JUSTQUANT: YOU DON'T NEED SMOOTHING, SVD, OR ROTATION FOR 4-BIT ACTIVATION QUANTIZATION

Anonymous authors
Paper under double-blind review

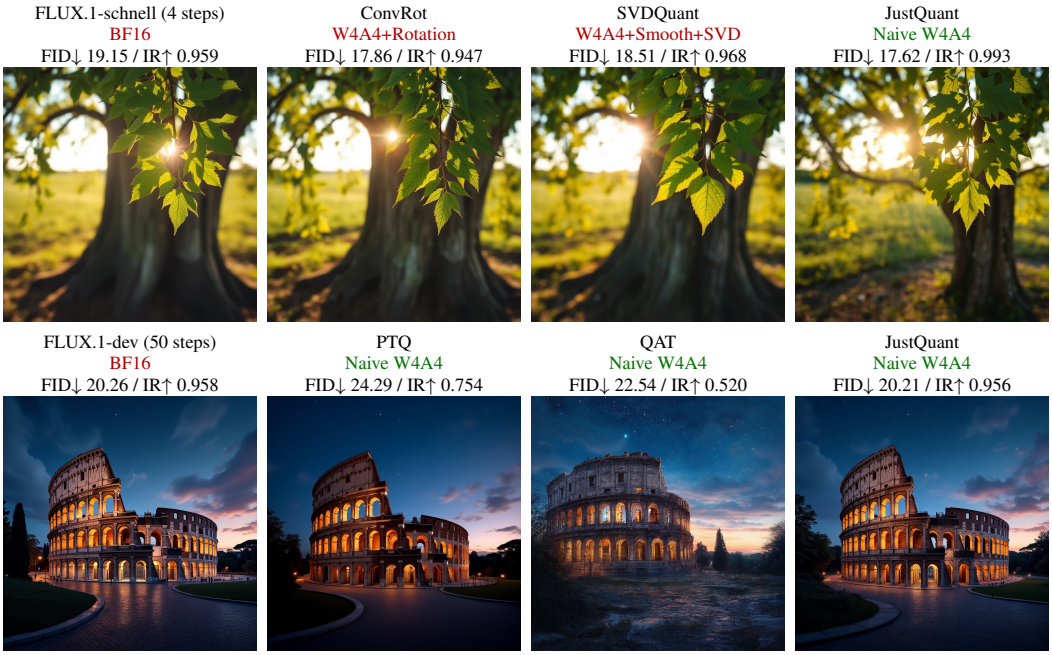

Figure 1: **FLUX-family 12B experiments.** JustQuant pushes naive W4A4 quality close to BF16 with only 18 hours on 4 H200 GPUs, while preserving a clean inference path.

ABSTRACT

Recent generative models have become increasingly powerful, but their inference cost continues to grow. Model quantization is a promising path for compressing these models and accelerating inference. However, when quantization is pushed to 4 bits, activation quantization becomes substantially more difficult than weight quantization. Recent PTQ (post-training quantization) and QAT (quantization-aware training) methods have made progress on 4-bit activation quantization by introducing smoothing, SVD branches, rotations, mixed precision, or advanced formats such as NVFP. Yet these operators and data types impose demanding requirements on inference engines and hardware, limiting the broad adoption of low-precision models. **Can we quantize using only plain low-bit operators?** To answer this question, we propose **JustQuant**, a simple yet effective framework that moves the complexity required for low-bit quantization from deployment-side operators into an acceptable training process. First, we revisit model quantization from the perspective of knowledge distillation, and show that a key reason existing PTQ and QAT methods fail is that they typically exploit supervision at only a single level. We then introduce **Theseus QAD** (quantization-aware distillation), a quantization-aware distillation method that progressively applies multi-level supervision, analogous to the gradual replacement process in the Ship of Theseus. Extensive experiments on DiT and dLLM show two regimes. For smaller models, Theseus QAD can serve as a light warm-up stage that substantially improves QAT with plain operators, while naive QAD can collapse in the same setting. For larger models, Theseus QAD provides a stronger distillation training path than ordinary QAD. Across these regimes, Theseus QAD improves plain quantization while avoiding the complex operators required by many PTQ methods.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

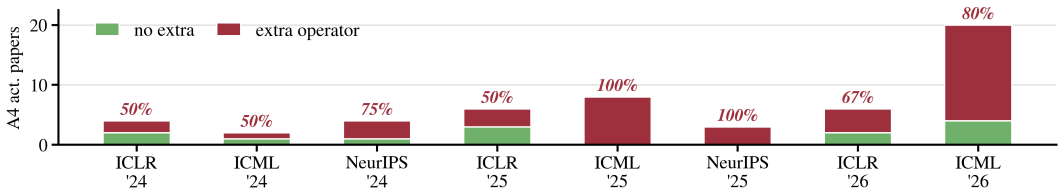

Figure 2: **The false prosperity of low-bit activation quantization.** Across eight recent conferences, ≤4-bit activation papers increasingly rely on extra operators. This deployment-side complexity can obscure the practical speedups promised by low-bit activations.

1 Introduction

Recent 4-bit weight quantization methods can already compress generative models with near-lossless quality and accelerate memory-bound inference scenarios by reducing weight bandwidth (Frantar et al., 2023; Lin et al., 2024b; Zhang et al., 2024; Li et al., 2025; Xue et al., 2026). However, weight-only quantization does not accelerate compute-bound transformer operations whose cost is dominated by activation-weight matrix multiplication. Without activation quantization, these operations cannot be executed as fully low-bit GEMMs. In diffusion transformers and large text-to-image backbones, this cost is repeated across many denoising steps (Peebles & Xie, 2023; Black Forest Labs, 2024). Pushing activations to 4 bits is therefore necessary for practical compute-bound acceleration, but it is also substantially more difficult than weight quantization. This bottleneck is repeatedly observed in recent work on DiT, video DiT, LLM, and FP4/MXFP4 quantization (Feng et al., 2025; Zhang et al., 2026c; Panferov et al., 2025; Zhao et al., 2026b; Shao et al., 2025). This is a fragile regime for generative models: activation errors are injected into every denoising or decoding step, shift internal feature distributions, reduce sample diversity, and can destabilize training.

	Small model		Large model	
Method	Quality	Deploy.	Quality	Deploy.
PTQ	✓	✗	✓	✗
QAT	✓	✗	✗	✓
QAD	✗	✗	✓	✓
JustQuant	✓	✓	✓	✓

Table 1: **Different regimes favor different quantization recipes.**

Existing 4-bit activation quantization methods follow two routes, both limited for plain deployment. PTQ methods often rely on smoothing, rotations, Hadamard transforms, channel permutations, SVD branches, mixed precision, extra normalization, or new formats such as NVFP4, imposing requirements on kernels and hardware (Xiao et al., 2023; Wu et al., 2024; Chen et al., 2025a; Zhao et al., 2025a; Li et al., 2025; Zhang et al., 2026c; Feng et al., 2025; Ashkboos et al., 2024; Liu et al., 2025; Saxena et al., 2025; Huang et al., 2025b; Zhao et al., 2026b; Shao et al., 2025; Xin et al., 2026). QAT appears more compatible with plain operators, but strong results often still use rotation, SVD, NVFP4, or related inference-side techniques (Yang et al., 2026; Wang et al., 2025; Panferov et al., 2025; Xue et al., 2026); without them, plain W4A4 or W1.58A4 models often fail to reach usable quality. Figure 2 summarizes the growing reliance on extra operators across recent conferences, highlighting the gap between low-bit formats and practical deployment. This raises a question:

Can we achieve 4-bit activation quantization using only naive low-bit operators?

We revisit model quantization from the perspective of **Net2Net knowledge distillation** (Du et al., 2024; Feng et al., 2025; Xin et al., 2026). Because the full-precision teacher and low-bit student share the same architecture, quantization naturally exposes supervision signals at different granularities. PTQ uses dense local reconstruction, which is cheap and stable but often leads to complex deployment operators, whereas direct QAT optimizes the final task objective but provides sparse and noisy supervision (Frantar et al., 2023; Wu et al., 2024; Chen et al., 2025a; Jacob et al., 2018; Yang et al., 2026). The intermediate region between these two extremes remains underexplored.

Based on this view, we propose JustQuant, a two-stage framework for training plain 4-bit-activation generative models. Its core method, Theseus QAD, starts from local distillation objectives and progressively expands supervision to longer low-bit computation paths before transitioning to standard end-to-end QAT or QAD. This moves the required complexity into training while leaving deployment as a plain low-bit operator graph.

We evaluate JustQuant in controlled and data-limited regimes. Our main experiments use DiT-XL/2 to study the optimization behavior of plain A4 quantization, and we further extend to larger text-

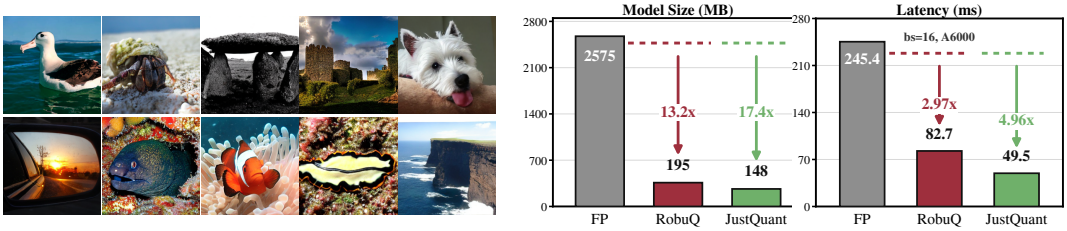

Figure 3: **Naive W1.58A4 ImageNet samples and deployment summary.** With a 24-hour GPT5.6-sol kernel-optimization budget, the clean JustQuant inference path realizes latency gains more easily than operator-heavy alternatives, even on a 0.6B DiT where small matrices and non-GEMM time make compute-bound speedup difficult.

to-image DiT and dLLM models under limited-budget and limited-data settings. Figure 3 gives a visual and deployment summary of the naive W1.58A4 setting.

Our main contributions are summarized as follows:

- **A distillation view of plain A4 quantization.** We frame 4-bit activation quantization as a Net2Net knowledge distillation problem for plain low-bit graphs. We analyze why PTQ and QAT methods struggle: PTQ often depends on complex deployment-side operators, while direct QAT suffers from stacked STE approximations and sparse supervision.
- **A progressive multi-level distillation algorithm.** We propose JustQuant and its core method, Theseus QAD, which uses progressive supervision levels as a warm-up before standard end-to-end QAT or QAD. The method moves the missing complexity into training while keeping the deployed model as a plain low-bit operator graph.
- **Controlled small-model regime.** On DiT-XL/2, Theseus QAD brings plain A4 quantization back to frontier-level generation quality. In the extreme W1.58A4 setting, it reduces FID-50K from 5.5698 to 3.3021 and raises IS-50K from 166.06 to 241.95. We further find that the progressive warm-up unexpectedly preserves generative diversity, addressing a previously unresolved failure mode of direct low-bit QAT.
- **Data-limited large-model regime.** We extend the same principle to larger, data-limited settings, including large text-to-image DiT and dLLM models. Theseus QAD outperforms PTQ and QAD baselines without adding new operators to the inference graph.

2 Background and Related Work

Specialized operators in PTQ and QAT. A major line of PTQ and QAT improves low-bit quality by changing the computation executed at inference. PTQ methods smooth or rotate activations, permute channels, factorize weights, or preserve mixed-precision branches to reduce outliers and reconstruction error (Xiao et al., 2023; Wei et al., 2022; Ashkboos et al., 2024; Liu et al., 2025; Lin et al., 2024a; Li et al., 2025; Shao et al., 2025). For LLMs, BWLA uses orthogonal transforms and low-rank refinement for binary weights with low-bit activations, while SpecQuant combines smoothing with frequency-domain truncation (Zhao et al., 2026d;a). Other ultra-low-bit PTQ approaches separate shared and expert-specific MoE weight components for vector quantization or use binary factorization under a strict bit budget (Xu et al., 2026; Zhao et al., 2026c). Other methods modify the operator itself: Quantizable Transformers changes attention to avoid learned outliers, SVDQuant and Q-DiT4SR add low-rank branches, and MXFP4/NVFP4-style work relies on specialized numerical formats (Bondarenko et al., 2023; Li et al., 2025; Zhang et al., 2026c; Xin et al., 2026; Xue et al., 2026). QAT methods follow the same pattern through Hadamard transforms, robust activation quantizers, modified normalization, rotation-based modules, or attention-specific kernels (Wang et al., 2025; Panferov et al., 2025; Yang et al., 2026; Huang et al., 2025a; Xu et al., 2025; Cheng et al., 2026; Zhang et al., 2026a). These techniques can recover accuracy, but their benefits come with extra transforms, branches, kernels, or format-specific hardware. Hardware-oriented designs such as QUARK additionally accelerate quantized Transformer nonlinearities through FPGA circuit sharing (Zhao et al., 2025b). We study the complementary setting: deployment remains a naive low-bit operator graph, and the missing complexity is handled during training.

The evolution of DiT and dLLM quantization. Quantization for generative transformers has progressed from weight-only compression toward joint weight–activation quantization. Early diffusion work identifies two recurring obstacles: activation distributions shift with timestep, and quantization error accumulates across denoising steps (Li et al., 2023; He et al., 2023; 2024). DiT-specific

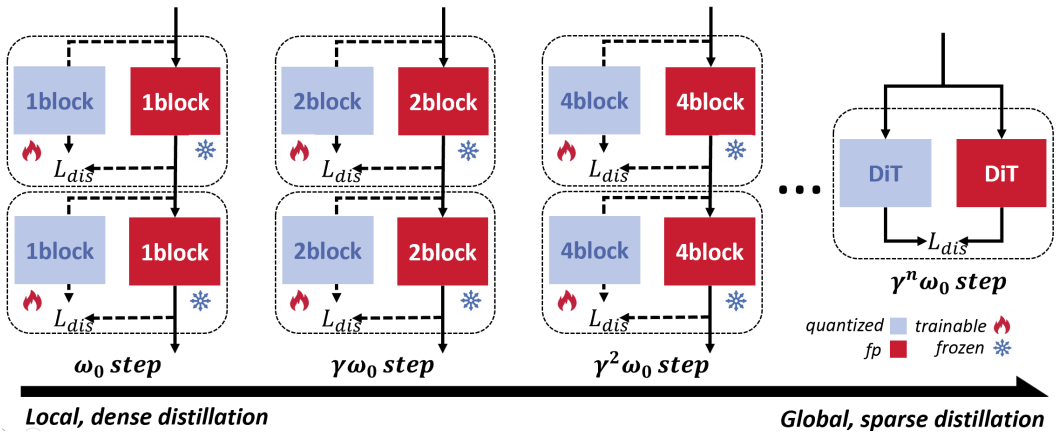

Figure 4: **Theseus QAD.** Theseus QAD expands distillation from local blocks to the full model while allocating training steps geometrically across path lengths.

PTQ then adds channel salience balancing, timestep-aware calibration, and token-aware granularity (Wu et al., 2024; Chen et al., 2025a; Zhao et al., 2025a). Later methods push toward more aggressive 4-bit or sub-4-bit settings with low-rank branches, rotations, temporal adaptation, binarization, and mixed-precision search (Li et al., 2025; Ryu et al., 2025; Zheng et al., 2024; 2025; Huang et al., 2025a; Xu et al., 2025; Yang et al., 2025; Zhang et al., 2026c). Training-based systems such as TerDiT and RobuQ further show that ternary or sub-4-bit DiT can be optimized, but still rely on specialized inference paths (Lu et al., 2024; Yang et al., 2026). Diffusion language models expose the same pattern in text generation: their masked-denoising activations and dynamic mask ratios make ordinary autoregressive-LLM calibration unreliable, motivating dLLM-specific PTQ methods (Zhang et al., 2026b; Chen et al., 2026). These developments leave open whether a plain W4A4 or W1.58A4 graph can be reached reliably without adding deployment operators.

Distillation for quantized models. Knowledge distillation transfers the behavior of a high-precision teacher to a compact student (Hinton et al., 2015). In quantization, the teacher can supervise logits, attention maps, hidden states, local blocks, or generative trajectories. Q-ViT uses distribution-guided distillation to repair attention distortion in fully quantized vision transformers, while token-scaled logit distillation adjusts the logit signal for ternary-weight generative language models (Li et al., 2022; Kim et al., 2023). Other work uses hidden-state or block-wise objectives to make compressed students easier to train (Dasgupta & Cohn, 2025; Chen et al., 2025b). In diffusion models, BiDM, BinaryDM, and Q-VDiT use spatial, low-rank, or temporal distillation to protect generative structure under aggressive quantization (Zheng et al., 2024; 2025; Feng et al., 2025). DilateQuant combines block-wise knowledge distillation (BKD) with Weight Dilation and a Temporal Parallel Quantizer (Liu et al., 2024). EfficientQAT trains all parameters block-wise before end-to-end training of quantization step sizes (Chen et al., 2025b). Our contribution is not block-wise alignment itself, but progressively merging teacher-fed student segments and allocating training steps across their lengths, while retaining plain deployed operators. This complements output-level QAD (Xin et al., 2026) and capacity-aware supervision (Harutyunyan et al., 2023).

3 Method

3.1 Quantization as Distillation

We view quantization as a Net2Net-style replacement guided by knowledge distillation. Let the full-precision network be

$$f_\theta(x) = B_L \circ B_{L-1} \circ \cdots \circ B_1(x),$$

where each B_i is a replaceable block or sub-block. Its quantized counterpart preserves the topology,

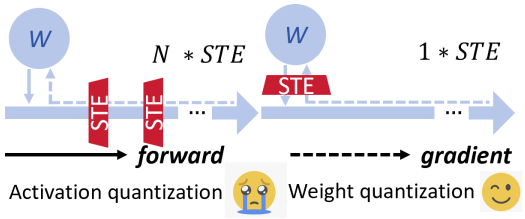

Figure 5: **Why activation quantization is hard to QAT.**

$$q_\phi(x) = \widetilde{B}_L \circ \widetilde{B}_{L-1} \circ \cdots \circ \widetilde{B}_1(x),$$

with each module replaced by a low-bit operator from the target deployment family. The goal is therefore to preserve network behavior, rather than merely reconstruct isolated matrices. This view separates the deployed operator from the training path: Theseus QAD keeps the deployed A4 operator plain and moves the additional guidance into training.

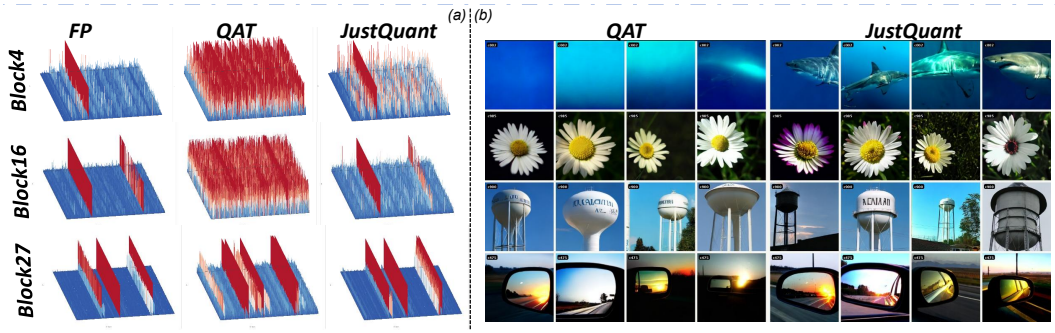

Figure 6: **Effect of Theseus QAD on W1.58A4 DiT.** (a) Intermediate activations after Theseus QAD warm-up and QAT resemble those of the FP model, whereas direct W1.58A4 QAT yields more disordered activations. (b) Theseus QAD substantially improves generation quality and diversity.

Why PTQ fails. PTQ mainly focuses on local linear layers or short blocks by matching their outputs to the corresponding full-precision activations. This signal is dense and cheap, but local agreement does not ensure that multiple replacements compose into a good network. Aggressive PTQ therefore often introduces smoothing, rotations, Hadamard transforms, SVD or low-rank branches, mixed precision which improve local approximation at the cost of a more complex deployed operator.

Why QAT fails. QAT optimizes the deployed low-bit model with the final task. Although aligned with final model, its correction signal is sparse and delayed. This is damaging for 4-bit activations, where the hidden state passes through straight-through estimator (STE) approximations, causing biased and noisy gradients (Figure 5). Quantization-aware distillation (QAD) adds a teacher, but forms still use one fixed alignment level: output-only QAD matches final behavior, while local QAD matches short segments. Neither setting exploits supervision at multiple granularities.

3.2 Distillation granularity and scheduling

We do not adopt a single fixed distillation width as a general design: a width suited to one model need not transfer to another architecture or quantization setting. Instead, we schedule distillation granularity according to three properties:

1. Training stability. Activation quantization makes optimization intrinsically unstable. Longer low-bit paths contain more stacked straight-through estimator (STE) approximations, so starting with end-to-end distillation can easily cause training to fail.

2. Model alignment. Extremely low-bit entries cannot match their full-precision counterparts exactly, while their composition can still approximate the overall transformation. An overly local target may therefore over-constrain intermediate values that the quantized network does not need to preserve.

3. Distillation density. Lower-level targets expose more intermediate states and provide denser supervision, allowing the student to receive useful updates more quickly. Global distillation provides the sparsest signal and is therefore the slowest to optimize.

These properties motivate progressive distillation granularity rather than a single fixed target. Our *Ship of Theseus* analogy refers to merging short low-bit segments into longer ones, not progressively replacing a mixed teacher–student forward path. Each level inherits the student parameters from the preceding level. Student blocks compose within each segment, but segment inputs remain teacher activations; student outputs are not passed between segments. Full-model distillation connects the entire student path. Subsequent QAT or QAD optimizes the model without changing its operators.

3.3 Theseus QAD design: local-to-global distillation

Theseus QAD uses a local-to-global, teacher-forced distillation schedule, illustrated in Figure 4. Figure 6 provides a qualitative comparison on W1.58A4 DiT: Theseus QAD warm-up followed by QAT yields intermediate activations closer to the full-precision reference and better generation quality than direct QAT. Stage 1 trains the low-bit student by matching teacher and student subnetworks at multiple depths. For a group of width w starting at block g, both networks receive the same teacher state h_g:

$$t_{g,w} = B_{g+w-1} \circ \cdots \circ B_g(h_g),$$
$$s_{g,w} = \widetilde{B}_{g+w-1} \circ \cdots \circ \widetilde{B}_g(h_g).$$

Stage 1 minimizes

$$\mathcal{L}_{\mathrm{S1}} = \mathbb{E}_{x,t,g,w}\left[d(s_{g,w}, t_{g,w})\right],$$

where t denotes task-specific conditioning, such as a diffusion timestep. In our DiT experiments, d is an activation reconstruction loss and the default implementation uses L_1 distance.

Algorithm 1 Theseus QAD: local-to-global distillation

func StageOne($f_\theta, q_\phi, \mathcal{D}, \gamma, T_1$)
Input: teacher f_θ, student q_ϕ, data $\mathcal{D}$, allocation ratio γ, budget T_1
$\phi \leftarrow \theta$, $(w_0, \ldots, w_K) \leftarrow$ BinaryMergeWidths(L)
$(N_0, \ldots, N_K) \leftarrow$ Allocate(T_1, γ) (see text)
for $k = 0, \ldots, K$ and $u = 1, \ldots, N_k$ **do**
  $(x, t) \sim \mathcal{D}$, $h \leftarrow x$, $\mathcal{L}_{\mathrm{S1}} \leftarrow 0$
  for $g = 0, w_k, 2w_k, \ldots, L - w_k$ **do**
    $a \leftarrow h$, $h \leftarrow B_{g+w_k-1} \circ \cdots \circ B_g(a)$
    $s \leftarrow \widetilde{B}_{g+w_k-1} \circ \cdots \circ \widetilde{B}_g(a)$
    $\mathcal{L}_{\mathrm{S1}} \leftarrow \mathcal{L}_{\mathrm{S1}} + d(s, h)$
  $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}_{\mathrm{S1}}$
return $q_\phi^{(1)} \leftarrow q_\phi$

Table 1 Stage-1 ratio sweep

γ	$N_{1,2,4,8,16,28}$	W4A4 FID↓	W4A4 IS↑	W1.58A32 FID↓	W1.58A32 IS↑
0	(10000, 0, 0, 0, 0, 0)	8.11	177	35.42	63
0.1	(9000, 900, 90, 9, 1, 0)	18.89	98	48.39	45
0.33	(6676, 2225, 742, 247, 82, 28)	14.20	124	23.00	91
0.5	(5079, 2540, 1270, 635, 317, 159)	9.23	162	17.16	113
1	(1667, 1667, 1667, 1667, 1666, 1666)	6.58	223	**12.87**	**146**
2	(159, 317, 635, 1270, 2540, 5079)	**6.28**	**225**	14.50	132
3	(28, 82, 247, 742, 2225, 6676)	7.32	213	14.83	127
10	(0, 1, 9, 90, 900, 9000)	6.96	218	15.69	113
∞	(0, 0, 0, 0, 0,10000)	188.26	4	18.05	103
Rev.	($\gamma_{\mathrm{W4A4}} = 2$, $\gamma_{\mathrm{W1.58A32}} = 1$)	8.70	169	29.50	82

Figure 7: **Theseus QAD warm-up and Stage-1 allocation.** The left panel shows the online Stage-1 loop: during one FP forward, each reached segment is immediately matched by its quantized counterpart, and the student is updated once after the accumulated loss. The right panel reports the same sweep, allocating 10k Stage-1 steps across replacement widths $(1, 2, 4, 8, 16, 28)$, followed by 10k-step official QAT.

All distillation paths use the same binary-merge schedule: starting from one-block segments, we repeatedly merge neighboring segments, giving the fixed widths $\mathcal{W} = (1, 2, 4, 8, 16, 28)$ for the 28-block DiT. The parameter γ does not define these widths. It only controls the geometric allocation of the Stage-1 budget across this fixed schedule:

$$N_k \propto \gamma^k, \qquad \sum_{k=0}^{K} N_k = T_1, \qquad 0 < \gamma < \infty.$$

Here k indexes the fixed widths from shortest to longest. Integer step allocations sum to the prescribed budget; the exact allocations used in the DiT sweep are listed in Figure 7. Thus, $\gamma = 1$ gives a uniform allocation, while larger γ values emphasize longer student segments; the $\gamma = 0$ and $\gamma = \infty$ rows are the corresponding shortest-path and longest-path endpoint schedules. For these endpoints, all T_1 steps are assigned to w_0 or w_K, respectively, and zero-budget levels are skipped. Thus, $\gamma = 0$ uses only single-block distillation in Stage 1, reflecting the local alignment principle in DilateQuant and EfficientQAT (Liu et al., 2024; Chen et al., 2025b). Conversely, $\gamma = \infty$ denotes full-model QAD (Xin et al., 2026), with all Stage-1 steps assigned to the longest path.

The sweep in Fig. 7 therefore tests our central hypothesis: the appropriate distillation signal should be scheduled across granularities, rather than selected once and held fixed. The two single-granularity extremes are both suboptimal: using only 1-block supervision ($\gamma = 0$) is too local, while jumping directly to full-model distillation ($\gamma = \infty$) is too unstable. This explains why prior fixed-granularity distillation can fail. In contrast, A4 targets benefit from allocating more exposure to wider distillation groups, whereas the W1.58A32 weight-only reference prefers a more balanced schedule. Reversing the order raises FID from 6.28 to 8.70 for W4A4 ($\gamma = 2$) and from 12.87 to 29.50 for W1.58A32 ($\gamma = 1$), supporting short-to-long distillation in these settings. Appendix C addresses common questions about our design choices, evaluation, and deployment scope.

4 Experiments

4.1 Setup

DiT is the primary testbed for validation and ablation; DLLM experiments test transfer across tasks.

Table 2: **Experimental summary.**

	Small model	Large model
DiT	DiT-XL/2, 0.6B (Peebles & Xie, 2023)	FLUX.1, 12B (Black Forest Labs, 2024)
DLLM	ELF-B, 105M (Hu et al., 2026)	LLaDA, 8B (Nie et al., 2025)

Datasets and evaluation metrics. We evaluate JustQuant in two regimes. The first uses class-conditional DiT-XL/2 (Peebles & Xie, 2023) on ImageNet (Russakovsky et al., 2015) 256×256, where data and compute are controllable enough to study plain A4 operators. FID-10K uses 10,000 generated images, CFG (Ho & Salimans, 2022) 1.5, and 50 sampling steps; FID-50K uses 50,000 images, CFG 1.5, and 250 steps. We report FID (Heusel et al., 2017), sFID (Salimans et al., 2016; Nash et al., 2021), and IS (Salimans et al., 2016; Barratt & Sharma, 2018) under the ADM protocol (Dhariwal & Nichol, 2021). The second uses FLUX.1-schnell and FLUX.1-dev (Black Forest Labs, 2024), where full low-bit retraining is expensive and the goal is recovery from limited data. We report MJHQ FID, CLIP-IQA (Wang et al., 2023), CLIP-Score (Hessel et al., 2021), ImageReward (Xu et al., 2023), and GenEval (Ghosh et al., 2023).

Table 3: **DiT-XL/2 on ImageNet** 256×256**.** Further evaluation details are provided in Appendix B.

Setting	Method	W/A	Type/Step	FID↓	sFID↓	IS↑	Additional Operator
10K samples 50 steps	FP (Peebles & Xie, 2023)	32/32	-/7000k	6.78	20.56	243.70	–
	SVDQuant (Li et al., 2025)	4/4	PTQ/0k	81.28	67.27	29.17	Smooth, SVD
	ConvRot (Huang et al., 2025a)	4/4	PTQ/0k	18.69	32.06	139.54	Rotation
	QuaRot (Ashkboos et al., 2024)	4/4	PTQ/0k	53.31	56.74	53.12	Rotation
	VETA-DiT (Xu et al., 2025)	4/4	PTQ/14.4k	9.87	25.52	202.55	Smooth, Rotation
	TreeQ (Yang et al., 2025)	4/4	PTQ/36.9k	6.92	20.86	219.66	MP, SVD, Rotation
	RobuQ (Yang et al., 2026)	4/4	QAT/20k	8.20	22.94	222.02	Norm, SVD, Rotation
	QAD (Xin et al., 2026)	4/4	QAD/20k	188.26	190.47	3.89	None
	QAT (Jacob et al., 2018)	4/4	QAT/20k	10.72	26.55	196.53	None
	JustQuant	4/4	QAT/20k	**6.48**	**19.31**	**238.31**	None
	TerDiT (Lu et al., 2024)	1.58/32	QAT/1750k	8.23	20.57	172.23	Norm
	JustQuant	1.58/32	QAT/110k	**6.56**	**19.84**	**249.65**	None
	RobuQ (Yang et al., 2026)	1.58/4	QAT/100k	7.76	20.81	195.98	Norm, SVD, Rotation
	QAT (Jacob et al., 2018)	1.58/4	QAT/100k	9.93	20.33	155.85	None
	JustQuant	1.58/4	QAT/110k	**7.23**	**19.88**	**223.99**	None
50K samples 250 steps	FP (Peebles & Xie, 2023)	32/32	-/7000k	2.27	4.55	277.83	–
	SVDQuant (Li et al., 2025)	4/4	PTQ/0k	60.01	37.85	39.26	Smooth, SVD
	ConvRot (Huang et al., 2025a)	4/4	PTQ/0k	7.98	13.36	187.96	Rotation
	VETA-DiT (Xu et al., 2025)	4/4	PTQ/14.4k	2.99	6.27	247.05	Smooth, Rotation
	RobuQ (Yang et al., 2026)	4/4	QAT/20k	**2.71**	**5.20**	**265.02**	Norm, SVD, Rotation
	QAT (Jacob et al., 2018)	4/4	QAT/20k	5.15	8.34	220.53	None
	JustQuant	4/4	QAT/20k	3.32	5.79	256.56	None
	TerDiT (Lu et al., 2024)	1.58/32	QAT/1750k	4.34	4.99	183.49	Norm
	JustQuant	1.58/32	QAT/110k	**3.05**	**4.59**	**266.83**	None
	RobuQ (Yang et al., 2026)	1.58/4	QAT/100k	**3.21**	4.86	217.04	Norm, SVD, Rotation
	QAT (Jacob et al., 2018)	1.58/4	QAT/100k	5.57	5.57	166.06	None
	JustQuant	1.58/4	QAT/110k	3.30	**4.51**	**241.95**	None

Table 4: **FLUX W4A4 block quantization on MJHQ and GenEval.**

Setting		MJHQ			GenEval				
Model	Method	FID↓	CLIP-IQA↑	ImageReward↑	Single↑	Count↑	Color↑	Overall↑	Add. Op.
FLUX.1-schnell (4 steps)	FP (Black Forest Labs, 2024)	19.15	0.936	0.959	0.9844	0.5938	0.7872	0.6677	–
	ConvRot (Huang et al., 2025a)	**17.86**	0.936	0.947	0.9812	**0.6406**	0.7819	**0.6817**	Rotation
	SVDQuant (Li et al., 2025)	18.51	**0.938**	**0.968**	**0.9906**	0.6219	**0.8138**	0.6775	SVD
	Naive INT4	18.37	0.929	0.902	0.9844	0.5813	0.7846	0.6753	None
	QAT (Jacob et al., 2018)	17.98	0.925	0.912	0.9844	0.6625	0.7872	0.6825	None
	JustQuant	**17.62**	**0.943**	**0.993**	**0.9938**	**0.6844**	**0.8351**	**0.7149**	None
FLUX.1-dev (50 steps)	FP (Black Forest Labs, 2024)	20.26	0.952	0.958	0.9875	**0.7281**	0.7926	**0.6707**	–
	ConvRot (Huang et al., 2025a)	19.78	**0.953**	0.948	0.9844	0.7094	0.7979	0.6662	Rotation
	SVDQuant (Li et al., 2025)	**19.71**	0.948	**0.961**	**0.9969**	0.6938	**0.8032**	0.6687	SVD
	Naive INT4	24.29	0.923	0.754	0.9844	0.7094	0.7952	0.6465	None
	QAT (Jacob et al., 2018)	22.54	0.917	0.520	0.9500	0.5312	0.6888	0.5302	None
	QAD (Xin et al., 2026)	20.44	**0.955**	0.898	**0.9906**	0.6906	0.7819	0.6585	None
	JustQuant	**20.21**	0.951	**0.956**	**0.9906**	**0.7188**	**0.8032**	**0.6643**	None

Compared methods. We compare against full-precision references, direct QAT and QAD using the same plain operator family, and PTQ methods that introduce smoothing, rotations, SVD branches, or mixed precision: SVDQuant (Li et al., 2025), ConvRot (Huang et al., 2025a), QuaRot (Ashkboos et al., 2024), VETA-DiT (Xu et al., 2025), and TreeQ (Yang et al., 2025). We also include trained low-bit systems such as TerDiT (Lu et al., 2024) and RobuQ (Yang et al., 2026), which add normalization, rotation, or SVD-based components. The tables identify these additional operators.

Training and quantization details. For small models, we use Theseus QAD followed by task-level QAT. For large models, we use Theseus QAD alone, because end-to-end QAT becomes unstable at the small batch sizes required by these experiments. A representative 20k-step W4A4 DiT-XL/2 run takes about 1.75 GPU-hours on four GPUs, while the FLUX experiments take about 0.75 days on four H200 GPUs. Further training and quantization details are provided in the appendix.

4.2 Main Results on Diffusion Models

DiT. Table 3 compares the 10K/50-step and 50K/250-step sampling protocols. Plain rows use low-bit operators without extra inference transformations. In W4A4, direct QAD fails to recover the model (FID-10K 188.26), and direct QAT reaches FID-10K 10.72. With staged replacement followed by QAT, Theseus QAD reduces FID-10K to 6.48, close to the FP value of 6.78, and improves IS from 196.53 to 238.31. The gain persists at 50K samples: FID drops from 5.15 to 3.32, and IS rises from 220.53 to 256.56. For W1.58A4, where both weight and activation precision are restricted, FID-50K improves from 5.57 to 3.30 and IS from 166.06 to 241.95 over direct QAT. The W1.58A32 result also improves on TerDiT despite using fewer training steps and no additional normalization operator. Operator-heavy methods can still yield a lower FID, as RobuQ does in W4A4. However, our method achieves greater sample diversity.

Table 5: **ELF-B OpenWebText generation.** W4A4 and W1.58A4 results are reported with generation perplexity from two judge models, following the evaluation protocol of ELF (Hu et al., 2026).

W/A	Arm	Qwen3-8B-Base		GPT-2-Large		Add. Op.
		gen-PPL↓(4seed)	ΔPPL vs FP	gen-PPL↓(4seed)	ΔPPL vs FP	
4/4	FP	14.729 ± 0.152	–	19.029 ± 0.305	–	–
	JustQuant	14.854 ± 0.344	$+0.125$	19.180 ± 0.417	$+0.152$	None
	RobuQ	16.751 ± 0.384	$+2.022$	22.089 ± 0.529	$+3.061$	Hadamard, SVD
	Direct QAT	19.625 ± 0.327	$+4.896$	25.011 ± 0.465	$+5.983$	None
1.58/4	FP	14.823 ± 0.208	–	19.254 ± 0.253	–	–
	JustQuant	14.626 ± 0.482	-0.197	19.038 ± 0.634	-0.216	None
	RobuQ	18.537 ± 0.265	$+3.714$	25.707 ± 0.414	$+6.453$	Hadamard, SVD
	Direct QAT	16.776 ± 0.255	$+1.953$	22.051 ± 0.391	$+2.797$	None

Table 6: **LLaDA-8B W4A4 on GSM8K and held-out transfer benchmarks.**

(a) GSM8K metrics by method

Method	Greedy↑	pass@1↑	pass@4↑	pass@16↑
FP	**0.6748**	**0.6307**	**0.8539**	**0.9325**
PTQ	0.6035	0.5704	0.8122	0.9113
QAT	0.6138	0.5836	0.8203	0.9163
QAD	0.6475	0.6015	0.8287	0.9196
JustQuant	0.6543	0.6137	0.8455	0.9249

(b) Transfer after GSM8K-only distillation

Method	PIQA	ARC-C	C-EVAL	HumanEval
FP	0.8300	**0.8662**	**0.6440**	**0.4512**
PTQ	0.8280	0.8495	0.6420	0.4268
QAT	0.8150	0.8200	0.6050	0.3512
QAD	0.8320	0.8562	**0.6440**	**0.4512**
JustQuant	**0.8420**	0.8562	0.6420	0.4451

FLUX. Following the transformer-focused quantization practice of SVDQuant (Li et al., 2025), we use W4A4 group64 for transformer-block linear layers. Table 4 reports MJHQ and representative GenEval sub-scores; the full breakdown is in the appendix. Dashed lines distinguish methods with extra inference operators from plain low-bit methods, and bold marks the best result within each group. On FLUX.1-schnell, Theseus QAD improves MJHQ FID from 18.37 for naive INT4 to 17.62 and raises GenEval overall from 0.6753 to 0.7149, exceeding the tested rotation and SVD baselines on the latter metric without their added operators. On FLUX.1-dev, direct QAT lowers GenEval overall to 0.5302, while standard QAD recovers it to 0.6585. Theseus QAD reaches 0.6643 and improves MJHQ FID from QAD's 20.44 to 20.21; its ImageReward (0.956) also approaches the FP value (0.958). QAD retains a slight CLIP-IQA advantage, so the improvement is not uniform across metrics. These results support staged distillation when full end-to-end QAT is unstable at the batch sizes available for larger models.

4.3 Results on Diffusion Language Models

ELF. ELF-B tests diffusion language generation under a larger training budget. The goal is to check whether Theseus QAD still gives an advantage at both W4A4 and W1.58A4 when the comparison moves beyond image generation. We compare against direct QAT and RobuQ under matched budgets. Table 5 shows that Theseus QAD is the strongest quantized method for both bit widths. In W4A4, its generation perplexity stays close to FP under both Qwen3-8B-Base and GPT-2-Large judges, while RobuQ and direct QAT introduce much larger gaps. In W1.58A4, Theseus QAD slightly improves over the FP reference within the reported variance. This suggests that, with enough training, the staged objective remains effective for DLLM generation and does not rely on DiTs.

LLaDA. We evaluate LLaDA-8B with NVFP4 W4A4, distilling only on GSM8K and testing both GSM8K and held-out transfer tasks. This experiment evaluates the transferability of Theseus QAD to another quantization format, rather than independence from specialized numerical formats. Table 6 shows that Theseus QAD achieves the highest reported quantized GSM8K greedy accuracy, improving from 0.6035 for PTQ and 0.6475 for QAD to 0.6543, alongside the highest quantized pass@1 and pass@16. Transfer performance remains broadly comparable to QAD:Theseus QAD improves PIQA, matches ARC-Challenge, stays within 0.002 of FP/QAD on C-EVAL, and remains close to QAD on HumanEval. Direct QAT, in contrast, reduces all four transfer scores relative to FP. These results suggest that Theseus QAD can improve target-task recovery while largely preserving the cross-task performance of ordinary QAD.

4.4 Ablation Studies

Learning-rate sensitivity. Direct W4A4 QAT is highly sensitive to the learning rate. Its FID improves from 24.60 at 10^{-4} to 10.72 at 5×10^{-5}, but this best tested setting still trails Theseus QAD by a wide margin: at the same learning rate, Theseus QAD reaches FID 6.72 versus 10.72 and IS 236.99 versus 196.53. In contrast, Theseus QAD is comparatively stable across the two tested learning rates, with FID 6.61 and 6.72. Thus, the main improvement is not explained by tuning.

Table 7: **Ablation Study.**

(a) Learning-rate sensitivity.

Method	LR	Steps	FID↓	sFID↓	IS↑
QAT	1e-4	20k	24.60	28.39	90.01
QAT	1e-6	20k	27.52	35.68	84.70
QAT	2e-6	20k	25.77	36.19	89.09
QAT	5e-6	20k	25.64	34.68	85.60
QAT	1e-5	20k	24.94	33.47	88.06
QAT	2e-5	20k	18.27	29.78	114.82
QAT	5e-5	20k	10.72	26.55	196.53
JustQuant	1e-4	20k	**6.61**	18.83	**243.60**
JustQuant	5e-5	20k	6.72	**18.65**	236.99

(b) Real-data allocation.

Stage-1 data	Stage-2 data	Best	FID↓	sFID↓	IS↑
1%	same 1%	5k	15.12	23.76	351.71
	full 100%	10k	6.90	18.66	238.20
5%	same 5%	5k	7.83	19.53	282.55
	full 100%	5k	6.59	18.72	231.79
10%	same 10%	5k	6.93	19.43	249.19
	full 100%	5k	6.39	18.78	227.25
20%	same 20%	5k	6.55	18.42	246.24
	full 100%	5k	6.94	18.86	216.13
50%	same 50%	5k	6.61	18.93	224.64
	full 100%	5k	6.59	18.71	226.93
100%	100%	5k	**6.28**	**18.32**	225.03

(c) Loss design.

Ablation	Loss variant	FID↓	sFID↓	IS↑
Stage-1	L1	**6.72**	18.65	**236.99**
	MSE/L2	6.85	**18.45**	232.30
Stage-2	task only	6.72	18.65	**236.99**
	convex mix	6.62	18.78	218.25
	additive mix	**6.51**	**18.46**	232.61

(d) Multi-stage Distilled W1.58A4 .

Teacher	γ	FID↓	sFID↓	IS↑
N/A	–	5.57	5.57	166.06
FP	1	3.64	4.70	227.23
W1.58A32	1	3.39	5.51	**244.00**
W1.58A32	2	**3.30**	**4.51**	241.95

(e) RobuQ with Theseus QAD .

Method	γ	FID↓	sFID↓	IS↑
W4A4, RobuQ	–	**2.71**	5.20	265.02
+ Theseus QAD	2	2.89	**4.48**	**267.22**
W1.58A4, RobuQ	–	**3.21**	4.86	217.04
+ Theseus QAD	1	3.16	**4.53**	**276.25**

Real-data allocation. The data sweep separates the role of Stage 1 from that of the final task objective. When Stage 2 has access to the full dataset, using only 5%-10% of the data in Stage 1 remains effective, reaching FID 6.59 and 6.39, respectively. By contrast, restricting both stages to the same 1% subset causes a large degradation to FID 15.12, while 5%-20% shared subsets are usable but remain weaker than the full-data run. These results show that Stage 1 can operate with limited coverage, but the final task stage still needs enough data to restore global behavior.

Loss design. For Stage 1, L_1 gives better FID and IS than MSE, while MSE gives slightly better sFID. This trade-off is consistent with both losses providing a local signal; the loss does not determine whether training succeeds. For Stage 2, the task-only objective gives the strongest IS, whereas adding a distillation term slightly improves FID or sFID but lowers IS. We therefore keep Stage 2 task-only in the main experiments, allowing the final stage to optimize the deployment task directly.

Multi-stage distillation order. Using a W1.58A32 student as the Stage-1 teacher for W1.58A4 improves FID-50K/IS-50K from 3.6419/227.23 to 3.3021/241.95, showing that extreme A4 quantization benefits from reducing weight precision before activation precision. The intermediate teacher matters: using the W1.58A32 teacher is stronger than using the full-precision teacher under the same uniform ($\gamma = 1$) budget-allocation schedule, improving FID from 3.64 to 3.39 and IS from 227.23 to 244.00. This supports the view that the replacement path should follow the target quantization problem rather than simply distill the final model from the full-precision network.

Training path under stronger operators. Table 7 shows that staged replacement improves RobuQ, depending on the metric. For W4A4, Theseus QAD improves sFID from 5.20 to 4.48 and IS from 265.02 to 267.22, with a modest FID trade-off from 2.71 to 2.89. For W1.58A4, it improves all three metrics, reducing FID from 3.21 to 3.16 and sFID from 4.86 to 4.53 while increasing IS from 217.04 to 276.25. This indicates that the training strategy is compatible with operator-enhanced baselines, even though its benefit is not uniformly expressed by every metric.

5 Conclusion

We presented JustQuant, a progressive distillation framework for deployable 4-bit-activation generative models, and Theseus QAD as its core training method. The central idea is to move the complexity required by aggressive A4 quantization from the inference graph into the training path. By aligning matched groups before returning to the original task objective, Theseus QAD trains plain W4A4 and W1.58A4 operators that would be unstable or less diverse under direct replacement. The DiT, FLUX, and DLLM results show that this approach recovers low-bit quality in both controlled and data-limited settings while keeping the deployed operator simple.

Takeaway. For smaller models, Theseus QAD supplies the warm-up stage missing from standard QAD, enabling subsequent end-to-end QAT of plain low-bit models. For larger models, where small-batch QAT is unstable, Theseus QAD is a more effective distillation strategy than QAD.

Limitations and future work. We have not reported experiments on autoregressive LLMs, nor have we evaluated large models with sufficient data and compute for extended end-to-end training. Testing whether the same training path remains effective in these settings is an important next step.

AI USE STATEMENT

We used generative AI tools to assist with code development and language polishing. The core tables, conclusions, and figures were prepared by the authors without generative AI assistance. All AI-assisted content and code were reviewed and verified by the authors. The authors take full responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies quantization methods for improving the efficiency of diffusion models. Quantized models may retain the biases and potential for misuse of their pretrained counterparts, and lower deployment costs may make these models more widely accessible. Our evaluations focus on generation quality and computational efficiency and do not establish the safety of the quantized models.

REPRODUCIBILITY STATEMENT

We will publicly release our code and model weights to facilitate reproducibility. Detailed experimental settings and implementation details will be provided in the appendix.

REFERENCES

8y. Pick-High-Dataset. Hugging Face Datasets, 2025. URL https://huggingface.co/datasets/8y/Pick-High-Dataset. Accessed 2026-09-25.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated llms. In *NeurIPS*, pp. 100213–100240, 2024.

Shane Barratt and Rishi Sharma. A note on the inception score, 2018.

Black Forest Labs. FLUX.1, 2024. URL https://blackforestlabs.ai.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. In *NeurIPS*, 2023.

Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jinyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-DiT: Accurate post-training quantization for diffusion transformers. In *CVPR*, pp. 28306–28315, 2025a.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient quantization-aware training for large language models. In *ACL*, 2025b.

Xuchen Chen, Zhixuan Chen, and Dawei Yang. DLLMQuant: A post-training quantization framework tailored for diffusion-based large language models. In *ICML*, 2026.

Yongsen Cheng, Kai Liu, Kaiwen Tao, Junxian Li, Zhixin Wang, Zhikai Chen, Renjing Pei, and Yulun Zhang. PermuQuant: Lowering per-group quantization error by reordering channels for diffusion models, 2026.

Sayantan Dasgupta and Trevor Cohn. Improving language model distillation through hidden state matching. In *ICLR*, 2025.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. BitDistiller: Unleashing the potential of sub-4-bit llms via self-distillation. In *ACL*, 2024.

Weilun Feng, Chuanguang Yang, Haotong Qin, Xiangqi Li, Yu Wang, Zhulin An, Libo Huang, Boyu Diao, Zixiang Zhao, Yongjun Xu, and Michele Magno. Q-VDiT: Towards accurate quantization and distillation of video-generation diffusion transformers. In *ICML*, 2025.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS*, 2023. URL https://mlanthology.org/neurips/2023/ghosh2023neurips-geneval/.

Hrayr Harutyunyan, Ankit Singh Rawat, Aditya Krishna Menon, Seungyeon Kim, and Sanjiv Kumar. Supervision complexity and its role in knowledge distillation. In *ICLR*, 2023.

Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. PTQD: Accurate post-training quantization for diffusion models. In *NeurIPS*, 2023.

Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. EfficientDM: Efficient quantization-aware fine-tuning of low-bit diffusion models. In *ICLR*, 2024.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows, 2026.

Feice Huang, Zuliang Han, Xing Zhou, Yihuang Chen, Lifei Zhu, and Haoqian Wang. ConvRot: Rotation-based plug-and-play 4-bit quantization for diffusion transformers, 2025a.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. SliM-LLM: Salience-driven mixed-precision quantization for large language models. In *ICML*, 2025b.

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 *CVPR*, pp. 2704–2713, 2018.

Minsoo Kim, Sihwa Lee, Janghwan Lee, Sukjin Hong, Du-Seong Chang, Wonyong Sung, and Jungwook Choi. Token-scaled logit distillation for ternary weight generative language models. In *NeurIPS*, 2023.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. SVDQuant: Absorbing outliers by low-rank components for 4-bit diffusion models. In *ICLR*, 2025.

Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-ViT: Accurate and fully quantized low-bit vision transformer. In *NeurIPS*, 2022.

Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. Q-DM: An efficient low-bit quantized diffusion model. In *NeurIPS*, 2023.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *NeurIPS*, pp. 87766–87800, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024b.

Xuewen Liu, Zhikai Li, Minhao Jiang, Mengjuan Chen, Jianquan Li, and Qingyi Gu. DilateQuant: Accurate and efficient diffusion quantization via weight dilation. *arXiv preprint arXiv:2409.14307*, 2024. URL https://arxiv.org/abs/2409.14307.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: Llm quantization with learned rotations. In *ICLR*, 2025.

Xudong Lu, Aojun Zhou, Ziyi Lin, Qi Liu, Yuhui Xu, Renrui Zhang, Yafei Wen, Shuai Ren, Peng Gao, Junchi Yan, et al. TerDiT: Ternary diffusion models with transformers, 2024.

Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W. Battaglia. Generating images with sparse representations. In *ICML*, 2021.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025.

Andrei Panferov, Jiale Chen, Soroush Tabesh, Roberto L. Castro, Mahdi Nikdan, and Dan Alistarh. QuEST: Stable training of llms with 1-bit weights and activations. In *ICML*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, pp. 4195–4205, 2023.

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(3):211–252, 2015.

Hyogon Ryu, NaHyeon Park, and Hyunjung Shim. DGQ: Distribution-aware group quantization for text-to-image diffusion models. In *ICLR*, 2025.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NeurIPS*, 2016.

Utkarsh Saxena, Sayeh Sharify, Kaushik Roy, and Xin Wang. ResQ: Mixed-precision quantization of large language models with low-rank residuals. In *ICML*, 2025.

Yuantian Shao, Peisong Wang, Yuanteng Chen, Chang Xu, Zhihui Wei, and Jian Cheng. Block rotation is all you need for MXFP4 quantization, 2025.

Hongyu Wang, Shuming Ma, and Furu Wei. BitNet v2: Native 4-bit activations with hadamard transformation for 1-bit llms, 2025.

Jianyi Wang, Kelvin C. K. Chan, and Chen Change Loy. Exploring CLIP for assessing the look and feel of images. In *AAAI*, 2023.

Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. QDrop: Randomly dropping quantization for extremely low-bit post-training quantization. In *ICLR*, 2022.

Junyi Wu, Haoxuan Wang, Yuzhang Shang, Mubarak Shah, and Yan Yan. PTQ4DiT: Post-training quantization for diffusion transformers. In *NeurIPS*, 2024.

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 *ICML*, pp. 38087–38099, 2023.

Meng Xin, Sweta Priyadarshi, Jingyu Xin, Bilal Kartal, Aditya Vavre, Asma Kuriparambil Thekkumpate, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Ido Shahaf, Akhiad Bercovich, Kinjal Patel, Suguna Varshini Velury, Chenjie Luo, Zhiyu Cheng, Jenny Chen, Chen-Han Yu, Wei Ping, Oleg Rybakov, Nima Tajbakhsh, Oluwatobi Olabiyi, Dusan Stosic, Di Wu, Song Han, Eric Chung, Sharath Turuvekere Sreenivas, Bryan Catanzaro, Yoshi Suhara, Tijmen Blankevoort, and Huizi Mao. Quantization-aware distillation for nvfp4 inference accuracy recovery, 2026.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. In *NeurIPS*, 2023.

Qinkai Xu, Yijin Liu, Yang Chen, Lin F. Yang, Li Erran Li, and Yuxiang Fu. VETA-DiT: Variance-equalized and temporally adaptive quantization for efficient 4-bit diffusion transformers. In *NeurIPS*, 2025.

Zukang Xu, Zhixiong Zhao, Xing Hu, Zhixuan Chen, and Dawei Yang. KBVQ-MoE: Klt-guided svd with bias-corrected vector quantization for moe large language models, 2026.

Bowen Xue, Zihan Min, Xingyang Li, Zhekai Zhang, Haocheng Xi, Lvmin Zhang, Maneesh Agrawala, Jun-Yan Zhu, Song Han, Yujun Lin, and Muyang Li. FourTune: Towards fully 4-bit efficient post-training for diffusion models, 2026.

Kaicheng Yang, Kaisen Yang, Baiting Wu, Xun Zhang, Qianrui Yang, Haotong Qin, He Zhang, and Yulun Zhang. TreeQ: Pushing the quantization boundary of diffusion transformer via tree-structured mixed-precision search, 2025.

Kaicheng Yang, Xun Zhang, Haotong Qin, Yucheng Lin, Kaisen Yang, Xianglong Yan, and Yulun Zhang. RobuQ: Pushing dits to w1.58a2 via robust activation quantization. In *ICML*, 2026.

Cheng Zhang, Jianyi Cheng, George A. Constantinides, and Yiren Zhao. LQER: Low-rank quantization error reconstruction for llms, 2024.

Peiyuan Zhang, Matthew Noto, Wenxuan Tan, Chengquan Jiang, Will Lin, Wei Zhou, and Hao Zhang. Attn-QAT: 4-bit attention with quantization-aware training. In *ICML*, 2026a.

Tianao Zhang, Zhiteng Li, Xianglong Yan, Haotong Qin, Yong Guo, and Yulun Zhang. QuantdLLM: Post-training extreme low-bit quantization for diffusion large language models. In *ICLR*, 2026b.

Xun Zhang, Kaicheng Yang, Hongliang Lu, Haotong Qin, Yong Guo, and Yulun Zhang. Q-DiT4SR: Exploration of detail-preserving diffusion transformer quantization for real-world image super-resolution. In *ICML*, 2026c.

Tianchen Zhao, Tongcheng Fang, Haofeng Huang, Enshu Liu, Rui Wan, Widyadewi Soedarmadji, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Nong, and Yu Wang. ViDiT-Q: Efficient and accurate quantization of diffusion transformers for image and video generation. In *ICLR*, 2025a.

Zhixiong Zhao, Haomin Li, Fangxin Liu, Yuncheng Lu, Zongwu Wang, Tao Yang, Li Jiang, and Haibing Guan. QUARK: Quantization-enabled circuit sharing for transformer acceleration by exploiting common patterns in nonlinear operations. In *ICCAD*, pp. 1–9, 2025b.

Zhixiong Zhao, Fangxin Liu, Junjie Wang, Chenyang Guan, Zongwu Wang, Li Jiang, and Haibing Guan. SpecQuant: Spectral decomposition and adaptive truncation for ultra-low-bit llms quantization. In *AAAI*, volume 40, pp. 28786–28794, 2026a.

Zhixiong Zhao, Zukang Xu, Zhixuan Chen, Xing Hu, Zhe Jiang, and Dawei Yang. TWLA: Achieving ternary weights and low-bit activations for llms via post-training quantization. In *ICML*, 2026b.

Zhixiong Zhao, Zukang Xu, Guangyu Sun, Lifeng Liu, and Dawei Yang. All for 1-bit: Towards genuine 1-bit post-training quantization for llms. In *EMNLP*, 2026c.

Zhixiong Zhao, Zukang Xu, and Dawei Yang. BWLA: Breaking the barrier of w1ax post-training quantization for llms. In *ACL*, pp. 19264–19290, 2026d. doi: 10.18653/v1/2026.acl-long.880. URL https://aclanthology.org/2026.acl-long.880/.

Xingyu Zheng, Xianglong Liu, Yichen Bian, Xudong Ma, Yulun Zhang, Jiakai Wang, Jinyang Guo, and Haotong Qin. BiDM: Pushing the limit of quantization for diffusion models. In *NeurIPS*, 2024.

Xingyu Zheng, Xianglong Liu, Haotong Qin, Xudong Ma, Mingyuan Zhang, Haojie Hao, Jiakai Wang, Zixiang Zhao, Jinyang Guo, and Michele Magno. BinaryDM: Accurate weight binarization for efficient diffusion models. In *ICLR*, 2025.

A VISUAL COMPARISONS

We provide additional visual comparisons to complement the main-paper results. Figure 8 presents two panels of DiT-XL/2 comparisons under W1.58A4 quantization, while Figure 9 presents FLUX.1-schnell comparisons under W4A4 INT4 quantization.

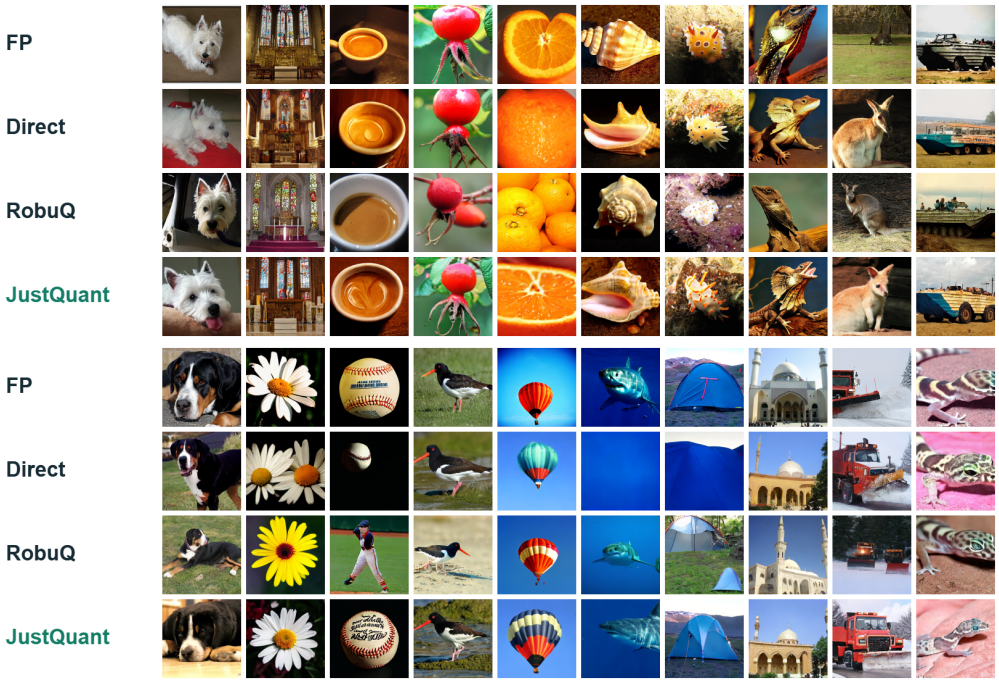

Figure 8: **DiT-XL/2 W1.58A4 qualitative comparisons.** The two panels show ImageNet 256 × 256 samples from the full-precision model and the compared low-bit methods under the W1.58A4 setting.

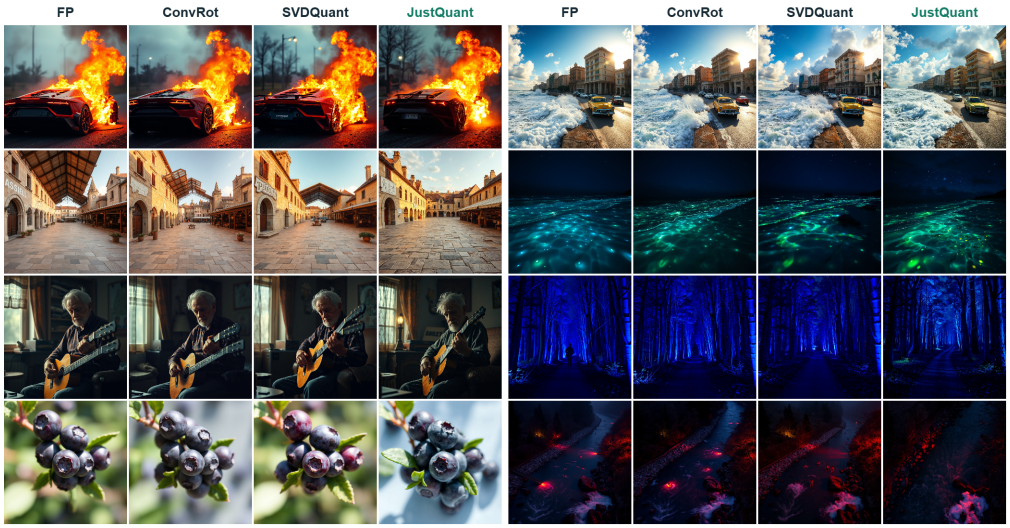

Figure 9: **FLUX.1-schnell W4A4 INT4 qualitative comparison.** Each row shows one prompt, and the columns compare FP, ConvRot, SVDQuant, and JustQuant under the W4A4 INT4 setting.

B EXPERIMENTAL DETAILS

This section records the configurations used for the experiments reported in the main paper. Each column of the table is one experimental setting and each row is one configuration field. Values marked as not recorded are not inferred from the available experiment logs.

Table 8: DiT-XL/2 experimental details.

	DiT-XL/2, ImageNet 256^2		
Setting	W1.58A32	W4A4	W1.58A4
Stage 1 (Theseus QAD)			
Dataset	ImageNet, 128k VAE latents	ImageNet, 128k VAE latents	ImageNet, 128k VAE latents
Learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}
Batch size	256	128	256
Training steps	10k	10k	10k
γ (budget)	1	2	2
Stage 2 (Official QAT)			
Dataset	ImageNet, 128k VAE latents	ImageNet, 128k VAE latents	ImageNet, 128k VAE latents
Learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}
Batch size	256	128	256
Training steps	100k	10k	100k
GPU hours	12 h $\times$ 8 H200 = 96 GPU-hours	1.75 h $\times$ 4 H200 = 7 GPU-hours	12 h $\times$ 8 H200 = 96 GPU-hours
Quantization function	Ternary weights, FP activations	INT4 weights/activations, group128	Ternary weights, INT4 activations, group128

Table 9: FLUX experimental details.

	FLUX (W4A4)	
Setting	FLUX-schnell	FLUX-dev
Stage 1 (Theseus QAD)		
Dataset	Pick-High 100K refined image-text subset (8y, 2025)	Pick-High 100K refined image-text subset (8y, 2025)
Learning rate	1×10^{-5}	1×10^{-5}
Batch size (global)	16	16
Training steps	10k	10k
Training epochs	2	2
γ (budget)	2	2
GPU hours	18 h $\times$ 4 H200 = 72 GPU-hours	18 h $\times$ 4 H200 = 72 GPU-hours
Quantization function	W4A4 INT4 block/group64 linears	W4A4 INT4 block/group64 linears

Table 10: ELF-B and LLaDA-8B experimental details.

	ELF-B		LLaDA-8B
Setting	W1.58A4	W4A4	W4A4
Stage 1 (Theseus QAD)			
Dataset	OpenWebText	OpenWebText	GSM8K train
Learning rate	2×10^{-3}, cosine $\rightarrow 1.25 \times 10^{-5}$	2×10^{-3}, cosine $\rightarrow 1.25 \times 10^{-5}$	1×10^{-5}
Batch size	512 effective	512 effective	8
Training steps	18,764	18,764	9.3k
γ (budget)	2	2	2
Stage 2 (Official QAT)			
Dataset	OpenWebText	OpenWebText	N/A
Learning rate	1×10^{-5}, constant	1×10^{-5}, constant	N/A
Batch size	512 effective	512 effective	N/A
Training steps	45,036	16,884	N/A
GPU hours	19 h $\times$ 8 H200 = 152 GPU-hours	12 h $\times$ 8 H200 = 96 GPU-hours	2 h $\times$ 4 H200 = 8 GPU-hours
Quantization function	INT4 weights/activations, group128	Ternary weights, INT4 activations, group128	NVFP4 W4A4

C Frequently Asked Questions and Clarifications

Q1: Why schedule distillation widths, and what does γ control? A single fixed width suited to one model need not transfer to other sizes or architectures. We use a binary-merge schedule, such as $(1, 2, 4, 8, 16, 28)$ for DiT-XL/2, and allocate training steps across its levels using $N_k \propto \gamma^k$. Thus, γ controls the budget, not the widths. $\gamma = 1$ gives a uniform allocation; $\gamma = 0$ corresponds to block-wise KD, while $\gamma = \infty$ corresponds to full-model distillation.

Q2: What is progressive, and must the smallest unit be a block? Student segments progressively grow by merging neighboring units, with parameters inherited across levels. Student blocks compose within each segment, but segment inputs remain teacher activations; student outputs are not passed between segments. The smallest unit need not be a full block: in FLUX, using MLP or attention modules yielded comparable performance.

Q3: What evidence supports the short-to-long order? Figure 7 includes reverse-order results. For W4A4 with $\gamma = 2$, reversing the order increases FID from 6.28 to 8.70; for W1.58A32 with $\gamma = 1$, it increases FID from 12.87 to 29.50. These results support the short-to-long order in the tested settings, not its superiority over every possible schedule.

Q4: Why not jointly optimize losses from all levels? We observed that distillation losses at different block locations can differ in magnitude by up to five orders of magnitude. Joint multi-level supervision therefore requires careful, model-dependent loss weighting. We schedule supervision across levels to reduce this tuning burden and improve transferability.

Q5: How does this differ from prior block-wise training? Block-wise alignment is not our novelty. DilateQuant combines BKD with Weight Dilation and a Temporal Parallel Quantizer (Liu et al., 2024); EfficientQAT uses block-wise training followed by end-to-end quantization-step-size training (Chen et al., 2025b). Our focus is progressively merging teacher-fed student segments and allocating steps across their lengths. The $\gamma = 0$ setting provides the block-wise KD baseline within our training framework.

Q6: Why use different training stages across models? For smaller models, Theseus QAD provides an initialization for subsequent task-level QAT. For FLUX and LLaDA-8B, we use Theseus QAD alone under the available budget and small-batch constraints. This choice does not imply that larger models cannot benefit from additional QAT.

Q7: Do plain operators eliminate the need for specialized kernels? No. Low-bit execution still requires suitable quantization, packing, and matrix-multiplication kernels. "Plain" means avoiding additional smoothing, rotation, normalization, or SVD branches introduced for quantization, not eliminating low-bit kernel engineering.

Q8: Does FLUX W4A4 cover the entire generation pipeline? No. W4A4 applies to transformer-block linear layers. Text encoders, the VAE, and boundary modules remain in high precision. This transformer-focused scope follows SVDQuant (Li et al., 2025); it does not mean that all parameters or operations are four-bit.

Q9: Why are multiple seeds not reported for every experiment? Repeating every training and evaluation run across seeds would exceed our available compute budget. We prioritize multi-seed evaluation for ELF generation, where results are seed-sensitive. Results without repeated runs do not quantify run-to-run variability.