# PTQ4VM: Post-Training Quantization for Visual Mamba

Younghyun Cho* Changhun Lee* Seonggon Kim Eunhyeok Park
Pohang University of Science and Technology (POSTECH), Republic of Korea
{yhcho97, changhun.lee, sungonuni, eh.park}@postech.ac.kr

## Abstract

*Visual Mamba is an approach that extends the selective space state model, Mamba, to vision tasks. It processes image tokens sequentially in a fixed order, accumulating information to generate outputs. Despite its growing popularity for delivering high-quality outputs at a low computational cost across various tasks, Visual Mamba is highly susceptible to quantization, which makes further performance improvements challenging. Our analysis reveals that the fixed token access order in Visual Mamba introduces unique quantization challenges, which we categorize into three main issues: 1) token-wise variance, 2) channel-wise outliers, and 3) a long tail of activations. To address these challenges, we propose Post-Training Quantization for Visual Mamba (PTQ4VM), which introduces two key strategies: Per-Token Static (PTS) quantization and Joint Learning of Smoothing Scale and Step Size (JLSS). To the our best knowledge, this is the first quantization study on Visual Mamba. PTQ4VM can be applied to various Visual Mamba backbones, converting the pretrained model to a quantized format in under 15 minutes without notable quality degradation. Extensive experiments on large-scale classification and regression tasks demonstrate its effectiveness, achieving up to 1.83× speedup on GPUs with negligible accuracy loss compared to FP16. Our code is available at* https://github.com/YoungHyun197/ptq4vm.

## 1. Introduction

The State Space Model (SSM) [8, 9] was introduced to address the quadratic computational cost of transformers and to process sequential data more efficiently. An enhanced version of SSM, called Mamba [7], further improves this by updating the internal states selectively. Mamba has outperformed transformers and other sub-quadratic models across various language tasks [6, 27, 28], offering higher accuracy with relatively low computational cost. Recently, there has been growing interest in extending Mamba's capabilities to vision tasks [12, 22, 36, 39], often referred to as Visual Mamba. These efforts have introduced new module designs and features, such as class tokens for image data, demonstrating the superiority of Mamba in visual tasks.

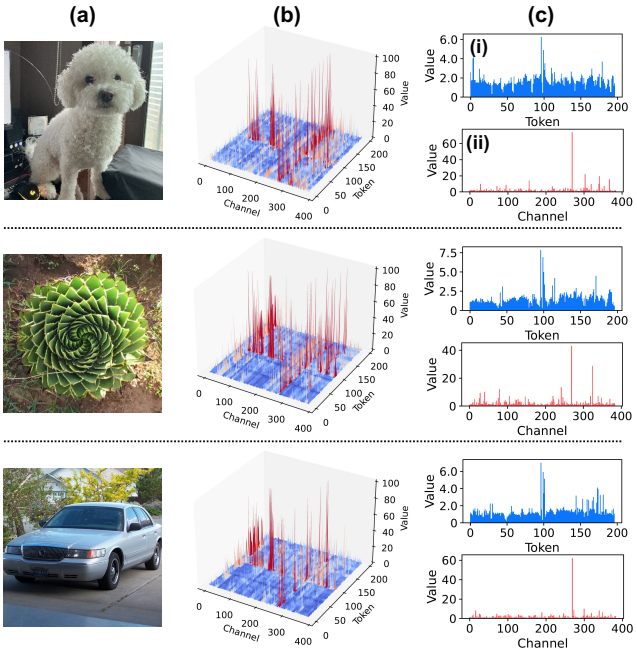



Figure 1. Distribution of the input activations of a 22nd out_proj layer in Vim-Ti. (a) Images from 3 categories, (b) their corresponding activation distributions, and (c) the average values across (i) the channel dimension and (ii) the token dimension.

In this study, we aim to enhance the cost-efficiency of Visual Mamba models through quantization. The performance advantage of the Visual Mamba can be further improved by reducing computational overhead and memory footprint via quantization. Our profiling results of existing Visual Mamba backbones revealed that a significant portion of execution time is dominated by linear operators (blue in Fig. 3), which are well-suited for low-precision computations. This analysis suggests that Visual Mamba can sufficiently benefit from quantization in practice.

However, our observations identified a significant quality degradation when applying traditional post-training quan-

*These authors contributed equally.

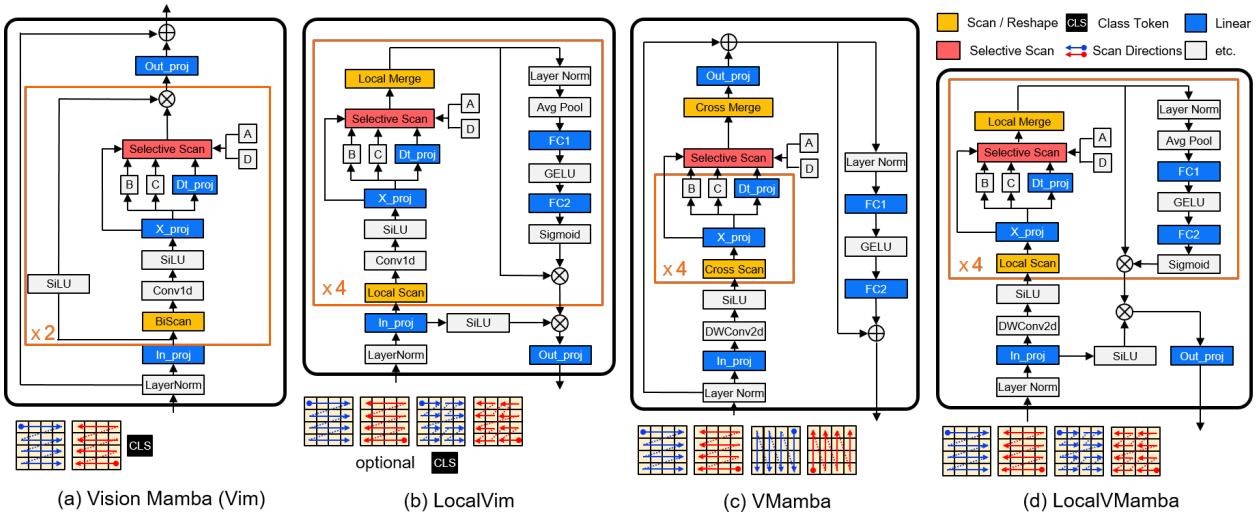



Figure 2. The Visual Mamba backbones consist of (a) Vision Mamba, (b) LocalVim, (c) VMamba, and (d) LocalVMamba. "x2" and "x4" indicate the repetition of operations based on scan directions. The square matrices beneath illustrate the scan method for each backbone.

tization (PTQ) [1, 24, 26] techniques to Visual Mamba. Specifically, the structure of Visual Mamba (see Fig. 2), which sequentially processes image data tokens in a fixed order, results in an activation distribution that is particularly susceptible to quantization. The vulnerabilities of Visual Mamba can be categorized into three key items: 1) token-wise variance (Fig. 1c (i)), 2) channel-wise outliers (Fig. 1c (ii)), and 3) the long tail of activations (Fig. 5). As these challenges are not adequately addressed by conventional quantization methods, resolving them is crucial for maintaining output quality after quantization in Visual Mamba.

The weaknesses identified are prevalent across all Visual Mamba backbones proposed thus far, indicating that addressing them could yield broad benefits across a variety of models. Building on these insights, we introduce PTQ4VM, an effective and efficient post-training quantization (PTQ) scheme for Visual Mamba. To the our best knowledge, PTQ4VM is the first comprehensive study on quantization techniques for Visual Mamba. It is founded on two key techniques: Per-Token Static Quantization (PTS) and Joint Learning of Smoothing Scale and Step Size (JLSS). PTS is specifically designed to handle per-token variance, and we have carefully crafted it to be compatible with existing SmoothQuant method [35], which are effective in managing outliers within channels. Moreover, JLSS jointly optimizes the quantization parameters for PTS and scales for SmoothQuant, ensuring minimal discrepancies in output feature maps and preserving the network's functionality after quantization. Both PTS and JLSS are meticulously designed to maximize throughput to realize acceleration in practice, and we demonstrate the versatility and superiority of PTQ4VM through extensive experiments.

## 2. Related Works

### 2.1. Selective State Space model

State Space Models (SSMs) [9, 10] are linear time-invariant (LTI) systems that process sequential data through internal state variables. Originally designed for natural language processing (NLP) tasks, they often use an input-independent discretized formulation, expressed as follows:

$$\bar{A} = \exp(\Delta A), \tag{1}$$

$$\bar{B} = (\Delta A)^{-1}(\exp(\Delta A) - I)\Delta B, \tag{2}$$

where $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times 1}$, and $C \in \mathbb{R}^{1\times N}$ are the parameters of the SSM, and $\Delta$ is the timescale parameter.

$$h_t = \bar{A}h_{t-1} + \bar{B}x_t, \tag{3}$$

$$y_t = Ch_t. \tag{4}$$

Recent research introduced Mamba [7], a general language backbone that eliminates the linear time-invariant (LTI) property of SSMs, making them input-dependent. The discretized parameters for each input sequence of length $t$ are computed as follows, based on the input $x$:

$$B_t, \Delta_t, C_t = \text{Linear}_x(x_t), \tag{5}$$

$$\bar{\Delta}_t = \text{softplus}\left(\text{Linear}_\Delta(\Delta_t)\right), \tag{6}$$

$$\bar{A}_t = \exp(\bar{\Delta}_t A), \tag{7}$$

$$\bar{B}_t = \bar{\Delta}_t B_t. \tag{8}$$

Mamba has gained significant attention for its ability to efficiently handle long sequence data with linear complexity. By introducing a selective scan that updates key information based on the input, it offers a distinct advantage over previous models that lacked this capability.

## 2.2. Visual Mamba Backbones

Vision Mamba (Vim) [39] represents the first attempt to apply Mamba directly to vision tasks. Vim employs a CLS token, which is essential for its classification tasks. VMamba [22], on the other hand, features a backbone architecture that closely resembles the Swin Transformer [23]. LocalMamba [12] introduces two variants—LocalVim and LocalVMamba—based on the Vim and VMamba architectures, respectively, with enhancements in scan directions. Notably, among these LocalMamba variants, the LocalVim-T† model makes use of the CLS token. Fig. 2 illustrates the detailed modular design of each backbone.

## 2.3. Post-training Quantization

Quantization is currently the most commercially successful optimization method for leveraging the benefits of low-precision representations. Early research primarily focused on quantization-aware training (QAT) [5, 18, 31, 38], but its popularity has waned due to the high cost of the training process. Consequently, post-training quantization (PTQ) [15, 19, 24, 25, 32, 34] has emerged as a key interest.

In this work, we aim to develop a PTQ scheme for linear quantization, focusing on accelerating computation through integer arithmetic while minimizing transformation costs. We adopt the conventional PTQ approach [13], utilizing per-channel symmetric quantization for weights and per-tensor asymmetric for activations. Given an input activation $X$, the quantized value $\hat{X}$ is calculated as follows:

$$\Delta_X = \frac{max(X) - min(X)}{2^b - 1}, \tag{9}$$

$$\epsilon_X = \frac{-min(X)}{\Delta_X}, \tag{10}$$

$$\hat{X} = \Delta_X \cdot \left( clip(\lceil \frac{X}{\Delta_X} \rfloor + \epsilon_X, 0, 2^b - 1) - \epsilon_X \right). \tag{11}$$

where $b$ represents the number of bits, $\Delta_X$ denotes the quantization step for the activation, and $\epsilon_X$ represents the quantization offset.

In the case of the given weight $W$, the quantized value $\hat{W}$ using symmetric quantization is calculated as follows:

$$\Delta_W = \frac{max(abs(W))}{2^{b-1} - 1}, \tag{12}$$

$$\hat{W} = \Delta_W \cdot clip(\lceil \frac{W}{\Delta_W} \rfloor, -2^{b-1} + 1, 2^{b-1} - 1). \tag{13}$$

where $\Delta_W$ represents the quantization step for the weight, and $\epsilon_W$ is omitted in symmetric quantization.

For the baseline, the quantization range is determined using min-max values [26]. While this method has been empirically proven to perform well in CNN-based networks, it is less effective for Visual Mamba, which exhibits significantly different characteristics.

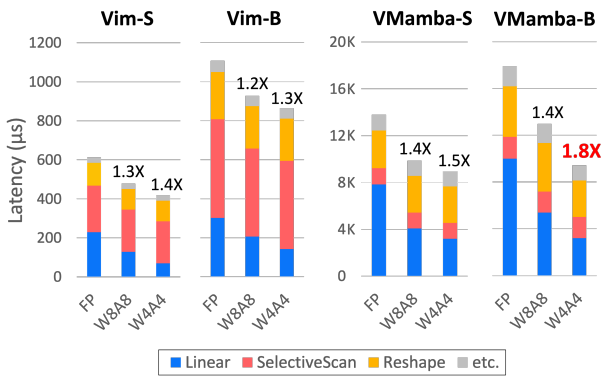



Figure 3. Profiling results of Visual Mamba backbones on an RTX 3090. The numbers above the bars indicate the speedup.

## 2.4. SmoothQuant

Quantization has also been actively studied in the context of Large Language Models (LLMs) [3, 29, 33, 37]. LLMs present unique challenges that complicate the quantization process, such as the presence of channel-wise outlier activations [4, 16, 17]. These outliers cause a significant increase in quantization error by enlarging the quantization step size. To address this, SmoothQuant [35] was proposed, aiming to mitigate the impact of outliers by shifting the quantization complexity from activations to weights, all without introducing additional computational overhead. In SmoothQuant, given the input activation $X$ and weight $W$, the normalization scale is calculated as:

$$s = \sqrt{\frac{max|X|}{max|W|}} \in \mathbb{R}^{D_{in}}, \tag{14}$$

where $D_{in}$ is the input channel. The computed scale is then applied to both $X$ and $W$ respectively, adjusting each tensor to ensure that the overall output remains consistent:

$$Y = (W diag(s)) \cdot (diag(s)^{-1} X). \tag{15}$$

This method mitigates activation outliers, thereby reducing quantization errors. According to our observation, in Visual Mamba, although the underlying causes may differ, we observed a similar phenomenon where outliers occur only in specific activation channels. The details of observation and the solution will be provided at Section 4.

# 3. Analysis for Quantization Target

Before applying quantization, we analyzed whether existing Visual Mamba backbones benefit from quantization. We profiled the two models among models presented in Fig. 2 on the GPU and analyzed which components would be most suitable for quantization.

As shown in Fig. 3, the operations that consistently consume the most time across all models are the linear layer

| Quant Target | Top-1 Acc. |
|---|---|
| FP16 | 76.1 |
| $h(t)$ only | 6.8 |
| Linear only | 57.8 |

| Method | Top-1 Acc. |
|---|---|
| FP16 | 76.1 |
| INT8 Baseline | 57.8 |
| + CLS Tok. FP16 | 73.6 |

Table 1. INT8 accuracy results (%) on Vim-Ti (Left) across different quantization targets, (Right) with FP16 CLS token.

(blue section in Fig. 3), selective scan [7] (red section), and reshape operation (yellow section). Our analysis of the quantization benefits for these three operations leads to the following conclusions: 1) For the linear operation, significant performance improvements can be achieved by applying quantization using INT8/INT4 operators [13]. In the case of the selective scan, it has been optimized to minimize memory bottlenecks through the use of registers and shared memory, so it is difficult to expect performance gains through quantization. More concerning, our experimental results, reported in Table 1 left, show that despite applying quantization only to the hidden state $h(t)$ in Eq. (3), it is far more vulnerable than anticipated, outweighing the expected performance gains. The reshape operation is used to change the data layout between adjacent operators for optimal speed or to rearrange data order for scan operations. However, the overhead caused by reshape operations is difficult to be mitigated by quantization.

Based on this analysis, we concluded that focusing on quantizing the linear layers is a reasonable approach. As illustrated in Fig. 3, applying PTQ4VM to the linear layers can reduce latency by up to 1.83× on the real GPUs, making it a highly appealing option.

## 4. Challenges of Linear Layer Quantization

Based on the previous analysis, we identified the linear operators in Visual Mamba as key targets for optimization. However, despite this focused optimization effort, Visual Mamba still shows a significant drop in quality, regardless of the backbone used. Further investigation revealed that its sequential access to visual tokens for information accumulation causes abnormal activation distributions, making quantization especially difficult. We categorized these difficulties into three main items: token-wise variance, channel-wise outliers, and the long tail of activations. In the following section, we will provide a more detailed explanation of the problems we identified.

### 4.1. Observation 1: Token-wise Variance

We began by analyzing the activation distribution by feeding various images into Visual Mamba. Fig. 1 illustrates three representative cases. The results show that specific token positions, such as position 97, consistently displayed similar activation patterns, regardless of the image class or features. This behavior appears to stem from Visual

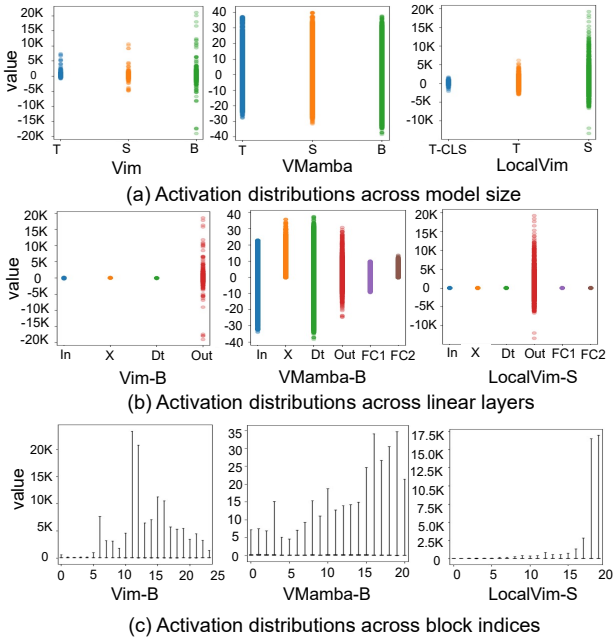
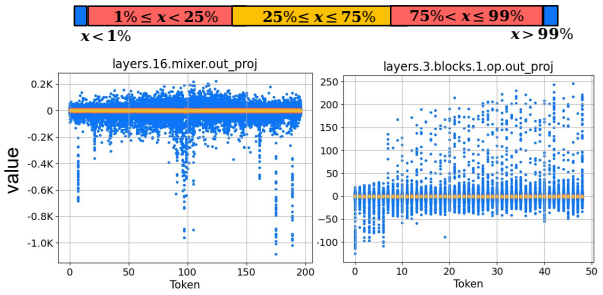



Figure 4. Observation on activation distributions across (a) model size, (b) types of linear layers , and (c) block indices. (c) shows the activation distribution of the out_proj layer.

Mamba's unique architecture, where image patches are processed sequentially in a predetermined order, as suggested by the scan directions in Fig. 2. Additionally, we observed that token-wise activation variance increases significantly in the middle to later blocks of the network compared to the early blocks, with the differences between tokens also becoming more pronounced (Fig. 4c). This suggests that the cumulative effect over the blocks amplifies the imbalance, making larger networks more challenging to quantize.

On the other hand, the Visual Mamba backbone can be categorized into two types of implementations: those incorporating the CLS token and those relying solely on visual tokens. In the case of the Vim model, which uses the CLS token, it is noticeable that the magnitude of the CLS token is significantly smaller than that of the visual tokens, regardless of the input. This presents a challenge, as the CLS token is crucial for downstream tasks like classification, making it particularly vulnerable to quantization errors. As shown in Table 1 right, preserving the CLS token in FP16 format substantially recovers most of the accuracy loss compared to fully quantizing the model's linear layers to INT8. For networks that use the CLS token, we should reduce the quantization error for this special token.

Due to the variation in token-wise activation, conventional tensor-wise quantization (as shown in Fig. 6b) is inevitably suboptimal for individual tokens. This becomes a key factor in increasing quantization error and highlights the need for an appropriate solution. One potential approach is per-token dynamic quantization, which adjusts the quan-




(a) LocalVim-Ti (w/ CLS Tok.) (b) LocalVmamba-S (w/o CLS Tok.)

Figure 5. Comparison of activation distribution on LocalMamba backbone depending on whether CLS Token is utilized or not.

tization range for each token based on its distribution during input processing. However, this method has a significant drawback: activation statistics must be computed online for each input, slowing down inference as shown in Table 4.

To fully preserve the benefits of low-precision arithmetic in the Visual Mamba backbone as well as maintain the quality of output, it is essential to explore methods that address token-wise variance to minimize the accuracy gap with per-token dynamic quantization while maintaining a speed comparable to per-tensor static quantization.

### 4.2. Observation 2: Channel-wise Outliers

The second observation is that activation outliers tend to occur in a few specific input channels, regardless of the input (Fig. 1c (ii)). This phenomenon is common across all backbones, where a small number of activation outliers lead to larger quantization steps, causing information loss in other tokens. It's important to note that this issue stems from a different dimension than the one discussed in the first observation. Even when per-token dynamic quantization, a potential solution for the previous issue, is applied, the same step size must be used for all channels within a token. As a result, it fails to mitigate the performance degradation caused by these activation outliers.

As discussed in Section 2.4, although the underlying causes may differ, activation outliers have also been observed in Large Language Models (LLMs) with transformer architectures. When we applied smoothing to Visual Mamba backbones, we noted quality improvements at 8-bit and 6-bit quantization (see Table 2). This suggests that smoothing should also be integrated into the quantization process for Visual Mamba. Accordingly, our proposed PTQ4VM is specifically designed to incorporate it.

### 4.3. Observation 3: Long tail of Activation

The activation distribution of Visual Mamba exhibits a distinctive characteristic. As illustrated in Fig. 5, approximately 98% of the activation values are concentrated within a very narrow range (represented by the red and yellow areas in the plot). The data spread across a wider range corresponds to the top and bottom 1% of values, indicating a long-tailed distribution. Similar to the outliers seen in Observation 2, this long tail extends the quantization range too far, leading to a loss of information from tokens near zero. Notably, as the backbones in the Vision Mamba series feature tails extending beyond 10,000, smoothing outliers with SmoothQuant alone is insufficient to fully resolve the issue.

Previous studies [2, 20] have emphasized the importance of introducing truncation to balance the trade-off between clipping and rounding errors. Given the long tail of activation, truncation is also crucial for Visual Mamba. However, similar to dynamic quantization, input-dependent truncation leads to significant performance degradation due to the associated online costs. To minimize quantization errors while preserving the acceleration benefits, it is essential to determine the optimal static truncation range that is generally applicable across elements within the same token position.

## 5. PTQ4VM

In this section, we propose PTQ4VM (Fig. 6d), a post-training quantization method designed to effectively address the three challenges identified in Section 4. It mainly consists of two parts: First, a Per-Token Static (PTS) quantization to handle Observation 1 and 2 (Section 5.1). Next, the JLSS method to find the optimal smooth scale and step size to address Observation 3 (Section 5.2).

### 5.1. Per-Token Static (PTS) Quantization

We propose Per-Token Static (PTS) quantization, a simple yet powerful method to address Observations 1 and 2 with minimal computational overhead. Since the token length in Visual Mamba is predetermined by the fixed input size, we can allocate the quantization step size and zero offset for each token of length $L$ using a calibration dataset. The weight step size $\Delta_W \in \mathbb{R}^{D_{out} \times 1}$ and activation step size $\Delta_X \in \mathbb{R}^{1 \times L}$ are determined by the tensor dimensions. After quantization, the integer-mapped weight $\bar{W} \in \mathbb{R}^{D_{out} \times D_{in}}$ and activation $\bar{X} \in \mathbb{R}^{D_{in} \times L}$ can be efficiently multiplied via integer operations. The final output is then generated by multiplying the result with the element-wise scales, produced by the outer product of $\Delta_W$ and $\Delta_X$:

$$WX = (\Delta_W \bar{W})(\Delta_X \bar{X}) = (\Delta_W \Delta_X)(\bar{W}\bar{X}). \quad (16)$$

Note that we omitted the zero offset ($\epsilon_X$) for simplicity of explanation. If $\Delta_W$ and $\Delta_X$ can be fused across adjacent linear layers, the output can be computed using only low-precision operations without element-wise scaling.

PTS predetermines the step size statically, resulting in significantly lower overhead compared to per-token dynamic methods, thereby gaining an advantage in acceleration. As reported in Table 4, our method demonstrates a 1.3× speed improvement over per-token dynamic ap-

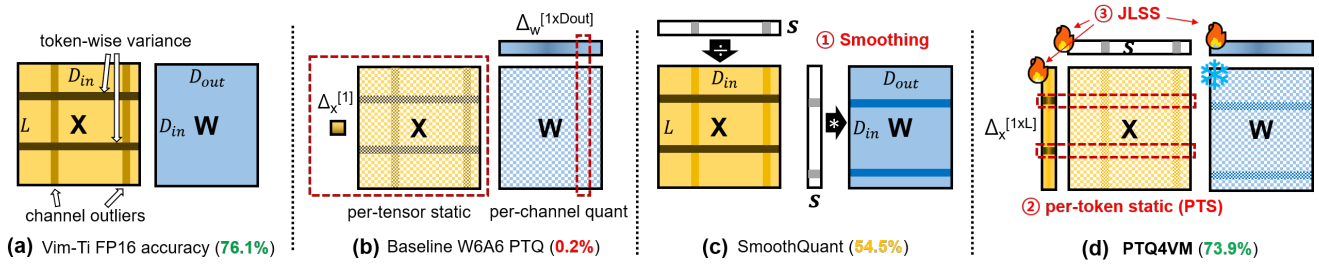



Figure 6. Comparisons of quantization methods. (b) and (c) uses per-tensor static quantization for activation.

proaches. Furthermore, PTS is a modification applied orthogonally to SmoothQuant, offering the advantage of compatibility with smoothing techniques. Through this combined approach, we can address the aforementioned problems from observation 1 and 2.

### 5.2. Joint Learning of Smoothing Scale and Step Size (JLSS)

To address the third challenge, the long tail of activations, we introduce Joint Learning of Smoothing Scale and Step Size (JLSS). The core objective of JLSS is to identify the optimal values for the smoothing scale $s$ and step sizes $\Delta_X$ and $\Delta_W$, ensuring minimal deviations in the output tensor. To achieve this, JLSS employs a three-stage method for concurrent optimization of these values.

In the first stage, we apply smoothing across all linear layers using a small calibration set to reduce outliers. The smoothing scale is initialized as described in Eq. (15). In the second stage, through a grid search, we initialize the $\Delta_X$ and $\Delta_W$ values to minimize the L2 loss for the calibration set. The loss for searching $\Delta_X$ is defined as the L2 distance between the FP16 values of $X$ and the quantized $\hat{X}$ for each layer, with the loss for $\Delta_W$ defined similarly. Finally, in the third stage, we sequentially tune $s$, $\Delta_X$, and $\Delta_W$ to minimize the quantization error, starting from the earlier blocks and updating the subsequent ones, using gradient descent. Specifically, the quantization error is defined as the cosine similarity between the FP16 and quantized block outputs. Throughout this process, the quantized integer weights $\bar{W}$ remain fixed, while only the $s$, $\Delta_X$, and $\Delta_W$ are learned. This approach initializes each value close to optimal in a layer-wise manner and then updates only those parameters with gradient descent in a block-wise manner for a few steps. PTQ4VM generates any quantized model within 15 minutes on a single RTX 3090 GPU.

The optimally learned quantization parameters play a key role in minimizing quantization errors and preserving network performance, even in the presence of outliers and long-tailed distributions. After training, these parameters facilitate acceleration using low-precision arithmetic, maximizing performance gains on actual GPU hardware.

## 6. Experiments

In order to demonstrate the superiority of the PTQ4VM, we conducted comprehensive experiments across various computer vision tasks, including image classification, object detection, and instance segmentation. We employed MinMax quantization, as described in Section 2.3, and SmoothQuant, detailed in Section 2.4, as our baselines. Both weights and activations of linear layers were quantized, with the notation W8A8 representing 8-bit weights and 8-bit activations, respectively.

### 6.1. Quantization Setting

For all tasks, we used the same calibration sets to minimize bias. For the Image Classification task, we randomly sampled 256 images from the ImageNet-1K [30] training set. For the Object Detection and Instance Segmentation tasks, we randomly sampled 16 images from the MSCOCO-2017 [21] training set as the calibration set. Detailed experimental configurations are provided in the supplementary.

### 6.2. Image Classification

To validate the universal applicability of our proposed method across various Visual Mamba backbones, we compared PTQ accuracy for the four architectures presented in Fig. 2. Table 2 shows the top-1 accuracy results on the ImageNet-1K validation set. PTQ4VM consistently achieved significantly higher accuracy compared to other methods across all backbones and quantization options.

The experimental results for the Vim family with CLS tokens reveal significant accuracy degradation when using MinMax, even at W8A8, as it fails to account for token-wise variance. Similarly, SmoothQuant struggles to retain essential CLS token information due to the same issue, leading to a 29.6% accuracy loss at 6-bit for Vim-B. In contrast, PTQ4VM addresses all the challenges posed by Visual Mamba, demonstrating strong performance with less than 0.5% accuracy loss at 8-bit for Vim-T/S and LocalVim$^{\dagger}$, while maintaining acceptable quality at 4-bit, where other models fall short. Despite Vim-B having higher FP16 accuracy than Vim-S, its quantized performance lags across all methods, primarily due to outliers exceeding magnitudes of 20K. These findings suggest that Visual Mamba models

| Model | Method | Top-1 Accuracy (%) | | | |
|---|---|---|---|---|---|
| | | FP16 | W8A8 | W6A6 | W4A4 |
| Vim-Ti | MinMax | 76.1 | 57.8 | 1.7 | 0.1 |
| | SmoothQuant | | 74.7 | 54.5 | 0.1 |
| | PTQ4VM (ours) | | **75.8** | **73.9** | **56.4** |
| Vim-S | MinMax | 80.5 | 79.4 | 27.3 | 0.1 |
| | SmoothQuant | | 80.1 | 73.7 | 0.2 |
| | PTQ4VM (ours) | | **80.5** | **79.7** | **69.6** |
| Vim-B | MinMax | 81.9 | 75.8 | 0.5 | 0.1 |
| | SmoothQuant | | 79.9 | 52.3 | 0.1 |
| | PTQ4VM (ours) | | **80.3** | **79.7** | **55.6** |
| VMamba-T | MinMax | 82.6 | 82.6 | 81.6 | 1.2 |
| | SmoothQuant | | 82.6 | 81.8 | 1.7 |
| | PTQ4VM (ours) | | **82.6** | **82.4** | **81.3** |
| VMamba-S | MinMax | 83.6 | 83.6 | 82.8 | 1.1 |
| | SmoothQuant | | 83.6 | 83.6 | 4.5 |
| | PTQ4VM (ours) | | **83.6** | **83.6** | **83.5** |
| VMamba-B | MinMax | 83.9 | 83.6 | 73.9 | 0.3 |
| | SmoothQuant | | 83.8 | 83.3 | 1.2 |
| | PTQ4VM (ours) | | **83.9** | **83.9** | **83.5** |
| LocalVim-T$^\dagger$ | MinMax | 78.1 | 76.2 | 42.4 | 0.1 |
| | SmoothQuant | | 76.7 | 52.6 | 0.2 |
| | PTQ4VM (ours) | | **77.6** | **76.1** | **55.0** |
| LocalVim-T | MinMax | 76.2 | 75.6 | 62.2 | 0.4 |
| | SmoothQuant | | 75.9 | 65.5 | 0.7 |
| | PTQ4VM (ours) | | **76.2** | **75.7** | **67.2** |
| LocalVim-S | MinMax | 81.1 | 80.9 | 63.5 | 0.2 |
| | SmoothQuant | | 81.0 | 69.7 | 0.6 |
| | PTQ4VM (ours) | | **81.1** | **80.5** | **64.5** |
| LocalVMamba-S | MinMax | 83.7 | 83.5 | 80.8 | 2.5 |
| | SmoothQuant | | 83.6 | 81.9 | 12.0 |
| | PTQ4VM (ours) | | **83.7** | **83.4** | **82.2** |

Table 2. ImgaeNet Top-1 validation accuracy comparison of quantization methods on various models. LocalVim$^\dagger$ indicates a model that uses the CLS token.

with excessively large activation magnitudes may be inherently vulnerable to compression techniques.

In the VMamba family, all methods show lossless quality at W8A8 compared to FP16, largely due to its significantly smaller activation ranges compared to other model families. This unique characteristic can be linked to the absence of gating functions, as seen in Fig. 2b, which differentiates it from other architectures. However, while other methods struggle with W4A4, only PTQ4VM manages to maintain quality on VMamba-S/B with less than 0.4% degradation, achieving lossless quality compared to FP16 models.

### 6.3. Object Detection and Instance Segmentation

To demonstrate that PTQ4VM performs in downstream tasks, we evaluated its performance in Object Detection and Instance Segmentation tasks. We used the official checkpoint trained with Mask R-CNN [11] 3X MS schedule using VMamba-T backbone for our experiments, evaluating on

| Backbone | Method | Bit | AP | |
|---|---|---|---|---|
| | | | $AP^b$ | $AP^m$ |
| VMamba-T | - | FP16 | 47.0 | 42.3 |
| | MinMax | W8A8 | 46.9 | 42.2 |
| | SmoothQuant | W8A8 | 46.9 | 42.2 |
| | PTQ4VM (ours) | W8A8 | **47.0** | **42.3** |
| | MinMax | W6A6 | 46.2 | 41.5 |
| | SmoothQuant | W6A6 | 45.5 | 40.9 |
| | PTQ4VM (ours) | W6A6 | **46.7** | **42.1** |
| | MinMax | W4A4 | 0.3 | 0.3 |
| | SmoothQuant | W4A4 | 0.5 | 0.4 |
| | PTQ4VM (ours) | W4A4 | **45.1** | **40.7** |

Table 3. Results of object detection and instance segmentation.

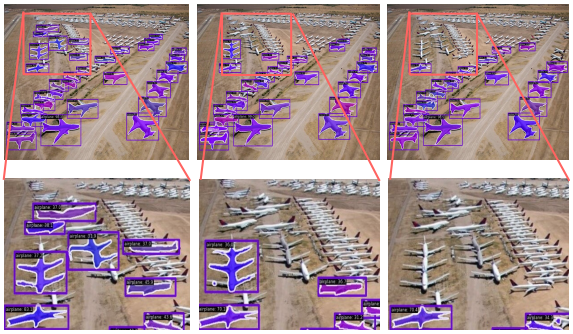



Figure 7. Qualitative results for the object detection and instance segmentation task. Applied W6A6 quantization to VMamba-T.

images cropped to 1280×800. As reported in Table 3, while MinMax and SmoothQuant both fail at W4A4, PTQ4VM demonstrates superior performance with less than 1.9% score degradation. At W6A6, SmoothQuant shows lower scores than MinMax, which can be attributed to the degradation caused by the difficulty imposed on weights by smoothing. In contrast, PTQ4VM achieves near-lossless performance with less than 0.3% $AP^b$ and 0.1% $AP^m$ degradation, due to optimal tuning by JLSS. Notably, the qualitative results in Fig. 7 validate that PTQ4VM maintains the network's quality well, preserving even small details.

### 6.4. Computation Acceleration

To demonstrate the efficiency of PTQ4VM from a hardware acceleration perspective, we implemented CUDA kernels and measured the execution latency. The kernels were built using CUTLASS [14], and all experiments were conducted on a single RTX 3090 with a batch size of 32.

Table 4 presents the end-to-end W4A4 latency when applying per-tensor static, per-token dynamic, and PTS quantization to the token dimension, with SmoothQuant applied by default. The results show that the proposed PTS achieves similar acceleration to per-tensor static across all models, while significantly outperforming per-tensor static in terms of accuracy, as seen in Table 2. In particular, for VMamba-

| Model | Method | Latency (ms) | | Speedup |
|---|---|---|---|---|
| | | FP16 | W4A4 | |
| VMamba-T | Per-tensor static | 60.33 | 33.87 | 1.78× |
| | Per-token dynamic | | 44.13 | 1.37× |
| | PTS (ours) | | 34.00 | 1.77× |
| VMamba-S | Per-tensor static | 118.96 | 68.88 | 1.73× |
| | Per-token dynamic | | 88.55 | 1.34× |
| | PTS (ours) | | 69.30 | 1.72× |
| VMamba-B | Per-tensor static | 165.18 | 89.93 | 1.84× |
| | Per-token dynamic | | 114.87 | 1.44× |
| | PTS (ours) | | 90.30 | 1.83× |
| Vim-T | Per-tensor static | 37.86 | 32.58 | 1.16× |
| | Per-token dynamic | | 39.15 | 0.97× |
| | PTS (ours) | | 32.74 | 1.16× |
| Vim-S | Per-tensor static | 85.43 | 67.75 | 1.26× |
| | Per-token-dynamic | | 75.67 | 1.13× |
| | PTS (ours) | | 67.86 | 1.26× |
| Vim-B | Per-tensor static | 219.48 | 150.02 | 1.46× |
| | Per-token dynamic | | 169.4 | 1.30× |
| | PTS (ours) | | 157.03 | 1.40× |

Table 4. Latency comparison for Visual Mamba backbone. The batch size is 32.

B, per-token dynamic quantization shows a 1.44× latency improvement compared to FP, while PTS achieves even greater acceleration at 1.83×. These improvements are detailed in Fig. 3, where PTS with W4A4 shows a significant improvement in the latency of the linear layer on VMamba-B, ultimately achieving a speedup of 1.83×. These results indicate that PTS can deliver optimal quantization quality with minimal overhead, comparable to the per-tensor static method. A comparison of accuracy between PTS and per-token dynamic is provided in Table 5.

### 6.5. Ablation Study

To evaluate the impact of each component in our proposed PTQ4VM, we conducted an ablation study (Table 6).

First, incorporating PTS into Vim-Ti allows us to account for token-wise variance, leading to notable accuracy improvements. Specifically, this method enables appropriate step size allocation to the crucial CLS token, proving particularly effective in models that use CLS tokens. As a result, we observe accuracy gains of 17.1% at 6-bit and 1% at 8-bit in Vim-Ti.

For VMamba, although there is no CLS token, accounting for token-wise variance still enhances accuracy at 6-bit and 4-bit levels. Additionally, by using a grid search to optimize the truncation level and minimize L2 loss at the layer level (labeled with +Truncation), both Vim and VMamba show significant accuracy improvements—approximately 40% and 50%, respectively, at 4-bit. This highlights the importance of proper truncation in static quantization. JLSS, designed to further optimize truncation, yields additional gains of 1.7% and 10% at 6-bit and 4-bit, respectively, while boosting accuracy by 10% in VMamba-T.

| Granularity | Vim-S (%) | VMamba-S (%) |
|---|---|---|
| Per-tensor static | 0.2 | 4.5 |
| Per-token dynamic | **72.6** | 82.4 |
| PTS | 69.6 | **83.5** |

Table 5. Top-1 accuracy results of INT8 quantization across different activation granularity schemes.

| Model | Method | Top-1 Accuracy (%) | | | |
|---|---|---|---|---|---|
| | | FP16 | W8A8 | W6A6 | W4A4 |
| Vim-Ti | SmoothQuant | 76.1 | 74.7 | 54.5 | 0.1 |
| | + PTS | | 75.7 | 71.6 | 5.4 |
| | + Truncation | | 75.8 | 72.2 | 46.4 |
| | + JLSS (ours) | | **75.8** | **73.9** | **56.4** |
| VMamba-T | SmoothQuant | 82.6 | 82.6 | 81.8 | 1.7 |
| | + PTS | | 82.6 | 82.2 | 19.8 |
| | + Truncation | | 82.6 | 82.3 | 73.3 |
| | + JLSS (ours) | | **82.6** | **82.4** | **81.3** |

Table 6. The effect of each component of PTQ4VM on the ImageNet top-1 validation accuracy.

## 7. Conclusion

In this paper, we identified three key challenges that complicate quantization for Visual Mamba: (i) token-wise variance, (ii) channel-wise outliers, and (iii) the long tail of activation. To address these challenges, we propose PTQ4VM, a post-training quantization technique designed to tackle all three issues. Through Per-Token Static (PTS) Quantization, we effectively address token-wise variance, while the integration of SmoothQuant with PTS mitigates channel-wise outliers. Additionally, by utilizing JLSS, which jointly optimizes smoothing scale and step size, we achieve higher post-quantization quality. To the our best knowledge, this is the first quantization study on Visual Mamba. PTQ4VM can generates any quantized model within 15 minutes and is widely applicable to various Visual Mamba architectures across different downstream tasks, all while enabling faster inference. Our techniques offer up to a 1.83× speedup on real GPUs, broadening the practical application of Visual Mamba backbones.

## Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2019-II191906, RS-2024-00396013, RS-2024-00457882).

## References


[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[2] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 696–697, 2020. 5

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 3

[4] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022. 3

[5] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 3

[6] Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*. 1

[7] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 1, 2, 4

[8] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*. 1

[9] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021. 1, 2

[10] Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Ré. How to train your hippo: State space models with generalized orthogonal basis projections. *arXiv preprint arXiv:2206.12037*, 2022. 2

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 7

[12] Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. *arXiv preprint arXiv:2403.09338*, 2024. 1, 3

[13] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018. 3, 4

[14] Andrew Kerr, Duane Merrill, Julien Demouth, and John Tran. Cutlass 3.5.1. https://github.com/NVIDIA/cutlass, 2017 (accessed September 10, 2024). 7

[15] Seonggon Kim and Eunhyeok Park. Hlq: Fast and efficient backpropagation via hadamard low-rank quantization. *arXiv preprint arXiv:2406.15102*, 2024. 3

[16] Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364, 2024. 3

[17] Changhun Lee, Jun-gyu Jin, YoungHyun Cho, and Eunhyeok Park. Qeft: Quantization for efficient fine-tuning of llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13823–13837, 2024. 3

[18] Changhun Lee, Hyungjun Kim, Eunhyeok Park, and Jae-Joon Kim. Insta-bnn: Binary neural network with instance-aware threshold. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17325–17334, October 2023. 3

[19] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*. 3

[20] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. 5

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 6

[22] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024. 1, 3

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 3

[24] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020. 2, 3

[25] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019. 3

[26] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 2, 3

[27] Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *The 2023 Conference on Empirical Methods in Natural Language Processing*. 1

[28] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023. 1

[29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 3

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 6

[31] Juncheol Shin, Junhyuk So, Sein Park, Seungyeop Kang, Sungjoo Yoo, and Eunhyeok Park. Nipq: Noise proxy-based integrated pseudo-quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3852–3861, 2023. 3

[32] Junhyuk So, Jungwon Lee, Daehyun Ahn, Hyungjun Kim, and Eunhyeok Park. Temporal dynamic quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[33] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 3

[34] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. In *International Conference on Learning Representations*. 3

[35] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023. 2, 3

[36] Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J Crowley. Plainmamba: Improving non-hierarchical mamba in visual recognition. *arXiv preprint arXiv:2403.17695*, 2024. 1

[37] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. 3

[38] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 3

[39] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024. 1, 3