# QUANTIZATION WITH PURPOSE: LOSS-AWARE BIT ALLOCATION FOR GRADIENT COMPRESSION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Gradient quantization is a critical technique for reducing communication overhead in large-scale distributed training. However, existing methods often employ fixed bit-width quantization or adaptive quantizers optimized with signal-level distortion metrics such as MSE, which poorly correlate with model performance. In this paper, we propose a novel layer-wise bit allocation framework for gradient quantization, formulated under a rate-distortion optimization (RDO) paradigm. Unlike prior approaches, our method introduces a loss-aware distortion metric that directly quantifies the impact of quantization on training loss, enabling task-aligned solution for bit allocation. A key insight of our work is the linear superposition property of cross-layer loss distortion, which we theoretically justify and empirically validate. This property allows us to decouple the original joint optimization problem and efficiently solve it via a Lagrangian optimization algorithm with linear complexity. Extensive experiments across vision and language tasks—using CNNs, ViTs, LSTMs, and Transformers—demonstrate the effectiveness of our approach. Moreover, our method integrates seamlessly with existing gradient compression techniques, yielding consistent performance gains.

## 1 INTRODUCTION

The rapid development of large-scale deep learning models has significantly reshaped the landscape of artificial intelligence. Foundation models such as GPT-4 (Achiam et al., 2023) and large-scale diffusion models like Stable Diffusion (Rombach et al., 2022) have demonstrated impressive generalization capabilities across a broad range of tasks. However, the enormous number of parameters in these models, often reaching hundreds of billions, necessitates distributed training across large GPU clusters. Within this paradigm, the synchronization of gradients or model parameters incurs substantial communication overhead, which has emerged as a primary bottleneck that limits both scalability and training efficiency (Wang et al., 2023; Tang et al., 2021).

To alleviate this bottleneck, gradient compression has become an essential component of modern distributed training systems. Existing approaches can be broadly categorized into three classes: sparsification (Stich et al., 2018; Lin et al., 2017; Wangni et al., 2018), low-rank decomposition (Vogels et al., 2019; Wang et al., 2018; Yu et al., 2018), and quantization. Among these, quantization is particularly appealing due to its simplicity and compatibility with contemporary hardware, making it a central focus of this study. Early studies, such as TernGrad (Wen et al., 2017) and QSGD (Alistarh et al., 2017), employed uniform quantization across fixed ranges, while more recent methods like Natural Compression (NC) (Horvóth et al., 2022) improved fidelity through non-uniform schemes. However, a common limitation of these methods is that their quantization levels are statically defined at the beginning of training and remain fixed throughout the training process, which is often suboptimal in practice due to the dynamic nature of gradient distributions. To address this, adaptive quantization methods have been investigated to dynamically adjust quantization strategies in response to the evolving characteristics of the training process. Prior work has primarily focused on intra-layer adaptation, where quantization parameters, such as clipping range or step size, are adjusted over time. However, these approaches typically overlook inter-layer heterogeneity, treating all layers uniformly despite their differing sensitivities to quantization. Representative methods, including AdaQS (Guo et al., 2020), AQG (Mao et al., 2022), and ALQ (Faghri et al., 2020), introduce various mechanisms for adaptive adjustment of quantization parameters. While these techniques

offer increased flexibility, they typically employ a uniform bit-width across all layers, thereby overlooking the varying sensitivity of gradients in different layers to quantization.

In light of this, mixed-precision quantization approaches have been explored to assign different bit-widths to gradients based on their relative sensitivity to compression. AC-SGD (Yan et al., 2022) adaptively adjusts the quantization bit-width over training iterations with respect to the norm of gradients, communication budget, and the remaining training steps. L-GreCo (Markov et al., 2024) employs dynamic programming to determine the minimum bit-width for each layer under a global quantization error constraint. Different from these two approaches, our work focuses on solving the layer-wise bit allocation problem under a fixed per-iteration communication budget constraint. Moreover, the above two methods rely on signal-level metrics, such as the L2 norm or mean squared error (MSE) of gradients, to estimate sensitivity to compression, which exhibit weak correlation with the actual impact of quantization on model performance.

In this paper, we propose a layer-wise bit allocation framework for adaptive gradient quantization. Specifically, we formulate the bit allocation problem within a rate-distortion optimization (RDO) paradigm, aiming to minimize task-related distortion under a fixed communication budget. Instead of optimizing signal-level quantization errors (typically heuristic metrics such as L2-norm or MSE), we introduce a loss-aware distortion metric that directly quantifies the impact of quantization on the training objective. This enables fine-grained, dynamic bit allocation guided by true optimization sensitivity. To decouple the RDO problem, we investigate and validate a linear superposition property of cross-layer loss distortion through theoretical and empirical analysis. Specifically, we show that the total degradation in training loss resulting from the joint quantization of multiple layers can be well approximated by the sum of individual distortions incurred when each layer is quantized independently. To solve the decoupled RDO problem, we further develop a Lagrangian optimization algorithm that identifies the optimal bit allocation while reducing computational complexity from exponential to linear in the number of layers. In addition, to efficiently perform bit allocation during model training without introduce unnecessary computational overhead, we propose a dynamic reallocation trigger that monitors changes in the gradient distribution and initiates bit assignment when significant shifts are detected.

The primary contributions of this paper are summarized as follows:

- We formulate the bit allocation problem for gradient quantization within a rate-distortion optimization (RDO) framework, which directly optimizes the training loss rather than widely-used signal-level quantization errors, thereby ensuring improved model training performance.
- We theoretically prove and empirically validate a linear superposition property of loss distortion, which enables efficient decomposition of the RDO problem. Based on this property, we propose a Lagrangian optimization method that achieves optimal bit allocation with linear computational complexity.
- We evaluate our method across a diverse set of model architectures, including CNNs, Vision Transformers (ViTs), Transformers, and LSTMs, on both image classification and language modeling tasks. Experimental results demonstrate that our approach consistently outperforms state-of-the-art static and adaptive quantization baselines.

## 2 METHODOLOGY

### 2.1 PROBLEM FORMULATION: AN RDO PERSPECTIVE

Gradient quantization aims to transform a full-precision gradient $g$ into a lower-bit representation $\tilde{g}$ to reduce communication overhead during distributed training. In layer-wise bit allocation problem, the key challenge is to determine how to distribute a limited communication budget $B_{\text{total}}$ (measured in bits) across the layers of a neural network. Consider a model with $L$ layers, where $\mathbf{g}^{(l)}$ denotes the gradient tensor of layer $l$, and $N_l$ is its number of parameters. The objective is to assign a bit-width $b_l \in \mathcal{B}_{\text{options}}$ to each layer such that the total quantization-induced distortion is minimized, subject to the overall budget constraint.

We formulate this as a classic RDO problem, where the "rate" $R$ refers to the total number of bits used for communication, and the "distortion" $D$ quantifies the performance degradation due to

quantization. The optimization objective is given by:

$$\min_{\{b_l\}\in\mathcal{B}_{\text{options}}} D_{\text{total}} = D(\mathbf{g}, Q(\mathbf{g}, \{b_l\})), \quad \text{s.t.} \quad R_{\text{total}} = \sum_{l=1}^{L} b_l \cdot N_l \leq B_{\text{total}}, \tag{1}$$

where $Q(\mathbf{g}, \{b_l\})$ denotes the quantization function applying $b_l$ bits to $\mathbf{g}^{(l)}$. To secure final training performance, the optimization objective (i.e., the distortion metric) should be carefully designed.

#### 2.1.1 The Loss-Aware Distortion Metric

Traditional distortion metrics, such as MSE, quantify the geometric deviation between the original and quantized gradients. While computationally efficient, these metrics often exhibit weak alignment with the ultimate objective of deep learning optimization: minimizing the training loss. Therefore, to better capture the true impact of quantization on model performance, we propose a Loss-Aware Distortion (LAD) metric, which directly measures the change in the training objective induced by quantizing the gradient of a specific layer.

Let $\mathbf{W}_t$ denote the model weights at iteration $t$ and $\eta$ denote the learning rate. Consider a set of $K$ data batches $\mathcal{S} = \{d_1, d_2, \ldots, d_K\}$ sampled from the training data, we first define the per-batch loss difference $\Delta\mathcal{L}_l(b_l; d)$ for layer $l$ with bit-width $b_l$ as:

$$\Delta\mathcal{L}_l(b_l; d) = \left| \mathcal{L}(\mathbf{W}_t - \eta \cdot \mathbf{g}^{(l)}_{\text{mixed},t}; d) - \mathcal{L}(\mathbf{W}_t - \eta \cdot \mathbf{g}_t; d) \right|, \tag{2}$$

where $\mathbf{g}_t = (\mathbf{g}_t^{(1)}, \ldots, \mathbf{g}_t^{(L)})^T$ is the gradient vector of all $L$ layers and $\mathbf{g}^{(l)}_{\text{mixed,t}} = (\mathbf{g}_t^{(1)}, \ldots, \tilde{\mathbf{g}}_t^{(l)}, \ldots, \mathbf{g}_t^{(L)})^T$ is a mixed gradient vector where only the gradient for layer $l$ (i.e. $\mathbf{g}_t^{(l)}$) is replaced by its quantized version $\tilde{\mathbf{g}}_t^{(l)}$. We then define the Loss-Aware Distortion $D_l(b_l)$ as the expected loss difference over the batch set $\mathcal{S}$:

$$D_l(b_l) = \mathbb{E}_{d\in\mathcal{S}} \left[ \Delta\mathcal{L}_l(b_l; d) \right]. \tag{3}$$

This expectation over multiple batches mitigates the effects of single-batch noise and yields a more stable and reliable signal for guiding bit allocation.

By quantifying distortion in terms of its impact on the training loss rather than gradient geometry, the proposed metric offers a task-aligned and dynamic measure of gradient sensitivity, better supporting optimization-aware quantization decisions. Our ablation study in Appendix B.1 confirms this, showing that using our LAD metric consistently yields higher accuracy than using a traditional MSE metric.

### 2.2 The Linear Superposition Property

A fundamental challenge in solving the RDO problem (Equation 1) lies in the combinatorial complexity of the joint optimization. Specifically, assigning bit-widths to $L$ layers from a discrete set $\mathcal{B}$ results in a search space of size $|\mathcal{B}|^L$, which grows exponentially with the number of layers. In deep neural networks, where $L$ may reach hundreds or more, this combinatorial explosion renders brute-force approaches (such as heuristic algorithms and greedy search methods) computationally infeasible. Moreover, the total distortion $D_{\text{total}}$ exhibits highly non-linear and coupled dependencies on the joint bit allocation $b_l$, as it reflects complex interactions among quantization errors across layers within the non-convex loss landscape of deep networks. It makes the joint bit allocation problem particularly challenging.

To address this challenge, we introduce and validate a linear superposition property, which asserts that the total change in training loss resulting from the simultaneous quantization of multiple layers can be closely approximated by the sum of the individual distortions incurred when each layer is quantized independently, i.e.:

$$D_{\text{joint}}(\{b_l\}) \approx \sum_{l=1}^{L} D_l(b_l). \tag{4}$$

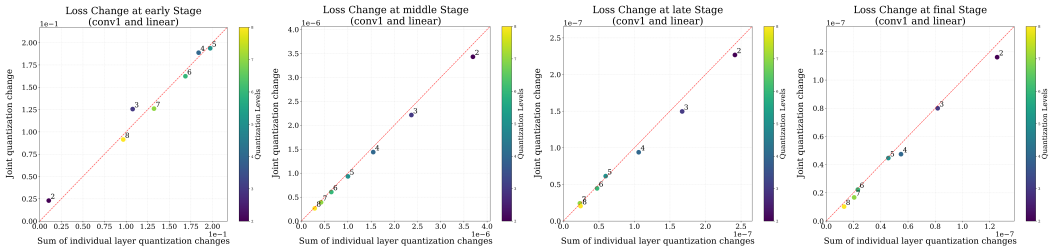


Figure 1: Empirical validation of the Linear Superposition property on ResNet-18. The joint loss change from quantizing two layers simultaneously (y-axis) versus the sum of their individual loss changes (x-axis). Each point represents a different quantization level (2-8 bits), evaluated at various stages of training.

### 2.2.1 Theoretical Proof

We begin by providing a theoretical justification for the linear superposition property based on a first-order Taylor expansion of the loss function. For a given layer $l$, we define the quantization error as $\mathbf{e}^{(l)} = (0, \ldots, 0, \mathbf{g}^{(l)} - \tilde{\mathbf{g}}^{(l)}, 0, \ldots, 0)^T$. When a subset $\mathcal{K}$ of layers is quantized, the resulting perturbation to the weights can be expressed as:

$$\delta\mathbf{W}^{(\mathcal{K})} = (\mathbf{W}_t - \eta\mathbf{g}^{(\mathcal{K})}_{\text{mixed}}) - (\mathbf{W}_t - \eta\mathbf{g}) = \eta \sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}. \tag{5}$$

Let $\mathbf{W}_{\text{orig}} = \mathbf{W}_t - \eta\mathbf{g}$ denote the pristine weight after a full-precision update. According to Equation 2, the distortion induced by quantizing layers in $\mathcal{K}$ is then:

$$D_{(\mathcal{K})} = \mathcal{L}(\mathbf{W}_{\text{orig}} + \delta\mathbf{W}^{(\mathcal{K})}) - \mathcal{L}(\mathbf{W}_{\text{orig}}). \tag{6}$$

Applying a first-order Taylor expansion to $\mathcal{L}(\mathbf{W}_{\text{orig}} + \delta\mathbf{W}^{(\mathcal{K})})$ around $\mathbf{W}_{\text{orig}}$ yields:

$$D_{(\mathcal{K})} \approx \nabla\mathcal{L}(\mathbf{W}_{\text{orig}})^T \delta\mathbf{W}^{(\mathcal{K})} = \mathbf{g}_{t+1}^T \left(\eta \sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}\right), \tag{7}$$

where $\mathbf{g}_{t+1} = \nabla\mathcal{L}(\mathbf{W}_{\text{orig}})$ is the gradient evaluated in updated full-precision weights. Given the linearity of the dot product, we can distribute it across the sum:

$$D_{(\mathcal{K})} \approx \sum_{l\in\mathcal{K}} \left(\eta \cdot \mathbf{g}_{t+1}^T \mathbf{e}^{(l)}\right). \tag{8}$$

The term $\eta \cdot \mathbf{g}_{t+1}^T \mathbf{e}^{(l)}$ corresponds to the first-order approximation of the individual distortion $D_l$, i.e., the distortion caused by quantizing only layer $l$. Therefore, we obtain the approximate linear superposition:

$$D_{(\mathcal{K})} \approx \sum_{l\in\mathcal{K}} D_l. \tag{9}$$

The approximation error stems from the higher-order terms in the Taylor expansion. The leading second-order term is given by $(\Delta\mathbf{W})^T\mathbf{H}(\Delta\mathbf{W})$, where $\mathbf{H}$ denotes the Hessian matrix. This term includes interaction effects between different error vectors and is quadratic in both the learning rate $\eta$ and the magnitude of the quantization error, making its contribution negligible in practice.

In addition, the above proof is based on the stochastic gradient descent (SGD). We also demonstrate that the linear superposition property remains valid for the AdamW (Loshchilov & Hutter, 2017) optimizer. The detailed proof is provided in Appendix A.

### 2.2.2 Empirical Validation

We further provide empirical evidence demonstrating that the higher-order terms in the Taylor expansion are indeed negligible. Figure 1 presents a comparison between the joint distortion ($D_{i,j}$) and the sum of individual distortions ($D_i + D_j$) for two representative layers, evaluated across bit-widths ranging from 2 to 8. The data points align closely with the $y = x$ diagonal, indicating strong agreement and validating the linear superposition property at various stages of training.

**Algorithm 1** Lagrangian Search for Layer-wise Bit Allocation

**Require:** Gradients $\{\mathbf{g}^{(l)}\}_{l=1}^{L}$, bit options $\mathcal{B}$, average quantization bit-width constraint $B_c$, data batches $\mathcal{D}$
**Ensure:** Bit allocation $\{b_l\}_{l=1}^{L}$

$B_{total} \leftarrow B_c \cdot \sum_{l=1}^{L} N_l$
**Compute R-D curves:**
**for** each layer $l \in \{1, \ldots, L\}$ **do**
  **for** each bit option $b \in \mathcal{B}$ **do**
    $d_{l,b} \leftarrow \text{LossDiff}(g_l, b, \mathcal{D})$ {Distortion per layer}
    $r_{l,b} \leftarrow b \cdot N_l$ {Rate per layer}
  **end for**
**end for**
**Estimate** $\lambda$ **range:** $\lambda_{low}, \lambda_{high}$ from R-D slopes
$\mathcal{A}^* \leftarrow \emptyset$
**while** $\lambda_{high} - \lambda_{low} > \epsilon$ **do**
  $\lambda_{mid} \leftarrow (\lambda_{low} + \lambda_{high})/2$
  $B_{used} \leftarrow 0$
  **for** each layer $l \in \{1, \ldots, L\}$ **do**
    $b_l^* \leftarrow \arg\min_{b \in \mathcal{B}} \{d_{l,b} + \lambda_{mid} \cdot r_{l,b}\}$
    $B_{used} \leftarrow B_{used} + b_l^* \cdot N_l$
  **end for**
  **if** $B_{used} > B_{total}$ **then**
    $\lambda_{low} \leftarrow \lambda_{mid}$
  **else**
    $\lambda_{high} \leftarrow \lambda_{mid}$
    $\mathcal{A}^* \leftarrow \{b_l^*\}_{l=1}^{L}$
  **end if**
**end while**
**return** $\mathcal{A}^*$

#### 2.2.3 Problem Decoupling.

Validating this linear superposition property enables the decoupling of the intractable joint optimization problem. By substituting the total distortion with the sum of individual LAD metrics (Equation 3), our objective function becomes separable:

$$\min_{\{b_l\} \in \mathcal{B}_{\text{options}}} \sum_{l=1}^{L} D_l(b_l), \quad \text{s.t.} \sum_{l=1}^{L} b_l \cdot N_l \leq B_{\text{total}}. \tag{10}$$

This transformation reduces the problem's complexity from exponential to linear with respect to the number of layers, enabling practical applicability even in deep neural networks. It serves as the foundation for our proposed bit allocation algorithm.

### 2.3 Solve Bit Allocation via Lagrangian Relaxation

To solve the decoupled constrained optimization problem efficiently, we adopt **Lagrangian relaxation** approach. By introducing a Lagrange multiplier $\lambda \geq 0$, we can convert the constrained problem into an unconstrained objective:

$$J(\lambda) = \sum_{l=1}^{L} \left[ D_l(b_l) + \lambda \cdot (b_l \cdot N_l) \right]. \tag{11}$$

The multiplier $\lambda$ can be interpreted as the trade-off factor between distortion $D$ and rate $R$ (i.e., bits). A greater $\lambda$ places a higher penalty on bit usage, favoring lower-bit allocations, while a smaller $\lambda$ prioritizes minimizing distortion. For a fixed $\lambda$, the optimal bit-width for each layer $b_l^*$ can be obtained independently by minimizing its layer-wise Lagrangian cost:

$$b_l^*(\lambda) = \arg\min_{b_l \in \mathcal{B}_{\text{options}}} \left[ D_l(b_l) + \lambda \cdot (b_l \cdot N_l) \right], \tag{12}$$

Table 1: Accuracy recovery and compression ratios for different compression methods with uniform and adaptive schemes on image classification tasks at 2-bit quantization. 'Acc.' refers to Top-1 accuracy and 'Comp. Ratio' refers to compression ratio.

| | ResNet-18 on CIFAR-10 | | ViT-small on ImageNet | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | 0–50 epoch | | 150–200 epoch | | 250–300 epoch | |
| Method | Acc.(%) | Comp. Ratio | Acc.(%) | Comp. Ratio | Acc.(%) | Comp. Ratio | Acc.(%) | Comp. Ratio |
| FP32 | 88.24 | - | 60.11 | - | 63.24 | - | 65.36 | - |
| Uniform | 77.33 | 16.00× | 47.88 | 16.00× | 62.84 | 16.00× | 65.24 | 16.00× |
| + Greedy | 88.09 | 16.35× | 51.13 | 16.07× | 64.03 | 16.07× | 65.30 | 16.08× |
| + Ours | **88.39** | **16.46**× | **52.66** | **16.28**× | **64.15** | **16.26**× | **65.33** | **16.24**× |

Table 2: Accuracy recovery and compression ratios for different compression methods with uniform and adaptive schemes on language modeling tasks at 3-bit quantization.

| Method | LSTM on PTB | | Transformer on WikiText-103 | |
|---|---|---|---|---|
| | Perplexity | Compression Ratio | Perplexity | Compression Ratio |
| FP32 | 82.32 | - | 77.18 | - |
| Uniform | 639.94 | 10.67× | 133.64 | 10.67× |
| + Greedy | 561.79 | 11.27× | 128.20 | 11.51× |
| + Ours | **388.08** | **12.37**× | **118.27** | **12.77**× |

since the solution space for each layer is small and discrete (e.g., 1 to 8 bits), this step can be performed efficiently.

The remaining challenge is to identify the optimal value of $\lambda^*$ that minimizes total distortion while satisfying the overall communication budget constraint. Given that the total rate $R_{\text{total}}(\lambda)$ is a monotonically non-increasing function of the Lagrange multiplier $\lambda$, we can efficiently identify the optimal value $\lambda^*$ using a bisection search. This process iteratively refines a bounded interval $[\lambda_{\text{low}}, \lambda_{\text{high}}]$, converging to a value that yields the total usage of bits as close as possible to, but not exceeding, the target budget $B_{\text{total}}$. This entire procedure allows us to find an optimal, per-layer bit allocation in a computationally efficient manner. The complete algorithm is described in Algorithm 1.

### 2.4 Perform Bit Allocation with A Dynamic Reallocation Trigger Mechanism

The previous sections detail how to optimally allocate bit-widths across layers under a fixed communication budget. However, computing the LAD metric (Equation 3) for all layers and candidate bit-widths incurs substantial computational overhead, primarily due to the need for multiple forward passes. Moreover, prior work has shown that the optimal bit allocation tends to remain relatively stable across consecutive training steps (Markov et al., 2024), making full reallocation at every iteration both computationally expensive and largely unnecessary.

This motivates the need for a mechanism to determine when bit reallocation should be performed. To this end, we propose a lightweight yet effective dynamic reallocation trigger that adaptively decides when to recompute the allocation based on changes in gradient statistics. Specifically, we monitor the distribution of L2-norms of the gradients across all $L$ layers, which serves as a compact and computationally efficient proxy for detecting shifts in the overall gradient landscape. Let $\mathbf{v}_t \in \mathbb{R}^L$ be the gradient norm vector at iteration $t$, where the $l$-th element is the L2-norm of the gradient of layer $l$:

$$\mathbf{v}_t = \left[ \|\mathbf{g}_t^{(1)}\|_2, \|\mathbf{g}_t^{(2)}\|_2, \ldots, \|\mathbf{g}_t^{(L)}\|_2 \right]^T. \tag{13}$$

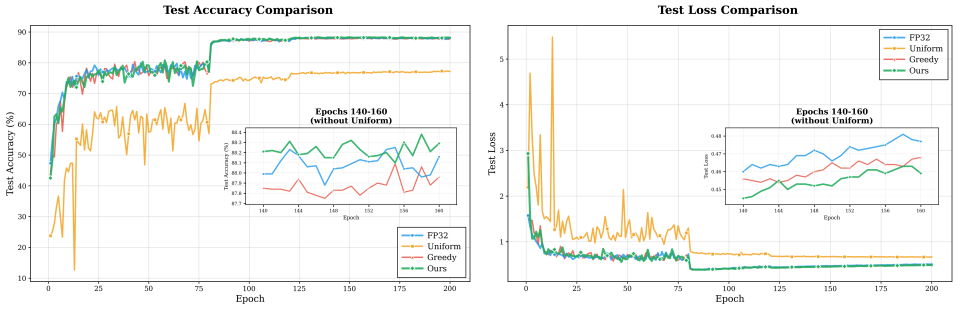


Figure 2: Learning curves of full-precision and different quantization methods for ResNet-18 on CIFAR-10 under 2-bit budget. The zommed-in views highlight test curves during Epochs 140–160.

The trigger mechanism works as follows:

1. **Anchoring:** After a bit allocation is performed at step $t_{\text{alloc}}$, we take the normalized gradient norm vector as an "anchor" vector $\mathbf{v}_{\text{anchor}}$. Normalization ensures that subsequent comparisons reflect changes in the distribution shape rather than overall magnitude:
$$\mathbf{v}_{\text{anchor}} = \frac{\mathbf{v}_{t_{\text{alloc}}}}{\|\mathbf{v}_{t_{\text{alloc}}}\|_2}. \tag{14}$$
2. **Monitoring:** At each subsequent training step $t > t_{\text{alloc}}$, we compute the current normalized gradient norm vector, $\mathbf{v}_{\text{current}} = \mathbf{v}_t / \|\mathbf{v}_t\|_2$.
3. **Similarity Check:** We then measure the change between the current and the anchor vector using **Cosine Similarity**:
$$\text{Similarity}(t) = \frac{\mathbf{v}_{\text{current}} \cdot \mathbf{v}_{\text{anchor}}}{\|\mathbf{v}_{\text{current}}\|_2 \|\mathbf{v}_{\text{anchor}}\|_2} \tag{15}$$
The denominator is unity since both vectors are L2-normalized.
4. **Triggering Condition:** A reallocation is triggered at step $t$ when both of the following two conditions are met: (a) the similarity between the current and anchor gradient norm vectors falls below a predefined threshold $\tau$ (e.g., $\tau = 0.92$), indicating a significant shift in the gradient distribution; and (b) a minimum number of iterations, $k_{\min}$, has elapsed since the last allocation at $t_{\text{alloc}}$ (i.e., $t - t_{\text{alloc}} \geq k_{\min}$).

When a trigger condition is met, the full adaptive bit allocation algorithm is executed, and the current gradient norm vector $\mathbf{v}_{\text{current}}$ is stored as the new anchor vector $\mathbf{v}_{\text{anchor}}$ for subsequent iterations. This dynamic trigger mechanism ensures that computational resources for reallocation are used only when necessary—that is, when the model's learning dynamics exhibit meaningful change. As a result, our approach remains both highly responsive and computationally efficient. Our ablation study in Appendix B.2 validates this, demonstrating that our dynamic trigger achieves superior accuracy with significantly lower overhead compared to fixed-interval strategies.

# 3 Experiments

## 3.1 Experimental Setup

**Datasets and models** We evaluate the effectiveness of our proposed method on two representative machine learning tasks: image classification and language modeling. For image classification, we train ResNet-18 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009) using momentum SGD with momentum 0.9, and ViT-Small (Dosovitskiy et al., 2020) on ImageNet (Deng et al., 2009) using AdamW with gradient clipping and weight decay. For language modeling, we employ a two-layer LSTM (Press & Wolf, 2016) on Penn Treebank (PTB) (Marcinkiewicz, 1994) trained with vanilla SGD and gradient clipping, and a four-layer Transformer (Vaswani et al., 2017) on WikiText-103 (Merity et al., 2016) using AdamW (Loshchilov & Hutter, 2017) with gradient clipping and weight

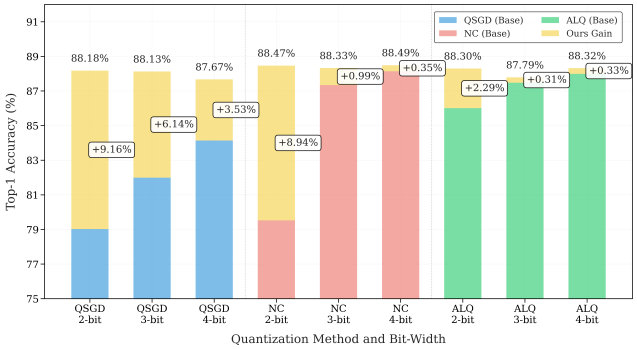


Figure 3: Performance boost from our bit allocation framework across various base quantizers on ResNet-18 using from 2 to 4 bits.

decay. This selection encompasses a broad range of model architectures, including convolutional and recurrent networks as well as modern attention-based designs.

For ViT-Small on ImageNet, we divide the full training process into three representative 50-epoch intervals: 0–50 (early stage), 150–200 (middle stage), and 250–300 (late stage). This setup enables us to evaluate each method's behavior across distinct training dynamics—from rapid parameter updates in early training to stable convergence in later stages—while significantly reducing computational requirement. The middle and late stages are initialized from FP32 checkpoints at epoch 150 and 250, respectively, and finetuned for 50 epochs using quantized gradient for training. This staged evaluation strategy ensures both fairness and efficiency, avoiding the need for full retraining under each configuration.

**Baselines** Given that the optimization objectives and constraint conditions in prior studies (Markov et al., 2024; Yan et al., 2022) on bit allocation for gradient compression differ from ours, as elaborated in the Introduction, we select the following three baselines for comparison: (1) ***'FP32'*** refers to full-precision training and serves as the benchmark for evaluating training accuracy; (2) ***'Uniform'*** applies a fixed bit-width uniformly across all gradient layers; (3) ***'Greedy'*** represents a heuristic-based adaptive bit allocation method. Specifically, the greedy strategy utilizes the same distortion metric as our method but differs in its approach by iteratively allocating bits to the layer with the highest quantization error until the communication budget is fully utilized, without accounting for global optimality. For a fair comparison, all methods are evaluated under the same average bit budget and are implemented using **uniform scalar quantization**.

**Infrastructure** All experiments are implemented in PyTorch 2.3.0 with CUDA 12.1 and cuDNN 8.9.0.2. We use `torchvision` 0.18.0 and `torchtext` 0.18.0 for data processing of image classification and language modeling, respectively. Model training is performed on NVIDIA RTX 4090 GPUs and AMD EPYC 7402 24-core CPUs at 2.8GHz.

### 3.2 Evaluation on Different Models and Datasets

We first conduct experiments on a variety of deep learning models and datasets to assess the effectiveness of our bit allocation framework. Table 1 and Table 2 present the comparison results between our method and baseline approaches on the image classification and language modeling tasks, respectively. The results demonstrate that, under the given bit constraint, our proposed layer-wise bit allocation method consistently enhances final model training performance, as measured by accuracy and perplexity. For instance, on ResNet-18, our method achieves 88.39% accuracy, representing improvements of 11.01% and 0.30% over the 'Uniform' and 'Greedy' baselines, respectively. In the case of ViT-small on ImageNet, our approach outperforms the baselines at every training stage, achieving higher accuracy even at more aggressive compression rates, particularly during the initial 50 epochs. For language modeling with LSTM on PTB, our method attains a perplexity of 388.08, reducing near 40% and 31% perplexity compared to 'Uniform' and 'Greedy' methods, respectively. The advantages of our approach are also evident in Transformer models, where, under a 3-bit quan-

tization setting, it achieves a perplexity of 118.27 at a compression rate of 12.77×. These results validate that our bit allocation framework consistently identifies more effective bit allocation strategies than both uniform and heuristic-based adaptive methods across diverse domains and training stages.

### 3.3 Comparison on Convergence Speed and Stability

To further investigate the impact of different quantization strategies on training trajectories under constrained communication budgets, we visualize the test loss and accuracy curves. As shown in Figure 2, our method consistently delivers competitive performance throughout the training process, demonstrating clear advantages in both convergence speed and final accuracy. During the early training phase, the 'Uniform' baseline exhibits significant instability and underfitting, as reflected by pronounced fluctuations and elevated test loss. This instability arises from its inability to accommodate layer-wise sensitivity variations under stringent bit constraints. In contrast, both 'Greedy' and our method substantially enhance training stability through adaptive bit allocation. Zoomed-in views of the training curves during the late stage (Epochs 140–160) reveal that while all methods achieve relative stability, our approach attains the highest accuracy and lowest loss with minimal fluctuation, outperforming both 'FP32' and 'Greedy'. These results underscore the efficacy of our bit allocation framework in dynamically adapting to the evolving gradient landscape and effectively prioritizing sensitive layers.

### 3.4 Enhanced Performance when Integrated with Existing Quantizers

To demonstrate the versatility of our bit allocation framework, we also apply our adaptive bit allocation strategy to several representative base quantizers, including QSGD (Alistarh et al., 2017), NC (Horvóth et al., 2022), and ALQ (Faghri et al., 2020). These methods span a diverse range of quantization paradigms, from fixed uniform schemes (QSGD) to non-uniform quantization (NC) and adaptive techniques (ALQ). As shown in Figure 3, we compare the accuracy of each quantizer with and without the integration of our bit allocation framework under an identical average bit budget. The results demonstrate that our framework consistently achieves significant performance gains across all quantizers, particularly under aggressive low-bit settings. For instance, applying our method to QSGD with a 2-bit budget elevates accuracy from 79.02% to 88.18%, effectively recovering 9.16% in performance and mitigating the adverse effects of severe quantization. Similarly, NC benefits from an accuracy increase from 82.93% to 88.11%, demonstrating that even advanced non-uniform schemes can be further enhanced by our strategy. Even for a strong baseline like ALQ, which already adapts its quantization range, our method further improves its performance from 86.01% to 88.30% at 2-bit setting. By integrating seamlessly with a wide range of existing quantization methods, our approach enables significant improvements in both gradient compression efficiency and final model performance.

## 4 Conclusion

In this paper, we proposed an efficient layer-wise bit allocation framework for gradient quantization based on a principled RDO formulation. Departing from prior works that naively optimize for signal-level quantization errors, we introduced a loss-aware distortion measure that captures the sensitivity of training loss to quantization, enabling task-aligned optimization. A key contribution of our work is the discovery and validation—both theoretical and empirical—of the linear superposition property of loss distortion. This property allows us to decompose the otherwise intractable joint bit allocation problem into a series of decoupled, per-layer subproblems. Building on this insight, we developed a Lagrangian-based optimization algorithm that finds the globally optimal bit allocation with linear computational complexity in the number of layers. Extensive experiments across a wide range of model architectures and learning tasks demonstrate the superiority and generality of our approach. Our method consistently outperforms both static and adaptive baselines, and can be seamlessly integrated with various quantization schemes to further enhance model performance under constrained communication budgets. Beyond empirical gains, this work offers theoretical insights into the structure of optimization-aware compression strategies, contributing toward scalable and intelligent gradient quantization for large-scale distributed deep learning.

## 5 Ethics Statement

This work adheres to the ICLR Code of Ethics. In this study, no human subjects or animal experimentation was involved. All datasets used were sourced in compliance with relevant usage guidelines, ensuring no violation of privacy. We have taken care to avoid any biases or discriminatory outcomes in our research process. No personally identifiable information was used, and no experiments were conducted that could raise privacy or security concerns. We are committed to maintaining transparency and integrity throughout the research process.

## 6 Reproducibility Statement

All source code required for conducting and analyzing the experiments **will be made publicly available upon publication of the paper** with a license that allows free usage for research purposes. The experimental setup, including model configurations and hardware details, is described in detail in the main paper (Section 3.1). Additionally, all datasets used in the paper are publicly available, ensuring consistent and reproducible evaluation results.

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. IEEE, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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*, 33:3174–3185, 2020.

Jinrong Guo, Wantao Liu, Wang Wang, Jizhong Han, Ruixuan Li, Yijun Lu, and Songlin Hu. Accelerating distributed deep learning by adaptive gradient quantization. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1603–1607. IEEE, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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.

Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, 273:31, 1994.

Ilia Markov, Kaveh Alimohammadi, Elias Frantar, and Dan Alistarh. L-greco: Layerwise-adaptive gradient compression for efficient data-parallel deep learning. *Proceedings of Machine Learning and Systems*, 6:312–324, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in neural information processing systems*, 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*, pp. 10118–10129. PMLR, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

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*, 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.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 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*, 30, 2017.

Guangfeng Yan, Tan Li, Shao-Lun Huang, Tian Lan, and Linqi Song. Ac-sgd: Adaptively compressed sgd for communication-efficient distributed learning. *IEEE Journal on Selected Areas in Communications*, 40(9):2678–2693, 2022.

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*, 31, 2018.

# Appendix

## A Proof of The Linear Superposition Property on AdamW

In our main paper, we provided a theoretical justification for the linear superposition property of distortion based on a first-order Taylor expansion, primarily framed around the SGD optimizer for clarity. Here, we extend this analysis to the AdamW optimizer (Loshchilov & Hutter, 2017), which is commonly used in practice. This justifies the application of our rate-distortion optimization framework to modern adaptive optimizers. The notation is kept consistent with the analysis for SGD.

### A.1 Preliminaries: The AdamW Update Rule

At timestep $t$, the AdamW optimizer updates the weights $\mathbf{W}_t$ to $\mathbf{W}_{t+1}$ based on the gradient $\mathbf{g}_t = \nabla\mathcal{L}(\mathbf{W}_t)$. The complete update rule is as follows:

$$\mathbf{m}_t = \beta_1\mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t, \tag{16}$$

$$\mathbf{v}_t = \beta_2\mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2, \tag{17}$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}, \tag{18}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t}, \tag{19}$$

$$\Delta\mathbf{W}_t = \eta\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}+\epsilon}, \tag{20}$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \Delta\mathbf{W}_t - \eta\lambda\mathbf{W}_t, \tag{21}$$

where $\mathbf{m}_t$ and $\mathbf{v}_t$ are the first and second moment estimates, $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ are their bias-corrected counterparts, and all operations involving vectors are element-wise. For this analysis, we focus on the perturbation to the main update term $\Delta\mathbf{W}_t$, as the decoupled weight decay term is independent of the gradient quantization.

### A.2 Defining Weight Perturbation due to Quantization

Our goal is to analyze the relationship between the gradient quantization error and the resulting perturbation in the weight space. Let $\Delta\mathbf{W}_{\text{orig},t}$ denote the weight update computed using the full-precision gradient $\mathbf{g}_t$ and $\Delta\mathbf{W}^{(\mathcal{K})}_{\text{mixed},t}$ denote the update computed using a mixed gradient where all layers in a set $\mathcal{K}$ are quantized. The weight perturbation, $\delta\mathbf{W}_t^{(\mathcal{K})}$, is the difference between these two updates:

$$\delta\mathbf{W}_t^{(\mathcal{K})} \triangleq \Delta\mathbf{W}^{(\mathcal{K})}_{\text{mixed},t} - \Delta\mathbf{W}_{\text{orig},t}. \tag{22}$$

The key to establishing the property is to determine if $\delta\mathbf{W}_t^{(\mathcal{K})} \approx \sum_{l\in\mathcal{K}}\delta\mathbf{W}_t^{(l)}$, where $\delta\mathbf{W}_t^{(l)}$ is the perturbation from quantizing only layer $l$.

### A.3 Additivity of Perturbations in Moment Estimates

First, we analyze how quantization errors propagate to the moment estimates.

**First Moment Perturbation** The perturbed first moment estimate, $\mathbf{m}^{(\mathcal{K})}_{\text{mixed},t}$, is:

$$\mathbf{m}^{(\mathcal{K})}_{\text{mixed},t} = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}^{(\mathcal{K})}_{\text{mixed},t} \tag{23}$$

$$= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\left(\mathbf{g}_t - \sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}_t\right) \tag{24}$$

$$= (\beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t) - (1-\beta_1)\sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}_t \tag{25}$$

$$= \mathbf{m}_{\text{orig},t} - (1-\beta_1)\sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}_t. \tag{26}$$

The total perturbation to the first moment is thus $\Delta\mathbf{m}^{(\mathcal{K})}_t = \mathbf{m}^{(\mathcal{K})}_{\text{mixed},t} - \mathbf{m}_{\text{orig},t} = -(1-\beta_1)\sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}_t$. Since the single-layer perturbation is $\Delta\mathbf{m}^{(l)}_t = -(1-\beta_1)\mathbf{e}^{(l)}_t$, we have:

$$\Delta\mathbf{m}^{(\mathcal{K})}_t = \sum_{l\in\mathcal{K}} \Delta\mathbf{m}^{(l)}_t. \tag{27}$$

**Second Moment Perturbation** The perturbed second moment estimate, $\mathbf{v}^{(\mathcal{K})}_{\text{mixed},t}$, is:

$$\mathbf{v}^{(\mathcal{K})}_{\text{mixed},t} = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)(\mathbf{g}^{(\mathcal{K})}_{\text{mixed},t})^2. \tag{28}$$

The total perturbation is $\Delta\mathbf{v}^{(\mathcal{K})}_t = \mathbf{v}^{(\mathcal{K})}_{\text{mixed},t} - \mathbf{v}_{\text{orig},t} = (1-\beta_2)[(\mathbf{g}^{(\mathcal{K})}_{\text{mixed},t})^2 - (\mathbf{g}_t)^2]$. We expand the squared term:

$$(\mathbf{g}^{(\mathcal{K})}_{\text{mixed},t})^2 = \left(\mathbf{g}_t - \sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}_t\right)^2 = (\mathbf{g}_t)^2 - 2\mathbf{g}_t\left(\sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}_t\right) + \left(\sum_{l\in\mathcal{K}} \mathbf{e}^{(l)}_t\right)^2. \tag{29}$$

Due to the zero-padding, the error vectors for different layers are orthogonal, i.e., $\mathbf{e}_t(i)\cdot\mathbf{e}_t(j) = 0$ for $i \neq j$. This implies that the square of the sum is the sum of the squares. The perturbation becomes:

$$\Delta\mathbf{v}^{(\mathcal{K})}_t = (1-\beta_2)\left[-\sum_{l\in\mathcal{K}}(2\mathbf{g}^{(l)}_t\mathbf{e}^{(l)}_t) + \sum_{l\in\mathcal{K}}(\mathbf{e}^{(l)}_t)^2\right] \tag{30}$$

$$= \sum_{l\in\mathcal{K}}(1-\beta_2)\left[-2(\mathbf{g}^{(l)}_t\mathbf{e}^{(l)}_t) + (\mathbf{e}^{(l)}_t)^2\right]. \tag{31}$$

The term inside the summation is precisely the single-layer second-moment perturbation, $\Delta\mathbf{v}^{(l)}_t$. Therefore, the additivity is also exact for the second moment:

$$\Delta\mathbf{v}^{(\mathcal{K})}_t = \sum_{l\in\mathcal{K}} \Delta\mathbf{v}^{(l)}_t. \tag{32}$$

The bias correction terms (Equation 18, 19) are scalar multiplications and do not affect the additivity of these perturbations.

### A.4 From Moment Perturbations to Weight Perturbation

The weight update $\Delta\mathbf{W}_t$ is a non-linear function $f(\mathbf{m}_t, \mathbf{v}_t)$. To analyze the weight perturbation $\delta\mathbf{W}_t$, we apply a first-order multi-variate Taylor expansion to $f(\cdot,\cdot)$ around the full-precision point $(\mathbf{m}_{\text{orig},t}, \mathbf{v}_{\text{orig},t})$. For simplicity, we absorb the bias correction scalars into $f$:

$$\delta\mathbf{W}^{(\mathcal{K})}_t = f(\mathbf{m}_{\text{orig},t} + \Delta\mathbf{m}^{(\mathcal{K})}_t, \mathbf{v}_{\text{orig},t} + \Delta\mathbf{v}^{(\mathcal{K})}_t) - f(\mathbf{m}_{\text{orig},t}, \mathbf{v}_{\text{orig},t}) \approx \frac{\partial f}{\partial \mathbf{m}}\Delta\mathbf{m}^{(\mathcal{K})}_t + \frac{\partial f}{\partial \mathbf{v}}\Delta\mathbf{v}^{(\mathcal{K})}_t. \tag{33}$$

The partial derivatives are evaluated at the full-precision point. Since the perturbations to the moments are additive, we substitute them into Equation 33:

$$\delta \mathbf{W}_t^{(\mathcal{K})} \approx \frac{\partial f}{\partial \mathbf{m}} \left( \sum_{l \in \mathcal{K}} \Delta \mathbf{m}_t^{(l)} \right) + \frac{\partial f}{\partial \mathbf{v}} \left( \sum_{l \in \mathcal{K}} \Delta \mathbf{v}_t^{(l)} \right) \tag{34}$$

$$= \sum_{l \in \mathcal{K}} \left( \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m}_t^{(l)} + \frac{\partial f}{\partial \mathbf{v}} \Delta \mathbf{v}_t^{(l)} \right). \tag{35}$$

The term inside the summation in Equation 35 is precisely the first-order approximation of the individual weight perturbation $\delta \mathbf{W}_t^{(l)}$ from quantizing only layer $l$. Thus, we have established the approximate additivity of weight perturbations:

$$\delta \mathbf{W}_t^{(\mathcal{K})} \approx \sum_{l \in \mathcal{K}} \delta \mathbf{W}_t^{(l)}. \tag{36}$$

### A.5 From Weight Perturbation to Loss Distortion

Finally, we connect the weight perturbation to the overall loss distortion. Let $\mathbf{W}_{\text{orig},t+1} = \mathbf{W}_t - \Delta \mathbf{W}_{\text{orig},t}$ be the weight state after a full-precision update (ignoring weight decay). The state after a quantized update is $\mathbf{W}^{(\mathcal{K})}_{\text{mixed},t+1} = \mathbf{W}_t - \Delta \mathbf{W}^{(\mathcal{K})}_{\text{mixed},t} = \mathbf{W}_{\text{orig},t+1} - \delta \mathbf{W}_t^{(\mathcal{K})}$.

The distortion $D_{(\mathcal{K})}$ is the change in loss between these two final states:

$$D_{(\mathcal{K})} = \mathcal{L}(\mathbf{W}^{(\mathcal{K})}_{\text{mixed},t+1}) - \mathcal{L}(\mathbf{W}_{\text{orig},t+1}) = \mathcal{L}(\mathbf{W}_{\text{orig},t+1} - \delta \mathbf{W}_t^{(\mathcal{K})}) - \mathcal{L}(\mathbf{W}_{\text{orig},t+1}). \tag{37}$$

Applying a first-order Taylor expansion to the loss function $\mathcal{L}(\cdot)$ around $\mathbf{W}_{\text{orig},t+1}$ yields:

$$D_{(\mathcal{K})} \approx \nabla \mathcal{L}(\mathbf{W}_{\text{orig},t+1})^T (-\delta \mathbf{W}_t^{(\mathcal{K})}) = -(\mathbf{g}_{t+1})^T \delta \mathbf{W}_t^{(\mathcal{K})}, \tag{38}$$

where $\mathbf{g}_{t+1} = \nabla \mathcal{L}(\mathbf{W}_{\text{orig},t+1})$ is the gradient evaluated at the updated weights.

By substituting the additivity of weight perturbations from Equation 36 and leveraging the linearity of the dot product, we arrive at the final result:

$$D_{(\mathcal{K})} \approx -(\mathbf{g}_{t+1})^T \left( \sum_{l \in \mathcal{K}} \delta \mathbf{W}_t^{(l)} \right) \tag{39}$$

$$= \sum_{l \in \mathcal{K}} \left( -(\mathbf{g}_{t+1})^T \delta \mathbf{W}_t^{(l)} \right) \tag{40}$$

$$\approx \sum_{l \in \mathcal{K}} D_l. \tag{41}$$

Therefore, the total distortion caused by quantizing multiple layers under the AdamW optimizer can be approximated by the sum of distortions from quantizing each layer individually. The approximation error stems from the higher-order terms in two Taylor expansions: one for the non-linear update rule and another for the loss function itself. This indicates that our rate-distortion framework remains theoretically applicable to adaptive optimizers.

## B Ablation Studies

### B.1 Effectiveness of the Distortion Metric

We validate the core principle of our approach through an ablation study designed to assess the effectiveness of the proposed LAD metric (Equation 3) compared with traditional signal-level metrics. Specifically, we evaluate both our Lagrangian-based allocation method and the Greedy baseline when guided by either MSE or LAD.

As shown in Table 3, the results provide strong evidence for the superiority of the LAD metric. First, for both allocation algorithms, replacing MSE with LAD consistently improves final accuracy,

Table 3: Ablation results for different distortion metrics on ResNet-18 at 2 and 3-bit quantization.

| Method | Distortion Metric | Acc. at 2-bit (%) | Acc. at 3-bit (%) |
|---|---|---|---|
| Uniform | - | 77.33 | 87.32 |
| + Greedy | MSE | 87.13 | 88.14 |
| | LAD (Ours) | **88.09** | **88.33** |
| + Ours | MSE | 87.24 | 88.21 |
| | LAD (Ours) | **88.39** | **88.49** |

Table 4: Ablation results for different bit reallocation strategies and trigger configurations. All methods are evaluated on ResNet-18 using 2-bit quantization. The "Strategy Explanation" column briefly summarizes the dynamic behavior of each setting.

| Strategy Type | Configuration | Accuracy (%) | Total Reallocations | Strategy Explanation |
|---|---|---|---|---|
| Static Baselines | $\tau = 0.00$ | 87.31 | 1 | Only allocates at the first iteration |
| Fixed Interval Baselines | Fixed-50 | 88.02 | 784 | Re-allocates every 50 iterations |
| | Fixed-100 | 88.26 | 392 | Re-allocates every 100 iterations |
| Dynamic Trigger (Ours) Sensitivity to $\tau$ (with $k_{\min} = 20$) | $\tau = 0.98$ | 87.80 | 373 | High sensitivity |
| | $\tau = 0.95$ | **88.32** | 91 | **Balanced trade-off** |
| | $\tau = 0.92$ | 88.28 | 59 | Low sensitivity |
| Dynamic Trigger (Ours) Sensitivity to $k_{\min}$ (with $\tau = 0.95$) | $k_{\min} = 0$ | **88.39** | 416 | No minimum reallocation interval |
| | $k_{\min} = 20$ | 88.32 | 91 | **Balanced trade-off** |
| | $k_{\min} = 50$ | 88.20 | 51 | Longer reallocation interval |

confirming that directly quantifying the impact on the training objective offers a more reliable signal for bit allocation. Second, we observe a pronounced synergy between our proposed metric and our optimization algorithm: when guided by the suboptimal MSE metric, the performance gap between our Lagrangian method and the greedy heuristic is relatively small, whereas under LAD guidance the advantage of our method becomes substantial. This demonstrates that the full potential of our principled optimization approach is realized when it is paired with its corresponding principled distortion measure.

### B.2 Analysis of the Dynamic Reallocation Trigger

We conduct an ablation study to evaluate the effectiveness of our proposed dynamic reallocation trigger and analyze its sensitivity to two key hyperparameters: the similarity threshold $\tau$ and the minimum reallocation interval $k_{\min}$. The parameter $\tau$ governs the sensitivity of the trigger to changes in the gradient distribution. Larger values make the system more responsive by lowering the similarity required to trigger reallocation. Conversely, $k_{\min}$ serves as a damping factor, enforcing a minimum number of iterations between successive reallocations; smaller values permit more frequent updates.

Table 4 presents the performance of different configurations. Notably, disabling reallocation after initialization by setting $\tau = 0$ results in a significant accuracy drop to 87.31%, highlighting the critical role of dynamic reallocation in preserving model performance under quantization constraints. We also examine fixed-interval reallocation strategies, such as triggering reallocation every 50 or 100 steps. While these strategies partially recover accuracy, they incur substantial computational overhead, requiring up to 784 reallocation events over the course of training. In contrast, dynamic strategies based on our trigger mechanism achieve superior trade-offs. Among dynamic configurations, $\tau = 0.95$ consistently provides a favorable trade-off between accuracy and efficiency. Although the setting with $\tau = 0.95$ and $k_{\min} = 0$ achieves the highest accuracy of 88.39%, increasing $k_{\min}$ to 20 reduces the number of reallocations from 416 to 91, with only a 0.07% decrease in accuracy. These results confirm the necessity of adaptive reallocation and the importance of tuning the trigger mechanism. Our proposed dynamic trigger significantly outperforms fixed or naive strategies, enabling efficient training with minimal accuracy loss under aggressive quantization.

## C THE USE OF LARGE LANGUAGE MODELS (LLMS)

We acknowledge the use of LLMs as writing assistance tools during the preparation of this paper. Specifically, we used LLMs for language polishing, whose role was limited to improving the grammatical correctness, refining sentence structure, and suggesting alternative academic phrasing to enhance the clarity and readability of our paper. The authors reviewed, edited, and take full responsibility for all content presented in this paper, including the final form of any text refined with LLMs assistance.