Post-Training Quantization of Image Tokenizers For Autoregressive Image Generation

Sukriti Paul
The University of Maryland, College Park
sukriti5@umd.edu

Abstract

We investigate post-training quantization techniques for optimizing image tokenizers in autoregressive generation models, where tokenizer computation remains a deployment bottleneck. Using the Cosmos tokenizer, we compare per-tensor and logarithmic quantization schemes on artistic images. Logarithmic quantization consistently outperforms per-tensor approaches, maintaining PSNR of 25.15 dB versus 14.84 dB at 6-bit quantization while achieving similar 4.9x compression ratios. Our analysis reveals asymmetric encoder-decoder sensitivity, enabling mixed-precision strategies that achieve up to 7.8x model size reduction while preserving image fidelity.

1. Introduction

The field of autoregressive image generation has recently garnered attention due to its computational efficiency in both FLOPS and inference speed. These models adapt the success of language modeling frameworks to predict sequential image tokens, often circumventing the computational overhead inherent to diffusion-based sampling.

In this paradigm, image patches are tokenized and quantized in continuous or discrete token spaces [1, 4, 7, 9, 10, 12, 16], with discrete approaches often building upon Vector Quantized Variational Autoencoders (VQVAE) [11]. This tokenization is followed by sequential prediction of image tokens, similar to how language models predict the next word in a sentence.

Given the potential of autoregressive models, it becomes crucial to further enhance their inference speed and optimize their computational efficiency. While recent works have explored various approaches to improve inference efficiency - from Li et al.'s [5] direct RGB prediction strategy that bypasses tokenization to MaskGIT's [2] parallel token prediction - the computational bottleneck of image tokenization remains underexplored. Recent advances have brought forth diverse approaches to image tokenization, with systems like HART [15] using hierarchical adaptive token generation and LLamaGen [6] employing multi-scale patch tokenization for enhanced detail preservation. Despite these advances, the impact of aggressive quantization of image tokenizers remains unstudiedin this context. NVIDIA's Cosmos tokenizer [8] has recently established itself as a robust foundation for high-quality image generation, making it the perfect choice for investigating post-training quantization strategies.

Successes in post-training quantization (PTQ) of Large Language Models [3, 13] suggest promising directions for optimizing image generation pipelines. PTQ enables model optimization through weight and activation quantization without requiring additional training data or fine-tuning. PTQ has been extensively studied in the language domain but its application to image tokenizers presents unique challenges and opportunities. These techniques offer dual benefits: accelerated inference speeds and reduced memory footprints, making deployment feasible on resource-constrained devices. In this work, we provide an analysis of static quantization strategies applied to the Cosmos tokenizer, investigating the balance between computational efficiency and perceptual quality in autoregressive image generation.

2. Methodology

2.1. Dataset

We use a subset of 5000 artistic images from the Boldbrush Artistic Image Dataset (BAID) [14]. BAID is suitable for evaluating tokenizer quantization due to its range of artistic styles featuring intricate brush strokes, complex textures, and fine-grained details - elements that are challenging to preserve under aggressive compression schemes.

2.2. Per Tensor Quantization

Our baseline approach employs a static quantization technique inspired by Min-Max quantization. We rescale and normalize weight values between w_{min} and w_{max} to fit within a discrete range determined by the bit width, particularly catering to distributions centered around zero. For

a given tensor w, the quantization process can be formalized as:

$$w_q = \text{round}\left(\frac{w - w_{\text{min}}}{s}\right) \tag{1}$$

where s is the quantization scale factor defined as:

$$s = \frac{w_{\text{max}} - w_{\text{min}}}{2^b - 1} \tag{2}$$

Here, b represents the target bit width, and w_q denotes the quantized tensor. The weights in each layer are flattened into tensors, with quantization parameters computed either globally (axis=-1) or along specific dimensions (axis=0,1) to preserve the structural information of convolutional and linear layers. For tensors containing zero values, we employ a masking mechanism to compute scale factors only over non-zero elements, ensuring robust quantization even in sparse regions.

2.3. Logarithmic Quantization

Logarithmic quantization applies non-uniform quantization levels. This approach works well for weights that span several orders of magnitude, as it allocates more quantization levels to smaller values while maintaining reasonable precision for larger values. For a given tensor w, we first separate the representation of magnitude and sign:

$$w_{\text{sign}} = \text{sign}(w), \quad w_{\text{abs}} = |w| \tag{3}$$

The logarithmic transformation is then applied to the absolute values:

$$w_{\text{log}} = \log(w_{\text{abs}} + \epsilon) \tag{4}$$

where ϵ is a small constant (typically 10^{-6}) to handle values near zero. The quantization process in log-space follows:

$$w_q = \text{round}\left(\frac{w_{\text{log}} - w_{\text{log,min}}}{s_{\text{log}}}\right) \tag{5}$$

where the logarithmic scale factor s_{log} is computed as:

$$s_{\text{log}} = \frac{w_{\text{log,max}} - w_{\text{log,min}}}{2^b - 2} \tag{6}$$

Here, b represents the target bit width, and we reserve one quantization level using $2^b - 2$ instead of $2^b - 1$ to explicitly encode zero values, which cannot be represented in log-space.

This approach provides several advantages over linear quantization: it naturally adapts to the exponential distribution commonly seen in neural network weights, preserves relative precision across different magnitudes, and maintains explicit handling of zero values which are crucial for model sparsity.

3. Experiments

We evaluate the impact of per-tensor quantization on the Cosmos tokenizer by analyzing PSNR, SSIM, and model size reduction across different bit-widths and components (encoder-only, decoder-only, and full model). As shown in Table 1, the decoder exhibits higher resilience to quantization, maintaining PSNR values > 25 at 6 bits, while the encoder requires 8 bits for comparable quality. SSIM results confirm this pattern, with all configurations achieving structural preservation (SSIM > 0.6) at 8 bits, and decoder-only quantization showing better fidelity at lower bit-widths.

Our mixed-precision analysis (Figure 1a, 1b) reveals that the decoder maintains acceptable performance at 6 bits while the encoder requires 8 bits, enabling asymmetric compression strategies that achieve 2.5× model size reduction without significant quality degradation. This asymmetric sensitivity enables more aggressive compression of the decoder without significant quality loss.

Logarithmic quantization (Figure 1c, 1d) shows advantages over per-tensor quantization, especially in low bit-width regimes. Even at 6-bit encoder and decoder settings, it achieves PSNR values of 25.153, while maintaining structural similarity of 0.673. When reducing decoder precision from 6 to 4 bits, the method maintains strikingly better quality with PSNR of 20.538, demonstrating resilience to aggressive compression. The corresponding SSIM values further support this finding, with graceful degradation even at lower bit-widths.

Visual inspection of the reconstructions in Figure 1 shows logarithmic quantization's superior quality preservation across all bit-width configurations. Both methods struggle at extreme compression (2-bit), however, logarithmic quantization shows markedly better preservation of fine brush strokes and color fidelity at moderate compression levels (6-bit). The advantage is particularly evident in preserving artistic details and texture patterns, with per-tensor quantization showing significant artifacts and color shifts at lower bit-widths.

4. Conclusion

Our analysis of static quantization strategies for the Cosmos image tokenizer reveals advantages of logarithmic over per-tensor quantization. Logarithmic quantization shows superior performance at lower bit-widths, with the quality gap most pronounced at 4-bit decoder settings (PSNR $>$ 20 vs $<$ 10 for per-tensor). While both methods converge at higher bit widths (8+), logarithmic quantization provides substantially better quality under aggressive compression (PSNR $\approx$ 20.5-25.2 vs 9.7-14.9 at 4-6 bits). This efficient bit utilization allows for more aggressive quantization without catastrophic quality loss, making it valuable for deployment. This work focuses on a single tokenizer, Cosmos. Fu-

Table 1. Comparison of Logarithmic and Per-Tensor Quantization Methods

Method	Enc Bits	Dec Bits	PSNR (dB)	SSIM	Original Size (MB)	Quantized Size (MB)	Comp. Ratio
Logarithmic	8	8	**26.54**	**0.686**	152.52	60.54	2.52x
Per-Tensor	8	8	25.91	0.639	152.52	60.19	2.53x
Logarithmic	6	6	**25.15**	**0.673**	152.52	30.92	4.93x
Per-Tensor	6	6	14.84	0.305	152.52	31.12	4.90x
Logarithmic	6	4	**20.54**	**0.593**	152.52	19.43	7.85x
Per-Tensor	6	4	9.69	0.119	152.52	19.51	7.82x
Logarithmic	2	2	6.55	**0.139**	152.52	2.56	59.61x
Per-Tensor	2	2	**7.83**	0.022	152.52	3.19	47.88x
Logarithmic	12	8	**26.75**	**0.687**	152.52	73.25	2.08x
Per-Tensor	12	8	25.90	0.640	152.52	72.83	2.09x
Logarithmic	4	12	**23.71**	**0.556**	152.52	58.34	2.61x
Per-Tensor	4	12	8.34	0.160	152.52	61.96	2.46x

ture analysis could extend to other tokenizers and explore advanced PTQ techniques, such as per-channel clustering, k-means quantization, power-of-two quantization, and additional optimizations to further improve model efficiency and deployment scalability.

References

[1] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023. 1

[2] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022. 1

[3] 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. 1

[4] Jiatao Gu, Yuyang Wang, Yizhe Zhang, Qihang Zhang, Dinghuai Zhang, Navdeep Jaitly, Josh Susskind, and Shuangfei Zhai. Dart: Denoising autoregressive transformer for scalable text-to-image generation. *arXiv preprint arXiv:2410.08159*, 2024. 1

[5] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*, 2024. 1

[6] Jie Liao, Xiaoyang Hu, Daquan Shi, Wenhan Liu, Chi-Wing Fu, and Jiaya Zhang. Llamagen: A visual foundation model for nighttime image generation from text. *arXiv preprint arXiv:2401.05252*, 2024. 1

[7] Zhuoyan Luo, Fengyuan Shi, Yixiao Ge, Yujiu Yang, Limin Wang, and Ying Shan. Open-magvit2: An open-source project toward democratizing auto-regressive visual generation. *arXiv preprint arXiv:2409.04410*, 2024. 1

[8] NVIDIA Research. Cosmos: An open source tokenizer for image generation. https://github.com/NVIDIA/Cosmos-Tokenizer, 2024. GitHub repository. 1

[9] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024. 1

[10] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. 1

[11] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 1

[12] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. 1

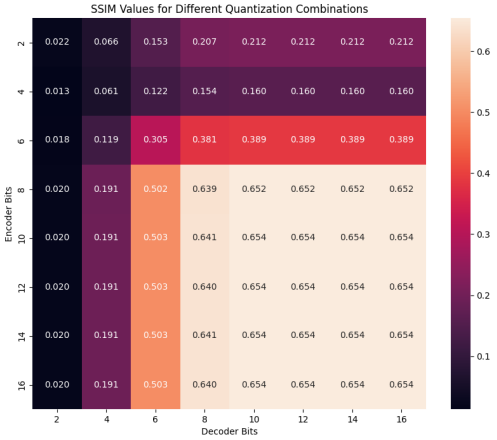
[13] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022. 1

[14] Ran Yi, Haoyuan Tian, Zhihao Gu, Yu-Kun Lai, and Paul L Rosin. Towards artistic image aesthetics assessment: a large-scale dataset and a new method. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22388–22397, 2023. 1

[15] Tiankai Yu, Chenlin Liang, Xiufeng Chen, Junhui Han, Yupeng Du, Wenbo Zhao, Thomas H Li, Jinjin Gu, and Radu Timofte. Hart: Hierarchical adversarial representation training for image synthesis. *arXiv preprint arXiv:2403.03348*, 2024. 1

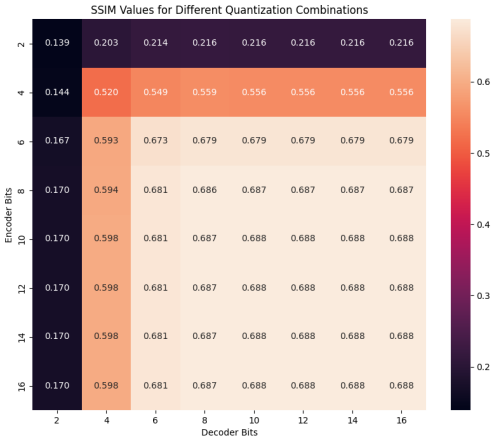
[16] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*, 2024. 1

(a) Per-tensor quantization: PSNR values across different encoder-decoder bit combinations.



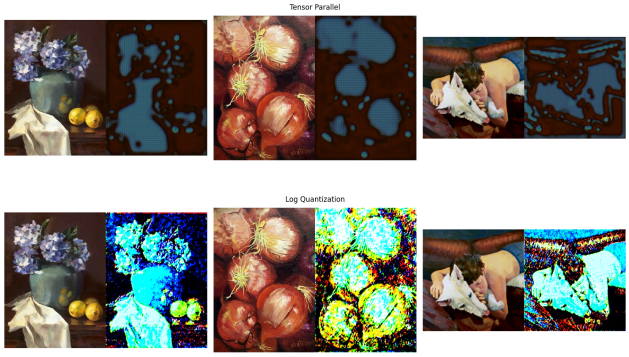
(b) Per-tensor quantization: SSIM values showing structural similarity preservation.

(c) Logarithmic quantization: PSNR values showing sensitivity to encoder precision.

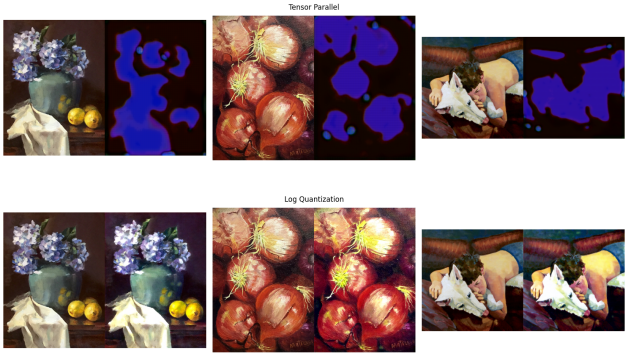


(d) Logarithmic quantization: SSIM values showing quality transitions.

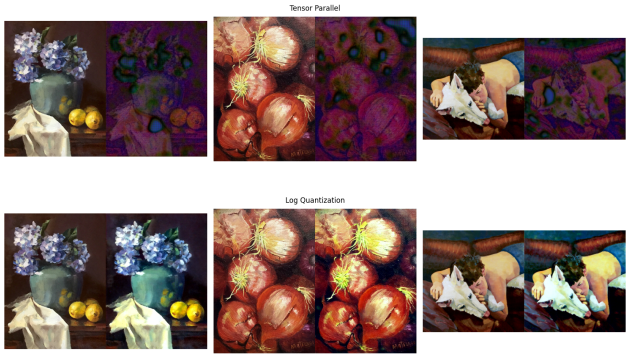
Figure 1. Comparison of PSNR and SSIM metrics between per-tensor (top) and logarithmic (bottom) quantization across different encoder-decoder bit-width combinations. Both methods show encoder sensitivity at lower bits, but logarithmic quantization maintains better quality.



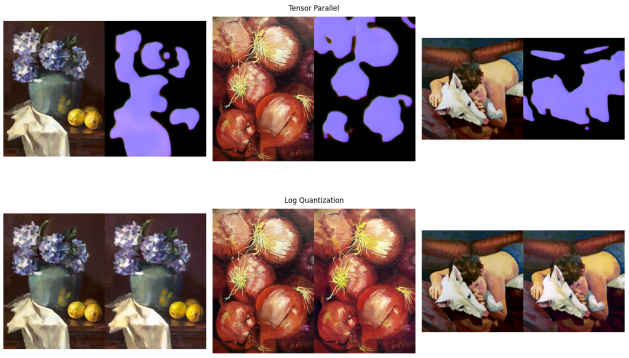
(a) 2-bit E, 2-bit D



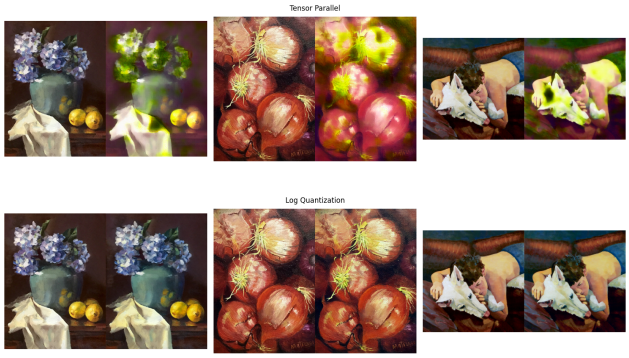
(b) 4-bit E, 4-bit D



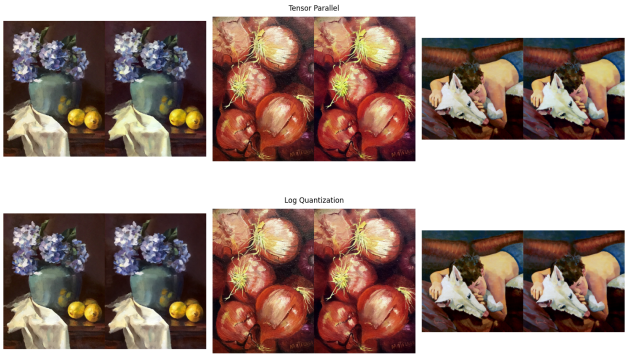
(c) 6-bit E, 4-bit D



(d) 4-bit E, 6-bit D



(e) 6-bit E, 6-bit D



(f) 8-bit E, 8-bit D

Figure 2. Qualitative comparison between per-tensor and logarithmic quantization across different bit-width configurations (E: encoder, D: decoder). Logarithmic quantization maintains better visual fidelity across all compression levels, particularly at lower bit-widths.