TRPQUANT: TOKEN REARRANGEMENT AND CHANNEL PRUNING FOR QUANTIZING LARGE VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

High memory and computation costs prevent the practical edge deployment of vision-language models (VLMs). Post-training quantization (PTQ) offers a promising solution without expensive retraining. Given the visual and textual inputs in VLMs, existing methods typically adopt modality-level token partitioning and processing. However, they treat tokens uniformly within each modality, overlooking intra-modality activation distribution differences. This coarse-grained strategy leads to degraded accuracy and redundant computation under low-bit quantization. We propose **TRPQuant**, a new framework that addresses these limitations through **T**oken **R**earrangement and Channel **P**runing, enabling fine-grained token grouping and computation reduction. Specifically, TRPQuant classifies tokens into Dense, Intermediate, and Sparse groups based on their activation patterns and applies tailored quantization strategies accordingly: dense tokens preserve full channels and go through the standard branch, sparse tokens are routed to the sparse branch with channel pruning to reduce computation, and intermediate tokens combine both mechanisms via dual-branch processing. Extensive experiments on mainstream VLMs demonstrate that TRPQuant achieves strong quantization accuracy. Notably, TRPQuant preserves 94.1% of FP16 performance under the challenging W3A3 setting (69.9 vs. 74.3) on Qwen2.5-VL-7B, pushing the frontier for ultra-low-bit quantization.

1 INTRODUCTION

Vision-language models (Bai et al., 2023; Liu et al., 2023) have demonstrated strong capabilities in understanding and reasoning over multimodal inputs (Antol et al., 2015; Li & Jia, 2016; You et al., 2016; Herdade et al., 2019; Lu et al., 2022; Yang et al., 2025), enabling applications ranging from visual question answering to embodied AI (Ma et al., 2026). However, their growing scale poses significant challenges for deployment on edge devices and resource-constrained environments. Post-training quantization (PTQ) compresses pretrained models by converting floating-point weights and activations into low-bit representations, reducing memory footprint and accelerating inference without costly retraining.

Despite its general effectiveness, applying PTQ to VLMs remains highly challenging due to the heterogeneous activation distributions. Existing VLM PTQ methods typically partition tokens by modality and apply separate quantization strategies to vision and text tokens (Li et al., 2025; Yu et al., 2025; Hu et al., 2026). Although effective, these methods overlook a crucial consideration: tokens within the same modality may exhibit significantly different activation patterns. After carefully studying the distributions of vision and text tokens, we have the following observations. Most text tokens show dense activations with high values across many channels, yet a minority display sparse, long-tailed distributions dominated by a few high-magnitude channels. In contrast, most vision tokens exhibit sparse distributions, while a subset shows dense activation patterns.

Ignoring such distributional differences and treating tokens based solely on their modality can lead to accuracy degradation under aggressive quantization. In addition, since modality-level partitioning is coarse-grained, most methods still perform full-channel quantization computation for all tokens, resulting in redundant computation and extra quantization errors.

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

To solve these problems, we propose **TRPQuant**, a token-adaptive PTQ approach for VLMs. At its core, TRPQuant introduces **T**oken **R**earrangement (**TR**), which moves beyond coarse modality-level partitioning by identifying a fixed active-channel set per layer and grouping tokens into *Dense*, *Intermediate*, and *Sparse* groups according to their distribution characteristics. Dense tokens follow the standard quantization path to preserve cross-channel dependencies, in contrast sparse tokens are routed to a sparse path equipped with the **C**hannel **P**runing (**CP**), which prunes inactive channels to reduce computation. Intermediate tokens are decomposed into active and inactive components and processed through both paths, preserving high-energy outliers and distributed features. By adapting the quantization paths and computation to each token group, TRPQuant achieves a favorable trade-off between efficiency and accuracy.

Overall, our contributions are summarized as follows:

- We identify substantial intra-modality activation heterogeneity in VLMs, where tokens within the same modality show distinct patterns beyond modality partitioning.
- We propose TRPQuant, a novel quantization framework that includes token rearrangement and channel pruning.
- We obtain strong and competitive results on Qwen2.5-VL, Qwen3-VL, and LLaVA-v1.5 across diverse benchmarks. TRPQuant delivers near-FP16 accuracy at W4A4 and remains robust under W3A3 quantization. On Qwen3-VL-8B, it scores 69.1 on average at W3A3, compared with 74.1 for FP16.

2 Related Work

2.1 LLM Quantization

Quantization is widely used to reduce the storage and inference cost of large language models. Depending on when the low-precision constraint is introduced, existing methods can be categorized into quantization-aware training (QAT) (Bondarenko et al., 2024; Chen et al., 2025; Liu et al., 2024b) and post-training quantization (PTQ) (Song et al., 2025; Wang et al., 2026a; Sharify et al., 2024). Recent QAT methods improve low-bit training from different perspectives. For example, Attn-QAT (Zhang et al., 2026) introduces FP4 attention-specific QAT to address the precision inconsistency in FlashAttention. Dremov et al. (2025) derive scaling laws to predict the optimal compute split between full-precision training and QAT under fixed budgets; BBQ (Yang & Aamodt, 2026) proposes an information-theoretic QAPT scheme for hardware-friendly low-precision formats. Compared with QAT, PTQ is more deployment-friendly since it does not require retraining. Early PTQ methods mainly focus on weight-only quantization while keeping activations in high precision (Dettmers et al., 2023; Lin et al., 2024b; Zheng et al., 2024; Hu et al., 2025a; Li et al., 2024a; Xu et al., 2025). Recently, weight-activation quantization has become a practical but challenging setting (Ma et al., 2024; Zhao et al., 2025; Li et al., 2023b; Hu et al., 2024; 2025b; Wei et al., 2022; Zhang et al., 2024; Yao et al., 2023). Representative methods alleviate quantization difficulty through smoothing transformations (Xiao et al., 2023), learnable clipping and equivalent transformations (Shao et al., 2023), or rotations (Liu et al., 2024c; Ashkboos et al., 2024).

2.2 VLM Quantization

Extending quantization from LLMs to VLMs is inherently challenging due to the heterogeneous statistics of vision and text tokens, as well as the markedly different quantization sensitivities of visual and language modules. Similar to vanilla LLM quantization, existing VLM quantization methods mainly follow QAT or PTQ. SPEED-Q (Guo et al., 2026) proposes a staged distillation-based QAT scheme to greatly reduce the sensitivity gap between visual and language modules. Recent PTQ methods exploit fine-grained token or modality differences carefully (Xie et al., 2024; Yu et al., 2025; Wang et al., 2026b). For instance, VLMQ (Xue et al., 2025) uses gradient-based token importance to preserve salient visual information, VLM-PTQ (Deng & Huang, 2026) improves Hessian-based weight compensation, MBQ (Li et al., 2025) balances quantization reconstruction loss according to the gradient sensitivities of visual and language modalities, and MASQuant (Hu et al., 2026) introduces modality-aware smoothing for intrinsically heterogeneous modalities. SplitQ (Zhong et al., 2026) addresses cross-modal heterogeneity through modality-specific outlier-channel decoupling and adaptive cross-modal calibration. QIG (Xiang et al., 2026) uses token-level sensitivity to

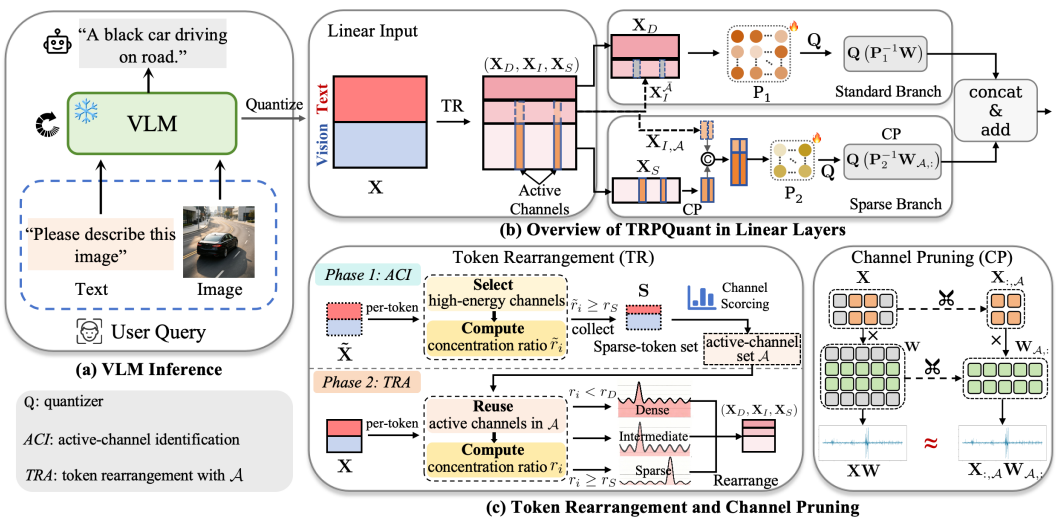


Figure 1: Overview of TRPQuant. **(a)** VLM inference pipeline. **(b)** TRPQuant in a linear layer: **T**oken **R**earrangement (TR) classifies input tokens into **Dense**, **Intermediate**, and **Sparse** groups through *ACI* and *TRA*; Dense tokens follow the standard branch; Sparse tokens undergo **C**hannel **P**runing (CP) and follow the sparse branch; Intermediate tokens are processed by both branches. Their outputs are then aggregated to produce the layer output. **(c)** Detailed processes of TR and CP. $\mathbf{P}_1$ and $\mathbf{P}_2$ denote the full-space (all channels) and subspace (only active channels) transformations.

weight calibration reconstruction errors, whereas Quant Experts (Jia et al., 2026) uses shared and routed low-rank experts for error compensation. TRPQuant instead uses activation energy to classify tokens into three groups: dense, intermediate, and sparse. This routing also enables inactive-channel pruning for sparse tokens, explicitly coupling token adaptation with computation reduction.

3 Preliminaries

Quantization maps high-precision tensors to low-precision representations. For uniform integer quantization (Jacob et al., 2018), given a floating-point tensor $\mathbf{X}$, the N-bit quantization-dequantization operator is defined as:

$$Q(\mathbf{X}) = \left(\mathrm{clamp}\left(\lfloor \mathbf{X}/\Delta \rceil + z, q_{\min}, q_{\max}\right) - z\right)\Delta, \tag{1}$$

where $\Delta = \frac{\max(|\mathbf{X}|)}{2^{N-1}-1}$ is the scaling factor, z is the zero-point, $\lfloor \cdot \rceil$ denotes the round-to-nearest operator, and $\mathrm{clamp}(\cdot)$ clips values to $[q_{\min}, q_{\max}]$. Consider a linear layer $\mathbf{Y} = \mathbf{X}\mathbf{W}$, where $\mathbf{X} \in \mathbb{R}^{T \times D_{\text{in}}}$ and $\mathbf{W} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$ denote the input tensor and weight matrix, respectively. The quantization of this layer can be represented as: $\widehat{\mathbf{Y}} = Q(\mathbf{X})Q(\mathbf{W})$.

Since $\mathbf{X}$ and $\mathbf{W}$ may contain outliers across channels, recent methods (Xiao et al., 2023; Sun et al., 2024) introduce an invertible matrix $\mathbf{P} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{in}}}$, given by:

$$\widehat{\mathbf{Y}} = Q(\mathbf{X}\mathbf{P})Q(\mathbf{P}^{-1}\mathbf{W}), \tag{2}$$

where $\mathbf{P}$ smooths or redistributes channels between activations and weights to improve quantization friendliness. The transformation $\mathbf{P}$ is typically optimized on calibration data by minimizing MSE between $\widehat{\mathbf{Y}}$ and $\mathbf{Y}$. For simplicity, we define $\mathrm{W}x\mathrm{A}y$ to denote x-bit weights and y-bit activations.

4 Methodology

4.1 Overview

TRPQuant is a PTQ framework for VLMs that addresses the limitations of modality-level partitioning in quantization. Instead of uniformly processing tokens within each modality, TRPQuant groups and rearranges tokens based on their energy concentration. This enables fine-grained, token-specific quantization paths tailored to their activation characteristics.

Without loss of generality, consider a linear layer with input activation $\mathbf{X} \in \mathbb{R}^{T \times D_{\text{in}}}$ and weight matrix $\mathbf{W} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$, where $\mathbf{X}$ contains T vision and text tokens. As illustrated in Fig. 1(b), the proposed TRPQuant processes this layer through the **T**oken **R**earrangement (**TR**) module, which classifies tokens into three groups based on their activation patterns: **Dense**, **Intermediate**, and **Sparse**. Tokens of each group are then rearranged to enable branch-specific quantized computation:

- **Dense** tokens exhibit broadly distributed activations across many channels. These tokens are processed by the *standard branch*, which applies a full-space transformation $\mathbf{P}_1$ to adapt the full-channel activations and weights.
- **Sparse** tokens concentrate most activation energy predominantly in a small subset of channels, and are routed to the *sparse branch*. Inactive channels of sparse tokens are first removed by **C**hannel **P**runing (**CP**) to effectively reduce computation. These tokens are further processed by transformation-based quantization with a learnable transformation matrix $\mathbf{P}_2$ on active-channel subspace.
- **Intermediate** tokens exhibit mixed characteristics: their active channels are processed by the sparse branch, while the remaining ones are processed by the standard branch.

The outputs from both branches are finally aggregated to produce the layer output. Fig. 1(c) further illustrates the detailed processes of TR and CP.

4.2 Token Rearrangement

Motivation. VLMs process two distinct input modalities. Accordingly, prior methods often adopt modality-level token partitioning, which first separates vision and text tokens into different groups and then applies modality-specific strategies to improve the performance of PTQ (Li et al., 2025; Yu et al., 2025; Hu et al., 2026). However, such coarse partitioning overlooks significant intra-modality differences in activation patterns. As illustrated in Fig. 2, we observe three representative phenomena: ① The majority of text tokens exhibit dense distributions, characterized by relatively high responses across many channels. In contrast, most vision tokens show sparse activations dominated by a few high-magnitude channels, with the remaining channels close to zero. ② A minority of tokens from both modalities display intermediate activation patterns lying between the typical dense and sparse patterns. ③ A small subset of tokens exhibits an activation pattern resembling that of the majority of tokens in the other modality, e.g., text tokens with sparse patterns. Table 1 summarizes modality-wise token proportions of three groups across all transformer blocks of various VLMs. The trends in all three models are consistent with the local observation in Fig. 2. These findings indicate that modality labels provide only a coarse descriptor for token-level distribution, whereas channel energy concentration provides a more precise criterion for distinguishing tokens.

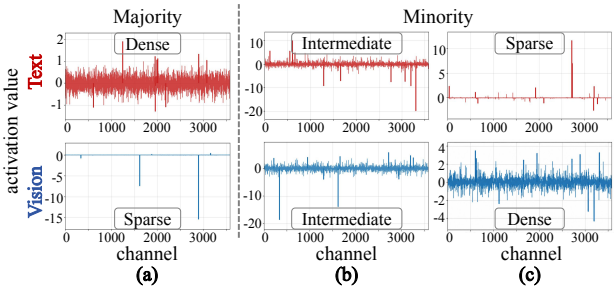


Figure 2: Activation distributions of the 11th transformer block of Qwen2.5-VL-7B. **(a)** The majority (86%) of text and vision tokens exhibit dense and sparse patterns, respectively. **(b)** A minority (9%) of tokens from both modalities show intermediate patterns. **(c)** A small subset (5%) of tokens show atypical patterns, i.e., sparse pattern for text tokens and dense pattern for vision tokens.

Motivated by this insight, we propose Token Rearrangement (TR), which categorizes tokens based on their channel-wise energy concentration. Compared with modality-level partitioning, TR produces more homogeneous token groups, thereby reducing intra-group distribution variance and facilitating subsequent branch-specific quantization.

Table 1: Proportions of dense, intermediate, and sparse tokens within each modality across all Transformer blocks of three models.

Model	Modality	Dense	Interm.	Sparse
Qwen2.5-VL-7B	Text	**85.0**%	6.7%	8.3%
	Vision	7.5%	7.2%	**85.3**%
Qwen3-VL-8B	Text	**87.1**%	8.0%	4.9%
	Vision	5.3%	10.5%	**84.2**%
LLaVA-v1.5-7B	Text	**80.3**%	13.1%	6.6%
	Vision	15.4%	20.5%	**64.1**%

TR Process. Token Rearrangement consists of two phases. ① Phase 1: runs once before PTQ calibration to obtain the fixed active-channel set $\mathcal{A}$ from activations on the calibration set. ② Phase 2:

runs during both PTQ calibration and inference, reusing the fixed $\mathcal{A}$ to classify tokens and rearrange them into Dense, Intermediate, and Sparse groups.

Phase 1: active-channel identification (ACI). The goal of ACI is to identify a fixed channel set $\mathcal{A}$ that contains active channels carrying dominant energy of the tokens. Using a fixed $\mathcal{A}$ provides a unified channel subspace for measuring token-wise energy concentration during both calibration and inference, which facilitates the subsequent branch-specific quantization strategy.

Before PTQ calibration, we perform a full-precision forward pass on the calibration set and randomly collect activation tokens $\tilde{\mathbf{X}} \in \mathbb{R}^{T \times D_{\text{in}}}$ for a linear layer. For the i-th sampled token $\tilde{\mathbf{X}}_{i,:}$, we first select its high-energy channels $\mathcal{H}_i$ by taking the top-ρ fraction of channels ranked by squared activation magnitude:

$$\mathcal{H}_i = \text{Top}_\rho \left(\left\{ \tilde{\mathbf{X}}_{i,j}^2 \right\}_{j=1}^{D_{\text{in}}} \right), \tag{3}$$

where $\text{Top}_\rho(\cdot)$ returns the indices of the largest ρD_{in} elements, and ρ is a small channel ratio. We then compute a token-wise energy concentration ratio: $\tilde{r}_i = \frac{\sum_{j \in \mathcal{H}_i} \tilde{\mathbf{X}}_{i,j}^2}{\sum_{j=1}^{D_{\text{in}}} \tilde{\mathbf{X}}_{i,j}^2}$. A *high* concentration ratio $\tilde{r}_i$ means that the token energy is concentrated in a few token-specific channels, indicating a *sparse* activation pattern. Tokens with $\tilde{r}_i \geq r_S$ are collected into a sparse-token set $\mathbf{S} \in \mathbb{R}^{K \times D_{\text{in}}}$, where K denotes the number of tokens being selected and $K < T$.

For each channel j, we compute its importance by the maximum squared activation over the sparse-token set $\mathbf{S}$:

$$g_j = \max_{i \in [0,K)} \mathbf{S}_{i,j}^2, \quad j = 1, \ldots, D_{\text{in}}. \tag{4}$$

The layer-wise active-channel set $\mathcal{A}$ is then defined as: $\mathcal{A} = \text{Top}_\rho \left(\{g_j\}_{j=1}^{D_{\text{in}}} \right)$.

Phase 2: token rearrangement with $\mathcal{A}$ (TRA). During PTQ calibration and inference, TR categorizes each input token using the fixed active-channel set $\mathcal{A}$. For the i-th token $\mathbf{X}_{i,:}$, we calculate the proportion of activation energy concentrated within the predefined active channels: $r_i = \frac{\sum_{j \in \mathcal{A}} \mathbf{X}_{i,j}^2}{\sum_{j=1}^{D_{\text{in}}} \mathbf{X}_{i,j}^2}$. Tokens satisfying $r_i < r_D$, $r_D \leq r_i < r_S$, and $r_i \geq r_S$ are classified and rearranged into $(\mathbf{X}_D, \mathbf{X}_I, \mathbf{X}_S)$ for subsequent branch-specific quantization. The thresholds r_D and r_S are hyperparameters. Notably, the fixed set $\mathcal{A}$ containing high-energy and important channels, also serves as the retained channel set in the subsequent channel pruning strategy.

4.3 Channel Pruning

Motivation. Compared with weights, activations in VLMs generally exhibit a wider dynamic range and stronger channel-wise variations. This activation–weight distribution difference is particularly evident for sparse tokens, as illustrated in Fig. 3. Since the weight values vary within a relatively narrow range, the token activation pattern becomes a key factor in determining the multiplication result of activation and weight. For sparse tokens identified by TR, only a few active channels dominate the output, whereas many inactive channels contribute little. This motivates pruning the inactive channels of sparse tokens to eliminate redundant computation while preserving the dominant contribution from active channels.

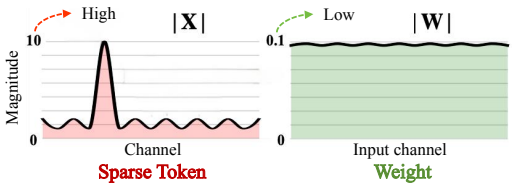


Figure 3: Distribution difference between sparse-token activation (left) and weight (right).

Theory. Recall that $\mathcal{A}$ denotes the set of active channel indices. Let $\bar{\mathcal{A}}$ be its complement, containing the indices of *inactive* channels. For the i-th token $\mathbf{X}_{i,:}$, the linear output can be decomposed into active- and inactive-channel contributions as

$$\mathbf{X}_{i,:}\mathbf{W} = \mathbf{X}_{i,\mathcal{A}}\mathbf{W}_{\mathcal{A},:} + \mathbf{X}_{i,\bar{\mathcal{A}}}\mathbf{W}_{\bar{\mathcal{A}},:}, \tag{5}$$

where $\mathbf{X}_{i,\mathcal{A}}$ and $\mathbf{W}_{\mathcal{A},:}$ are the sub-matrices of $\mathbf{X}_{i,:}$ and $\mathbf{W}$ corresponding to the active channels in $\mathcal{A}$. The terms $\mathbf{X}_{i,\bar{\mathcal{A}}}$ and $\mathbf{W}_{\bar{\mathcal{A}},:}$ are defined analogously for the inactive channels.

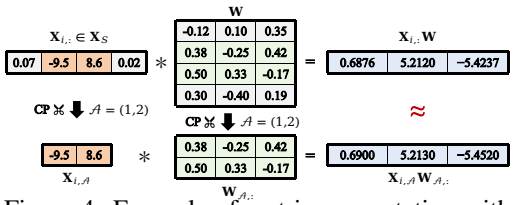


Figure 4: Example of matrix computation with channel pruning (CP) for a sparse token. Matrix computation *w.* or *w.o.* CP gives similar results.

Discarding the inactive-channel contribution introduces the following approximation error:

$$\|\mathbf{X}_{i,:}\mathbf{W} - \mathbf{X}_{i,\mathcal{A}}\mathbf{W}_{\mathcal{A},:}\|_2 = \left\|\mathbf{X}_{i,\bar{\mathcal{A}}}\mathbf{W}_{\bar{\mathcal{A}},:}\right\|_2 \leq \left\|\mathbf{X}_{i,\bar{\mathcal{A}}}\right\|_2 \left\|\mathbf{W}_{\bar{\mathcal{A}},:}\right\|_2, \tag{6}$$

where $\|.\|_2$ denotes the standard spectral norm. This bound shows that the pruning error is jointly controlled by inactive-channel activation energy and the norm of the associated weight submatrix.

According to the energy concentration ratio r_i, the inactive-channel activation energy satisfies

$$\left\|\mathbf{X}_{i,\bar{\mathcal{A}}}\right\|_2^2 = (1 - r_i)\,\|\mathbf{X}_{i,:}\|_2^2. \tag{7}$$

For a sparse token $\mathbf{X}_{i,:} \in \mathbf{X}_S$, we have $r_i \geq r_S$. Therefore,

$$\|\mathbf{X}_{i,:}\mathbf{W} - \mathbf{X}_{i,\mathcal{A}}\mathbf{W}_{\mathcal{A},:}\|_2 \leq \sqrt{1 - r_S}\,\|\mathbf{X}_{i,:}\|_2 \left\|\mathbf{W}_{\bar{\mathcal{A}},:}\right\|_2. \tag{8}$$

When the activation energy of a sparse token is highly concentrated in $\mathcal{A}$ (i.e., $r_S \approx 1$), its inactive-channel contribution has a limited influence on the output. We therefore approximate its output by retaining only the active-channel contribution:

$$\mathbf{X}_{i,:}\mathbf{W} \approx \mathbf{X}_{i,\mathcal{A}}\mathbf{W}_{\mathcal{A},:}. \tag{9}$$

As illustrated in Fig. 4, CP implements this approximation by retaining only the active activation channels and corresponding rows of the weight matrix, thereby greatly reducing the overall computation. In the following, we formally use $\mathbf{X}_{S,\mathcal{A}}$ to denote the *pruned* sparse tokens with active channels $\mathcal{A}$ being selected from $\mathbf{X}_S$.

4.4 Branch-specific Quantization

After TR and CP, the standard and sparse branches operate in two distinct channel spaces. The standard branch preserves the full D_{in}-dimensional channel space, whereas the sparse branch performs computation only in the $|\mathcal{A}|$-dimensional active-channel subspace. We therefore introduce specific transformations to separately adapt the activation/weight distributions in the two channel spaces.

According to the transformation-based quantization in Eq. 2, we assign a full-space transformation $\mathbf{P}_1 \in \mathbb{R}^{D_{\text{in}} \times D_{\text{in}}}$ to the standard branch and a subspace transformation $\mathbf{P}_2 \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{A}|}$ to the sparse branch. For dense tokens $\mathbf{X}_D$, the quantized output is computed through the standard branch as

$$\widehat{\mathbf{Y}}_D = Q(\mathbf{X}_D\mathbf{P}_1)\,Q\big(\mathbf{P}_1^{-1}\mathbf{W}\big). \tag{10}$$

For pruned sparse tokens $\mathbf{X}_{S,\mathcal{A}}$, the quantized output is computed in the active-channel subspace as

$$\widehat{\mathbf{Y}}_S = Q(\mathbf{X}_{S,\mathcal{A}}\mathbf{P}_2)\,Q\big(\mathbf{P}_2^{-1}\mathbf{W}_{\mathcal{A},:}\big). \tag{11}$$

Intermediate tokens exhibit mixed characteristics of dense and sparse tokens. To deal with this distribution, we route them through the two branches separately. Let $\mathbf{M}_{\mathcal{A}} \in \{0,1\}^{D_{\text{in}}}$ denote the binary mask of the active channels and let $\mathbf{M}_{\bar{\mathcal{A}}} = \mathbf{1} - \mathbf{M}_{\mathcal{A}}$. We define the full-dimensional residual component as $\mathbf{X}_I^{\bar{\mathcal{A}}} = \mathbf{X}_I \odot \mathbf{M}_{\bar{\mathcal{A}}}$, where $\odot$ denotes element-wise multiplication with $\mathbf{M}_{\bar{\mathcal{A}}}$ broadcast across intermediate tokens. Before quantization, the output of intermediate tokens can be decomposed as $\mathbf{X}_I\mathbf{W} = \mathbf{X}_I^{\bar{\mathcal{A}}}\mathbf{W} + \mathbf{X}_{I,\mathcal{A}}\mathbf{W}_{\mathcal{A},:}$. Accordingly, the residual part $\mathbf{X}_I^{\bar{\mathcal{A}}}$ is processed by the standard branch, while the active-channel contribution is processed by the sparse branch:

$$\widehat{\mathbf{Y}}_I = Q\Big(\mathbf{X}_I^{\bar{\mathcal{A}}}\mathbf{P}_1\Big)\,Q\big(\mathbf{P}_1^{-1}\mathbf{W}\big) + Q(\mathbf{X}_{I,\mathcal{A}}\mathbf{P}_2)\,Q\big(\mathbf{P}_2^{-1}\mathbf{W}_{\mathcal{A},:}\big). \tag{12}$$

5 Experiments

5.1 Experimental Setups

We evaluate TRPQuant on six VLMs, including Qwen2.5-VL (Bai et al., 2025b) (3B/7B), Qwen3-VL (Bai et al., 2025a) (2B/8B), and LLaVA-v1.5 (Liu et al., 2024a) (7B/13B). Following recent works (Wang et al., 2024; Li et al., 2025; Hu et al., 2026; Deng & Huang, 2026), we quantize only the LLM component. We compare TRPQuant with relevant and competitive methods including RTN (Nagel et al., 2020), GPTQ (Frantar et al., 2022), FlatQuant (Sun et al., 2024), SmoothQuant (Xiao et al., 2023), MBQ (Li et al., 2025), MASQuant (Hu et al., 2026),

Table 2: Quantization results on Qwen2.5-VL-3B and Qwen2.5-VL-7B. SQ denotes SmoothQuant. SQ, MBQ, and MASQuant are *not reported* at W3A3 because they do *not* produce meaningful results. *BFs* (*Bit-FLOPs*) measures the theoretical computational cost per token, normalized to the FP16 baseline. *Lower is better for BFs*. The best quantization results are shown in **bold**.

Method	Bits	Qwen2.5-VL-3B								Qwen2.5-VL-7B							
		MMMU	SEED	OCRB.	VizWiz	SciQA	TextVQA	Avg.	BFs	MMMU	SEED	OCRB.	VizWiz	SciQA	TextVQA	Avg.	BFs
FP16	W16A16	42.2	69.9	79.3	69.1	81.9	77.9	70.0	1.0×	46.7	73.0	83.8	70.8	88.4	82.9	74.3	1.0×
SQ	W4A8	25.6	55.7	66.9	57.5	72.1	63.9	56.9	0.38×	37.8	62.7	70.2	61.5	83.3	71.1	64.4	0.36×
MBQ		41.2	58.2	66.9	65.0	76.7	73.4	63.5	0.38×	43.3	67.7	74.1	64.3	86.0	74.8	68.3	0.36×
MASQuant		**46.7**	59.7	67.2	62.7	77.9	69.2	63.9	0.41×	43.4	69.5	72.8	66.4	85.7	77.0	69.1	0.40×
QuaRot		45.9	69.2	78.6	67.2	80.1	77.4	69.7	0.38×	46.7	72.4	83.2	68.6	87.4	82.7	73.5	0.36×
SpinQuant		46.2	69.4	78.7	67.4	80.3	**77.6**	69.9	0.38×	47.1	72.5	83.4	68.7	87.5	**83.0**	73.7	0.36×
FlatQuant		46.6	**69.7**	78.8	67.6	80.7	77.0	70.0	0.49×	46.8	72.6	83.1	68.8	**87.8**	82.1	73.5	0.44×
TRPQuant		46.6	69.5	**79.0**	**68.2**	**81.3**	77.4	**70.3**	**0.31×**	**49.1**	**72.9**	**83.5**	**69.0**	**87.8**	82.7	**74.1**	**0.29×**
SQ	W4A4	23.3	0.0	0.0	0.0	0.0	0.0	3.9	0.25×	24.8	0.0	0.2	0.0	0.7	0.0	4.3	0.25×
MBQ		25.0	0.0	0.0	0.0	0.0	0.0	4.2	0.25×	26.7	3.0	0.5	0.0	0.9	0.0	5.2	0.25×
MASQuant		26.7	0.0	7.7	0.0	0.0	0.0	5.7	0.27×	25.0	0.6	13.2	0.0	7.1	0.4	7.7	0.28×
QuaRot		39.8	65.1	75.1	62.7	75.6	73.0	65.2	0.25×	44.0	69.9	80.9	66.6	84.9	79.7	71.0	0.25×
SpinQuant		40.5	65.9	75.7	63.1	76.3	73.7	65.9	0.25×	44.6	70.3	81.3	66.9	85.3	80.0	71.4	0.25×
FlatQuant		41.3	67.2	76.9	65.0	78.4	75.9	67.5	0.36×	44.7	72.0	81.8	68.5	87.2	81.2	72.5	0.33×
TRPQuant		**43.6**	**68.9**	**78.7**	**65.5**	**81.0**	**77.2**	**69.2**	**0.22×**	**46.8**	**72.9**	**83.2**	**69.3**	**87.5**	**82.3**	**73.7**	**0.21×**
QuaRot	W3A3	27.2	48.3	59.1	43.8	47.6	52.0	46.3	**0.19×**	31.6	58.7	67.2	54.8	62.0	61.5	56.0	0.20×
SpinQuant		29.7	52.4	62.1	47.3	51.6	54.9	49.7	**0.19×**	33.1	60.4	69.1	56.8	65.2	64.0	58.1	0.20×
FlatQuant		34.0	60.3	66.1	55.7	61.6	61.4	56.5	0.30×	35.1	65.1	73.2	60.9	71.5	68.4	62.3	0.27×
TRPQuant		**40.1**	**67.0**	**75.3**	**64.2**	**71.5**	**71.9**	**65.0**	0.20×	**43.3**	**70.7**	**78.8**	**65.8**	**81.1**	**79.6**	**69.9**	**0.18×**

Table 3: Quantization results on Qwen3-VL-2B and Qwen3-VL-8B. RTN and GPTQ are *not reported* at W4A4 and W3A3 because they do *not* produce meaningful results.

Method	Bits	Qwen3-VL-2B								Qwen3-VL-8B							
		MMMU	SEED	OCRB.	VizWiz	SciQA	TextVQA	Avg.	BFs	MMMU	SEED	OCRB.	VizWiz	SciQA	TextVQA	Avg.	BFs
FP16	W16A16	40.3	71.7	74.2	67.2	82.4	79.6	69.2	1.0×	52.9	73.8	75.8	68.8	92.8	80.9	74.1	1.0×
RTN	W4A8	30.8	62.3	63.1	54.7	65.2	61.9	56.3	0.38×	47.1	70.1	65.6	64.4	86.1	71.9	67.5	0.36×
GPTQ		32.3	62.6	63.7	55.1	65.7	62.4	57.0	0.38×	46.6	70.3	64.7	64.9	85.5	70.7	67.1	0.36×
QuaRot		37.7	68.3	72.4	63.1	72.8	76.9	65.2	0.38×	50.7	71.9	75.1	66.3	88.6	78.4	71.8	0.36×
SpinQuant		38.7	69.3	72.9	64.1	74.4	77.8	66.2	0.38×	51.2	72.4	75.6	66.8	89.6	78.9	72.4	0.36×
FlatQuant		38.5	70.7	74.1	65.9	76.0	78.3	67.2	0.47×	51.5	73.0	76.8	67.7	90.9	79.2	73.2	0.42×
TRPQuant		**39.9**	**71.4**	**74.6**	**67.2**	**80.5**	**79.9**	**68.9**	**0.33×**	**52.4**	**73.7**	**77.0**	**68.8**	**92.3**	**81.2**	**74.3**	**0.29×**
QuaRot	W4A4	29.2	57.8	64.1	52.1	57.9	64.1	54.2	0.25×	44.2	66.8	69.7	58.9	71.8	70.2	63.6	0.25×
SpinQuant		33.2	63.4	69.1	56.8	66.1	69.9	59.8	0.25×	47.7	69.8	72.6	62.3	76.1	73.5	67.0	0.25×
FlatQuant		35.9	68.3	74.2	63.5	74.9	76.0	65.4	0.34×	51.3	72.2	**77.1**	65.7	88.9	78.4	72.2	0.31×
TRPQuant		**37.6**	**71.0**	**75.0**	**66.3**	**75.3**	**78.1**	**67.2**	**0.24×**	**52.0**	**73.6**	76.4	**68.6**	**91.2**	**79.8**	**73.6**	**0.22×**
QuaRot	W3A3	27.0	23.3	52.0	10.9	20.1	32.2	27.6	**0.19×**	35.2	58.8	67.7	55.3	62.9	60.9	56.8	0.20×
SpinQuant		28.5	22.1	51.5	12.0	16.2	32.7	27.2	**0.19×**	38.2	63.9	70.3	58.8	69.9	65.9	61.2	0.20×
FlatQuant		27.1	17.5	49.2	12.2	13.2	31.1	25.0	0.28×	39.1	68.2	71.8	63.9	76.1	72.6	65.2	0.26×
TRPQuant		**29.9**	**61.0**	**73.4**	**59.8**	**59.2**	**70.5**	**59.0**	**0.19×**	**45.0**	**70.3**	**73.8**	**65.8**	**84.6**	**75.0**	**69.1**	**0.18×**

QuaRot (Ashkboos et al., 2024), SpinQuant (Liu et al., 2024c), Q-VLM (Wang et al., 2024), DuQuant (Lin et al., 2024a), and QSVD (Wang et al., 2025). For a fair comparison, all methods are evaluated using their recommended configurations, with calibration samples drawn from the same ShareGPT4V-enhanced subset of COCO and under the same random seeds.

The experiment benchmarks include MMMU (Yue et al., 2024), SEED-Bench (Li et al., 2023a), OCRBench (Fu et al., 2024), VizWiz (Gurari et al., 2018), ScienceQA (Lu et al., 2022), and TextVQA (Singh et al., 2019). For the transformation matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ in TRPQuant, we adopt the learnable form from FlatQuant (Sun et al., 2024). We set the hyperparameter ρ in TR to 0.05 and use $r_D = 0.75$ across all settings. For r_S, we use a default value of 0.98. For simplicity and to avoid introducing additional stage-specific hyperparameters, we use the same r_S in both phases.

5.2 Main Results

Results on Qwen2.5-VL. Table 2 reports quantization results on Qwen2.5-VL-3B/7B across six benchmarks, with Bit-FLOPs (BFs) measuring theoretical cost by accounting for operation counts and bit-widths. At W4A8, TRPQuant achieves the highest average scores of 70.3/74.1, closely matching the FP16 baselines of 70.0/74.3, while reducing BFs relative to FlatQuant by 36.7%/34.1% through channel pruning. At W4A4, while SQ, MBQ, and MASQuant mostly collapse to near-zero scores, TRPQuant achieves 69.2/73.7 (98.9%/99.2% of FP16) with only 0.22×/0.21× BFs.

It also outperforms FlatQuant by 1.7/1.2 points with 38.9%/36.4% lower cost. At W3A3, TRPQuant achieves the highest average scores among the considered baselines, including QuaRot and SpinQuant, improving upon FlatQuant from 56.5/62.3 to 65.0/69.9 while reducing BFs from 0.30×/0.27× to 0.20×/0.18×.

Results on Qwen3-VL. Table 3 reports quantization results on Qwen3-VL-2B/8B. At W4A8, TRPQuant surpasses all baselines on both model variants, achieving average scores of 68.9 (2B) and 74.3 (8B), which are nearly on par with the FP16 reference (69.2 and 74.1, respectively). At W4A4, both RTN and GPTQ produce meaningless results, and FlatQuant exhibits noticeable accuracy drops. TRPQuant remains the best-performing method, narrowing the FP16 gap to within one point on the 8B model (73.6 vs. 74.1). At W3A3, QuaRot, SpinQuant, and FlatQuant all suffer severe performance degradation on the 2B model, whereas TRPQuant achieves 59.0 on 2B and 69.1 on 8B, preserving 85.3% and 93.3% of their FP16 average scores, respectively.

Results on LLaVA-v1.5. We further verify the generality of TRPQuant on the widely adopted LLaVA-v1.5 architecture with 7B/13B parameter sizes, as shown in Table 4. TRPQuant achieves the highest average score under both W4A8 and W4A4 settings. On the 13B model, TRPQuant retains 99.4% of the FP16 average score at W4A4 (66.4 vs. 66.8), substantially outperforming DuQuant, Q-VLM, and QSVD. At W3A3, all baselines experience significant accuracy drops, while TRPQuant maintains competitive performance with average scores of 60.0 on 7B and 63.2 on 13B models.

Table 4: Quantization results on LLaVA-v1.5-7B/13B.

Method	Bits	LLaVA-v1.5-7B					LLaVA-v1.5-13B				
		SEED	VizWiz	SciQA	Avg.	BFs	SEED	VizWiz	SciQA	Avg.	BFs
FP16	W16A16	66.2	54.3	70.0	63.5	1.0×	68.3	57.3	74.7	66.8	1.0×
DuQuant	W4A8	54.4	50.6	55.3	53.4	0.40×	66.1	56.5	72.3	65.0	0.39×
Q-VLM	W4A8	46.1	48.7	53.2	49.3	0.38×	64.2	55.7	68.4	62.8	0.38×
QSVD	W4A8	57.8	53.5	63.6	58.3	0.38×	66.8	56.9	**75.0**	**66.2**	0.38×
TRPQuant	W4A8	**65.0**	**56.6**	**69.8**	**63.8**	**0.35**×	**67.7**	**57.0**	74.0	**66.2**	**0.36**×
DuQuant	W4A4	51.5	49.8	54.8	52.0	0.27×	64.6	55.3	67.2	62.4	0.27×
Q-VLM	W4A4	37.2	48.9	53.1	46.4	0.25×	48.3	50.9	65.0	54.7	0.25×
QSVD	W4A4	55.1	53.6	57.7	55.5	0.25×	67.0	56.8	67.8	63.9	0.25×
TRPQuant	W4A4	**64.8**	**55.5**	**69.5**	**63.3**	**0.24**×	**67.7**	**57.5**	**74.1**	**66.4**	**0.24**×
DuQuant	W3A3	16.5	7.1	10.6	11.4	0.21×	27.8	19.5	33.2	26.8	0.20×
Q-VLM	W3A3	7.4	0.0	0.8	2.7	**0.19**×	10.3	9.9	15.1	11.8	**0.19**×
QSVD	W3A3	21.9	11.5	13.2	15.5	**0.19**×	26.4	22.3	37.2	28.6	**0.19**×
TRPQuant	W3A3	**60.6**	**56.0**	**63.4**	**60.0**	0.20×	**64.0**	**56.6**	**69.1**	**63.2**	0.21×

5.3 Ablation Study

Stability of Active-channel Set $\mathcal{A}$. PTQ typically relies on a limited calibration set. We evaluate the stability of ACI by computing the Jaccard similarity between active-channel sets selected under different calibration set sizes and those selected using the default 128 calibration samples. As shown in Table 5, ACI produces highly consistent active-channel sets, with all similarities at least 92.9%.

Table 5: Similarity of ACI results under different calibration set sizes (N) compared with calibration size of 128 (default).

Sim. (%)	**32**	**64**	**256**	**512**
Qwen2.5-VL-7B	95.5	95.7	96.4	96.3
Qwen3-VL-8B	95.2	94.9	95.5	95.8
LLaVA-v1.5-7B	92.9	93.4	93.2	94.1

Ablation of TR and CP. Table 6 evaluates the effectiveness of TR and CP. We start with the simple baseline FlatQuant, which uses one shared transformation for vision and language tokens. For "FlatQ+Modal", we separate tokens into vision and language groups, which are processed by two different FlatQuant transformations.

It shows that this modality-based token partitioning improves the average score from 59.9 to 63.3. Based on FlatQuant, we add our proposed token rearrangement module, separating tokens into dense, sparse and intermediate groups, which are then processed by two different FlatQuant transformations. The result shows the superiority of our TR over the modality-level token partitioning. The reason is that TR performs fine-grained token partitioning and each group of tokens exhibits a more homogeneous distribution, improving quanti-

Table 6: Ablation study of core components (TR and CP) of TRPQuant at W3A3 on Qwen2.5-VL-7B. FlatQ (FlatQuant) is the baseline, for which all tokens (vision and language) share one transformation. "FlatQ+Modal" means that tokens are partitioned into two groups based on modality, which are processed by two different FlatQuant transformations.

Method	MMMU	OCRB.	SciQA	Avg.	BFs
FlatQ	35.1	73.2	71.5	59.9	0.27×
FlatQ+Modal	38.9	75.5	75.6	63.3	0.27×
FlatQ+TR	**43.6**	**78.8**	**81.5**	**68.0**	0.27×
FlatQ+TR+CP (TRPQuant)	43.3	**78.8**	81.1	67.7	**0.18**×

zation friendliness. Finally, we add channel pruning on "FlatQ+TR", giving our full model TRPQuant. Compared with FlatQ+TR, incorporating CP technique maintains a comparable average score while clearly reducing BFs from $0.27\times$ to $0.18\times$.

Sensitivity of r_S and r_D. Table 7 reports the effect of r_S on the pruning ratio and SciQA score, where the pruning ratio is the fraction of removed FLOPs over the original FLOPs. Setting $r_S = 0.98$ removes 30.8% of the FLOPs while retaining a SciQA score of 81.1, only 0.4 points below the unpruned setting. Further decreasing r_S leads to more aggressive pruning but causes a substantial performance degradation. We therefore set $r_S = 0.98$ by default. As shown in Table 7, the model is relatively robust to r_D over the range from 0.45 to 0.75, with $r_D = 0.75$ achieving the highest SciQA score of 81.1. Accordingly, we adopt $r_D = 0.75$ as the default.

Table 7: Sensitivity to r_S and r_D on Qwen2.5-VL-7B at W3A3.

r_S	Threshold	1.00	0.98	0.96	0.94
	Pruning	0.0%	30.8%	45.3%	53.9%
	Score	81.5	81.1	75.6	68.1
r_D	Threshold	0.90	0.75	0.60	0.45
	Score	79.3	**81.1**	80.9	81.0

5.4 Efficiency Evaluation

Single Kernel Efficiency. We evaluate the per-layer efficiency of TRPQuant on different linear-layer shapes from Qwen3-VL-8B and LLaVA-v1.5-7B. As shown in Table 8, TRPQuant consistently accelerates linear computation across all evaluated shapes. The speedup primarily arises from reduced memory traffic in W4A16 weight loading and the reduced FLOP count of the sparse branch on active channels. TRPQuant achieves up to $4.06\times$ speedup on Qwen3-VL-8B and $3.89\times$ on LLaVA-v1.5-7B.

Table 8: Speedup of the fused W4A16 kernel on linear layers of different shapes ($D_{\text{in}} \times D_{\text{out}}$). All experiments are conducted on an NVIDIA RTX 4090 GPU with batch size 1 and sequence length 2048.

Model	$D_{\text{in}} \times D_{\text{out}}$	FP16 (ms)	TRPQuant (ms)	Speedup
Qwen3-VL-8B	4096×1024	0.110	0.049	**2.24**×
	4096×4096	0.422	0.104	**4.06**×
	4096×12288	1.252	0.370	**3.38**×
	12288×4096	1.260	0.363	**3.47**×
LLaVA-v1.5-7B	4096×4096	0.428	0.110	**3.89**×
	4096×11008	1.138	0.340	**3.35**×
	11008×4096	1.127	0.343	**3.28**×

End-to-End Efficiency. To evaluate end-to-end efficiency, we implement a custom W4A4 CUDA kernel following the fused-kernel design in (Li et al., 2024b), which integrates projection and quantization operators. Table 9 compares prefill latency, GPU memory consumption, and quantization accuracy on Qwen2.5-VL-7B. Compared with FP16, TRPQuant achieves a $2.60\times$ prefill speedup and a $2.69\times$ memory reduction, while retaining an average score of 73.7, close to the FP16 score of 74.3. TRPQuant also runs faster than MASQuant and FlatQuant, and achieves substantially higher accuracy than MASQuant. Compared with FlatQuant, TRPQuant provides lower prefill latency and higher accuracy, with only a slight increase in memory usage.

Table 9: Efficiency comparison on Qwen2.5-VL-7B using an RTX 4090 at W4A4. Latency and memory are measured with batch size 1 and sequence length 2048, while avg. score is averaged over six evaluation benchmarks.

Method	Prefill (ms)	Speedup	Mem (GB)	Mem Reduction	Avg. Score
FP16	191.82	1.0×	13.73	1.0×	74.3
MASQuant	77.10	2.49×	5.37	2.56×	7.7
FlatQuant	84.23	2.28×	**4.96**	**2.77**×	72.5
TRPQuant	**73.72**	**2.60**×	5.10	2.69×	**73.7**

6 Conclusion

We presented TRPQuant, a token-adaptive PTQ framework that addresses the limitations of modality-level partitioning in VLM quantization. By leveraging channel-wise energy concentration, TRPQuant dynamically classifies tokens into Dense, Intermediate, and Sparse groups and applies tailored strategies: dense tokens retain all channels and follow the standard branch, sparse tokens undergo channel pruning and are processed via a sparse branch, and intermediate tokens are decomposed and handled by both branches. Extensive experiments demonstrate that TRPQuant achieves strong quantization accuracy while reducing computational overhead. Future work will explore hardware-aware optimizations for further inference acceleration on edge devices.

AI USE STATEMENT

In this work, we used generative AI tools for tasks including translation, and qualitative and thematic data analysis. We have not used generative AI tools for tasks such as developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, and interpreting results, and tasks including generating synthetic data sets, and cleaning dataset are not applicable to this work. Additionally, we used generative AI tools for tasks like brainstorming, and searching for information. We have reviewed all AI-assisted work. For example, we checked LLM-generated information through a manual fact check. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the methodological and experimental details necessary to reproduce TRPQuant throughout the paper. Section 4 describes the complete framework, including active-channel identification, token rearrangement, channel pruning, and branch-specific quantization. The experimental setup specifies the evaluated models, benchmarks, baseline configurations, calibration data, random seeds, and hyperparameters. Section 5.3 further reports calibration-set stability, component ablations, and threshold sensitivity, while the efficiency evaluation provides the hardware and runtime settings used for latency measurements. All baselines are evaluated using their recommended configurations under the same calibration subset and random seeds.

REFERENCES

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In *NeurIPS*, 2024.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b.

Yelysei Bondarenko, Riccardo Del Chiaro, and Markus Nagel. Low-rank quantization-aware training for llms. *arXiv preprint arXiv:2406.06385*, 2024.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. In *ACL*, 2025.

Juncan Deng and Kejie Huang. Vlm-ptq: Efficient post-training quantization for large vision-language models. In *CVPR*, pp. 24696–24705, 2026.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Aleksandr Dremov, David Grangier, Angelos Katharopoulos, and Awni Hannun. Compute-optimal quantization-aware training. *arXiv preprint arXiv:2509.22935*, 2025.

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

Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, et al. Ocrbench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. *arXiv preprint arXiv:2501.00321*, 2024.

Tianyu Guo, Shanwei Zhao, Shiai Zhu, and Chenguang Ma. Speed-q: Staged processing with enhanced distillation towards efficient low-bit on-device vlm quantization. In *AAAI*, volume 40, pp. 21486–21494, 2026.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *CVPR*, 2018.

Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words. *NeurIPS*, 32, 2019.

Lulu Hu, Wenhu Xiao, Xin Chen, Xinhua Xu, Bowen Xu, Kun Li, and Yongliang Tao. Masquant: Modality-aware smoothing quantization for multimodal large language models. *arXiv preprint arXiv:2603.04800*, 2026.

Xing Hu, Yuan Cheng, Dawei Yang, Zhihang Yuan, Jiangyong Yu, Chen Xu, and Sifan Zhou. I-llm: Efficient integer-only inference for fully-quantized low-bit large language models. *arXiv preprint arXiv:2405.17849*, 2024.

Xing Hu, Zhixuan Chen, Dawei Yang, Zukang Xu, Chen Xu, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. Moequant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. *arXiv preprint arXiv:2505.03804*, 2025a.

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

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, pp. 2704–2713, 2018.

Chenwei Jia, Baoting Li, Xuchong Zhang, Mingzhuo Wei, Bochen Lin, and Hongbin Sun. Quant experts: Token-aware adaptive error reconstruction with mixture of experts for large vision-language models quantization. *arXiv preprint arXiv:2602.24059*, 2026.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023a.

Liang Li, Qingyuan Li, Bo Zhang, and Xiangxiang Chu. Norm tweaking: High-performance low-bit quantization of large language models. In *AAAI*, 2024a.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024b.

Qingyuan Li, Yifan Zhang, Liang Li, Peng Yao, Bo Zhang, Xiangxiang Chu, Yerui Sun, Li Du, and Yuchen Xie. Fptq: Fine-grained post-training quantization for large language models. *arXiv preprint arXiv:2308.15987*, 2023b.

Ruiyu Li and Jiaya Jia. Visual question answering with question representation update (qru). *NeurIPS*, 29, 2016.

Shiyao Li, Yingchun Hu, Xuefei Ning, Xihui Liu, Ke Hong, Xiaotao Jia, Xiuhong Li, Yaqi Yan, Pei Ran, Guohao Dai, et al. Mbq: Modality-balanced quantization for large vision-language models. In *CVPR*, 2025.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. In *NeurIPS*, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *MLSys*, 2024b.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024a.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. In *ACL*, 2024b.

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

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*, 2022.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision–language–action models for embodied ai. *TNNLS*, 37(7):3031–3051, 2026. doi: 10.1109/TNNLS.2025.3650584.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Affinequant: Affine transformation quantization for large language models. *arXiv preprint arXiv:2403.12544*, 2024.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *ICML*, pp. 7197–7206. PMLR, 2020.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

Sayeh Sharify, Utkarsh Saxena, Zifei Xu, Ilya Soloveychik, Xin Wang, et al. Post training quantization of large language models with microscaling formats. *arXiv preprint arXiv:2405.07135*, 2024.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, 2019.

Siqing Song, Chuang Wang, Rui-Qi Wang, Yi Yang, and Xu-Yao Zhang. Achieving binary weight and activation for llms using post-training quantization. In *ACL*, 2025.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024.

Changyuan Wang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Q-vlm: Post-training quantization for large vision-language models. In *NeurIPS*, 2024.

Shigeng Wang, Chao Li, Yangyuxuan Kang, Jiawei Fan, Zhonghong Ou, and Anbang Yao. Sliderquant: Accurate post-training quantization for llms. *arXiv preprint arXiv:2603.25284*, 2026a.

Xijun Wang, Rayyan Abdalla, Junyun Huang, Chengyuan Zhang, Ruiqi Xian, and Dinesh Manocha. Bi-vlm: Binary post-training quantization for vision-language models. In *AAAI*, 2026b.

Yutong Wang, Haiyu Wang, and Sai Qian Zhang. Qsvd: Efficient low-rank approximation for unified query-key-value weight compression in low-precision vision-language models. *arXiv preprint arXiv:2510.16292*, 2025.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. In *NeurIPS*, 2022.

Ziwei Xiang, Fanhu Zeng, Hongjian Fang, Rui-Qi Wang, Renxing Chen, Yanan Zhu, Yi Chen, Peipei Yang, and Xu-Yao Zhang. Fine-grained post-training quantization for large vision language models with quantization-aware integrated gradients. *arXiv preprint arXiv:2603.17809*, 2026.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *ICML*, 2023.

Jingjing Xie, Yuxin Zhang, Mingbao Lin, Liujuan Cao, and Rongrong Ji. Advancing multimodal large language models with quantization-aware scale learning for efficient adaptation. In *ACM MM*, 2024.

Chen Xu, Yuxuan Yue, Zukang Xu, Xing Hu, Jiangyong Yu, Zhixuan Chen, Sifan Zhou, Zhihang Yuan, and Dawei Yang. Rwkvquant: Quantizing the rwkv family with proxy guided hybrid of scalar and vector quantization. *arXiv preprint arXiv:2505.03803*, 2025.

Yufei Xue, Yushi Huang, Jiawei Shao, and Jun Zhang. Vlmq: Efficient post-training quantization for large vision-language models via hessian augmentation. *arXiv preprint arXiv:2508.03351*, 2025.

Ningfeng Yang and Tor M Aamodt. Boosting entropy with bell box quantization. *arXiv preprint arXiv:2603.01599*, 2026.

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, et al. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. In *ICCV*, 2025.

Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation. *arXiv preprint arXiv:2303.08302*, 2023.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *CVPR*, pp. 4651–4659, 2016.

JiangYong Yu, Sifan Zhou, Dawei Yang, Shuoyu Li, Shuo Wang, Xing Hu, Chen Xu, Zukang Xu, Changyong Shu, and Zhihang Yuan. Mquant: Unleashing the inference potential of multimodal large language models via static quantization. In *ACM MM*, 2025.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *CVPR*, 2024.

Peiyuan Zhang, Matthew Noto, Wenxuan Tan, Chengquan Jiang, Will Lin, Wei Zhou, and Hao Zhang. Attn-qat: 4-bit attention with quantization-aware training. *arXiv preprint arXiv:2603.00040*, 2026.

Ying Zhang, Peng Zhang, Mincong Huang, Jingyang Xiang, Yujie Wang, Chao Wang, Yineng Zhang, Lei Yu, Chuan Liu, and Wei Lin. Qqq: Quality quattuor-bit quantization for large language models. *arXiv preprint arXiv:2406.09904*, 2024.

Weibo Zhao, Yubin Shi, Xinyu Lyu, Wanchen Sui, Shen Li, and Yong Li. Aser: activation smoothing and error reconstruction for large language model quantization. In *AAAI*, 2025.

Zhen Zheng, Xiaonan Song, and Chuanjie Liu. Mixllm: Llm quantization with global mixed-precision between output-features and highly-efficient system design. *arXiv preprint arXiv:2412.14590*, 2024.

Yi Zhong, Haotong Qin, Xindong Zhang, Lei Zhang, and Guolei Sun. Breaking modality heterogeneity in low-bit quantization for large vision-language models. *arXiv preprint arXiv:2605.19929*, 2026.