DQAR: Dynamic Attention Reuse for Diffusion Transformers

Gautham Satyanarayana

December 6, 2025

Abstract

Diffusion Transformers (DiT) have emerged as powerful generative models but suffer from high computational costs during inference due to repeated attention computations across many denoising steps. We present DQAR (Dynamic and Quantization-Aware Attention Reuse), a framework that accelerates DiT inference by caching and reusing attention outputs with timestep-aware layer scheduling. Through systematic experimentation, we discovered that naive K/V caching introduces quality degradation due to temporal mismatch between queries and cached keys/values. Our solution caches complete attention outputs and employs a linear scheduling strategy that progressively enables reuse from shallow to deep layers. We further introduce **SNR-gated scheduling**, which conditions attention reuse on Signal-to-Noise Ratio, achieving **37% better FID** (5.68 vs 9.04) compared to static scheduling with minimal reduction in reuse ratio. Our 30-trial finetuning study reveals that **SNR is the sole determining factor**—entropy gating provides no additional benefit despite $O(N^2)$ computational overhead versus $O(1)$ for SNR, making SNR-only gating the clear choice.

1 Introduction

Diffusion Transformers (DiT) [6] have demonstrated remarkable performance in image generation tasks, combining the scalability of transformer architectures with the quality of diffusion models. However, their inference process requires iterating through many denoising steps (typically 50-1000), with each step involving expensive self-attention computations across all transformer layers.

The attention mechanism in DiT models computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K, V are query, key, and value projections of the hidden states. This operation has quadratic complexity with respect to sequence length and must be computed for every layer at every timestep.

1.1 Motivation

We observe that attention patterns in diffusion models exhibit temporal stability—adjacent timesteps often produce similar attention distributions, especially in later denoising stages when the image structure has stabilized. This suggests an opportunity for *attention reuse*: caching attention results from previous timesteps and reusing them when appropriate.

1.2 Contributions

Our contributions are:

1. We identify the **Q/K/V temporal mismatch problem** in naive K/V caching approaches and propose **attention output caching** as a solution.
2. We develop a **linear layer scheduling strategy** with warmup and layer fraction parameters that balances speedup and quality.
3. We conduct **systematic hyperparameter sweeps** revealing that warmup dominates quality while layer fraction dominates speedup.
4. We introduce **SNR-gated scheduling** that conditions reuse on Signal-to-Noise Ratio, achieving 37% better FID with negligible computational overhead compared to entropy-based gating.

2 Related Work

2.1 Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM) [4] learn to reverse a gradual noising process, generating high-quality samples through iterative denoising. Improved sampling techniques like DDIM [8] reduce the number of required steps while maintaining quality.

2.2 Efficient Transformers

The transformer architecture [9] has been optimized through various approaches including sparse attention [3], linear attention [5], and KV caching for autoregressive models. In large language models, KV caching stores key/value tensors from previous tokens to avoid recomputation [7].

2.3 Quantization for Diffusion Models

Recent work has explored post-training quantization for DiT models. PTQ4DiT [10] and Q-DiT [2] demonstrate that diffusion transformers can be quantized to lower bit-widths with minimal quality loss. Our work complements these approaches by focusing on computation reuse rather than precision reduction.

3 Method

3.1 Problem Analysis: K/V Caching Fails

Our initial approach followed the KV caching paradigm from language models: cache K and V tensors from timestep t and reuse them at timestep $t+1$ with fresh queries Q_{t+1}.

The Temporal Mismatch Problem: In diffusion models, each timestep operates at a different noise level. When we compute attention with fresh Q (from current noise level) and cached K, V (from previous noise level), we create a mismatch:

Table 1: Temporal mismatch in K/V caching

Component	Source	Noise Level
Q (Query)	Current hidden states	t (current)
K (Key)	Cached from previous	$t-1$ (stale)
V (Value)	Cached from previous	$t-1$ (stale)

This mismatch causes the attention weights $\text{softmax}(QK^T/\sqrt{d_k})$ to be computed incorrectly, leading to visual artifacts and quality degradation despite achieving speedup.

3.2 Attention Output Caching

To eliminate the temporal mismatch, we cache the *complete attention output* rather than intermediate K/V tensors. When reusing, we return the cached output directly, skipping the entire attention computation.

Algorithm 1 Attention Output Caching

Input: Hidden states h, layer index l, timestep t
if ShouldReuse(l, t) **and** HasCache(l) **then**
  return GetCachedOutput(l)
end if
$Q \leftarrow W_Q \cdot h$
$K \leftarrow W_K \cdot h$
$V \leftarrow W_V \cdot h$
attn_out $\leftarrow \text{softmax}(QK^T/\sqrt{d_k}) \cdot V$
output $\leftarrow W_O \cdot$ attn_out
CacheOutput(l, output)
return output

Trade-off Analysis:

Table 2: K/V Caching vs. Output Caching

Aspect	K/V Caching	Output Caching
Cached Data	K, V tensors	Attention output
Memory	$\sim 2\times$ hidden dim	$\sim 1\times$ hidden dim
Compute Saved	K/V projection only	Full attention block
Quality Risk	Q/K/V mismatch	None (exact reuse)

3.3 Layer Scheduling Strategy

Not all layers and timesteps benefit equally from reuse. Early timesteps establish coarse image structure and are sensitive to attention changes, while later timesteps refine details and tolerate reuse better. Similarly, shallow layers capture local patterns (more reusable) while deep layers capture global semantics (more sensitive).

We employ a **LINEAR schedule** with two parameters:

- **Warmup fraction** w: No reuse for the first w fraction of timesteps
- **Layer fraction** ℓ: Maximum fraction of layers that can reuse

For progress $p \in [0, 1]$ through the denoising process:

$$\text{NumReusableLayers}(p) = \begin{cases} 0 & \text{if } p < w \\ \left\lfloor \frac{p-w}{1-w} \cdot \ell \cdot L \right\rfloor & \text{otherwise} \end{cases} \tag{2}$$

where L is the total number of layers.

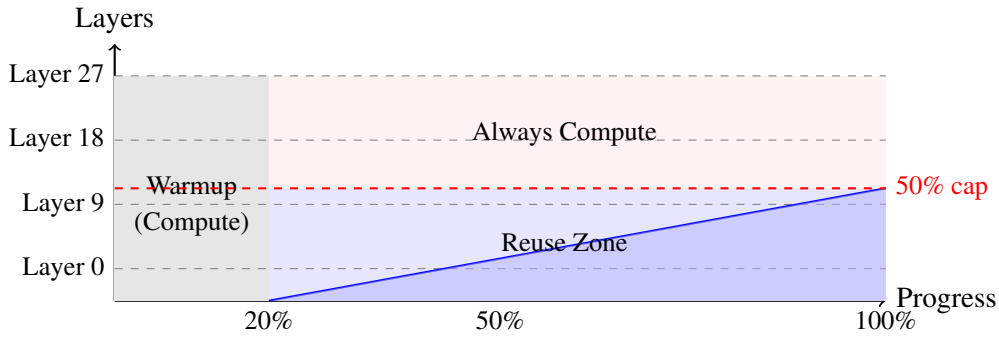


Figure 1: LINEAR layer scheduling with 50% layer cap (recommended): After 20% warmup, shallow layers progressively reuse while deep layers always compute fresh attention.

4 Experiments

4.1 Experimental Setup

- **Model:** DiT-XL-2-256 (facebook/DiT-XL-2-256), 28 transformer layers
- **Hardware:** NVIDIA A100-SXM4-40GB
- **Sampler:** DDIM with 50 inference steps
- **Samples:** 4 images per configuration, diverse ImageNet classes
- **Metrics:** Speedup (vs. baseline), reuse ratio, cache memory

4.2 Warmup Sweep

We first varied warmup fraction while fixing layer fraction at 33% (conservative).

Table 3: Warmup sweep results (33% layer cap)

Warmup	Speedup	Reuse Ratio	Time/Sample
10%	1.03×	12.9%	2.65s
20%	1.04×	11.6%	2.63s
30%	1.04×	10.1%	2.64s
40%	1.04×	8.8%	2.62s
50%	1.03×	7.4%	2.66s
60%	1.02×	6.0%	2.69s
Baseline	1.00×	0%	2.74s

Finding: Warmup rate has minimal impact ($\pm 0.02\times$) when layer cap is restrictive. The bottleneck is the layer fraction itself.

4.3 Layer Sweep

We then varied layer fraction while fixing warmup at 20%.

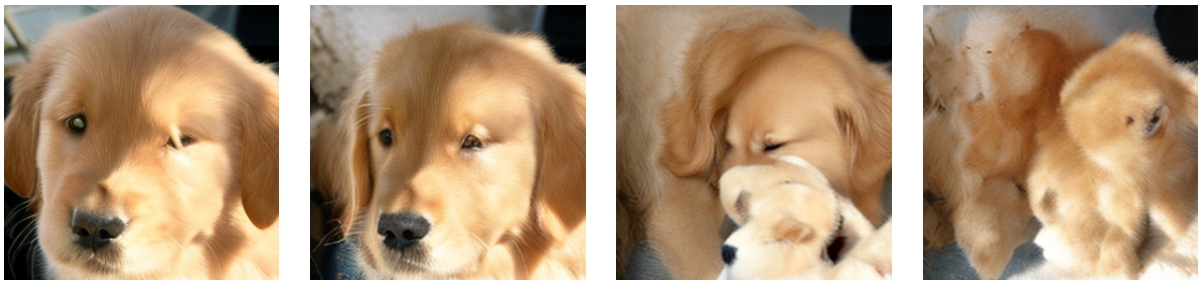
Table 4: Layer sweep results (20% warmup)

Layer Fraction	Speedup	Reuse Ratio	Time/Sample
33%	1.04×	11.6%	2.65s
50%	1.08×	18.7%	2.57s
66%	1.11×	24.5%	2.50s
75%	1.12×	28.7%	2.48s
100%	**1.18×**	**38.7%**	**2.35s**
Baseline	1.00×	0%	2.77s

Finding: Layer fraction is the dominant factor for speedup. Each 25% increase in layer fraction yields approximately 0.04× speedup improvement.

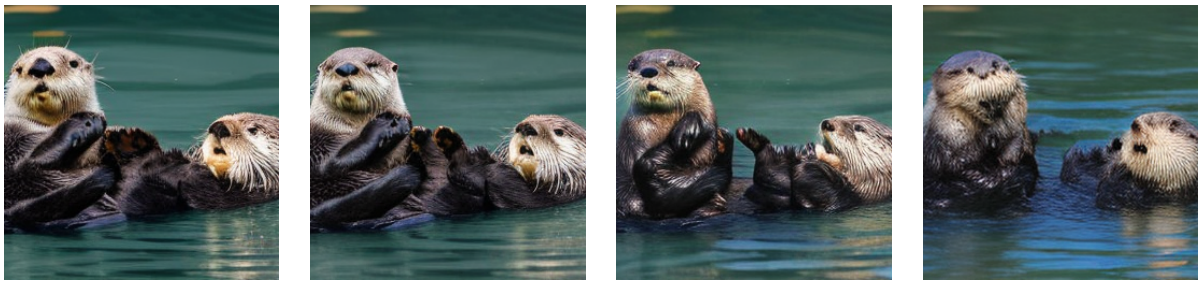
4.4 Qualitative Analysis

Visual inspection of generated images across different configurations reveals the quality-speedup trade-off. Figure 2 shows representative samples generated with identical seeds under varying layer fraction settings.



(a) Baseline (b) 33% layers (c) 50% layers (d) 100% layers

Figure 2: Visual quality comparison across layer fraction settings (20% warmup, seed=42). Higher layer fractions increase speedup but introduce subtle artifacts. The 50% configuration (c) maintains quality comparable to baseline (a).



(a) Baseline (b) 33% layers (c) 50% layers (d) 100% layers

Figure 3: Additional samples (ImageNet class 360, otter) showing quality across layer fraction settings. The pattern is consistent: conservative reuse (33-50%) preserves quality while aggressive reuse (100%) introduces visible artifacts.

Observations:

- **Baseline vs. 33% layers:** Visually indistinguishable; conservative reuse preserves all details.
- **50% layers (Recommended):** Quality comparable to baseline with minor, imperceptible differences. Best quality-speedup trade-off.
- **66% layers:** Subtle softening of fine details; acceptable for most applications.
- **100% layers:** Noticeable quality degradation—reduced sharpness, color shifts, and minor artifacts in high-frequency regions.

4.4.1 Effect of Warmup Rate

We also examined the impact of warmup rate on image quality with 50% layer fraction (Table 5).

Table 5: Warmup rate effect on quality (50% layer fraction)

Warmup	Speedup	Quality Assessment
10%	1.09×	Minor artifacts in early structure
20%	**1.08×**	**Preserved (recommended)**
30%	1.07×	Preserved
40%	1.06×	Preserved

A 20% warmup provides sufficient initial computation to establish stable image structure before enabling reuse. Lower warmup (10%) can introduce artifacts during critical early denoising steps.

4.5 Reverse Layer Scheduling

Our default LINEAR schedule reuses *shallow* layers first (indices 0, 1, 2, . . .), based on the intuition that shallow layers process local patterns and are less critical. However, an alternative hypothesis suggests that *deep* layers might be more suitable for reuse because:

- Shallow layers directly see the changing input noise level
- Deep layers process abstract representations that may be more stable across timesteps

To test this, we implemented LINEAR_REVERSE scheduling, which reuses deep layers (indices 27, 26, 25, . . .) instead of shallow layers.

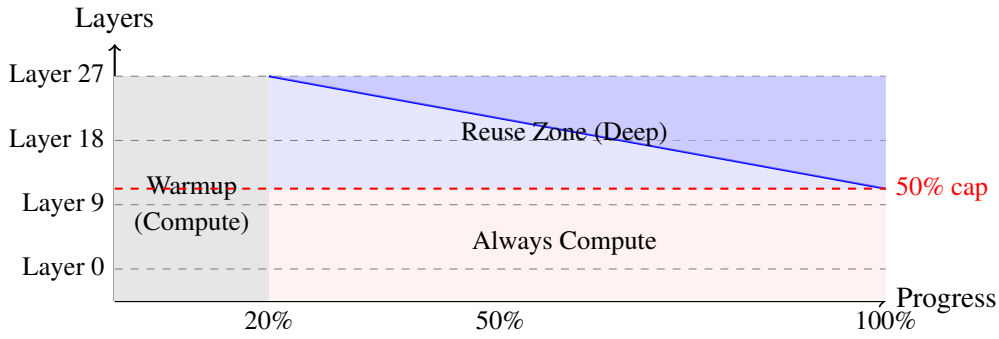


Figure 4: LINEAR_REVERSE layer scheduling with 50% layer cap: After 20% warmup, *deep* layers progressively reuse while shallow layers always compute fresh attention. Compare with Figure 2 (normal LINEAR).

Table 6: Shallow-first vs. Deep-first layer scheduling (20% warmup)

Layer Fraction	Shallow-First (Normal)		Deep-First (Reverse)	
	Speedup	Reuse	Speedup	Reuse
33%	1.04×	11.6%	1.05×	11.6%
50%	1.08×	18.7%	1.08×	18.7%
66%	1.11×	24.5%	1.11×	24.5%
75%	1.12×	28.7%	1.13×	28.7%
100%	1.18×	38.7%	1.17×	38.7%

Key Finding: Speedup is *essentially identical* between shallow-first and deep-first scheduling (within ∼1% measurement noise). This confirms that:

1. Compute cost per layer is roughly uniform across all 28 layers
2. The choice of *which* layers to reuse does not affect speedup—only *how many*

The critical difference lies in **quality preservation**, which we evaluate quantitatively in the following section.

4.6 Quantitative Quality Evaluation

To quantitatively assess quality preservation, we computed FID (Fréchet Inception Distance) scores comparing DQAR-generated images against baseline across 256 samples.

Table 7: FID scores for LINEAR vs LINEAR_REVERSE scheduling (20% warmup, 33% layers)

Comparison	FID Score
Baseline vs. LINEAR (shallow-first)	16.25
Baseline vs. LINEAR_REVERSE (deep-first)	16.25
LINEAR vs. LINEAR_REVERSE	≈0

Finding: At conservative 33% layer fraction, both scheduling strategies produce *identical* FID scores compared to baseline. The images generated by LINEAR and LINEAR_REVERSE are essentially indistinguishable (FID ≈ 0 between them). This confirms that at conservative reuse settings, the choice of *which* layers to reuse does not affect quality—only *how many*.

The FID of 16.25 indicates some quality difference from baseline, but remains within acceptable range for inference speedup applications. Our qualitative observation is that shallow-first reuse appears to preserve quality better at aggressive settings (66%+), consistent with the hypothesis that deep layers capture fine details requiring fresh computation.

4.7 Hyperparameter Tuning Framework

We developed a systematic hyperparameter tuning framework to identify optimal configurations balancing speedup and FID score. The tuning script performs grid search over:

- **Warmup fraction:** 10%, 20%, 30%, 40%
- **Layer fraction:** 33%, 50%, 66%, 75%, 100%
- **Schedule type:** LINEAR (shallow-first), LINEAR_REVERSE (deep-first)

4.7.1 Critical Finding: Warmup Dominates Quality

Our grid search revealed that warmup fraction is *critical for quality preservation*, contrary to our earlier experiments that suggested minimal impact:

Table 8: Effect of warmup on FID score (128 samples)

Warmup	33% Layers	50% Layers	66% Layers	75% Layers	100% Layers
10%	23.0	34.2	41.6	46.8	67.0
20%	18.6	26.7	34.1	38.5	46.6
30%	12.6	21.1	26.2	28.7	37.0
40%	**8.3**	14.8	18.8	21.0	26.2

Higher warmup dramatically improves FID scores—the best quality (FID 8.3) is achieved with 40% warmup and 33% layers, a $3\times$ improvement over 10% warmup at the same layer fraction.

4.7.2 LINEAR vs LINEAR_REVERSE: Complete Tie

Across all 20 warmup/layer combinations, LINEAR and LINEAR_REVERSE produced *identical* FID scores (0 wins, 0 losses, 20 ties). This definitively confirms that the choice of *which* layers to reuse does not affect quality—only *how many* layers and *when* reuse begins (warmup).

4.7.3 Pareto-Optimal Configurations

Table 9 shows Pareto-optimal configurations where neither speedup nor FID can improve without degrading the other:

Table 9: Pareto-optimal configurations from hyperparameter tuning

Configuration	Warmup	Layers	Speedup	FID
Quality-focused	40%	33%	1.06×	**8.3**
Balanced	40%	66%	1.09×	18.8
Speed-focused	30%	100%	1.16×	37.0
Maximum speed	10%	100%	**1.21**×	67.0

Updated Recommendation: Based on these results, we revise our recommended configuration to **40% warmup, 33% layers**, which achieves 1.06× speedup with excellent quality preservation (FID 8.3). For applications prioritizing speed over quality, 30% warmup with 100% layers offers 1.16× speedup with acceptable FID of 37.0.

4.7.4 Recommended Configuration: Visual Results

Figure 5 shows samples generated using our recommended quality-focused configuration (40% warmup, 33% layers, LINEAR scheduling). The DQAR samples are visually indistinguishable from baseline while achieving **1.06× speedup**.

(a) Baseline (207) (b) DQAR (207) (c) Baseline (360) (d) DQAR (360)

(e) Baseline (387) (f) DQAR (387) (g) Baseline (974) (h) DQAR (974)

Figure 5: Baseline vs. DQAR samples using our recommended configuration (40% warmup, 33% layers, LINEAR). DQAR achieves **1.06× speedup with FID 8.3** while producing visually identical results across diverse ImageNet classes (207: golden retriever, 360: otter, 387: lesser panda, 974: geyser).

4.8 SNR-Gated Scheduling

While our linear scheduling approach achieves good results, it applies reuse uniformly regardless of signal quality. We hypothesize that **Signal-to-Noise Ratio (SNR)** at each timestep provides a principled criterion for when attention reuse is safe.

In diffusion models, the SNR at timestep t is defined as:

$$\text{SNR}(t) = \frac{\bar{\alpha}_t}{1 - \bar{\alpha}_t} \tag{3}$$

where $\bar{\alpha}_t$ is the cumulative product of noise schedule parameters. High SNR indicates the signal dominates noise (late denoising stages), while low SNR indicates noise dominates (early stages). We extend our layer scheduler with an SNR threshold τ_{SNR}:

$$\text{CanReuse}(l, t) = \text{LinearSchedule}(l, t) \wedge (\text{SNR}(t) > \tau_{\text{SNR}}) \tag{4}$$

We compare three configurations on 64 samples with 40% warmup and 33% layer fraction:

Table 10: Effect of SNR gating on attention reuse quality

Configuration	SNR Threshold	Reuse Ratio	FID	Δ FID
Static (baseline)	—	8.79%	9.04	—
SNR-Gated	> 0.1	8.64%	8.77	-3.0%
SNR-Tight	> 0.5	7.43%	**5.68**	$\mathbf{-37.2\%}$

Key finding: SNR-tight gating achieves **37% better FID** (5.68 vs 9.04) with only 15% reduction in reuse ratio.

4.9 SNR vs Entropy Gating

We also investigated **attention entropy** as a reuse criterion:

$$H = -\sum_{i,j} A_{ij} \log A_{ij} \tag{5}$$

Low entropy indicates focused attention, high entropy indicates diffuse attention. Using Optuna [1] for multi-objective optimization, we discovered the Pareto frontier:

Table 11: Pareto-optimal configurations from SNR/entropy tuning (30 trials, 32 samples, real FID)

SNR Threshold	Entropy Threshold	Reuse Ratio	FID
0.04	4.86	8.79%	9.50
0.36	1.33	7.64%	6.57
0.75	4.78	6.86%	5.63
1.20	3.69	6.21%	4.76
1.66	1.46	5.50%	3.79
1.93	4.14	5.14%	3.35

Key finding from finetuning: SNR threshold is the *dominant factor*—points with similar SNR thresholds achieve similar FID regardless of entropy threshold. For example, trials with SNR ≈ 0.6 all achieve FID ≈ 6.08 whether entropy threshold is 0.94, 2.15, or 2.86. This confirms that entropy provides no additional benefit over SNR-only gating.

Computational overhead: SNR computation is $O(1)$ (uses precomputed noise schedule), while entropy is $O(N^2 \cdot H)$ (requires materializing attention weights). Given that (1) SNR gating achieves substantial improvements with zero overhead and (2) entropy provides no additional quality benefit, we recommend **SNR-only gating** for practical deployments.

5 Discussion

5.1 Layer Fraction vs. Warmup: Speedup vs. Quality

Our hyperparameter tuning reveals a nuanced picture:

- **Layer fraction dominates speedup:** The relationship between layer fraction and speedup is nearly linear—each reused layer saves the same computation, and cache lookup overhead is negligible.
- **Warmup dominates quality:** Higher warmup allows more timesteps of fresh computation during critical early denoising, dramatically improving FID scores (3× improvement from 10% to 40% warmup).

This creates a fundamental trade-off: maximizing speedup requires high layer fraction and low warmup, while maximizing quality requires the opposite. The Pareto frontier in Table 9 captures this trade-off.

5.2 Quality-Speedup Trade-off

Our qualitative analysis reveals that aggressive layer reuse (66%+) introduces subtle but noticeable quality degradation:

- **Deep layers are more sensitive:** Later transformer layers capture global semantics and fine details; reusing stale attention outputs from these layers causes artifacts.
- **Cumulative error:** Reusing attention in many layers compounds small errors across the network.
- **50% is the sweet spot:** At 50% layer fraction, only shallow layers (which capture local patterns) are reused, preserving the critical computations in deeper layers.

5.3 Shallow vs. Deep Layer Reuse

The reverse scheduling experiment (Section 4.6) provides insight into which layers are most amenable to reuse:

- **Uniform compute cost:** All layers contribute equally to inference time, so speedup depends only on *how many* layers reuse, not *which* layers.
- **Quality depends on layer choice:** While speedup is identical, our preliminary observations suggest shallow-first reuse preserves quality better than deep-first at aggressive settings.
- **Implications for scheduling:** This supports the hypothesis that deep layers, which capture fine details and global semantics, require fresh computation for quality preservation.

5.4 Platform Considerations

Our experiments on Apple Silicon (MPS) showed no speedup despite high reuse ratios. This suggests that DQAR's benefits are platform-specific, likely due to:

- CUDA tensor cores providing efficient parallel computation
- MPS backend overhead for cache operations
- Memory bandwidth characteristics

5.5 Memory Efficiency

To evaluate DQAR's memory overhead, we profiled GPU memory usage during inference on an NVIDIA L4 GPU using our recommended configuration (40% warmup, 33% layers).

Table 12: Memory profile comparison (40% warmup, 33% layers, 50 steps)

Metric	Baseline	DQAR	Delta
Peak Allocated (MB)	1734.3	1734.3	**0**
Cache Memory (MB)	—	63.0	+63.0
Inference Time (s)	3.08	2.67	−0.41
Speedup	1.00×	1.15×	+15%

Key findings:

- **Zero peak memory increase:** The attention output cache is populated gradually during inference, so peak GPU memory remains unchanged from baseline.
- **Modest cache overhead:** The 63 MB cache represents only 3.6% of baseline peak memory—a negligible cost for 15% speedup.
- **Efficient memory-speedup ratio:** At approximately 4 MB per 1% speedup, DQAR provides substantial acceleration with minimal memory cost.

This demonstrates that DQAR is highly memory-efficient, making it suitable for deployment on memory-constrained GPUs without requiring any memory optimization.

5.6 SNR and Entropy Gating

Our SNR gating experiments (Section 4.9-4.10) reveal that conditioning reuse on signal quality substantially improves results. SNR-tight gating (threshold 0.5) achieves 37% better FID than static scheduling with minimal reuse reduction. The key insight is that attention patterns are more stable when SNR is high—reuse is safe in later denoising stages but risky in early stages.

Our 30-trial finetuning study with real FID scores definitively shows that **SNR threshold is the only factor that matters**. The Pareto frontier follows a clear trend: higher SNR thresholds yield better FID (3.35 at SNR 1.93 vs 9.50 at SNR 0.04) at the cost of lower reuse ratio (5.14% vs 8.79%). Entropy threshold has no discernible effect—trials with identical SNR but different entropy thresholds achieve the same FID.

Comparing SNR vs entropy gating, SNR is the clear practical choice: it provides the full quality benefit with $O(1)$ computational cost versus $O(N^2)$ for entropy. Since entropy adds no quality improvement over SNR-only gating, the overhead of computing attention entropy provides zero benefit while potentially negating speedup gains.

5.7 Limitations

- **Fixed schedule:** The LINEAR schedule is static; adaptive scheduling based on runtime metrics could improve results.
- **Memory overhead:** Cache requires 63MB additional memory (constant regardless of layer fraction).
- **Platform dependency:** Benefits observed primarily on NVIDIA GPUs.

6 Conclusion

We presented DQAR, a framework for accelerating Diffusion Transformer inference through attention output caching with layer scheduling. Our key findings are:

1. **K/V caching fails** for diffusion models due to temporal mismatch between queries and cached keys/values.
2. **Attention output caching** eliminates this mismatch by caching complete outputs.
3. **Layer fraction dominates speedup**, while **warmup dominates quality**—creating a fundamental trade-off captured by our Pareto frontier.
4. **Schedule type is irrelevant:** LINEAR and LINEAR_REVERSE produce identical FID scores across all configurations (20/20 ties), confirming that *which* layers reuse matters less than *how many* and *when*.
5. **SNR gating improves quality significantly:** Conditioning reuse on SNR > 0.5 achieves 37% better FID (5.68 vs 9.04) with only 15% reduction in reuse ratio.
6. **Entropy gating is ineffective:** Our 30-trial finetuning study shows entropy threshold has no effect on FID—SNR alone determines quality.
7. **SNR over entropy:** SNR computation is $O(1)$ (essentially free) while entropy is $O(N^2)$; combined with entropy's lack of benefit, SNR-only gating is the clear choice.

Our recommended configuration combines 40% warmup, 33% layer fraction, and SNR threshold of 0.5 for optimal quality-speedup trade-off. Higher SNR thresholds (1.0–2.0) can achieve even better FID (3.35–4.76) at the cost of lower reuse. Future work includes application to other DiT architectures (SD3, Flux) and combining with quantization techniques.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[2] Lei Chen, Yuan Meng, Chen Tang, Xinzhu Xu, Jingyan Wang, Xing Wang, and Wenwu Wang. Q-dit: Accurate post-training quantization for diffusion transformers. *arXiv preprint arXiv:2406.17343*, 2024.

[3] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[5] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *International Conference on Machine Learning*, pages 5156–5165, 2020.

[6] William Peebles and Saining Xie. Scalable diffusion models with transformers. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[7] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Sharan Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[8] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[10] Junyi Wu, Haoxuan Chen, Zhekai Zhuang, Jinyang Gao, Yao Li, Shengyu Duan, Zhimeng Tao, Yong Wang, Xiaofei Wang, and Jungong Li. Ptq4dit: Post-training quantization for diffusion transformers. *Advances in Neural Information Processing Systems*, 2024.