Vision Model Optimization with Quantization and Efficient Attention

Puneeth Kotha   Jayraj Pamnani
New York University
pk3058@nyu.edu   jmp10051@nyu.edu

***Abstract*—Vision Transformers (ViTs) achieve strong performance in image classification but remain challenging to deploy due to high memory consumption and expensive attention operations. This work studies the impact of low-bit weight quantization and efficient attention kernels on the deployability of a ViT-L/16 model. We evaluate controlled variants combining 4-bit and 8-bit quantization with standard scaled dot-product attention (SDPA) and FlashAttention-2. Results show that 4-bit quantization yields large compression and VRAM savings with a moderate Top-1 accuracy drop, while 8-bit quantization preserves baseline accuracy with substantial memory reduction. Layer-wise sensitivity analysis reveals that deeper encoder layers (20–23) are most susceptible to quantization errors. This motivates targeted mixed-precision strategies over simple heuristics. Overall, the study highlights practical accuracy–efficiency trade-offs for deploying large ViTs on resource-constrained hardware.**

***Index Terms*—Vision Transformer, quantization, FlashAttention-2, SDPA, LoRA, GPU profiling, model compression**

I. Introduction

A. Background and Motivation

Vision Transformers (ViTs) have become a strong alternative to convolutional networks for image classification due to their scalability and ability to model long-range interactions [2]. However, large variants such as ViT-L/16 incur substantial memory and compute overhead, making deployment difficult under strict VRAM and latency constraints. As foundation models grow in size, the ability to run inference on consumer-grade hardware (e.g., single GPUs with 16GB–24GB VRAM) becomes increasingly critical for democratizing access to state-of-the-art vision capabilities.

B. Problem Statement

We study whether two systems-oriented techniques can improve deployability without changing the ViT architecture: (i) low-bit weight quantization (8-bit and 4-bit), and (ii) efficient attention kernels, specifically FlashAttention-2 compared with standard scaled dot-product attention (SDPA). A key challenge is understanding how these optimizations interact—specifically, whether the computational overhead of dequantizing weights at runtime negates the speed benefits of optimized attention kernels.

C. Objectives and Scope

The model architecture and training pipeline are kept fixed across all experiments. Observed differences in accuracy and efficiency are attributed to numerical precision and attention-kernel implementation. The goal is to characterize trade-offs across accuracy, memory footprint, and inference efficiency.

II. Literature Review

A. Review of Relevant Literature

ViT showed that transformer backbones can perform strongly on image classification when trained at scale [2]. Low-bit quantization is widely used to reduce model size and memory traffic; practical implementations such as bitsandbytes provide 8-bit and 4-bit weight-only quantization for PyTorch [7]. PyTorch exposes optimized attention backends through SDPA [4], while FlashAttention-2 improves attention performance through IO-aware computation and work partitioning [3].

Parameter-efficient fine-tuning methods such as LoRA stabilize adaptation by training low-rank adapters while freezing base weights [5]. QLoRA popularized NF4 quantization and adapter-based fine-tuning principles that motivate 4-bit strategies in transformer settings [6].

B. Quantization Paradigms

It is important to distinguish between Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ). While QAT generally yields higher accuracy by simulating quantization errors during training, it requires expensive retraining of the full model. In contrast, PTQ approaches—such as the weight-only quantization used in this work—offer a more accessible path for adapting pre-trained models to resource-constrained environments without extensive computational resources.

C. Identification of Gaps

Quantization and efficient attention kernels are often evaluated separately. Their combined deployment impact on a fixed large ViT backbone—including VRAM usage, latency, and kernel-level behavior under low precision—is less commonly reported in a controlled comparison.

III. Methodology

A. Data Collection and Preprocessing

Experiments use the Oxford Flowers-102 dataset [1]. All variants share identical preprocessing and augmentation (resizing, normalization, and standard augmentations such as random crops and horizontal flips). Dataset splits and data-loader settings are fixed across experiments.

B. Model Selection

All experiments use ViT-L/16 as the shared backbone. The model contains a patch embedding layer, positional encodings, a stack of transformer encoder blocks (multi-head self-attention and MLP with residual connections and layer normalization), and a linear classification head.

C. Optimization Procedures

This project primarily optimizes *inference efficiency* (memory and latency) while maintaining classification quality.

- **Weight-only quantization:** 8-bit (int8) and 4-bit (NF4) applied to transformer encoder weights via bitsandbytes [7]. Activations remain in higher precision.
- **Efficient attention:** SDPA baseline versus FlashAttention-2; the attention formulation is unchanged, only the kernel implementation differs.
- **Stabilization:** Parameter-efficient fine-tuning using LoRA-style adapters, with base weights frozen [5].

D. LoRA Configuration

To stabilize training under low precision, we employed Low-Rank Adaptation (LoRA). We injected trainable rank decomposition matrices into the query (W_q) and value (W_v) projection layers of the attention mechanism. The pre-trained backbone weights were frozen in their quantized state (4-bit or 8-bit). This approach significantly reduces the memory footprint of gradients during optimization, allowing for fine-tuning on hardware that would otherwise be unable to accommodate full model training.

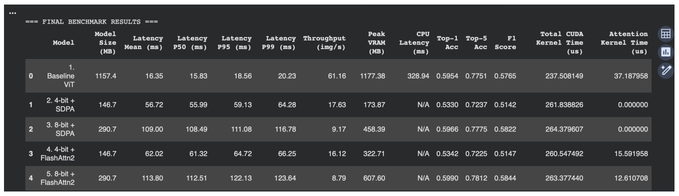
E. Profiling Tools and Methods

Latency and memory measurements are performed on an NVIDIA A100 GPU with batch size 32. In addition to end-to-end inference latency, we examine kernel-level execution behavior using CUDA profiling outputs. This helps distinguish gains from efficient attention implementations from overheads introduced by low-bit inference (e.g., dequantization and data layout conversions).

IV. Experimental Results

A. Performance Comparison

Fig. 1 compares Top-1 accuracy and mean latency across variants. Fig. 2 compares disk model size and peak VRAM usage.



```
=== FINAL BENCHMARK RESULTS ===
```

	Model	Model Size (MB)	Latency Mean (ms)	Latency P50 (ms)	Latency P95 (ms)	Latency P99 (ms)	Throughput (img/s)	Peak VRAM (MB)	CPU Latency (ms)	Top-1 Acc	Top-5 Acc	F1 Score	Total CUDA Kernel Time (us)	Attention Kernel Time (us)
0	1. Baseline ViT	1157.4	16.35	15.83	18.56	20.23	61.16	1177.38	328.94	0.5954	0.7751	0.5765	237.508149	37.187958
1	2. 4-bit + SDPA	146.7	56.72	55.99	59.13	64.28	17.63	173.87	N/A	0.5330	0.7237	0.5142	261.838826	0.000000
2	3. 8-bit + SDPA	290.7	109.00	108.49	111.08	116.78	9.17	458.39	N/A	0.5966	0.7775	0.5822	264.379607	0.000000
3	4. 4-bit + FlashAttn2	146.7	62.02	61.32	64.72	66.25	16.12	322.71	N/A	0.5342	0.7225	0.5147	260.547492	15.591958
4	5. 8-bit + FlashAttn2	290.7	113.80	112.51	122.13	123.64	8.79	607.60	N/A	0.5990	0.7812	0.5844	263.377440	12.610708

Fig. 1. Top-1 accuracy vs. mean latency. Quantized variants can exhibit higher latency due to dequantization overhead despite smaller model size.

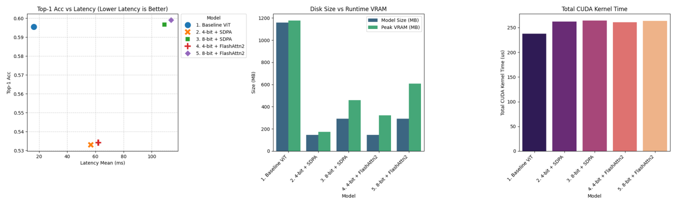


Fig. 2. Disk model size and peak runtime VRAM across model variants. Low-bit quantization yields large memory savings.

B. Analysis of Results

Accuracy vs. memory: 8-bit quantization preserves accuracy close to the baseline while reducing memory usage substantially. 4-bit quantization provides more aggressive compression (over 80% reduction in disk size) but incurs a moderate accuracy drop.

Latency trade-offs: While FlashAttention-2 reduces attention-kernel overhead, end-to-end latency can increase in low-bit settings. This occurs because weight-only quantization requires weights to be dequantized to FP16 before computation, an operation that is often memory-bandwidth bound. For the batch size evaluated (32), this overhead outweighs the speedup from faster attention kernels.

Sensitivity analysis: We perform layer-wise ablation to identify where quantization errors concentrate. Fig. 3 shows that deeper encoder layers (approximately 20–23) are most sensitive to 4-bit quantization.

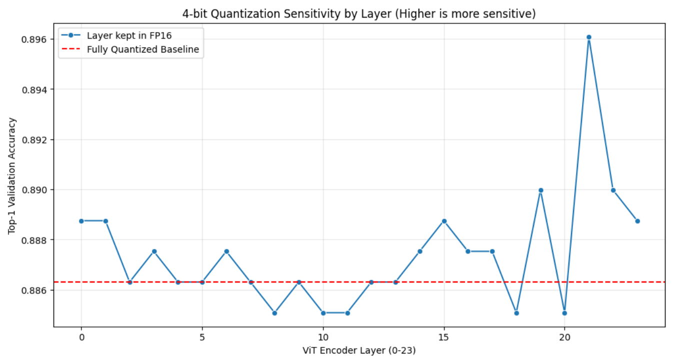


Fig. 3. Layer-wise sensitivity to 4-bit quantization via selective precision ablation.

V. Discussion

A. Interpretation of Results

The results highlight a Pareto frontier: 8-bit quantization is a strong general-purpose setting that maintains accuracy while reducing memory footprint, whereas 4-bit quantization is most attractive under extreme memory constraints. Efficient attention reduces attention overhead, but its impact on end-to-end latency depends on low-bit kernel support and pipeline overheads.

B. Evaluation of Mixed Precision Heuristics

Given the accuracy drop in 4-bit models, we investigated whether a simple mixed-precision heuristic could recover performance. A common intuition in transformer optimization is that the first and last layers are the most critical for feature extraction and prediction, respectively. We evaluated a configuration where the first and last encoder blocks were kept in FP16, while the intermediate blocks were quantized to 4-bit.

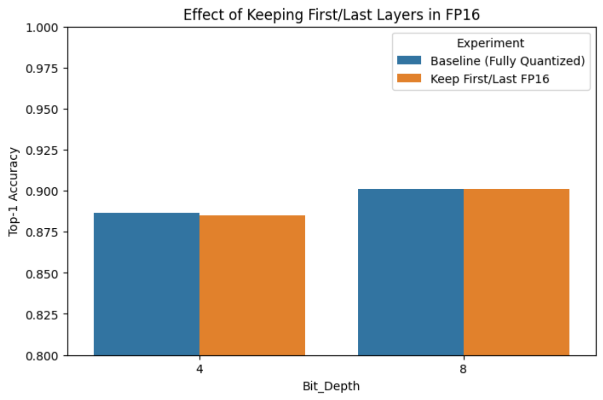


Fig. 4. Effect of keeping First/Last layers in FP16. The improvement over the fully quantized baseline was negligible.

As shown in Fig. 4, this heuristic yielded negligible accuracy improvement (less than 0.1%) compared to the fully quantized baseline. This result is consistent with our sensitivity analysis (Fig. 3), which indicates that the most critical layers for this architecture and dataset are the deeper layers (20–23), not necessarily the first or last layers. This suggests that effective mixed-precision strategies should be data-driven (based on sensitivity profiling) rather than relying on architectural position heuristics.

C. Comparison with Previous Studies

Our findings align with prior observations that fused attention kernels can reduce attention overhead [4], [3]. However, the latency penalty observed in weight-only quantization highlights a practical limitation: without fully fused low-bit compute paths, the dequantization step can remain a bottleneck.

D. Challenges and Limitations

Latency and memory results depend on hardware and batch regime. This work focuses on weight-only quantization; activation quantization and end-to-end integer inference are not explored. Results are reported on a single dataset and may vary across tasks and data distributions.

VI. Conclusion

We evaluated controlled ViT-L/16 variants combining 4-bit/8-bit quantization with SDPA and FlashAttention-2. 8-bit quantization preserves accuracy close to baseline while providing substantial memory savings. 4-bit quantization yields stronger compression at the cost of moderate accuracy degradation. Our results demonstrate that generic mixed-precision heuristics (e.g., preserving first/last layers) are ineffective for this architecture; instead, selective precision should target the highly sensitive deeper layers (20–23). Future work should focus on co-designing quantization kernels with attention mechanisms to reduce the dequantization latency bottleneck.

References

[1] M.-E. Nilsback and A. Zisserman, *102 Category Flower Dataset*, Visual Geometry Group, University of Oxford, 2008. Accessed: Dec. 2025. [Online]. Available: https://www.kaggle.com/datasets/crawford/oxford-102-flowers

[2] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” in *Proc. ICLR*, 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[3] T. Dao, “FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning,” in *Proc. ICLR*, 2024. [Online]. Available: https://arxiv.org/abs/2307.08691

[4] PyTorch Documentation, “Scaled Dot-Product Attention,” Accessed: Dec. 2025. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.functional.scaled_dot_product_attention.html

[5] E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models,” in *Proc. ICLR*, 2022. [Online]. Available: https://arxiv.org/abs/2106.09685

[6] T. Dettmers *et al.*, “QLoRA: Efficient Finetuning of Quantized LLMs,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. [Online]. Available: https://arxiv.org/abs/2305.14314

[7] T. Dettmers *et al.*, “bitsandbytes: k-bit Quantization for PyTorch,” Accessed: Dec. 2025. [Online]. Available: https://github.com/bitsandbytes-foundation/bitsandbytes