Diffusia: Quantization-Aware Training for Diffusion Models

Mohd Arham
Department of Artificial Intelligence & Data Science
Vivekananda Institute of Professional Studies (VIPS TC)
Delhi, India
arham@example.com

***Abstract*—This paper studies post-training and quantization-aware strategies for compressing diffusion models — specifically Stable Diffusion v1.5 — to enable efficient inference on resource-constrained edge devices. We design and implement a Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) workflow targeted at the U-Net backbone, evaluate the impact of precision reduction (FP32, FP16, INT8, mixed precision) on generative quality (FID, SSIM, LPIPS) and computational performance (latency, memory, throughput), and present practical deployment recommendations. Experiments on a laptop-class NVIDIA RTX 2050 (4 GB) demonstrate that INT8 quantization reduces model size and GPU memory use significantly while retaining acceptable image quality; QAT further improves perceptual metrics compared to PTQ. We report Pareto trade-offs and layer-sensitivity analyses that inform mixed-precision strategies for edge deployment.**

***Index Terms*—Stable Diffusion, Post-Training Quantization, Quantization-Aware Training, U-Net, Edge Deployment, Model Compression, INT8**

I. Introduction

Diffusion models are a class of generative models that learn to generate data by reversing a gradual noising process, effectively denoising random noise into coherent samples. They have produced high-quality outputs across images, audio, and text, and enable tasks such as text-to-image generation, image-to-image translation, inpainting, and super-resolution. Stable Diffusion, a latent diffusion variant with CLIP conditioning, has democratized high-resolution image generation due to its efficiency and modularity.

However, diffraction-based pipelines face computational challenges that limit deployment on edge devices: large model sizes, high memory footprints, iterative inference with many denoising steps, and energy consumption. Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) are promising compression approaches to address these constraints without full retraining. This work focuses on applying and evaluating PTQ and QAT on the U-Net backbone of Stable Diffusion v1.5 to determine practical accuracy-efficiency trade-offs for edge deployment.

A. Problem Statement

Deploying diffusion models to edge devices is limited by:

- Substantial GPU/VRAM demands and latency constraints.
- The need for compression without expensive retraining.
- The lack of comprehensive studies on PTQ/QAT effects on diffusion U-Net architectures and cross-attention sensitivity.

B. Contributions

1) A PTQ and QAT framework applied to the U-Net backbone of Stable Diffusion v1.5 (FP32 $\rightarrow$ FP16/INT8/mixed).
2) Quantitative evaluation across FID, SSIM, LPIPS, latency, GPU memory, and throughput.
3) Layer-wise sensitivity analysis and Pareto trade-off curves leading to deployment recommendations (mixed-precision strategies).
4) Demonstration of practical inference on a laptop-class GPU (RTX 2050, 4 GB).

II. Related Work

Denoising Diffusion Probabilistic Models (DDPM) introduced iterative denoising, often requiring hundreds of steps. DDIM and latent diffusion methods accelerate sampling or move diffusion into compressed latent spaces. Prior compression approaches include pruning, knowledge distillation, low-rank adaptation, and quantization. While PTQ works well for discriminative models, generative diffusion models present unique challenges: quantization error accumulation across timesteps, dynamic activation distributions, and cross-attention sensitivity. Recent works propose timestep-aware calibration and careful noise accounting for diffusion quantization.

III. Background

A. Forward and Reverse Diffusion

The forward process gradually injects noise; notation commonly used:

$$dX_t = -\tfrac{1}{2} X_t \, dt + dW_t, \tag{1}$$

and the reverse denoising model is trained by minimizing the noise-prediction loss:

$$\mathcal{L}_{\text{diffusion}} = \mathbb{E}_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]. \tag{2}$$

B. Stable Diffusion and U-Net

Stable Diffusion performs diffusion in a VAE latent space, conditioning via CLIP embeddings using cross-attention within a U-Net architecture. Cross-attention computes

$$\text{CrossAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \quad (3)$$

enabling text-image alignment.

IV. Methodology

This section summarizes the experimental environment and the quantization workflow used in our study. The content and experimental design follow the uploaded draft (DIFFUSIA_DRAFT_01.docx). :contentReference[oaicite:2]index=2

A. Experimental Setup

a) Hardware: Representative laptop-class environment selected to emulate constrained edge deployment:

- OS: Windows 11 (64-bit)
- CPU: Intel Core i5 (multi-core)
- GPU: NVIDIA GeForce RTX 2050 (4 GB GDDR6)
- CUDA Driver: 560.70, CUDA compute capability 12.6 (notes: in-experiment recorded compute capability 12.1 for some libraries)

b) Software: Key libraries and versions used (validated for Python 3.10): `PyTorch 2.5.1`, `diffusers 0.30.3`, `transformers 4.46.0`, `accelerate 0.34.2`, `bitsandbytes 0.43.3`, `optimum 1.21.4`, `safetensors 0.4.4`, `xformers 0.0.26.post1`. The full environment details follow the source document. :contentReference[oaicite:3]index=3

c) Model: Stable Diffusion v1.5 (runwayml/stable-diffusion-v1-5) loaded via Hugging Face Diffusers. The text encoder (CLIP) and VAE decoder remain frozen; U-Net is targeted for quantization.

d) Inference Settings:

- Scheduler: DPMSolverMultistepScheduler
- Inference steps: 25 (reduced from 50 for memory constraints)
- Guidance scale: 7.5
- Seed: 42
- Batch size: 1

Memory optimizations: XFormers attention (when available), attention slicing, and low-CPU-memory loading were enabled.

B. Quantization Workflow

a) Layer selection: Quantizable targets: Conv2D layers, linear projections in attention, multi-head attention weight matrices. Excluded or kept higher precision: timestep embeddings, normalization layers (LayerNorm/InstanceNorm), and some residual path computations due to high sensitivity.

b) Methods: We implement two main strategies:

1) **Static PTQ:** calibrate a representative dataset and convert weights to INT8 (per-channel where applicable) and activations using collected statistics.
2) **QAT:** simulate quantization during fine-tuning (lightweight) to adapt the model parameters to quantization noise.

c) Calibration: A 100-image calibration set (diverse categories) was used to compute activation ranges and percentiles (99.9th percentile calibration used as a balance against outliers).

C. Validation and Testing

Post-conversion checks include tensor shape and dtype verification, numerical stability forward passes, peak-GPU-memory analysis, and test image generation for sanity-checks.

V. Evaluation Protocol

A. Metrics

- **Latency:** end-to-end time (s) per image, averaged over 10 runs (with GPU sync).
- **Model size:** storage of model parameters (MB).
- **GPU memory:** peak VRAM consumption (MB).
- **Throughput:** images per hour.
- **Quality:** FID, SSIM, LPIPS measured over test images.

B. Reproducibility

All experiments were executed with fixed seed 42 and identical prompts (example: “a futuristic city skyline at sunset, cinematic lighting, high detail”).

VI. Results

A. Model size and memory

Table I summarizes the model size reductions.

TABLE I: Model size reduction by precision

Precision	Size (MB)	Reduction
FP32 baseline	4891	—
FP16	2446	50.0%
INT8 (PTQ)	1223	75.0%

Peak GPU memory usage (reported against a 4 GB device) is shown in Table II.

TABLE II: Peak GPU memory (MB) and utilization

Precision	Peak Memory (MB)	% of 4GB
FP32 baseline	3847	96.2%
FP16	2156	53.9%
INT8 (PTQ)	1524	38.1%

B. Inference speed

Table III gives wall-clock latency and throughput.

TABLE III: Latency and throughput (batch size 1)

Precision	Latency (s)	Throughput (images/hr)
FP32 baseline	38.2	94.2
FP16	24.5	146.9
INT8 (PTQ)	18.7	192.5

TABLE IV: Quality metrics (100 test images)

Metric	FP32	FP16	INT8 PTQ	INT8 QAT
FID (lower better)	0.00	0.18	0.34	0.28
SSIM (higher better)	1.00	0.968	0.942	0.955
LPIPS (lower better)	0.00	0.052	0.087	0.071

C. Image quality

Aggregated perceptual metrics over 100 test images are shown in Table IV.

QAT consistently improves over PTQ, bringing SSIM back above the 0.95 threshold in many categories. Category-wise benchmarks (landscape, portraits, abstract, technical) show portraits and technical images are most sensitive to quantization.

VII. Discussion

A. Layer-wise sensitivity

Attention and normalization layers are the most sensitive to precision reduction. Convolution layers are comparatively robust and constitute prime targets for INT8 conversion. Thus, a mixed-precision scheme (Conv INT8 + Attention FP16) often yields the best trade-off.

B. Trade-offs

We observe a logarithmic-like quality-efficiency trade-off; INT8 provides a large efficiency gain for modest perceptual quality loss. Beyond INT8, degradation accelerates.

C. Limitations

- Calibration dataset size impacts PTQ quality; 100 images were used for speed, but 500–1000 samples reduce FID further.
- Attention-layer quantization remains problematic with static calibration due to dynamic activation ranges.
- Some perceptual metrics incompletely capture human-perceived quality; subjective evaluation remains valuable.

VIII. Practical Recommendations

- **Mobile / very constrained devices:** Full INT8 with expected quality trade-offs.
- **Laptop GPUs (4–8 GB):** Mixed-precision — INT8 for convolutions, FP16 for attention and norms.
- **Workstations:** FP16 is recommended for higher fidelity.
- **Calibration:** Use 100–500 semantically stratified images; expand to 1000 only if resources permit.

IX. Conclusion

This work demonstrates that PTQ and QAT applied to Stable Diffusion's U-Net achieve substantial model size and memory reductions enabling inference on laptop-class GPUs. Quantization-aware fine-tuning improves perceptual metrics over static PTQ. Mixed-precision strategies that keep attention and normalization layers at higher precision are practical for production deployment. Future work will investigate timestep-aware calibration and improved attention quantization strategies.

Acknowledgments

We thank the authors of the open-source Stable Diffusion and Hugging Face Diffusers projects. The experiments and the draft text are based on the user's uploaded document (DIFFUSIA_DRAFT_01.docx). :contentReference[oaicite:4]index=4

References

[1] J. Ho, A. Jain, P. Abbeel, "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.
[2] J. Song, S. Ermon, "Denoising Diffusion Implicit Models," *ICLR*, 2021.
[3] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," *CVPR*, 2022.
[4] S. Choukroun et al., "A Survey on Neural Network Quantization," *arXiv:XXXX.YYYY*.