LampQ: Towards Accurate Layer-wise Mixed Precision Quantization for Vision Transformers

Minjun Kim[1], **Jaeri Lee**[2], **Jongjin Kim**[1], **Jeongin Yun**[2], **Yongmo Kwon**[2], **U Kang**[1, 2*],

[1]Department of Computer Science and Engineering, Seoul National University
[2]Interdisciplinary Program in Artificial Intelligence, Seoul National University
{minjun.kim, jlunits2, j2kim99, yji00828, rnjsdydah, ukang}@snu.ac.kr

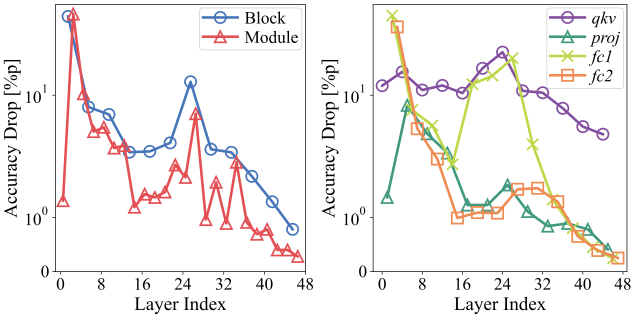
Abstract

How can we accurately quantize a pre-trained Vision Transformer model? Quantization algorithms compress Vision Transformers (ViTs) into low-bit formats, reducing memory and computation demands with minimal accuracy degradation. However, existing methods rely on uniform precision, ignoring the diverse sensitivity of ViT components to quantization. Metric-based Mixed Precision Quantization (MPQ) is a promising alternative, but previous MPQ methods for ViTs suffer from three major limitations: 1) coarse granularity, 2) mismatch in metric scale across component types, and 3) quantization-unaware bit allocation. In this paper, we propose **LampQ** (**La**yer-wise **M**ixed **P**recision **Q**uantization for Vision Transformers), an accurate metric-based MPQ method for ViTs to overcome these limitations. LampQ performs layer-wise quantization to achieve both fine-grained control and efficient acceleration, incorporating a type-aware Fisher-based metric to measure sensitivity. Then, LampQ assigns bit-widths optimally through integer linear programming and further updates them iteratively. Extensive experiments show that LampQ provides the state-of-the-art performance in quantizing ViTs pre-trained on various tasks such as image classification, object detection, and zero-shot quantization.

Code — https://github.com/snudatalab/LampQ

Introduction

How can we compress a pre-trained Vision Transformer model while maintaining accuracy? Vision Transformers (ViTs) (Dosovitskiy et al. 2021; Touvron et al. 2021) have recently gained significant attention due to their superior performance across a wide range of computer vision problems (Li et al. 2022a; Gao et al. 2022; Liu et al. 2024). Despite their success, ViTs are difficult to deploy on resource-limited devices due to their complex architecture, along with significant memory and computational demands (Li et al. 2022c; Liu et al. 2023b). Model quantization (Kim, Kim, and Kang 2025; Kim et al. 2025) mitigates these challenges by converting models into a low-bit format, which enables higher compression rate and faster inference with minimal performance loss over other compression methods such as pruning (Park, Choi, and Kang 2024; Park et al. 2025b), knowledge distillation (Kim, Jung, and Kang 2021; Cho and Kang 2022; Jeon et al. 2023), and low-rank approximation (Jang et al. 2023). Among the two approaches of quantization, Post-Training Quantization (PTQ) (Zhong et al. 2024; Park et al. 2025a) is more suitable for ViTs since quantization-aware training (Li et al. 2022b; Yu et al. 2023) requires training that may take up to several days or weeks.



(a) Block and module

(b) Layer

Figure 1: Accuracy when quantizing a single component of the DeiT-S model to 1-bit following Wu et al. (2024) while keeping the others unchanged. Sensitivity varies significantly across (a) blocks and modules, and (b) layers.

Previous PTQ studies (Li et al. 2023; Liu et al. 2023a; Moon et al. 2024) have achieved promising performance by tackling ViT-specific problems such as inter-channel variation of post-LayerNorm activations (Li et al. 2023), power-law distribution of post-Softmax activations (Wu et al. 2024), and outliers of block outputs (Ma et al. 2024). However, most of these methods employ uniform precision, applying the same bit-width across all components. This approach leads to suboptimal performance since different ViT components (such as blocks, modules, and layers) exhibit varying sensitivities to quantization (see Figures 1, 2, and 6).

Assigning different quantization bit-widths to each component based on their sensitivity, known as Mixed Precision Quantization (MPQ) (Koryakovskiy et al. 2023; Rakka et al. 2024), aims to improve performance by allocating higher bit-widths to critical network layers. Metric-based methods (Piao, Cho, and Kang 2022; Sun et al. 2022; Ma et al.

*Corresponding author.

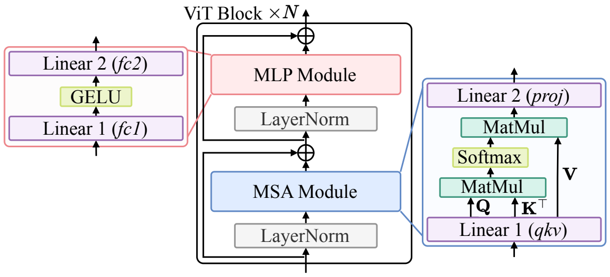


Figure 2: Illustration of a ViT model with N blocks. Each block consists of two modules: MSA (red) and MLP (blue), and four layers: *qkv*, *proj*, *fc1*, and *fc2* (purple).

2023) emerge as the most practical choice among the MPQ techniques. Other MPQ techniques such as gradient (Huang et al. 2022), reinforcement learning (Kim et al. 2024), and neural architecture search (Wang et al. 2025), lack scalability as they demand substantial computational resources; they take 10 to 300 GPU hours even for a small model such as ResNet-18 with only 11M parameters (Kim et al. 2024).

While metric-based MPQ has a potential to be an optimal solution, the only existing MPQ method for ViTs (Liu et al. 2021b) exhibits limited performance due to its simplistic design of the following key components (see Table 1):

- **Granularity.** A module-wise MPQ approach applies the same bit-width across an entire module, disregarding the varying sensitivity of different layers (see Figure 1(b)). This coarse-grained granularity results in suboptimal bit allocation, harming layers that require finer precision.
- **Metric.** The nuclear norms of attention and feature maps vary significantly in scale depending on the module type, such as MSA and MLP (see Figure 5(a)). This discrepancy hinders direct comparison of metrics across modules, leading to inaccurate sensitivity assessments.
- **Bit assignment.** The Pareto frontier approach in prior quantization works is costly and relies on a fixed metric, which fails to reflect error changes during quantization and results in suboptimal decisions (see Figure 6).

In this paper, we propose **LAMPQ** (**La**yer-wise **M**ixed **P**recision **Q**uantization for Vision Transformers), an accurate PTQ method for ViTs by a layer-wise metric-based MPQ. By choosing layers as its quantization granularity, LAMPQ ensures both fine-grained control and efficient acceleration via kernel support. LAMPQ quantifies each layer's sensitivity with a type-aware Fisher-based metric, enabling direct comparison between layers with different types. LAMPQ assigns bits initially through integer linear programming and then updates them iteratively to dynamically reflect changes in sensitivity. LAMPQ is powerful and versatile since it is easily integrated with any PTQ method for ViTs, achieving the state-of-the-art performance.

Our main contributions are summarized as follows:

- **Observation.** We observe three major challenges in designing a metric-based MPQ method for ViTs each associated with a specific component: granularity, metric, and bit assignment (see Figures 1, 5, and 6).

Table 1: Comparison of the key components in metric-based MPQ between VT-PTQ (Liu et al. 2021b) and LAMPQ.

Component	**VT-PTQ**	**LAMPQ (Proposed)**	
Granularity	Module-wise	Layer-wise	
Metric	Nuclear norm of attention / feature maps	Trace of Fisher info.	Layer-wise recon. error
Bit assign.	Pareto frontier	Integer linear programming	Iterative bit update

- **Method.** We propose LAMPQ, an accurate MPQ method for ViTs. LAMPQ carefully assigns quantization bits to achieve both efficiency and accuracy (see Figure 4). LAMPQ considers the differences between types of layers by a type-aware Fisher-based metric, and incorporates the quantization feedback by iterative bit updates.
- **Experiments.** We experimentally show that LAMPQ consistently outperforms its competitors on various models and datasets in image classification, object detection, and zero-shot quantization tasks (see Tables 3, 4, and 5).

Preliminaries

Problem Definition

Given a pre-trained model, a small calibration dataset, and quantization bits, Post-Training Quantization (PTQ) targets to optimize the quantized model to maintain performance.

Problem 1 (Post-training Quantization for Vision Transformers (ViTs) (Zhong et al. 2024; Wu et al. 2024))**.**

- ***Input****: a ViT model f_θ with parameters θ pre-trained on a target task $\mathcal{T}$, a sample dataset $\mathbb{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{S}$ of size S, and quantization bits b_t.*
- ***Output****: a quantized model $f_{\theta'}$ with parameters θ' within the b_t-bit limit minimizing performance degradation on $\mathcal{T}$.*

Vision Transformers

ViTs (Dosovitskiy et al. 2021) are deep learning models that apply self-attention to capture image context and improve feature representation. Figure 2 illustrates the architecture of a standard ViT block, which we reformulate in a simplified manner. Each block consists of two modules: Multi-head Self-Attention (MSA) and Multi-Layer Perceptron (MLP).

Simplified Formulation. To facilitate layer-wise quantization analysis, we abstract each ViT block as consisting of four core linear layers: *qkv*, *proj*, *fc1*, and *fc2*. While this formulation omits the inner workings of the attention mechanism, we empirically validate its effectiveness later (see Section *'Type-aware Fisher-based Metric'*). This abstraction supports a metric both principled and effective for ViTs.

Model Quantization

Model quantization involves converting a model into a lower bit precision. We focus on asymmetric uniform quantization, following previous researches (Li et al. 2023; Wu et al. 2024). Given a matrix $\mathbf{M}$, the B-bit quantized matrix $\mathbf{M}' = \lfloor \mathbf{M}/s - z + 0.5 \rfloor$, where $s = (r_{max} - r_{min})/(2^B - 1)$ is the

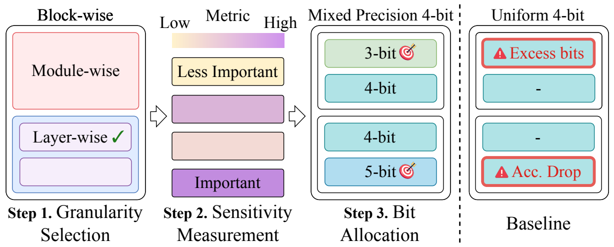


Figure 3: Illustration of how metric-based MPQ works. They first partition model parameters into groups and measure their sensitivity. More bits are allocated to sensitive groups, reducing model size while maintaining performance.

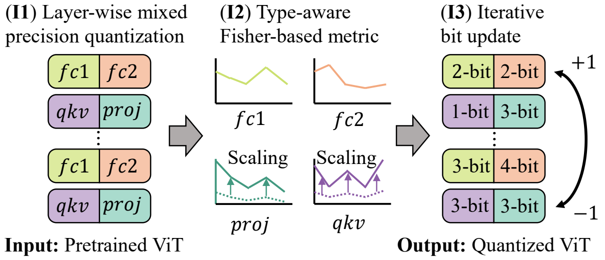


Figure 4: Overall architecture of LAMPQ. Our main ideas are I1) layer-wise mixed-precision quantization, I2) type-aware Fisher-based metric, and I3) iterative bit update.

scaling factor, $z = r_{min}/s + 2^{B-1}$ is the integer offset, and (r_{min}, r_{max}) are the lower and upper bounds of $\mathbf{M}$. The corresponding dequantized value is given by $\widehat{\mathbf{M}} = s(\mathbf{M}' + z)$. While some methods quantize only the weights, we quantize both weights and activations for efficient inference.

Metric-based Mixed Precision Quantization

In this work, we follow a metric-based approach for Mixed Precision Quantization (MPQ) (Sun et al. 2022; Ma et al. 2023). As shown in Figure 3, the three key components of metric-based MPQ are granularity, metric, and bit assignment. Granularity determines the level (e.g., per-layer, per-channel, or per-tensor) at which bit-width allocation is applied, and is selected in the first step of MPQ to define the partitioning unit. Sensitivity[1] metric evaluates how each partition responds to quantization and is used to assess the impact of different bit-widths. Finally, bit assignment determines the bit-width allocation based on sensitivity, prioritizing important segments under a given budget.

Hessian-based Importance. In weight-only quantization of CNN models, researchers introduce the trace $\text{tr}(\mathbf{H}_i)$ of each layer's Hessian matrix $\mathbf{H}_i$ as a potential metric, as shown in Lemma 1 (Dong et al. 2020; Yao et al. 2021).

Lemma 1 (Layer importance and Hessian trace**).** *Assume that for all layers l_i with weight vector $\mathbf{W}_i$, its gradient $\mathbf{g}_i = \mathbf{0}$, Hessian $\mathbf{H}_i$ is positive semi-definite for target loss $\mathcal{L}$, and $\exists \alpha \in \mathbb{R}, \Delta_{\mathbf{W}_i} = \widehat{\mathbf{W}}_i - \mathbf{W}_i = \alpha \sum_k \mathbf{v}_{ik}$, where $\widehat{\mathbf{W}}_i$ is a dequantized form of the quantized weight $\mathbf{W}'_i$ and $\{\mathbf{v}_{ik}\}$ are the eigenvectors of $\mathbf{H}_i$. For two layers l_i and l_j, if $\|\Delta_{\mathbf{W}_i}\|_2^2 = \|\Delta_{\mathbf{W}_j}\|_2^2$ and $\text{tr}(\mathbf{H}_i) > \text{tr}(\mathbf{H}_j)$, then,*

$$\mathcal{L}(\widehat{\mathbf{W}}_i) \geq \mathcal{L}(\widehat{\mathbf{W}}_j).$$

Proof. Refer to Lemma 1 of HAWQ-V2 (Dong et al. 2020). □

Proposed Method

We propose **LAMPQ** (**La**yer-wise **M**ixed **P**recision **Q**uantization for Vision Transformers), an accurate mixed-precision PTQ method for ViTs. These are the three main challenges that must be tackled:

C1. Coarse-grained granularity. Previous methods enforce uniform quantization within each module, limiting precision flexibility at deeper levels. How can we apply finer granularity while ensuring acceleration?

C2. Mismatch in metric scale across component types. Sensitivity scores vary widely in scale across component types, hindering direct comparison. How can we design a metric that enables fair comparison across heterogeneous components?

C3. Quantization-unaware bit allocation. Bit allocation based on one-shot full-precision metrics is suboptimal, as it overlooks how quantizing earlier layers affects the sensitivity of later ones. How can we incorporate quantization feedback without compromising efficiency?

We propose three main ideas to address the challenges:

I1. Layer-wise mixed precision quantization. We propose an MPQ strategy that flexibly allocates bit-widths at layer level, enhancing both inference speed and performance of the quantized model.

I2. Type-aware Fisher-based metric. We measure the sensitivity of each layer with the trace of its Fisher information matrix and introduce type-aware scaling to address inter-layer range mismatches.

I3. Iterative bit update. We assign initial bit-widths via Integer Linear Programming (ILP) and update them iteratively based on estimated error to correct misallocations.

Figure 4 depicts the overall procedure of LAMPQ. Given a pre-trained full-precision ViT with N blocks, we structure it into $4N$ layers, with N layers for each layer types: *qkv*, *proj*, *fc1*, and *fc2*. Next, we quantify the sensitivity of each layer towards quantization by evaluating the trace of its Fisher information matrix, scaled according to the layer type. With this sensitivity metric, we determine the initial bit-widths by solving an ILP problem. After quantizing the ViT with the initial bit-widths, LAMPQ iteratively updates the bit allocation with the layer-wise reconstruction error, ensuring that the updated bit allocation reflects the quantization-induced error. Algorithm of LAMPQ is formulated as Alg. 1 (see Section *'Algorithm'*). For clarity, we reference the relevant lines for each idea. Note that LAMPQ is compatible with any quantization method for ViTs; we adopt AdaLog (2024) as the baseline for best performance.

[1] In the remainder of the paper, we use the term 'sensitivity' to refer to 'quantization sensitivity' for simplicity.

Layer-wise Mixed Precision Quantization

Observation. We present an empirical observation that layer-wise sensitivity varies significantly, making module-based MPQ approaches suboptimal. We prepare a full-precision DeiT-S (2021) model and quantize each component following AdaLog (Wu et al. 2024) while keeping other components unchanged to ignore their effects. Figure 1 shows that accuracy degradation varies significantly not only across (a) blocks and modules, but also (b) individual layers. Within each module, *qkv* and *proj* in MSA, and *fc1* and *fc2* in MLP show notably different sensitivities, especially in the middle layers (16–32). This variation in sensitivity is consistently observed across different models (see Section *'Further Experiments'*). However, module-wise MPQ methods (Liu et al. 2021b) assign the same bits to all components within a module, ignoring this sensitivity variation.

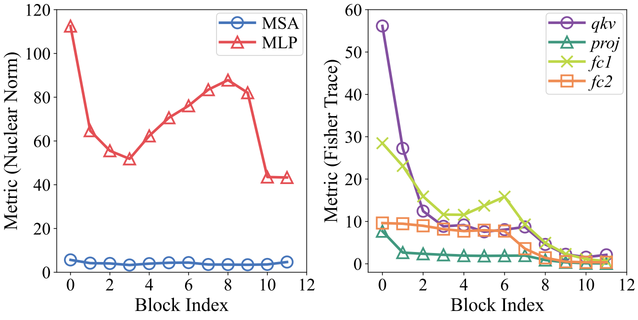
Solution. Motivated by this observation, we allocate varying bit-widths for each layer to reflect their sensitivity differences. To achieve this, we propose a layer-wise MPQ scheme that determines its granularity at the layer level. Finer-grained quantization beyond the module level improve sensitivity estimation, but often introduce hardware inefficiencies. In particular, granularity finer than the layer level (e.g., per-channel or per-weight) introduces varying bit-widths within the same weight or activation matrix, requiring frequent conversions to full precision during computation, leading to substantial runtime overhead. By contrast, layer-wise quantization offers a good trade-off between inference speed and performance, as layers are the smallest units compatible with low-bit kernels (Park et al. 2024).

Type-aware Fisher-based Metric

Observation. Designing an accurate and efficient metric to estimate each component's quantization sensitivity is crucial for metric-based MPQ for ViTs. VT-PTQ estimates the quantization sensitivity of MSA and MLP modules by evaluating the nuclear norm of attention and feature maps, respectively. However, this approach has two major limitations. First, the metric relies on empirical intuition, which lacks a solid theoretical foundation. A greater nuclear norm indicates more information in the matrix, which may suggest an important layer, but it does not directly correlate with the performance. Second, the metric exhibits a large scale gap between two module types, MSA and MLP. Figure 5(a) illustrates the metric values measured on a DeiT-S model with VT-PTQ. The metric values for MSA and MLP differ by 10 to 40 times, making direct comparison between inter-type modules challenging. Hence, developing a metric with theoretical justification and scaling consistency is essential.

Solution. In metric-based MPQ, a sensitivity score Ω_i is required to quantify the expected degradation from quantizing each layer l_i; Ω_i later guides the bit-width allocation. Our idea is to obtain this score from the trace $\text{tr}(\mathbf{F})$ of the Fisher information matrix with type-aware scaling α_t for fair comparison across different layer types (lines 1-10 of Alg. 1).

As shown in Lemma 1, the Hessian trace $\text{tr}(\mathbf{H})$ is an effective sensitivity metric for weight-only MPQ in CNNs (Dong et al. 2020; Yao et al. 2021). Motivated by this, we adopt the Hessian trace as our initial basis for layer sensitivity of ViTs under a simplified four-linear-layer formulation. However, directly applying this metric into ViTs faces two challenges. First, computing $\mathbf{H}_i$ for each layer l_i requires second-order derivatives with respect to all parameters, leading to significant overhead. Second, as ViTs contain four layer types while CNNs have only one, assuming that all types exhibit the same accuracy change per unit metric value is invalid.



(a) VT-PTQ (2021b) (b) LAMPQ (Proposed)

Figure 5: Comparison of metric values between (a) VT-PTQ (2021b) and (b) LAMPQ for a DeiT-S model.

To address these challenges, we first replace $\text{tr}(\mathbf{H}_i)$ with $\text{tr}(\mathbf{F}_i)$, noting that the expected Hessian $\mathbf{H}_i$ of each layer approximates its Fisher information matrix $\mathbf{F}_i$ as in Lemma 2.

Lemma 2 (Hessian and Fisher information matrices)**.** *Assume that loss $\mathcal{L}$ is the negative log-likelihood. Let θ, $p(y|x,\theta)$, $q(x)$ as the model parameters, conditional probability of y given input x, and an empirical distribution of x, respectively. Then, the expected Hessian of layer l_i*

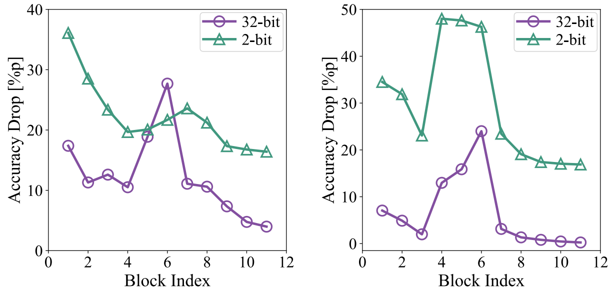
$$\mathbb{E}_{x\sim q(x)}\mathbb{E}_{y\sim p(y|x,\theta)}[\mathbf{H}_i] \approx \mathbf{F}_i,$$

where $\mathbf{F}_i$ denotes the Fisher information matrix of layer l_i.

Proof. Refer to Section *'Theoretical Analysis'*. □

Next, we need to normalize sensitivity metrics across layer types, so that their values correspond to an equivalent amount of accuracy degradation. To achieve this, we introduce a type-specific scaling factor α_t, which converts a unit metric value into an equivalent amount of accuracy drop for each layer type. We define the type-scaled sensitivity metric Ω_i for layer l_i as $\Omega_i = \alpha_t \text{tr}(\mathbf{F}_i)$, where $\alpha_t \in \{\alpha_{qkv}, \alpha_{proj}, \alpha_{fc1}, \alpha_{fc2}\}$ is assigned based on the layer type t. This scaling enables fair comparison of sensitivity across heterogeneous components by expressing their metric values in terms of expected accuracy impact.

Then, how can we determine α_t without investigating all layers which leads to an excessive computation? Our idea is to approximate it from a few sampled layers of each type. Specifically, we define $\alpha_t = A^{(t)}/\text{tr}(\mathbf{F}^{(t)})$ as the ratio between the average accuracy drop $A^{(t)}$ and the average Fisher trace $\text{tr}(\mathbf{F}^{(t)})$ when μ layers of type t are quantized to β-bit ($\mu \leq L$ and $\beta < 32$). As a result, LAMPQ effectively scales metric values across different types, as shown in Figure 5(b).



(a) *qkv* (b) *fc1*

Figure 6: Accuracy drop by 1-bit quantization of (a) *qkv* or (b) *fc1* layers. 32-bit and 2-bit denote a DeiT-S model and its variant with the first block quantized to 2-bit, respectively.

Iterative Bit Update

Observation. We present an empirical observation that quantized models show different sensitivity patterns compared to full-precision models. Figure 6 depicts the accuracy drop in a full-precision (32-bit) DeiT-S model and its variant (2-bit), where only the first block is quantized to 2-bit, when quantizing each layer to 1-bit following AdaLog while keeping others unchanged. The sensitivity trend of the 2-bit variant differs significantly (specifically in blocks 4-8) from its original, highlighting the importance of accounting for quantization effects. However, previous approaches evaluate sensitivity only once and assign bit-widths accordingly, thereby failing to reflect this trend difference.

Solution. To mitigate this challenge, we design a two-step scheme that 1) first assigns bit-widths by solving an Integer Linear Programming (ILP) problem, and then 2) iteratively updates them based on the estimated reconstruction error. In this way, LAMPQ dynamically adapts to evolving sensitivity patterns, leading to more accurate bit configurations.

Step 1. Initial assignment. Our goal is to quickly initialize bit-widths that minimize quantization error while satisfying the average bit-width constraint. That is, we want to determine the bit allocation $\{B_i^*\}$ that minimizes total error:

$$\{B_i^*\} = \operatorname*{argmin}_{\{B_i\}} \sum_{i=1}^{4N} \Omega_i^{(B_i)}, \text{s.t.} \sum_{i=1}^{4N} c_i B_i \le b_t \sum_{i=1}^{4N} c_i,$$

where $\Omega_i^{(B_i)}$ denotes the estimation error at B_i-bit and c_i is the number of parameters of layer l_i. The key challenge is to define $\Omega_i^{(B_i)}$ as a proper proxy for quantization error and to optimize bit allocation efficiently under this definition. We begin by identifying the key requirements for $\Omega_i^{(B_i)}$: it should (1) reflect the sensitivity of each layer so that more critical layers receive higher bit-widths, (2) increase the penalty as bit-width decreases reflecting the amplified error at lower precisions, and (3) be computationally efficient to evaluate. To meet these requirements, we extend our type-aware Fisher-based sensitivity metric Ω_i with an additional penalty to model error growth at lower bit-widths, defining $\Omega_i^{(B_i)} = \gamma^{-B_i}\Omega_i$ with penalizing factor $\gamma > 1$. With the defined $\Omega_i^{(B_i)}$, the bit allocation problem becomes an ILP task. We adopt a fast ILP solver to efficiently find the global optimum under the average bit-width constraint, providing a practical initialization for subsequent iterative refinement. Compared to existing methods that construct a Pareto frontier, LAMPQ accelerates the allocation process over 250× (see Section *'Further Experiments'*).

Table 2: Relative values of estimated $\mathcal{L}_{recon}^{(B_i)}(l_i)$ when activation $\mathbf{X}_i$ follows the Gaussian distribution $\mathcal{N}(0,1)$.

B_i	$\mathcal{L}_{recon}^{(B_i)}(l_i)$	$\frac{\mathcal{L}_{recon}^{(B_i-1)}(l_i)}{\mathcal{L}_{recon}^{(B_i)}(l_i)}$
1	1.000 E+0	-
2	1.209 E−2	× 82.74
3	1.835 E−3	× 6.59
4	3.903 E−4	× 4.70
5	9.093 E−5	× 4.29
6	2.199 E−5	× 4.13
7	5.339 E−6	× 4.12
8	1.324 E−6	× 4.03

Step 2. Error-based iterative updates. Our goal is to further update bit assignments to reduce quantization error, while keeping the same average bit-widths assigned from the initial assignment step. For the goal, we iteratively update bit assignments; at each iteration, we select two layers, and increase the first layer's bit-width by 1, while decreasing the second layer's bit-width by 1. In this way, we guarantee that the average bit assignment remains the same.

Then, how can we find the best two layers to increase/decrease their bit-widths? Our idea is to make decision based on gain and degradation which we define for this purpose. The gain of a layer l_i is defined to be the reduction in error when increasing the bit-width of l_i by 1, while the degradation of it is the increase in error when decreasing the bit-width of l_i by 1. Specifically, the gain and the degradation of l_i are given by $\mathcal{L}_{recon}^{(B_i-1)}(l_i) - \mathcal{L}_{recon}^{(B_i)}(l_i)$ and $\mathcal{L}_{recon}^{(B_i)}(l_i) - \mathcal{L}_{recon}^{(B_i+1)}(l_i)$, respectively, where $\mathcal{L}_{recon}^{(B_i)}(l_i)$ is the layer-wise reconstruction error of l_i at B_i-bit:

$$\mathcal{L}_{recon}^{(B_i)}(l_i) = \|\widehat{\mathbf{W}}_i^{(B_i)}\widehat{\mathbf{X}}_i^{(B_i)} - \mathbf{W}_i\mathbf{X}_i\|_F^2/\|\mathbf{W}_i\mathbf{X}_i\|_F^2,$$

where $\mathbf{W}_i$ and $\mathbf{X}_i$ are weights and activations of layer l_i, respectively, and $\widehat{\mathbf{W}}_i^{(B_i)}$ and $\widehat{\mathbf{X}}_i^{(B_i)}$ are dequantized weights and activations after quantization, respectively. This updates are performed iteratively until the model accuracy on the sample dataset $\mathbb{D}$ stops improving (lines 22-24 of Alg. 1).

Efficient estimation of error gain and degradation. Naïvely evaluating gain and degradation requires measuring loss for all bit-width reallocation pairs, which is prohibitively expensive for ViTs. To make this tractable, we approximate gain and degradation using statistical estimation, eliminating the need to recompute reconstruction errors for every bit-width change. Concretely, we approximate the ratio $\mathcal{L}_{recon}^{(B_i-1)}(l_i)/\mathcal{L}_{recon}^{(B_i)}(l_i)$ with its expectation $\mathbb{E}(\cdot)$ over weight and activation distributions, as formalized in Lemma 3. For example, the gain $\mathcal{L}_{recon}^{(B_i-1)}(l_i) - \mathcal{L}_{recon}^{(B_i)}(l_i)$ can be rewritten as $\mathcal{L}_{recon}^{(B_i)}(l_i)\big(\frac{\mathbb{E}(\mathcal{L}_{recon}^{(B_i-1)}(l_i))}{\mathbb{E}(\mathcal{L}_{recon}^{(B_i)}(l_i))} - 1\big)$, where the ratio term is substituted with the pre-computed expectation.

Lemma 3 (Expected ratio of reconstruction losses)**.** *Assume that given a layer l_i, its weight $\mathbf{W}_i$ and activation*

Table 3: Image classification accuracy [%] of quantized ViTs on the ImageNet dataset. WBAB denotes that weights and activations are quantized into B-bit. MP indicates MPQ. Note that LampQ achieves the highest accuracy in all cases.

Method	W/A	ViT		DeiT			Swin		Average
		ViT-S	**ViT-B**	**DeiT-T**	**DeiT-S**	**DeiT-B**	**Swin-S**	**Swin-B**	
Full-Precision	32/32	81.38	84.53	72.13	79.83	81.80	83.23	85.27	81.17
QDrop (2022)	4/4	17.77	21.72	31.65	35.79	65.47	78.92	80.49	47.40
PTQ4ViT (2022)	4/4	42.57	30.69	36.96	34.08	64.39	76.09	74.02	51.26
PD-Quant (2023a)	4/4	32.64	34.86	58.50	64.85	73.76	77.04	75.84	59.64
RepQ-ViT (2023)	4/4	65.05	68.48	57.43	69.03	75.61	79.45	78.32	70.48
OPTQ (2023)	4/4	67.59	75.12	58.96	70.85	76.10	80.17	81.08	72.84
ERQ (2024)	4/4	68.91	76.63	60.29	72.56	78.23	80.74	82.44	74.26
AdaLog (2024)	4/4	72.75	79.68	63.52	72.06	78.03	80.77	82.47	75.61
VT-PTQ† (2021b)	4MP/4MP	73.69	80.10	63.90	72.78	78.30	80.96	82.80	76.07
LampQ (Proposed)	4MP/4MP	**74.02**	**81.91**	**65.71**	**75.40**	**79.24**	**81.76**	**83.87**	**77.42**
QDrop (2022)	3/3	4.44	8.00	30.73	22.67	24.37	60.89	54.76	29.41
PTQ4ViT (2022)	3/3	0.01	0.01	0.04	0.01	0.27	0.35	0.29	0.14
RepQ-ViT (2023)	3/3	0.43	0.14	0.97	4.37	4.84	8.84	1.34	2.99
AdaLog (2024)	3/3	13.88	37.91	31.56	24.47	57.45	64.41	69.75	42.78
VT-PTQ† (2021b)	3MP/3MP	16.62	42.13	32.98	26.37	60.14	69.80	73.51	45.94
LampQ (Proposed)	3MP/3MP	**23.06**	**48.53**	**37.54**	**45.38**	**61.44**	**70.91**	**75.82**	**51.81**

†: AdaLog quantization with bit allocation by VT-PTQ.

$\mathbf{X}_i$ are independent. Then, the expected ratio between the reconstruction error at $(B_i - 1)$-bit and B_i-bit is:

$$\frac{\mathbb{E}(\mathcal{L}_{recon}^{(B_i-1)}(l_i))}{\mathbb{E}(\mathcal{L}_{recon}^{(B_i)}(l_i))} = \frac{k(B_i-1;l_i)}{k(B_i;l_i)},$$

where $k(\cdot) = \mathbb{E}\Big(\big(\Delta_W^{(B_i)}X + W\Delta_W^{(B_i)} + \Delta_W^{(B_i)}\Delta_X^{(B_i)}\big)^2\Big)$, and $\mathbb{E}(\cdot)$ denotes the expectation over random variables W, X (distributed as elements of $\mathbf{W}_i, \mathbf{X}_i$) and $\Delta_W^{(B_i)}, \Delta_X^{(B_i)}$ (distributed as elements of $\widehat{\mathbf{W}}_i^{(B_i)} - \mathbf{W}_i, \widehat{\mathbf{X}}_i^{(B_i)} - \mathbf{X}_i$).

Proof. Refer to Section *'Theoretical Analysis'*. □

We compute the expectations in Lemma 3 by modeling weights and activations as Gaussian $\mathcal{N}(0,1)$, following prior observations (Yuan et al. 2022; Li et al. 2023). Pre-computed expectations for Gaussian distribution are summarized in Table 2. We detail the derivation process in Section *'Theoretical Analysis'* and Table 9.

Experiments

We present experimental results to answer the questions:

Q1. Image classification accuracy. How accurate are the quantized ViTs with LampQ in image classification?

Q2. Object detection precision. How accurate are the quantized ViTs by LampQ in object detection?

Q3. Application to zero-shot quantization. How accurate are quantized ViTs in zero-shot quantization?

Q4. Ablation study. Does each component of LampQ help improve performance?

Q5. Case study on bit allocation. Does LampQ assign bits according to quantization sensitivity?

Experimental Setup

We briefly introduce the experimental setup. Refer to Section *'Experimental Setup'* for further details.

Setup. We evaluate our method with ViT (2021), DeiT (2021), and Swin (2021a) models on ImageNet (2009) dataset for image classification and zero-shot quantization, and MS COCO (2014) dataset for object detection.

Competitors and Details. We compare LampQ with nine existing PTQ methods for image classification and object detection tasks. For zero-shot quantization, we set PSAQ-ViT (2022d) as our baseline. We follow the settings from PSAQ-ViT (2022d) and AdaLog (2024) for fair comparison.

Image Classification Accuracy (Q1)

We evaluate the image classification accuracy of ViTs quantized by LampQ against existing quantization methods in Table 3. LampQ consistently improves quantized models across diverse bit-widths and architectures, achieving up to 5.87%p higher average accuracy. Notably, LampQ becomes increasingly effective as the bit-width decreases, highlighting its robustness in challenging low-precision settings.

Object Detection Precision (Q2)

We investigate the effectiveness of LampQ for quantized models on object detection and instance segmentation. Table 4 shows the average precision of each quantized model's bounding box (AP$^{\text{box}}$) and segmentation mask (AP$^{\text{mask}}$). LampQ achieves the state-of-the-art performance across all settings, validating its generalization to detection tasks.

Application to Zero-shot Quantization (Q3)

We investigate the effectiveness of LampQ in settings without any real data. Table 5 shows the zero-shot quantization accuracy of four ViT models. LampQ enhances the quantization accuracy across various models, achieving up to 0.62%p increase when applied to PSAQ-ViT (Li et al.

Table 4: Precision of quantized models on MS-COCO dataset. Note that LAMPQ achieves the best performance in all cases.

Method	**W/A**	**Mask R-CNN**				**Cascade Mask R-CNN**				**Average**
		Swin-T		**Swin-S**		**Swin-T**		**Swin-S**		
		AP^{box}	AP^{mask}	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}	
Full-Precision	32/32	46.0	41.6	48.5	43.3	50.4	43.7	51.9	45.0	46.3
QDrop (2022)	4/4	12.4	12.9	42.7	40.2	23.9	21.2	24.1	21.4	24.9
PD-Quant (2023a)	4/4	17.7	18.1	32.2	30.9	35.5	31.0	41.6	36.3	30.4
RepQ-ViT (2023)	4/4	36.1	36.0	44.2	40.2	47.0	41.4	49.3	43.1	42.2
OPTQ (2023)	4/4	36.3	36.3	42.9	40.2	47.1	41.5	49.2	43.2	42.1
ERQ (2024)	4/4	36.8	36.6	43.4	40.7	47.9	42.1	50.0	43.6	42.6
AdaLog (2024)	4/4	39.1	37.7	44.3	41.2	48.2	42.3	50.6	44.0	43.4
VT-PTQ† (2021b)	4MP/4MP	39.2	37.7	44.3	41.3	48.3	42.5	50.9	44.2	43.6
LAMPQ (Proposed)	4MP/4MP	**39.8**	**38.4**	**44.9**	**41.8**	**49.0**	**43.1**	**51.1**	**44.5**	**44.1**

†: AdaLog quantization with bit allocation by VT-PTQ.

Table 5: Zero-shot quantization accuracy [%] on ImageNet dataset. LAMPQ consistently shows the best performance.

Method	**W/A**	**DeiT**		**Swin**	
		DeiT-T	**DeiT-S**	**Swin-T**	**Swin-S**
Original	32/32	72.21	79.85	81.35	83.20
PSAQ-ViT	4/8	65.57	72.04	69.78	75.03
VT-PTQ†	4MP/8MP	65.65	72.18	69.91	75.09
LAMPQ	4MP/8MP	**66.27**	**72.71**	**70.24**	**75.49**
PSAQ-ViT	8/8	71.56	75.97	73.54	76.68
VT-PTQ†	8MP/8MP	71.58	76.02	73.63	76.72
LAMPQ	8MP/8MP	**71.77**	**76.20**	**73.76**	**76.85**

†: PSAQ-ViT quantization with bit allocation by VT-PTQ.

Table 6: Ablation study of our proposed ideas in LAMPQ. All ideas of LAMPQ effectively enhance the performance.

Method	**MPQ**	**I1**	**I2**	**I3**	**Accuracy**
Base: AdaLog (2024)	✗	✗	✗	✗	24.47
Base + VT-PTQ (2021b)	✓	✗	✗	✗	26.37
Base + I1 + I2	✓	✓	✓	✗	44.43
Base + I1 + I3	✓	✓	✗	✓	27.87
LAMPQ (Proposed)	✓	✓	✓	✓	**45.38**

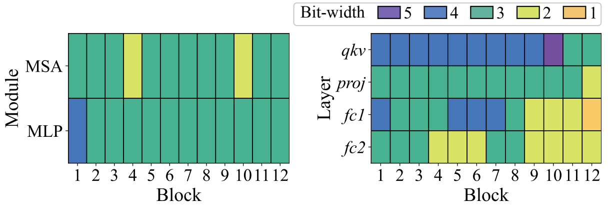
2022d). The results show that LAMPQ is robust towards dataset quality, effective both for real and synthetic datasets.

Ablation Study (Q4)

We perform an ablation study to show that each main idea of LAMPQ improves the performance. Table 6 summarizes the 3-bit results of a DeiT-S model pre-trained on ImageNet. Our analysis shows that all ideas of LAMPQ contribute to the improved performance, with type-aware Fisher-based metric (I2) having the strongest impact of 19.96%p.

Case Study on Bit Allocation (Q5)

We conduct a case study to examine how LAMPQ allocates bits and whether it reflects each layer's sensitivity. Figure 7 compares the bit allocation of a 3-bit DeiT-S model by (a) VT-PTQ and (b) LAMPQ. LAMPQ captures layer-level sensitivity variations within modules that VT-PTQ misses. Fine-grained allocation shows *qkv* and *fc1* requiring higher bits than *proj* and *fc2*, consistent to our analysis in Figure 1(b).



(a) VT-PTQ (b) LAMPQ (Proposed)

Figure 7: Bit allocation of 3-bit quantized DeiT-S models.

Related Work

MPQ (Rakka et al. 2024) quantizes different components of a model with varying bit-precisions. There are four main approaches: learning, Reinforcement Learning (RL), Neural Architecture Search (NAS), and metric-based solutions. Learning-based methods (Huang et al. 2022; Shin et al. 2023) treat bit-widths as trainable parameters and update them based on loss gradients. RL-based methods (Lou et al. 2020; Kim et al. 2024) leverage a RL agent to determine the allocation policy. NAS-based solutions (Yu et al. 2020; Wang et al. 2025) explore the bit selection space through an automated search process. Metric-based methods rely on statistical properties such as quantization entropy (Sun et al. 2022) and orthogonality (Ma et al. 2023). Among them, LAMPQ quantifies sensitivity based on the trace of Fisher information matrix. In this way, LAMPQ more precisely allocates bit-widths to layers and achieves better performance.

Conclusion

We propose **LAMPQ**, an accurate Mixed Precision Quantization (MPQ) method for Vision Transformers, which is adaptable to any existing quantization method. LAMPQ outperforms existing methods in diverse tasks such as image classification, object detection, and zero-shot quantization. Future works include extending LAMPQ into various settings such as vision language models and generation tasks.

Acknowledgments

This work was supported by Mobile eXperience (MX) Business, Samsung Electronics Co., Ltd. U Kang is the corresponding author.

References

Cho, I.; and Kang, U. 2022. Pea-KD: Parameter-efficient and accurate Knowledge Distillation on BERT. *PLOS ONE*, 17(2): 1–12.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

Ding, Y.; Qin, H.; Yan, Q.; Chai, Z.; Liu, J.; Wei, X.; and Liu, X. 2022. Towards accurate post-training quantization for vision transformer. In *ACM MM*.

Dong, Z.; Yao, Z.; Arfeen, D.; Gholami, A.; Mahoney, M. W.; and Keutzer, K. 2020. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. In *NeurIPS*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

Forrest, J.; and Lougee-Heimer, R. 2005. CBC user guide. In *Emerging theory, methods, and applications*, 257–277. INFORMS.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *ICLR*.

Gao, Z.; Tan, C.; Wu, L.; and Li, S. Z. 2022. Simvp: Simpler yet better video prediction. In *CVPR*.

Huang, X.; Shen, Z.; Li, S.; Liu, Z.; Xianghong, H.; Wicaksana, J.; Xing, E.; and Cheng, K.-T. 2022. SDQ: Stochastic differentiable quantization with mixed precision. In *ICML*.

Jang, J.; Quan, C.; Lee, H. D.; and Kang, U. 2023. Falcon: lightweight and accurate convolution based on depthwise separable convolution. *Knowl. Inf. Syst.*, 65(5): 2225–2249.

Jeon, H.; Park, S.; Kim, J.-G.; and Kang, U. 2023. PET: Parameter-efficient Knowledge Distillation on Transformer. *PLOS ONE*, 18(7): 1–21.

Kim, H.-B.; Lee, J. H.; Yoo, S.; and Kim, H.-S. 2024. MetaMix: Meta-state Precision Searcher for Mixed-precision Activation Quantization. In *AAAI*.

Kim, J.; Jung, J.; and Kang, U. 2021. Compressing deep graph convolution network with multi-staged knowledge distillation. *PLOS ONE*, 16(8): 1–18.

Kim, M.; Choi, J.; Lee, J.; Cho, W.; and Kang, U. 2025. Zero-shot Quantization: A Comprehensive Survey. In *IJCAI*.

Kim, M.; Kim, J.; and Kang, U. 2025. SynQ: Accurate Zero-shot Quantization by Synthesis-aware Fine-tuning. In *ICLR*.

Koryakovskiy, I.; Yakovleva, A.; Buchnev, V.; Isaev, T.; and Odinokikh, G. 2023. One-shot model for mixed-precision quantization. In *CVPR*.

Li, Y.; Mao, H.; Girshick, R.; and He, K. 2022a. Exploring plain vision transformer backbones for object detection. In *ECCV*.

Li, Y.; Xu, S.; Zhang, B.; Cao, X.; Gao, P.; and Guo, G. 2022b. Q-vit: Accurate and fully quantized low-bit vision transformer. In *NeurIPS*.

Li, Y.; Yuan, G.; Wen, Y.; Hu, J.; Evangelidis, G.; Tulyakov, S.; Wang, Y.; and Ren, J. 2022c. Efficientformer: Vision transformers at mobilenet speed. In *NeurIPS*.

Li, Z.; Ma, L.; Chen, M.; Xiao, J.; and Gu, Q. 2022d. Patch similarity aware data-free quantization for vision transformers. In *ECCV*.

Li, Z.; Xiao, J.; Yang, L.; and Gu, Q. 2023. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In *ICCV*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2024. Visual instruction tuning. In *NeurIPS*.

Liu, J.; Niu, L.; Yuan, Z.; Yang, D.; Wang, X.; and Liu, W. 2023a. Pd-quant: Post-training quantization based on prediction difference metric. In *CVPR*.

Liu, X.; Peng, H.; Zheng, N.; Yang, Y.; Hu, H.; and Yuan, Y. 2023b. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *CVPR*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021a. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*.

Liu, Z.; Wang, Y.; Han, K.; Zhang, W.; Ma, S.; and Gao, W. 2021b. Post-training quantization for vision transformer. In *NeurIPS*, volume 34.

Lou, Q.; Guo, F.; Kim, M.; Liu, L.; and Jiang., L. 2020. AutoQ: Automated Kernel-Wise Neural Network Quantization. In *ICLR*.

Ma, Y.; Jin, T.; Zheng, X.; Wang, Y.; Li, H.; Wu, Y.; Jiang, G.; Zhang, W.; and Ji, R. 2023. Ompq: Orthogonal mixed precision quantization. In *AAAI*.

Ma, Y.; Li, H.; Zheng, X.; Ling, F.; Xiao, X.; Wang, R.; Wen, S.; Chao, F.; and Ji, R. 2024. Outlier-aware Slicing for Post-Training Quantization in Vision Transformer. In *ICML*.

Mitchell, S.; OSullivan, M.; and Dunning, I. 2011. Pulp: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, 65: 25.

Moon, J.; Kim, D.; Cheon, J.; and Ham, B. 2024. Instance-Aware Group Quantization for Vision Transformers. In *CVPR*.

Park, S.; Bae, J.; Kwon, B.; Kim, M.; Kim, B.; Kwon, S. J.; Kang, U.; and Lee, D. 2025a. Unifying Uniform and Binary-coding Quantization for Accurate Compression of Large Language Models. In *ACL*.

Park, S.; Choi, H.; and Kang, U. 2024. Accurate Retraining-free Pruning for Pretrained Encoder-based Language Models. In *ICLR*.

Park, S.; Lee, S.; Kim, J.; Lee, J.; Jo, H.; and Kang, U. 2025b. Accurate Sublayer Pruning for Large Language Models by Exploiting Latency and Tunability Information. In *IJCAI*.

Park, Y.; Hyun, J.; Cho, S.; Sim, B.; and Lee, J. W. 2024. Any-Precision LLM: Low-Cost Deployment of Multiple, Different-Sized LLMs. In *ICML*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Piao, T.; Cho, I.; and Kang, U. 2022. SensiMix: Sensitivity-Aware 8-bit index & 1-bit value mixed precision quantization for BERT compression. *PLOS ONE*, 17(4): 1–22.

Rakka, M.; Fouda, M. E.; Khargonekar, P.; and Kurdahi, F. 2024. A Review of State-of-the-Art Mixed-Precision Neural Network Frameworks. *IEEE TPAMI*.

Shin, J.; So, J.; Park, S.; Kang, S.; Yoo, S.; and Park, E. 2023. Nipq: Noise proxy-based integrated pseudo-quantization. In *CVPR*.

Sun, Z.; Ge, C.; Wang, J.; Lin, M.; Chen, H.; Li, H.; and Sun, X. 2022. Entropy-driven mixed-precision quantization for deep network design. In *NeurIPS*.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*.

Wang, M.; Meng, Y.; Tang, C.; Zhang, W.; Qin, Y.; Yao, Y.; Li, Y.; Feng, T.; Wang, X.; Guan, X.; et al. 2025. JAQ: Joint Efficient Architecture Design and Low-Bit Quantization with Hardware-Software Co-Exploration. In *AAAI*.

Wei, X.; Gong, R.; Li, Y.; Liu, X.; and Yu, F. 2022. QDrop: Randomly Dropping Quantization for Extremely Low-bit Post-Training Quantization. In *ICLR*.

Wightman, R. 2019. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models.

Wu, Z.; Chen, J.; Zhong, H.; Huang, D.; and Wang, Y. 2024. AdaLog: Post-Training Quantization for Vision Transformers with Adaptive Logarithm Quantizer. In *ECCV*.

Yao, Z.; Dong, Z.; Zheng, Z.; Gholami, A.; Yu, J.; Tan, E.; Wang, L.; Huang, Q.; Wang, Y.; Mahoney, M.; et al. 2021. Hawq-v3: Dyadic neural network quantization. In *ICML*.

Yu, C.; Chen, T.; Gan, Z.; and Fan, J. 2023. Boost vision transformer with gpu-friendly sparsity and quantization. In *CVPR*.

Yu, H.; Han, Q.; Li, J.; Shi, J.; Cheng, G.; and Fan, B. 2020. Search what you want: Barrier panelty nas for mixed precision quantization. In *ECCV*.

Yuan, Z.; Xue, C.; Chen, Y.; Wu, Q.; and Sun, G. 2022. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In *ECCV*.

Zhong, Y.; Hu, J.; Huang, Y.; Zhang, Y.; and Ji, R. 2024. ERQ: Error Reduction for Post-Training Quantization of Vision Transformers. In *ICML*.

Appendix

We provide additional information on LAMPQ including frequently used notations, detailed experimental setup, algorithm, theoretical analysis, and further experimental results.

Notation

We summarize the notations in the paper as Table 7.

Table 7: Notation description.

Symbol	Description
f_θ	Pre-trained model with parameters θ
$f_{\theta'}$	Quantized model with parameters θ'
$\mathcal{I} \in \mathbb{R}^{H\times W\times C}$	Input image
H	Height of the image $\mathcal{I}$
W	Width of the image $\mathcal{I}$
C	Channels of the image $\mathcal{I}$
P	Patch size
$L = HW/P^2$	Number of patches
N	Number of ViT blocks
s	Scaling factor in quantization
z	Integer offset in quantization
(r_{min}, r_{max})	Lower and upper bounds of a given matrix $\mathbf{M}$
$\mathbf{M}'$	Quantized matrix of $\mathbf{M}$
$\widehat{\mathbf{M}}$	Dequantized matrix of $\mathbf{M}'$
$\mathbf{W}_i$	Weight matrix of layer l_i in the model
$\mathbf{X}_i$	Activation of layer l_i in the model
$\Delta_{\mathbf{W}_i}, \Delta_{\mathbf{X}_i}$	Perturbations of $\mathbf{W}_i$ and $\mathbf{X}_i$ after quantization
B_i	Bit-width assigned to layer l_i
$\mathbf{H}_i$	Hessian matrix of layer l_i
$\mathbf{F}_i$	Fisher information matrix of layer l_i
tr$(\cdot)$	Trace operator
$\mathbb{E}(\cdot)$	Expectation function
Ω_i	Proxy of the loss $\mathcal{L}(\widehat{\mathbf{W}}_i)$ of layer l_i
α_{type}	Scaling factor for each type of layer
μ	Number of layers for α_{type} calculation
β	Bit-width for α_{type} calculation
$\mathbb{B}$	Pre-defined set of possible bit selections
γ	Hyperparameter controlling the degree of MPQ

Experimental Setup

We describe the experimental setup, including datasets, competitors, hyperparameters, and implementation details.

Datasets

We utilize ImageNet (ILSVRC 2012) (Deng et al. 2009) dataset to evaluate the classification accuracy of the quantized model obtained by LAMPQ. We evaluate quantized models on MS-COCO (Lin et al. 2014) dataset to check their performance on object detection. For zero-shot quantization task, we utilize real datasets (e.g. ImageNet) only for evaluation purposes.

Table 8: Hyperparameter ranges for LAMPQ.

Hyperparameter	Range
μ	$[\frac{1}{4}N, \frac{1}{2}N, \frac{3}{4}N, N]$
$\mathbb{B}$	[{1, 2, 3, 4, 5}, {1, 2, 3, 4}, {2, 3, 4, 5}, {2, 3, 4}]
γ	[2, 4, 6, 8, 10, 15, 20]
β	[1, 2, 3, 4]

Competitors

We summarize the competitors of LAMPQ as follows:

- **PTQ4ViT** (Yuan et al. 2022) observes the power-law distribution of post-Softmax activations and introduce twin uniform quantization to reduce the quantization error in such activation distributions.
- **APQ-ViT** (Ding et al. 2022) introduces bottom-elimination blockwise calibration to apply quantization in a blockwise manner, and Matthew-effect preserving quantization to maintain the power-law distribution of post-Softmax activations.
- **QDrop** (Wei et al. 2022) randomly drops the quantization of activation for smoother loss landscapes in test samples.
- **PD-Quant** (Liu et al. 2023a) mitigates the overfitting of quantized models toward calibration samples by considering global information from the prediction difference between quantized and full precision models.
- **RepQ-ViT** (Li et al. 2023) decouples the quantization and inference processes, by initially applying $\log\sqrt{2}$ quantization and reparameterizing the scales to hardware-friendly $\log 2$ quantization in inference stage.
- **OPTQ** (Frantar et al. 2023) is an approximate second-order method for quantizing Transformer-based models, by utilizing second-order information.
- **ERQ** (Zhong et al. 2024) reduces activation and weight quantization error by formulating the quantization process as a ridge regression problem.
- **AdaLog** (Wu et al. 2024) is a non-uniform quantizer that optimizes the logarithmic base to accommodate the power-law distribution of activations.

Additionally, we compare LAMPQ with PSAQ-ViT (Li et al. 2022d) for the zero-shot quantization experiment.

Hyperparameters

We conduct a grid search to validate hyperparameters, and select the set with the best performance. Table 8 reports the searched hyperparameter ranges of LAMPQ. For competitors, we search within the range described in each paper.

Implementation Details and Machine

We follow the settings from PSAQ-ViT (Li et al. 2022d), RepQ-ViT (Li et al. 2023), and AdaLog (Wu et al. 2024) for fair comparison. The size S of sample dataset $\mathbb{D}$ is 1,024 with batch size of 32. The average bit-width of each model is computed as the mean across all parameters, with each layer

Table 9: Expected values for Gaussian distribution $\mathcal{N}(0, 1)$.

B_i	$\mathbb{E}(X^2)$	$\mathbb{E}(X\Delta_X)$	$\mathbb{E}((\Delta_X)^2)$
1	1	1.396 E+0	5.212 E+0
2	1	1.655 E−2	3.359 E−1
3	1	7.123 E−4	6.109 E−2
4	1	1.723 E−4	1.330 E−2
5	1	4.123 E−5	3.113 E−3
6	1	1.003 E−5	7.538 E−4
7	1	2.472 E−6	1.855 E−4
8	1	6.134 E−7	4.601 E−5

assigned a uniform bit-width. We implement LAMPQ with PyTorch (Paszke et al. 2019) and timm (Wightman 2019) libraries in Python. We exploit the default solver provided by the PuLP (Mitchell, OSullivan, and Dunning 2011) library, which leverages CBC (COIN-OR Branch-and-Cut) (Forrest and Lougee-Heimer 2005) algorithm to solve the ILP problem. The ILP optimization completes within a few seconds on a standard CPU, thus we do not observe runtime bottlenecks. For the other methods, we reproduce the result using their open-source code if possible and implement them otherwise. All of our experiments are done at a workstation with Intel Xeon Silver 4310 and RTX 4090.

Theoretical Analysis

Proof of Lemma 2

Proof. We consider a statistical model defined by the conditional probability distribution $p(y \mid x, \theta)$, which gives the probability of observing the target y given an input x and is parameterized by the model parameters θ. When the loss function is the negative log-likelihood, the expected Hessian of layer l_i with respect to θ can be expressed as:

$$\begin{aligned}
&\mathbb{E}_{x\sim q(x)}\mathbb{E}_{y\sim p(y|x,\theta)}[\mathbf{H}_i] \\
&= \mathbb{E}_{x\sim q(x)}\Big[\mathbb{E}_{y\sim p(y|x,\theta)}\Big[-\nabla^2_\theta \log p\big(y \mid x,\theta\big)\Big]\Big] \\
&= \mathbb{E}_{x\sim q(x)}\Big[\mathbb{E}_{y\sim p(y|x,\theta)}\Big[-\frac{\nabla^2_\theta p(y \mid x,\theta)}{p(y \mid x,\theta)} \\
&\quad + \nabla_\theta \log p(y \mid x,\theta)\nabla_\theta \log p(y \mid x,\theta)^\top\Big]\Big] \\
&\approx \mathbb{E}_{x\sim q(x)}\Big[\mathbb{E}_{y\sim p(y|x,\theta)}\Big[-\frac{\nabla^2_\theta p(y \mid x,\theta)}{p(y \mid x,\theta)}\Big]\Big] + \mathbf{F}_i \\
&= \mathbb{E}_{x\sim q(x)}\Big[\sum_{y\in\mathcal{C}} -\frac{\nabla^2_\theta p(y \mid x,\theta)}{p(y \mid x,\theta)} p(y \mid x,\theta)\Big] + \mathbf{F}_i \\
&= \mathbb{E}_{x\sim q(x)}\Big[-\nabla^2_\theta \sum_{y\in\mathcal{C}} p(y \mid x,\theta)\Big] + \mathbf{F}_i \\
&= \mathbf{F}_i,
\end{aligned}$$

where $q(x)$ and $\mathcal{C}$ denote an empirical prior distribution regarding the dataset and the class set, respectively. The second equality follows from the derivative identity:

$$\begin{aligned}
-\nabla^2_\theta \log p\big(y \mid x,\theta\big) &= -\nabla_\theta\Big[\frac{\nabla_\theta p\big(y \mid x,\theta\big)}{p\big(y \mid x,\theta\big)}\Big] \\
&= -\frac{\nabla^2_\theta p(y \mid x,\theta)}{p(y \mid x,\theta)} \\
&\quad + \nabla_\theta \log p(y \mid x,\theta)\nabla_\theta \log p(y \mid x,\theta)^\top.
\end{aligned}$$

The approximation step assumes that the empirical prior distribution $q(x)$ is sufficiently close to the prior distribution. □

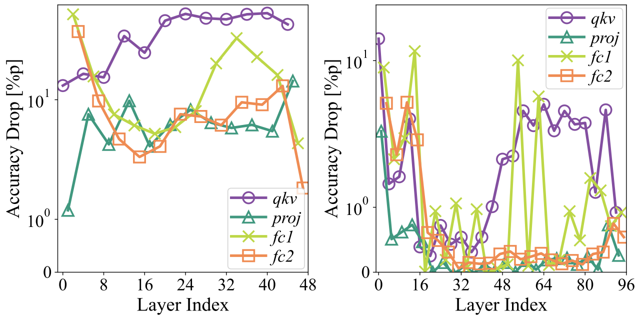
Proof of Lemma 3

Proof. We first calculate the expected value of the reconstruction error $\mathcal{L}^{(B_i)}_{recon}(l_i)$ for the i-th layer l_i with bit-width B_i. For weights $\mathbf{W}_i \in \mathbb{R}^{D'\times D}$ and activations $\mathbf{X}_i \in \mathbb{R}^{D\times L}$ of the given layer l_i, let $\Delta^{(B_i)}_{\mathbf{W}_i} = \widehat{\mathbf{W}}^{(B_i)}_i - \mathbf{W}_i$ and $\Delta^{(B_i)}_{\mathbf{X}_i} = \widehat{\mathbf{X}}^{(B_i)}_i - \mathbf{X}_i$ be the quantization errors of weights and activations, respectively. Then, the expected reconstruction error is obtained as follows:

$$\begin{aligned}
&\mathbb{E}\Big(\mathcal{L}^{(B_i)}_{recon}(l_i)\Big) \\
&= \frac{1}{\|\mathbf{W}_i\mathbf{X}_i\|^2_F}\mathbb{E}\big(\|(\mathbf{W}_i + \Delta^{(B_i)}_{\mathbf{W}_i})(\mathbf{X}_i + \Delta^{(B_i)}_{\mathbf{X}_i}) \\
&\qquad - \mathbf{W}_i\mathbf{X}_i\|^2_F\big) \\
&= \frac{1}{\|\mathbf{W}_i\mathbf{X}_i\|^2_F}\mathbb{E}\big(\|\Delta^{(B_i)}_{\mathbf{W}_i}\mathbf{X}_i + \mathbf{W}_i\Delta^{(B_i)}_{\mathbf{X}_i} \\
&\qquad + \Delta^{(B_i)}_{\mathbf{W}_i}\Delta^{(B_i)}_{\mathbf{X}_i}\|^2_F\big) \\
&= \frac{1}{\|\mathbf{W}_i\mathbf{X}_i\|^2_F}\mathbb{E}\Big(\sum_{r=1}^{D'}\sum_{c=1}^{L}\big((\Delta^{(B_i)}_{\mathbf{W}_i}\mathbf{X}_i)_{rc} + (\mathbf{W}_i\Delta^{(B_i)}_{\mathbf{X}_i})_{rc} \\
&\qquad + (\Delta^{(B_i)}_{\mathbf{W}_i}\Delta^{(B_i)}_{\mathbf{X}_i})_{rc}\big)^2\Big) \\
&= \frac{1}{\|\mathbf{W}_i\mathbf{X}_i\|^2_F}\mathbb{E}\Big(\sum_{r=1}^{D'}\sum_{c=1}^{L}\sum_{k=1}^{D}\big((\Delta^{(B_i)}_{\mathbf{W}_i})_{rk}(\mathbf{X}_i)_{kc} \\
&\qquad + (\mathbf{W}_i)_{rk}(\Delta^{(B_i)}_{\mathbf{X}_i})_{kc} \\
&\qquad + (\Delta^{(B_i)}_{\mathbf{W}_i})_{rk}(\Delta^{(B_i)}_{\mathbf{X}_i})_{kc}\big)^2\Big).
\end{aligned}$$

We further simplify the summation by the assumption that the weights and activations are independent, which allows us to separate the expectations of the individual terms. Remind that random variables $W, X, \Delta^{(B_i)}_W$, and $\Delta^{(B_i)}_X$ are distributed as elements of $\mathbf{W}_i, \mathbf{X}_i, \Delta^{(B_i)}_{\mathbf{W}_i}$, and $\Delta^{(B_i)}_{\mathbf{X}_i}$, respectively. Then, the expected value of the reconstruction error is expressed as follows:

$$\begin{aligned}
&\mathbb{E}\Big(\mathcal{L}^{(B_i)}_{recon}(l_i)\Big) \\
= \;&\frac{D'LD}{\|\mathbf{W}_i\mathbf{X}_i\|^2_F}\mathbb{E}\Big(\big(\Delta^{(B_i)}_W X + W\Delta^{(B_i)}_X + \Delta^{(B_i)}_W\Delta^{(B_i)}_X\big)^2\Big).
\end{aligned}$$



a ViT-S b Swin-S

Figure 8: Impact on accuracy when quantizing a single layer of the 32-bit (a) ViT-S and (b) Swin-S models to 1-bit following AdaLog (Wu et al. 2024) while keeping the other layers unchanged. Sensitivity towards quantization varies significantly across layers.

Note that the term $\frac{D'LD}{\|\mathbf{W}_i\mathbf{X}_i\|_F^2}$ is a constant factor that is independent of the quantization bit-width B_i. Hence, the ratio between the expected reconstruction errors for different bit-widths B_i and $B_i - 1$ is obtained as follows:

$$\frac{\mathbb{E}(\mathcal{L}_{recon}^{(B_i-1)}(l_i))}{\mathbb{E}(\mathcal{L}_{recon}^{(B_i)}(l_i))} = \frac{\mathbb{E}\Big(\big(\Delta_W^{(B_i-1)}X + W\Delta_X^{(B_i-1)} + \Delta_W^{(B_i-1)}\Delta_X^{(B_i-1)}\big)^2\Big)}{\mathbb{E}\Big(\big(\Delta_W^{(B_i)}X + W\Delta_X^{(B_i)} + \Delta_W^{(B_i)}\Delta_X^{(B_i)}\big)^2\Big)}. \tag{1}$$

Thus, we obtain the lemma as stated, finishing the proof. □

Calculating the Relative Reconstruction Error

We calculate the Equation (1) to obtain the relative reconstruction error for each bit width B_i. Note that weight W and activation X are independent and both follows a Gaussian distribution. Since the scale of each distribution cancels out in the ratio in Equation (1), we consider only the standard normal distribution $\mathcal{N}(0, 1)$ for the Gaussian distribution. We assume that the clipping range of quantization is between -3σ and 3σ. Then, the expectation value of Equation (1) is expressed as follows since the weights and the activations are independent.

$$\begin{aligned}
&\mathbb{E}\Big(\big(\Delta_W^{(B_i)}X + W\Delta_X^{(B_i)} + \Delta_W^{(B_i)}\Delta_X^{(B_i)}\big)^2\Big) \\
&= \mathbb{E}\big((\Delta_W^{(B_i)})^2\big)\mathbb{E}\big(X^2\big) + \mathbb{E}\big(W^2\big)\mathbb{E}\big((\Delta_X^{(B_i)})^2\big) \\
&+ \ \mathbb{E}\big((\Delta_W^{(B_i)})^2\big)\mathbb{E}\big((\Delta_X^{(B_i)})^2\big) + 2\,\mathbb{E}\big(W\Delta_W^{(B_i)}\big)\mathbb{E}\big(X\Delta_X^{(B_i)}\big) \\
&+ 2\,\mathbb{E}\big(W\Delta_W^{(B_i)}\big)\mathbb{E}\big((\Delta_X^{(B_i)})^2\big) + 2\,\mathbb{E}\big((\Delta_W^{(B_i)})^2\big)\mathbb{E}\big(X\Delta_X^{(B_i)}\big)
\end{aligned}$$

We compute the value of each term numerically for each bit width B_i, as summarized in Table 9. Then, we obtain the relative reconstruction errors for each bit width.

Algorithm

Algorithm 1: Quantization procedure of LAMPQ

```
Input: A pre-trained full-precision ViT model $f_\theta$ with $N$
    blocks, target bit-width $b_t$, sample dataset $\mathbb{D}$, and hyper-
    parameters $\beta$, $\gamma$, $\mu$, and $\mathbb{B}$.
Output: Quantized ViT model $f_{\theta'}$ within the $b_t$ bit limit
    /* Calculation of type-aware Fisher-based metric */
Assess the initial accuracy of the pre-trained model
Randomly sample $\mu$ blocks as $\mathbb{L}_\mu$
// Obtain scaling factors $\alpha_t$ for each layer type $t$
for each type $t$ of layer do
    Calculate the average accuracy loss $A^{(t)}$ compared
      to the initial accuracy of the pre-trained model
      when quantizing a layer of type $t$ in $\mathbb{L}_\mu$ to $\beta$ bit
    Calculate the average trace tr$(\mathbf{F}^{(t)})$ of Fisher
      information matrix of the quantized layer when
      quantizing a layer of type $t$ in $\mathbb{L}_\mu$ to $\beta$ bit
    Obtain the average accuracy loss per sensitivity
      $\alpha_t$ as $A^{(t)}/$tr$(\mathbf{F}^{(t)})$
end for
for each layer $l$ in $f_\theta$ do
    Estimate error $\Omega_l$ with scaling factor $\alpha_t$
end for

/* Bit allocation process */
// Step 1: Initial assignment
Initialize the bit allocation $\{B_i^*\}_{i=1}^{4N}$ for each layer by
  solving an ILP problem
Quantize $f_\theta$ to $f_{\theta'}$ with $\{B_i^*\}_{i=1}^{4N}$

// Step 2: Error-based iterative updates
while True do
    for each layer $l$ in $f_{\theta'}$ do
        Obtain the current reconstruction loss $\mathcal{L}_{recon}^{(B_i)}(l)$
      // Estimation using Table 2
        Estimate the accuracy gain $L_l^{(up)}$ when
        increasing the bit-width of $l$ by 1
        Estimate the accuracy loss $L_l^{(down)}$ when
        decreasing the bit-width of $l$ by 1
    end for
    Find the layer $u$ with the highest accuracy gain
    Find the layer $d$ with the lowest accuracy loss
    Obtain $f_{\theta^{cal}}$ by increasing the bit-width of $u$ by 1
      and decreasing the bit-width of $d$ by 1 from $f_{\theta'}$
    if $f_{\theta^{cal}}$ is less accurate than $f_{\theta'}$ then
        break
    end if
    $f_{\theta'} \leftarrow f_{\theta^{cal}}$
end while
return $f_{\theta'}$
```

We present the overall algorithm for LAMPQ in Algorithm 1. Note that any ViT quantization technique whose quantization granularity is not larger than a layer such as RepQ-ViT (Li et al. 2023), ERQ (Zhong et al. 2024), and

Table 10: Runtime analysis between AdaLog (Wu et al. 2024), VT-PTQ (Liu et al. 2021b), and LAMPQ (Proposed). hrs. and sec. indicate hours and seconds, respectively. LAMPQ is up to 335× faster than VT-PTQ.

Method	**W/A**	**ViT-S**	**DeiT-S**	**Swin-S**
AdaLog (2024)	3/3	165.3 sec.	278.9 sec.	648.8 sec.
VT-PTQ (2021b)†	3MP/3MP	23.8 hrs.	40.2 hrs.	93.5 hrs.
LAMPQ (Proposed)	3MP/3MP	323.6 sec.	484.5 sec.	1002.6 sec.

†: AdaLog quantization with bit allocation by VT-PTQ.

AdaLog (Wu et al. 2024) is applicable to quantize the model according to the bit allocation found by LAMPQ.

Further Experiments

Layer-wise Sensitivity of Different Models

We empirically analyze that layer-wise sensitivity to quantization varies significantly, making module-based mixed-precision approaches ineffective for optimal performance. We investigate the layer-wise sensitivity for other ViT variants, such as ViT-S (Dosovitskiy et al. 2021) and Swin-S (Liu et al. 2021a), and present the results in Figure 8. As shown in the figure, the quantization sensitivity differs greatly between layers, regardless of the model type. This observation confirms that existing module-wise MPQ approaches fail to capture this layer-wise sensitivity, leading to suboptimal performance.

Runtime Analysis

We analyze the computational overhead of LAMPQ by measuring the quantization time of Algorithm 1. Table 10 compares the total quantization time between AdaLog (Wu et al. 2024), VT-PTQ (Liu et al. 2021b), and LAMPQ. Note that AdaLog relies on uniform precision, while others are MPQ methods. The results indicate that LAMPQ consistently achieves speedups of 250× or more, with maximum gains of 335× over VT-PTQ. Compared to VT-PTQ, which assigns bits by exploring a Pareto frontier and testing hundreds of cases, LAMPQ initializes bit assignments with a type-aware Fisher-based metric and iteratively updates them based on estimated reconstruction error, significantly reducing computational overhead.