompressed `FP32`.

Table 3: `GPT2-xl` throughput (tokens / s).

Network Bandwidth	`FP32`	`AQ-SGD` fw4 bw8	`TAH-Q` fw~4 bw6
1Gbps	2600	4749	5650
500Mbps	2482	4311	5749
300Mbps	1761	4369	5120
100Mbps	751	3310	4045

Micro-benchmarks. To further characterize the impact of system parameters, we sweep network bandwidth and micro batch size while fixing the global batch size to 32, and measure end-to-end throughput (tokens/s).

[2]We further discuss backward-gradient quantization and numerical stability in Appendix G.

The speedup of TAH-QUANT is larger under lower bandwidth, and is maximized at moderate micro-batch sizes (mbs=2–4). With very large mbs, the number of micro-batches decreases and the idle time of pipeline bubbles increases, slightly reducing the speedup. Full results are reported in Appendix D.

5.3 Ablation Study

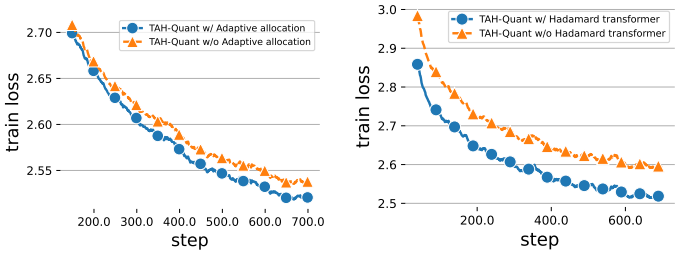
To evaluate the contributions of each module in TAH-QUANT to reduce quantization error under pipeline parallel training, we perform a series of ablation studies and enumerate the experimental results below:

First, to study how the **tile-wise quantization group size** influences statistical efficiency, we vary the group size to 8, 32, and 128 and compare SFTed `Qwen2.5-3B` models over the set of benchmarks. Table 4 shows that smaller groups (TS=8 and TS=32) achieve nearly identical scores on both MMLU and ARC, whereas TS=128 measurably degrades MMLU (from 64.88 to 64.34) and ARC (from 50.34 to 49.66). We attribute this to the within-group statistical heterogeneity that grows with TS: as the group spans more channels, the single shared scale factor must accommodate increasingly disparate activation magnitudes, inflating the per-element quantization error. We therefore adopt TS=64 as our default, which retains the accuracy of small groups while reducing the per-tile metadata overhead relative to TS=8 by an 8× factor.

Table 4: Ablation: tile-wise quantization group size (TS).

TS	MMLU	ARC
8	64.60	50.34
32	64.88	49.91
128	64.34	49.66

Second, to examine the effectiveness of the **entropy-guided adaptive bit allocation**, we compare TAH-QUANT with adaptive bit allocation enabled against a variant without adaptive allocation. Results in Figure 3a show that adaptive allocation accelerates training convergence — the training loss consistently decreases faster with adaptive bit allocation, reflecting reduced quantization error during compression. These observations validate our design of incorporating entropy-based tile-wise bit-width allocation.



(a) Adaptive bit allocation. (b) Hadamard transform.

Figure 3: Ablation studies on TAH-QUANT components.

Third, we evaluate the necessity of the **Hadamard-based outlier suppression** component in TAH-QUANT. Comparing training performance between setups with and without the Hadamard transform reveals that including this transform notably improves training stability and convergence speed — in Figure 3b, the training loss is substantially lower across the training phase when the Hadamard transform is applied, underscoring its effectiveness in mitigating quantization errors from outliers. This finding confirms the value of combining pivot element swapping with the Hadamard transform.

6 Conclusion

We present TAH-QUANT, a novel activation quantization method that alleviates communication bottlenecks in decentralized pipeline-parallel training of LLMs. TAH-QUANT integrates fine-grained tile-wise quantization for localized error control, entropy-guided tile-wise bit allocation that refines token-level allocation, and a Hadamard-based transform with pivot swapping to mitigate outliers. We theoretically show that pipeline-parallel training with TAH-QUANT preserves the same convergence rate ($\mathcal{O}(1/\sqrt{T})$) as standard SGD. Empirical results demonstrate that TAH-QUANT compresses activations to 3–4 bits without degrading convergence, while improving throughput over uncompressed communication, reducing wall-clock time relative to AQ-SGD, avoiding activation-cache overhead, and maintaining robust generalization across fine-tuning, instruction tuning, and from-scratch pretraining.

References

Ucloud. https://www.ucloud.cn/en/.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Andrei Bersatti, Nima Shoghi Ghalehshahi, and Hyesoon Kim. Neural network weight compression with nnw-bdi. In *The International Symposium on Memory Systems*, pp. 335–340, 2020.

Song Bian, Dacheng Li, Hongyi Wang, Eric Xing, and Shivaram Venkataraman. Does compressing activations help model parallel training? *Proceedings of Machine Learning and Systems*, 6:239–252, 2024.

Nikolay Blagoev, Lydia Yiyu Chen, and Oğuzhan Ersoy. Skippipe: Partial and reordered pipelining framework for training llms in heterogeneous networks. *arXiv preprint arXiv:2502.19913*, 2025.

Alexander Borzunov, Max Ryabinin, Tim Dettmers, Quentin Lhoest, Lucile Saulnier, Michael Diskin, and Yacine Jernite. Training transformers together. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 335–342. PMLR, 2022.

Alexander Borzunov, Max Ryabinin, Artem Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, and Colin A Raffel. Distributed inference and fine-tuning of large language models over the internet. *Advances in neural information processing systems*, 36:12312–12331, 2023.

Ayan Chakrabarti and Benjamin Moseley. Backprop with approximate activations for memory-efficient network training. *Advances in Neural Information Processing Systems*, 32, 2019.

Jianfei Chen, Lianmin Zheng, Zhewei Yao, Dequan Wang, Ion Stoica, Michael Mahoney, and Joseph Gonzalez. Actnn: Reducing training memory footprint via 2-bit activation compressed training. In *International Conference on Machine Learning*, pp. 1803–1813. PMLR, 2021a.

Joya Chen, Kai Xu, Yuhui Wang, Yifei Cheng, and Angela Yao. Dropit: Dropping intermediate tensors for memory-efficient dnn training. In *The Eleventh International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021b.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Michael Diskin, Alexey Bukhtiyarov, Max Ryabinin, Lucile Saulnier, Anton Sinitsin, Dmitry Popov, Dmitry V Pyrkin, Maxim Kashirin, Alexander Borzunov, Albert Villanova del Moral, et al. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34:7879–7897, 2021.

Arthur Douillard, Yanislav Donchev, Keith Rush, Satyen Kale, Zachary Charles, Zachary Garrett, Gabriel Teston, Dave Lacey, Ross McIlroy, Jiajun Shen, et al. Streaming diloco with overlapping communication: Towards a distributed free lunch. *arXiv preprint arXiv:2501.18512*, 2025.

Alexander Erben, Ruben Mayer, and Hans-Arno Jacobsen. How can we train deep learning models across clouds and continents? an experimental study. *arXiv preprint arXiv:2306.03163*, 2023.

R David Evans and Tor Aamodt. Ac-gc: Lossy activation compression with guaranteed convergence. *Advances in Neural Information Processing Systems*, 34, 2021.

R David Evans, Lufei Liu, and Tor M Aamodt. Jpeg-act: accelerating deep learning via transform-based lossy compression. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 860–873. IEEE, 2020.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Fangcheng Fu, Yuzheng Hu, Yihan He, Jiawei Jiang, Yingxia Shao, Ce Zhang, and Bin Cui. Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript. In *International Conference on Machine Learning*, pp. 3304–3314. PMLR, 2020.

Rohan Gandhi, Karan Tandon, Debopam Bhattacherjee, Venkata N Padmanabhan, et al. Improving training time and gpu utilization in geo-distributed language model training. *arXiv preprint arXiv:2411.14458*, 2024.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Georgios Georgiadis. Accelerating convolutional neural networks via activation map compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7085–7095, 2019.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. 2016.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Zhihang Yuan, Jiangyong Yu, Chen Xu, Zhe Jiang, and Sifan Zhou. Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. *arXiv preprint arXiv:2501.13987*, 2025.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

Ziyu Jiang, Xuxi Chen, Xueqin Huang, Xianzhi Du, Denny Zhou, and Zhangyang Wang. Back razor: Memory-efficient transfer learning by self-sparsified backpropagation. *Advances in neural information processing systems*, 35:29248–29261, 2022.

Geon-Woo Kim, Junbo Li, Shashidhar Gandham, Omar Baldonado, Adithya Gangidi, Pavan Balaji, Zhangyang Wang, and Aditya Akella. Halos: Hierarchical asynchronous local sgd over slow networks for geo-distributed large language model training. *arXiv preprint arXiv:2506.04531*, 2025.

Zonghang Li, Wenjiao Feng, Weibo Cai, Hongfang Yu, Long Luo, Gang Sun, Hongyang Du, and Dusit Niyato. Accelerating geo-distributed machine learning with network-aware adaptive tree and auxiliary route. *IEEE/ACM Transactions on Networking*, 32(5):4238–4253, 2024.

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*.

Hwijoon Lim, Juncheol Ye, Sangeetha Abdu Jyothi, and Dongsu Han. Accelerating model training in multi-cluster environments with consumer-grade gpus. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 707–720, 2024.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024b.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL `https://aclanthology.org/2022.acl-long.229/`.

Xiaoxuan Liu, Lianmin Zheng, Dequan Wang, Yukuo Cen, Weize Chen, Xu Han, Jianfei Chen, Zhiyuan Liu, Jie Tang, Joey Gonzalez, et al. Gact: Activation compressed training for generic network architectures. In *International Conference on Machine Learning*, pp. 14139–14152. PMLR, 2022.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 467–484, 2024a.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024b.

Zirui Liu, Kaixiong Zhou, Fan Yang, Li Li, Rui Chen, and Xia Hu. Exact: Scalable graph neural networks training via extreme activation compression. In *International Conference on Learning Representations*, 2021.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. In *International Conference on Machine Learning*, pp. 32332–32344. PMLR, 2024c.

Lin Lu, Chenxi Dai, Wangcheng Tao, Binhang Yuan, Yanan Sun, and Pan Zhou. Position: exploring the robustness of pipeline-parallelism-based decentralized training. In *Forty-first International Conference on Machine Learning*, 2024.

Yixuan Mei, Yonghao Zhuang, Xupeng Miao, Juncheng Yang, Zhihao Jia, and Rashmi Vinayak. Helix: Serving large language models over heterogeneous gpus and network via max-flow. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 586–602, 2025.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. Pipedream: generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pp. 1–15, 2019.

Deepak Narayanan, Amar Phanishayee, Kaiyu Shi, Xie Chen, and Matei Zaharia. Memory-efficient pipeline-parallel dnn training. In *International Conference on Machine Learning*, pp. 7937–7947. PMLR, 2021.

Sameera Ramasinghe, Aleksandr Shevchenko, Gil Avraham, and Ajanthan Thalaiyasingam. Subspace networks: Scaling decentralized training with communication-efficient model parallelism. In *Advances in Neural Information Processing Systems*, 2025.

Minsoo Rhu, Mike O'Connor, Niladrish Chatterjee, Jeff Pool, Youngeun Kwon, and Stephen W Keckler. Compressing dma engine: Leveraging activation sparsity for training deep neural networks. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 78–91. IEEE, 2018.

Max Ryabinin and Anton Gusev. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. *Advances in Neural Information Processing Systems*, 33:3659–3672, 2020.

Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. Swarm parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, pp. 29416–29440. PMLR, 2023.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Foteini Strati, Paul Elvinger, Tolga Kerimoglu, and Ana Klimovic. Ml training with cloud gpu shortages: Is cross-region the answer? In *Proceedings of the 4th Workshop on Machine Learning and Systems*, pp. 107–116, 2024.

Sergios Theodoridis and Konstantinos Koutroumbas. Feature generation i: data transformation and dimensionality reduction. *Pattern recognition*, pp. 323–409, 2009.

Guanchu Wang, Zirui Liu, Zhimeng Jiang, Ninghao Liu, Na Zou, and Xia Hu. Division: memory efficient training via dual activation precision. In *International Conference on Machine Learning*, pp. 36036–36057. PMLR, 2023a.

Jue Wang, Binhang Yuan, Luka Rimanic, Yongjun He, Tri Dao, Beidi Chen, Christopher Ré, and Ce Zhang. Fine-tuning language models over slow networks using activation quantization with guarantees. *Advances in Neural Information Processing Systems*, 35:19215–19230, 2022.

Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Re, and Ce Zhang. Cocktailsgd: Fine-tuning foundation models over 500mbps networks. In *International Conference on Machine Learning*, pp. 36058–36076. PMLR, 2023b.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.

Jaeseong You, Minseop Park, Kyunggeun Lee, Seokjun An, Chirag Patel, and Markus Nage. How to parameterize asymmetric quantization ranges for quantization-aware training. *arXiv preprint arXiv:2404.16898*, 2024.

Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 35:25464–25477, 2022.

Zhengyan Zhang, Chaojun Xiao, Qiujieli Qin, Yankai Lin, Zhiyuan Zeng, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. Exploring the benefit of activation sparsity in pre-training. In *International Conference on Machine Learning*, pp. 60040–60056. PMLR, 2024.

A Experimental Details

Downstream evaluation protocol (Table 2). We evaluate the SFTed `Qwen2.5-3B` models on four standard benchmarks: ARC Clark et al. (2018), TruthfulQA (TQ) Lin et al. (2022), WinoGrande (WG) Sakaguchi et al. (2021), and HumanEval (HE) Chen et al. (2021b). All evaluations are conducted in a zero-shot setting using the lm-evaluation-harness Gao et al. (2024) framework. We report normalized accuracy for ARC-Challenge, accuracy for WinoGrande, mc2 for TruthfulQA, and pass@1 for HumanEval. **Fine-tuning.** We fine-tune the `GPT2-xl` on `WikiText-2` and `ArXiv21` for 10 epochs. Specifically, we set the learning rate to 5.0e-6, the batch size to 32, and the micro-batch size to 1, max sequence length to 1024 for both datasets. The learning rate decays linearly after the warm-up stage.

Instruction-tuning. We perform instruction tuning on `Qwen2.5-3B` using `Open-Platyups` and `Magicoder-110K` for 1 and 2 epochs, respectively. The learning rate is set to 2.0e-5, with a batch size of 32 for both datasets. We use a cosine learning rate scheduler for `Open-Platyups`, and a cosine scheduler with a minimum learning rate of 2.0e-6 for `Magicoder-110K`.

Pretraining (Task (v)). We pretrain `LLaMA-3.2-1B` from scratch on the `SlimPajama-6B` corpus (a deduplicated 6 B-token subset of SlimPajama) tokenized with the LLaMA-3.2 tokenizer. The model has 16 transformer layers, hidden size 2048, 32 attention heads, 8 KV heads, and FFN hidden size 8192; we use sequence length 4096 and a global batch size of 256 (i.e. $\sim$1.05 M tokens / step). The peak learning rate is 1.5×10^{-4} with cosine decay to a minimum of 1×10^{-5}, weight decay 0.1, β_1=0.9, β_2=0.95, gradient clipping at 1.0, and a small linear warmup. Pipeline depth is PP=4 (one transformer stage per GPU). The `FP16` baseline and the `TAH-Quant` arm run concurrently on the same 8-GPU server (4 GPUs each), sharing the same hardware and wall-clock budget. The main text reports training through $\sim$6 B tokens, where both arms reach the same loss and validation perplexity within noise.

A supplementary `LLaMA-8B` pretraining run on `Proof-Pile` is reported in Appendix C: $20k$ iterations at global batch size 131,072 tokens (i.e. $\sim$2.5 B tokens, >30% of the corpus), peak learning rate 1.5×10^{-4} with cosine decay to 1×10^{-5}, and weight decay 0.1.

B Complete Proofs

In this section, we provide detailed proofs for Theorem 4.5. We first prove the following lemma.

Lemma B.1 (Descent lemma)**.** *Under Assumption 4.2 and the update rule 7, it holds that*

$$f(\mathbf{x}^{t+1}) \leq f(\mathbf{x}^t) - \frac{\eta}{2}\|\nabla f(\mathbf{x}^t)\|_2^2 - \left(\frac{1}{2\eta} - \frac{L}{2}\right)\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2 + \frac{\eta}{2}\|\mathbf{m}^t - \nabla f(\mathbf{x}^t)\|_2^2. \tag{11}$$

Proof. by Assumption 4.2 we have

$$\begin{aligned} f(\mathbf{x}^{t+1}) \leq & f(\mathbf{x}^t) + \eta\langle \nabla f(\mathbf{x}^t), \frac{1}{\eta}(\mathbf{x}^{t+1} - \mathbf{x}^t)\rangle + \frac{L}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2 \\ = & f(\mathbf{x}^t) - \frac{\eta}{2}\|\nabla f(\mathbf{x}^t)\|_2^2 - \frac{1}{2\eta}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2 + \frac{\eta}{2}\|\nabla f(\mathbf{x}^t) - \mathbf{m}^t\|_2^2 + \frac{L}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2. \end{aligned} \tag{12}$$

where the second equality uses $2\langle a, b\rangle = \|a\|_2^2 + \|b\|_2^2 - \|a - b\|_2^2$ □

Lemma B.2 (momentum contraction)**.** *Under Assumptions 4.1–4.4, if $\delta \in (0, 1)$, it holds that*

$$\begin{aligned} \mathbb{E}[\|\mathbf{m}^t - \nabla f(\mathbf{x}^t)\|_2^2] \leq & \left(1 - \beta_1\left(1 - \frac{\delta}{2}\right)\right)\mathbb{E}[\|\mathbf{m}^{t-1} - \nabla f(\mathbf{x}^{t-1})\|_2^2] + \frac{2L^2}{\delta\beta_1}\mathbb{E}[\|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2^2] \\ & + (\beta_1 + 6\beta_1^2)(1 - \delta)\mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2] + 3(2 - \delta)\beta_1^2\sigma^2. \end{aligned} \tag{13}$$

Proof. According to the update of momentum 7, we have

$$\mathbf{m}^t - \nabla f(\mathbf{x}^t) = (1 - \beta_1)(\mathbf{m}^{t-1} - \nabla f(\mathbf{x}^{t-1}) + \nabla f(\mathbf{x}^{t-1}) - \nabla f(\mathbf{x}^t)) + \beta_1(\hat{\mathbf{g}}^t - \nabla f(\mathbf{x}^t)).$$

Taking expectation we have

$$\begin{aligned}\mathbb{E}[\|\mathbf{m}^t - \nabla f(\mathbf{x}^t)\|_2^2] =& \mathbb{E}[\|(1-\beta_1)(\mathbf{m}^{t-1} - \nabla f(\mathbf{x}^{t-1}) + \nabla f(\mathbf{x}^{t-1}) - \nabla f(\mathbf{x}^t)) + \beta_1(\mathbb{E}[\hat{\mathbf{g}}^t] - \nabla f(\mathbf{x}^t))\|_2^2] \\ &+ \beta_1^2 \mathbb{E}[\|\hat{\mathbf{g}}^t - \mathbb{E}[\hat{\mathbf{g}}^t]\|_2^2]. \end{aligned} \tag{14}$$

For the first term, applying Jensen's inequality yields

$$\begin{aligned}&\mathbb{E}[\|(1-\beta_1)(\mathbf{m}^{t-1} - \nabla f(\mathbf{x}^{t-1}) + \nabla f(\mathbf{x}^{t-1}) - \nabla f(\mathbf{x}^t) + \beta_1(\mathbb{E}[\hat{\mathbf{g}}^t] - \nabla f(\mathbf{x}^t))\|_2^2] \\ \leq& (1-\beta_1)\mathbb{E}[\|\mathbf{m}^{t-1} - \nabla f(\mathbf{x}^{t-1}) + \nabla f(\mathbf{x}^{t-1}) - \nabla f(\mathbf{x}^t)\|_2^2] + \beta_1 \mathbb{E}[\|\mathbb{E}[\hat{\mathbf{g}}^t] - \nabla f(\mathbf{x}^t)\|_2^2]. \end{aligned} \tag{15}$$

By Young's inequality, we have

$$\begin{aligned}\mathbb{E}[\|\mathbf{m}^{t-1} - \nabla f(\mathbf{x}^{t-1}) + \nabla f(\mathbf{x}^{t-1}) - \nabla f(\mathbf{x}^t)\|_2^2] \leq& \left(1 + \frac{\delta\beta_1}{2}\right) \mathbb{E}[\|\mathbf{m}^{t-1} - \nabla f(\mathbf{x}^{t-1})\|_2^2] \\ &+ \left(1 + \frac{2}{\delta\beta_1}\right) \mathbb{E}[\|\nabla f(\mathbf{x}^t) - \nabla f(\mathbf{x}^{t-1})\|_2^2]. \end{aligned} \tag{16}$$

For the second term, applying Cauchy's inequality yields

$$\begin{aligned}\mathbb{E}[\|\hat{\mathbf{g}}^t - \mathbb{E}[\hat{\mathbf{g}}^t]\|_2^2] \leq& 3\mathbb{E}\|\hat{\mathbf{g}}^t - \mathbf{g}^t\|_2^2 + 3\mathbb{E}[\|\mathbf{g}^t - \nabla f(\mathbf{x}^t)\|_2^2] + 3\mathbb{E}[\|\nabla f(\mathbf{x}^t) - \mathbb{E}[\hat{\mathbf{g}}^t]\|_2^2] \\ \leq& 3(1-\delta)\mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2] + 3(1-\delta)\mathbb{E}[\|\mathbf{g}^t\|_2^2] + 3\sigma^2, \\ \leq& 6(1-\delta)\mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2] + 3(2-\delta)\sigma^2, \end{aligned} \tag{17}$$

where the inequality uses Assumption 4.3 and 4.4. Applying (15)(16)(17) to (14) and using Assumption 4.2 and 4.4, we obtain B.2. □

Remark. From this proof, it is evident that both inequalities in Assumption 4.4 are necessary. In particular, the second inequality is essential for bounding the variance of $\hat{\mathbf{g}}^t$, which plays a crucial role in the overall convergence analysis.

Now we are ready to prove Theorem 4.5. We first restate the theorem in Theorem B.3.

Theorem B.3. *Under Assumptions 4.1–4.4, if* $\beta_1 \in (0, \delta/(24 - 12\delta))$, $\delta_1 \in (0,1)$ *and* $\eta \leq \min\{1/2L, \sqrt{(\delta\beta_1^2)/(8L^2)}\}$, *TAH-Quant with momentum SGD converges as*

$$\frac{1}{T+1}\sum_{t=0}^{T} \mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2] \leq \frac{8[f(\mathbf{x}^0) - \inf_{\mathbf{x}} f(\mathbf{x})]}{\delta\eta(T+1)} + \frac{8\|\mathbf{m}^0 - \nabla f(\mathbf{x}^0)\|_2^2}{\delta\beta_1(T+1)} + \frac{24\beta_1\sigma^2}{\delta}. \tag{18}$$

Proof. By Lemma B.1, we have

$$f(\mathbf{x}^{t+1}) - f(\mathbf{x}^t) \leq -\left(\frac{1}{2\eta} - \frac{L}{2}\right)\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2 + \frac{\eta}{2}\|\nabla f(\mathbf{x}^t) - \mathbf{m}^t\|_2^2 - \frac{\eta}{2}\|\nabla f(\mathbf{x}^t)\|_2^2. \tag{19}$$

Taking expectation and summing (19) for $t = 0, 1, \cdots, T$ yields

$$\begin{aligned}\inf_{\mathbf{x}} f(\mathbf{x}) - f(\mathbf{x}^0) \leq& \frac{\eta}{2}\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\mathbf{x}^t) - \mathbf{m}^t\|_2^2] - \left(\frac{1}{2\eta} - \frac{L}{2}\right)\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2] \\ &- \frac{\eta}{2}\sum_{t=0}^{T}\mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2]. \end{aligned} \tag{20}$$

summing the inequality in Lemma B.2 for $t = 1, 2, \cdots, T$ we have

$$\begin{aligned}\beta_1\left(1 - \frac{\delta}{2}\right)\sum_{t=0}^{T}\mathbb{E}[\|\mathbf{m}^t - \nabla f(\mathbf{x}^t)\|_2^2] \leq& \|\mathbf{m}^0 - \nabla f(\mathbf{x}^0)\|_2^2 + \frac{2L^2}{\delta\beta_1}\sum_{t=1}^{T}\|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2^2 \\ &+ (1-\delta)(\beta_1 + 6\beta_1^2)\sum_{t=1}^{T}\mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2] + 3T(2-\delta)\beta_1^2\sigma^2. \end{aligned} \tag{21}$$

noting that $\delta \in (0,1)$ we obtain

$$\sum_{t=0}^{T} \mathbb{E}[\|\mathbf{m}^t - \nabla f(\mathbf{x}^t)\|_2^2] \leq \frac{2\|\mathbf{m}^0 - \nabla f(\mathbf{x}^0)\|_2^2}{\beta_1} + \frac{4L^2}{\delta\beta_1^2} \sum_{t=1}^{T} \|\mathbf{x}^t - \mathbf{x}^{t-1}\|_2^2 + \left(1 - \frac{\delta}{2}\right)(1 + 6\beta_1) \sum_{t=1}^{T} \mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2] + 6T\beta_1\sigma^2. \tag{22}$$

Applying 22 to (20) and noting that $\beta_1 \in (0, \delta/(24 - 12\delta))$ implies $(1 - \delta/2)(1 + 6\beta_1) \leq 1 - \delta/4$, we obtain

$$\frac{1}{T+1} \sum_{t=0}^{T} \mathbb{E}[\|\nabla f(\mathbf{x}^t)\|_2^2] \leq \frac{8[f(\mathbf{x}^0) - \inf_{\mathbf{x}} f(\mathbf{x})]}{\delta\eta(T+1)} + \frac{8\|\mathbf{m}^0 - \nabla f(\mathbf{x}^0)\|_2^2}{\delta\beta_1(T+1)} + \frac{24\beta_1\sigma^2}{\delta} - \frac{8}{\delta\eta}\left(\frac{1}{2\eta} - \frac{L}{2} - \frac{2\eta L^2}{\delta\beta_1^2}\right) \sum_{t=0}^{T} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|_2^2. \tag{23}$$

Since $\eta \leq \min\{1/2L, \sqrt{(\delta\beta_1^2)/(8L^2)}\}$ implies $1/(4\eta) \geq L/2$ and $1/(4\eta) \geq (2\eta L^2)/(\delta\beta_1^2)$, (18) is a direct result of (23). □

C Supplementary Pretraining: `LLaMA-8B` on `Proof-Pile`

For completeness we also include a from-scratch pretraining run of `LLaMA-8B` on the `Proof-Pile` mathematical corpus, using the configuration described in the "Pretraining" paragraph of Appendix A ($20k$ iterations, global batch size 131,072 tokens, ∼2.5 B tokens total, peak learning rate 1.5×10^{-4} with cosine decay, weight decay 0.1, PP=4).

We present this larger-model run as a supplementary scale check and use the smaller `LLaMA-3.2-1B` run as the main-text pretraining benchmark. The reason is statistical efficiency rather than model size alone: under Chinchilla-style scaling laws Hoffmann et al. (2022), the `LLaMA-3.2-1B` run receives substantially more tokens per parameter (∼5) than the `LLaMA-8B` run (∼0.3), making it the more informative setting for detecting mature pretraining-quality differences under the available token budget.

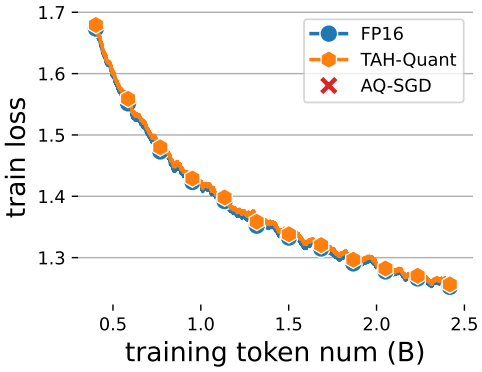
Figure 4 reports the training loss and the held-out validation perplexity for this run. The training-loss curve (Figure 4a) covers the full ∼0.4–2.5 B-token horizon after the initial warmup transient is dropped (matching the slicing convention used in Wang et al. (2022)). The validation-PPL curve (Figure 4b) shows perplexity at 0.5 B-spaced checkpoints over the 0.5–2.5 B-token range. Across the entire horizon, TAH-Quant is essentially indistinguishable from the uncompressed `FP16` baseline: the validation-PPL gap stays within ±0.013 at every measured checkpoint and shows no upward drift, indicating that the per-step quantization error of TAH-Quant does not visibly accumulate even at the 8B-parameter scale.

D Hyper-Parameter Tuning: Sweeping Micro-batch Size and Bandwidth

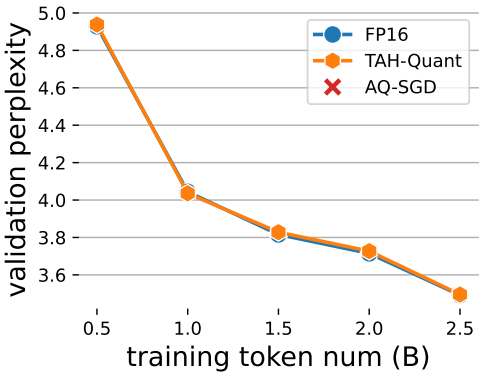
We fix the global batch size to 32 and sweep the micro-batch size (mbs) and inter-stage bandwidth. Tables 5–6 report throughput (tokens/s) and speedup.

Table 5: Throughput and speedup under 500 Mbps (global batch = 32).

tokens / s	w/o TAH-Quant	w/ TAH-Quant	Speedup
mbs = 1	2406	4289	1.78×
mbs = 2	1710	4016	2.35×
mbs = 4	1317	3023	2.30×
mbs = 8	981	2099	2.14×



(a) Training loss vs. tokens.



(b) Validation perplexity vs. tokens.

Figure 4: Pretraining `LLaMA-8B` from scratch on `Proof-Pile` (PP=4, global batch size 32, sequence length 4096). TAH-Quant (orange) tracks the uncompressed `FP16` baseline (blue) on both training loss (a) and held-out validation perplexity (b). `AQ-SGD` is shown in the legend for reference but is infeasible on this single-epoch pretraining workload (its activation cache would require storing per-microbatch forward activations for the entire corpus).

Table 6: Throughput and speedup under 1 Gbps (global batch = 32).

tokens / s	w/o TAH-Quant	w/ TAH-Quant	Speedup
mbs = 1	3449	5177	1.50×
mbs = 2	2576	4655	1.81×
mbs = 4	2003	3475	1.73×
mbs = 8	1485	2436	1.64×

Discussion. TAH-Quant yields larger speedups under lower bandwidth since inter-stage communication dominates (up to 2.35× at 500 Mbps). Speedup increases with micro-batch size and peaks at mbs=2–4, where communicated activation volume per step is larger. At a very large micro-batch size (mbs=8), increased pipeline idle time slightly reduces the speedup.

E Sensitivity of Outlier Threshold τ

We evaluate the robustness of the outlier detection threshold τ in Eq. 3 by varying τ. $\tau = 0$ corresponds to always applying the transform (treating every tile as containing outliers), while $\tau = \infty$ corresponds to never applying it. Table 7 shows that $\tau = 2.0$ yields the most stable and fastest loss decrease.

Loss	Step 100	Step 200	Step 300	Step 400	Step 500
$\tau = 0$ (always transform)	2.97	2.79	2.63	2.60	2.60
$\tau = 2$ (ours)	**2.72**	**2.63**	**2.59**	**2.56**	**2.55**
$\tau = 4$	2.73	2.65	2.61	2.58	2.57
$\tau = \infty$ (never transform)	2.82	2.72	2.68	2.64	2.63

Table 7: Effect of the outlier detection threshold τ on training loss.

F Additional Bit-Allocation Studies

F.1 Varying INT4/INT3 Ratios

Table 8 reports training losses under different INT4/INT3 splitting ratios. The mixed setting lies between the two extremes, and increasing the INT4 fraction yields more stable convergence in this sweep.

Loss	step 50	step 100	step 150	step 200	step 300	step 400	step 500	step 600
100% INT4	2.78	2.70	2.66	2.61	2.56	2.52	2.51	2.49
50% INT4 + 50% INT3	2.82	2.74	2.71	2.66	2.62	2.58	2.57	2.55
100% INT3	2.85	2.77	2.74	2.69	2.65	2.62	2.60	2.58

Table 8: Loss trajectories under different INT4/INT3 ratio settings.

F.2 Extremely Low-bit Configuration with INT2

We additionally evaluate a more aggressive configuration involving INT2. Table 9 shows that the INT2-involved setting exhibits degraded behavior in this experiment, consistent with instability under extremely low-bit quantization.

Loss	step 0	step 100	step 200	step 300	step 400	step 500	step 600
80% INT2 + 20% INT3	3.31	3.22	3.20	3.19	3.17	3.18	3.15

Table 9: Loss trajectories with INT2-involved quantization, where training becomes unstable.

G Backward Gradients under Ultra-low Precision

In the main text, we quantize backward gradients with a higher-bit fixed-point compressor to strike a practical balance between system efficiency and numerical stability. Here we stress-test backward quantization under an ultra-low precision budget (4-bit). As shown in Table 10, the naive 4-bit fixed-point scheme diverges, whereas applying TAH-Quant to backward gradients remains stable and continues to converge.

Loss	step 0	step 100	step 200	step 300	step 400	step 500	step 600
`TAH-Quant-4bit`	2.87	2.71	2.63	2.59	2.56	2.55	2.52
`NAIVE-4bit`	2.87	3.04	3.16	3.21	3.49	4.53	6.04

Table 10: Training loss with 4-bit backward gradient quantization.

Numerical stability in long-horizon training. During long-horizon pretraining, we observed a single instability event where the loss became `NaN` around 1.9B tokens when using an 8-bit fixed-point compressor for backward gradients. Our investigation indicates that the issue is not caused by the forward TAH-Quant design (activation quantization), but by insufficient numerical precision in the backward fixed-point path. Increasing the backward precision from 8-bit to 10-bit and resuming from the checkpoint eliminated the issue and restored stable training. This motivates studying more robust backward quantization under aggressive bit budgets; as shown in Table 10, TAH-Quant remains stable even at 4-bit, whereas naive fixed-point quantization diverges.

Gradient scaling. Gradient magnitudes can be small in practice (e.g., on the order of 10^{-5} in our setting), which makes ultra-low-bit quantization particularly sensitive to rounding and underflow. To improve numerical robustness, we rescale gradients before quantization and invert the scaling after dequantization:

$$\hat{\mathbf{g}} = Q(c\mathbf{g})/c, \tag{24}$$

where $Q(\cdot)$ denotes the chosen quantizer and c is a constant scaling factor.