

Quantizing Masked Diffusion Language Models: Storage–Quality Trade-offs and Runtime Diagnostics

Pawan Kumar
IIIT, Hyderabad, India
pawan.kumar@iiit.ac.in

Neel Mishra
Microsoft, India
neelmishra@microsoft.com

Abstract

Masked diffusion language models reuse one denoiser across many steps, so post-training quantization (PTQ) must balance predictive quality, serialized storage, and runtime memory. We evaluate LLaDA-8B-Instruct and run transfer checks on Dream-7B, MDLM-OWT, encoder-only masked LMs, LLaDA-MoE, and LLaDA2.0-mini. The 24-text calibration set is distinct from the main $n=500$ -example-per-task evaluation. Three findings emerge. First, AWQ-4 is the closest tested 4-bit quality point at the evaluated 1.1–2.3 percentage-point resolution, although ARC-Challenge shows a detectable loss. TurboLDM-(3+1) instead provides the smallest serialized checkpoint (4.82 GiB, 3.55 effective bits) while losing quality, making it a storage-compression and diagnostic result, not a live-memory result. NF4 alone reduces peak VRAM from roughly 15 GiB to 6 GiB in our tested PyTorch/bitsandbytes runtime; this is a runtime-specific outcome, not an inherent format advantage. Second, individual attention projections are inexpensive to quantize, whereas jointly quantizing the attention block is damaging. Third, more denoising steps partly recover GSM8K loss but not multiple-choice accuracy. Code and logs are available at <https://github.com/misterpawan/turbo-quant-paper.git>.

1 Introduction

Masked diffusion language models (MDLMs) are becoming relevant for industrial NLP systems that need controllable latency, batch generation, or smaller stored checkpoints. Recent industrial work such as Mercury frames diffusion LMs as commercial-scale systems for fast APIs and on-prem serving (Inception Labs, 2025; Labs et al., 2025). MDLM clarified the objective (Sahoo et al., 2024), scaling studies showed the paradigm improves with size (Nie et al., 2025), Block Diffu-

sion connected diffusion and autoregressive generation (Arriola et al., 2025), and LLaDA, Dream-7B, and LLaDA2.0 moved the family into the 7B–100B regime (Nie et al., 2026; Ye et al., 2025; Bie et al., 2025). Their deployment cost is unusual for resource-constrained settings: the same Transformer is evaluated at many denoising steps, so a fixed weight perturbation is reused along the whole trajectory.

Autoregressive PTQ methods such as GPTQ, AWQ, SmoothQuant, SpQR, QuIP#, and AQLM (Frantar et al., 2023; Xiao et al., 2023; Lin et al., 2025; Dettmers et al., 2024; Tseng et al., 2024; Egiazarian et al., 2024) assume a prompt-prefix calibration regime. Image-diffusion PTQ methods such as PTQ4DM, PTQD, TFMQ-DM, and Q-Diffusion (Shang et al., 2023; He et al., 2023; Huang et al., 2024; Li et al., 2023) show that timestep-aware calibration can matter, but target continuous U-Net denoisers rather than discrete masked-token Transformers. This leaves practitioners with a concrete deployment question: which parts can be compressed, how checkpoint size, quality, and runtime memory differ, and which findings carry across diffusion-LM families.

This paper. We use TurboQuant (Zandieh et al., 2026) inside a stage-aware base-residual wrapper, TurboLDM, and compare it with practical baselines. Our goal is to compress weights and identify which quantization choices are usable for deployment. We use the codec as a diagnostic to derive practical rules for storage-constrained model distribution: where masked diffusion LMs tolerate PTQ, where they fail, and which runtime metric should drive the choice. We study per-module sensitivity, bit-budget choice, mixed-bit recipes, stage-aware calibration, step-count interactions, storage accounting, runtime memory, and cross-family transfer.

Contributions. We make four claims. First, LLaDA-8B-Instruct has a repeatable PTQ profile across module families, depth, denoising steps, and seeds. Second, among BF16, uniform int quantization, GPTQ-dia/full, AWQ, NF4, Q-Diffusion-Text, and TurboLDM, AWQ-4 is closest to BF16 among the tested 4-bit rows at our evaluation resolution. NF4 is the only integrated low-bit path that lowers live VRAM in our tested runtime, whereas TurboLDM is a compact-checkpoint result. Third, two plausible scheduling ideas fail under larger evaluations: stage-stratified calibration and sensitivity-ranked residual placement. Fourth, quality, serialized storage, and live VRAM lead to different choices.

2 TurboLDM Method

This section describes TurboLDM, our stage-aware base-residual wrapper around TurboQuant. Let x_t be the partially masked sequence at denoising step t and $f_\theta(x_t, t)$ the masked-token denoiser. We partition denoising states into masking-ratio stages $\{\mathcal{D}_s\}_{s=1}^S$ and minimize a stage-weighted masked-token KL surrogate,

$$\mathcal{L}_{\text{cal}}(\hat{\theta}) = \sum_{s=1}^S \pi_s \mathbb{E}_{(x_t, t) \sim \mathcal{D}_s} [\text{KL}(p_\theta(\cdot | x_t, t) \| p_{\hat{\theta}}(\cdot | x_t, t))]. \quad (1)$$

Each quantized matrix is split into fixed-size blocks, rotated by a Hadamard-sign map, and encoded with a symmetric Gaussian codebook at b_0 base bits. A residual code at b_1 bits can be stored once per layer and activated by stage. The resulting effective storage lies between b_0 and $b_0 + b_1$; TurboLDM-(3+1) measures 3.55 effective bits on LLaDA-8B.

The scheduler estimates a per-(layer, stage) sensitivity $\alpha_{\ell, s}$ by temporarily quantizing layer ℓ and measuring the masked-token KL increase. It greedily activates residuals by

$$\rho_{\ell, s} = \pi_s \alpha_{\ell, s} (\delta_\ell^{(0)} - \delta_\ell^{(1)}) / c_\ell,$$

where $\delta_\ell^{(0)}$ and $\delta_\ell^{(1)}$ are base-only and after-residual distortions and c_ℓ is the layer-once residual cost. This is a heuristic for a non-additive cost: once any stage uses a residual, the layer pays the storage cost. Appendix C gives the full algorithm and the local error-propagation motivation.

3 Experimental Setup

Models and data. The main model is LLaDA-8B-Instruct (Nie et al., 2026). The main tasks are MMLU (Hendrycks et al., 2021), ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and GSM8K (Cobbe et al., 2021). Transfer checks cover Dream-7B (Ye et al., 2025), MDLM-OWT (Sahoo et al., 2024), BERT/RoBERTa, LLaDA-MoE-7B-A1B, and LLaDA2.0-mini.

Baselines and decoding. We compare BF16, uniform 2/3/4-bit group quantization, GPTQ-dia and GPTQ-full, AWQ, NF4 through bitsandbytes, Q-Diffusion-Text, and TurboLDM-(4+1) and TurboLDM-(3+1). MCQ tasks use $T=16$ denoising steps and GSM8K uses $T=64$ unless otherwise stated. Calibration uses 24 WikiText-103 texts at most 256 tokens each.

Calibration versus evaluation size. The 24 texts are used only to calibrate PTQ parameters; they are not the samples on which accuracy is reported. Main accuracy uses $n=500$ examples *per task*. A separate calibration-size sweep uses 8/24/64/128/256 texts while holding its $n=75$ evaluation slice fixed (Appendix F).

Compute and statistics. Experiments run on one NVIDIA RTX 5090 with 32 GB VRAM, PyTorch 2.11, CUDA 13.0, and transformers 4.55. The main $n=500$ CI table uses three seeds for non-degenerate non-NF4 rows and paired-bootstrap confidence intervals against BF16. The main NF4 row was rerun with seeds 17/42/101 and produced identical scores under deterministic decoding, but it is not included in the paired-CI comparison. Weak baselines, storage-audit, transfer, and trajectory analyses are single-seed unless explicitly labeled otherwise.

4 Results

4.1 Main Quality-Storage Trade-off

Table 1 condenses the main $n=500$ result. AWQ-4 is closest to BF16 among the tested quality-oriented 4-bit rows at a task-dependent resolution of roughly 1.1–2.3 percentage points. Intervals crossing zero on MMLU, HellaSwag, and GSM8K indicate near-parity at this sample size, not proven equivalence; ARC-C has a detectable 1.5-point loss (95% CI $[-2.7, -0.4]$). NF4 produces the same scores in three deterministic reruns and is the only main end-to-end row integrated

Method	Bits	Total	Peak	MMLU	ARC-C	H.Swag	GSM8K
BF16	16.00	14.93	15.6	61.8	86.0	69.2	68.6
AWQ-4	4.25	5.38	15.6	62.4	84.5	68.4	69.5
NF4	6.38	7.11	6.0	60.2	85.4	68.8	69.0
Q-Diff.-4	4.48	5.57	15.6	62.0	84.2	66.8	68.5
TLDM (4+1)	4.56	5.64	18.1	60.8	84.0	65.3	68.5
TLDM (3+1)	3.55	4.82	18.0	58.6	80.1	60.8	60.4

Table 1: Compact view of the main LLaDA-8B-Instruct result at $n=500$. AWQ-4 is closest to BF16 on quality, NF4 lowers live VRAM in our tested runtime, and TLDM trades accuracy for smaller storage. Q-Diff.-4 is the 4-bit Q-Diffusion-Text row. Weak baselines, GPTQ variants, and confidence intervals appear in Appendix B. TLDM abbreviates TurboLDM; $(b+1)$ means b base bits plus a 1-bit residual code. Peak and storage are GiB.

Configuration	MMLU	ARC-C	H.Swag	GSM8K
BF16 reference	61.8	86.0	69.2	68.6
attn_all	55.6	77.6	60.0	55.2
mlp_all	60.8	83.6	66.0	67.6

Table 2: Corrected compound-family result on LLaDA-8B-Instruct. Each non-BF16 row jointly quantizes every projection in the named family at 3-bit with no residual (single seed, $n=500$). In particular, `attn_all` is a joint $\{q, k, v, \text{attn_out}\}$ rerun, not a duplicate of the isolated `attn_out` row.

with a low-bit kernel in our tested runtime, reducing peak inference VRAM to 6.0 GiB. Optimized AWQ and GPTQ kernels exist in other runtimes, so this VRAM result is not an inherent advantage of the NF4 format. TurboLDM-(3+1) gives the smallest serialized checkpoint but is significantly below BF16 on all four main tasks.

4.2 What Compresses?

The per-family map does not say that attention is always fragile. Isolated Q, K, V, and output projections lose at most about 2 pp on any task at 3-bit. Quantizing all attention projections together is much worse, dropping 6.2/8.4/9.2/13.4 pp on MMLU/ARC-C/HellaSwag/GSM8K; compound MLP quantization is milder at 1.0/2.4/3.2/1.0 pp. Table 2 reports the corrected joint-attention rerun. Perturbing Q and K together changes the bilinear attention logits before softmax, while simultaneous V and output-projection errors compound in the aggregated values; the same perturbations are then reused over denoising steps. This is an explanation consistent with the measurements, not a causal proof.

Figure 1 shows the fine-grained stage-by-layer view.

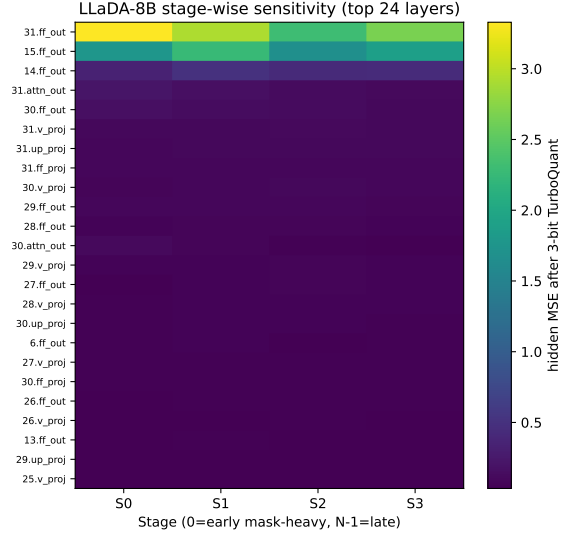


Figure 1: Stage-wise sensitivity for the 24 most-sensitive LLaDA-8B linear layers. Rows are layers; columns are masking-ratio stages.

Method / task	$T=8$	$T=16$	$T=64$	$T=128$
BF16 / MMLU	43.5	43.5	43.5	43.5
Turbo (3+1) / MMLU	41.5	41.5	41.5	41.5
BF16 / GSM8K	26.0	33.0	69.0	73.0
Turbo (3+1) / GSM8K	12.0	14.0	57.0	65.0

Table 3: Step count interacts with task type. MCQ accuracy is flat in T , while GSM8K recovers substantially as denoising compute increases. Turbo (3+1) means TurboLDM with 3 base bits plus a 1-bit residual code.

Mixed-bit recipes in Table 8 follow the same pattern. The best hand recipe, `ATTN@3 + MLP@4`, saves storage relative to W4/AWQ-4 but is not accuracy-equivalent under the three-seed evaluation. It is a storage trade-off instead of a parity recipe.

4.3 What Breaks?

Two apparent signals disappear under larger evaluations. First, stage-stratified calibration ($S=4$) looked like a +5 pp GSM8K win in a single-seed $n=40$ slice, but at $n=100$ the gain disappears. Second, sensitivity-ranked residual placement does not beat base-only or static-layer residuals at matched average bits. The repeatable lever is residual *capacity*: (3+1) is better than lower-bit settings because it stores more residual information, not because stage placement is reliably optimal.

Step count is task-conditioned. Table 3 shows that MMLU is essentially flat as T varies from 8 to 128, while GSM8K improves sharply. More de-

Model	Method	Bits	Storage	Peak	MMLU
LLaDA-MoE	BF16	16.00	13.70	14.4	50.0
LLaDA-MoE	NF4	4.00	4.33	5.4	51.7
LLaDA-MoE	AWQ-4	4.25	3.32	14.6	51.7
LLaDA-MoE	Turbo (3+1)	3.45	2.70	16.6	45.3
LLaDA2-mini	BF16	16.00	30.28	26.2	64.0
LLaDA2-mini	Lazy 4-bit	4.25	7.72	17.1	22.7
LLaDA2-mini	Lazy 3-bit	3.25	5.91	17.1	17.3

Table 4: Scaling snapshot. NF4 and AWQ-4 retain LLaDA-MoE MMLU quality in this slice; TurboLDM-(3+1) is smaller but lower quality. The LLaDA2-mini rows test memory feasibility only: without matched AWQ/NF4 baselines, their quality collapse is not attributable specifically to lazy reconstruction. “Lazy 4-bit” and “Lazy 3-bit” use compact codes and on-the-fly dequantization. Storage and peak memory are GiB; $(b+1)$ means b base bits plus a 1-bit residual code.

noising steps can buy back generative reasoning quality, but they do not help MCQ scoring.

4.4 What Transfers?

Table 4 summarizes the largest-model transfer checks. Dream-7B is more PTQ-fragile than LLaDA-8B under matched $n=500$ checks: AWQ-4 loses about 7 pp on MMLU and 9 pp on GSM8K, while TurboLDM-(3+1) loses much more on ARC-C and GSM8K. The compound-attention pattern still transfers: individual attention projections are cheap, while quantizing all attention projections together is damaging. MDLM-OWT changes which MLP sublayer is most fragile. On LLaDA-MoE, NF4 and AWQ-4 keep MMLU close to BF16, but TurboLDM-(3+1) loses quality even though it is compact. The LLaDA2.0-mini experiment shows only that lazy reconstruction can fit a 16B sparse model into about 17 GiB VRAM. We did not run 16B AWQ or NF4 baselines, so its quality collapse cannot be separated from model- or implementation-specific difficulty. Each architecture needs its own module map.

4.5 Deployment Lessons

Table 5 summarizes the deployment rules. Serialized checkpoint size and live inference memory are different axes. AWQ-4 is the closest tested quality-oriented 4-bit point at our resolution, NF4 is the memory point in our tested runtime, and TurboLDM-(3+1) is a compact storage point rather than a parity method.

Question	Finding	Deployment rule
Tested 4-bit points	AWQ-4 is closest to BF16; NF4 lowers VRAM only in the tested runtime.	Match the method and optimized runtime to the binding constraint.
Sub-4-bit storage	TurboLDM-(3+1) is compact but not parity.	Treat it as a storage trade-off.
Modules	Isolated attention is cheap; compound attention is harmful.	Measure family maps before recipes.
Stages	Stage scheduling is not robust at larger sample sizes.	Spend bits on capacity first.
Steps	GSM8K recovers with T ; MCQ does not.	Tune T by task.

Table 5: Deployment lessons supported by the main and appendix results. TurboLDM-(3+1) means 3 base bits plus a 1-bit residual code.

Method	Bits	Quant. GiB	Total GiB	Compression
BF16	16.00	13.00	14.93	1.00×
W4/AWQ-4	4.25	3.45	5.38	2.77×
Turbo (4+1)	4.56	3.71	5.64	2.65×
Turbo (3+1)	3.55	2.89	4.82	3.10×

Table 6: Full deployed-checkpoint accounting. W4 denotes uniform 4-bit group quantization; AWQ-4 is activation-aware 4-bit quantization. Every PTQ row keeps the same 1.93 GiB BF16 residual outside the quantized linears; Turbo $(b+1)$ means b base bits plus a 1-bit residual code.

4.6 Storage, Runtime, and Recipes

Nominal bits are not enough for deployment decisions. Table 6 reports full checkpoint storage, and Table 7 reports live decoder memory. NF4 is nominally 4-bit, but its serialized effective storage is 6.38 bits per quantized parameter once block scales and zero-points are counted. TurboLDM-(3+1) has the smallest serialized checkpoint in the main comparison but does not lower live memory in our reference runtime because quantized linears cache BF16 dequantized buffers. The deployment choice splits three ways in this implementation: AWQ-4 for quality, bitsandbytes NF4 for live VRAM, and TurboLDM-(3+1) for checkpoint size. Optimized AWQ/GPTQ runtimes can produce different live-memory rankings.

Table 8 shows that mixed-bit recipes help mainly as storage trade-offs. The best recipe, ATTN@3 + MLP@4, saves about 250 MiB relative to uniform 4-bit linears, but it is not accuracy-equivalent to AWQ-4 or W4 under the three-seed

	64 tok		128 tok		256 tok	
Method	tok/s	VRAM	tok/s	VRAM	tok/s	VRAM
BF16	195.3	15.24	138.6	15.45	76.3	15.88
Turbo (3+1)	164.3	17.90	122.0	18.11	70.7	18.53
NF4	65.4	5.72	58.3	5.83	43.6	6.23

Table 7: Decoder throughput and active VRAM on LLaDA-8B. In this tested PyTorch/Transformers stack, NF4 is the row integrated with a mature low-bit inference kernel; the comparison does not imply that the NF4 format inherently uses less memory than optimized AWQ/GPTQ runtimes. Turbo (3+1) means TurboLDM with 3 base bits plus a 1-bit residual code.

Recipe	Bits	GiB	MMLU	ARC-C	GSM8K
ATTN@3 + MLP@4	3.94	3.20	39.3	79.0	60.0
ATTN@4 + MLP@3	3.56	2.89	39.3	78.0	54.7
ATTN@3	+	3.48	2.83	39.3	76.0
UP/GATE@3	+				
DOWN@4					
ATTN@2 + MLP@4	3.63	2.95	17.3	27.0	0.0

Table 8: Mixed-bit recipe comparison on LLaDA-8B-Instruct. In a recipe name, ATTN@3 means attention projections at 3 bits, MLP@4 means MLP projections at 4 bits, and UP/GATE@3 + DOWN@4 splits the MLP sublayers by bit width. The best storage recipe saves space but is not a quality-parity 4-bit replacement.

evaluation. More aggressive attention compression, such as ATTN@2 + MLP@4, collapses all tasks. When storage is the hard constraint, the module map can guide the recipe, but mixed bits should not be treated as a drop-in replacement for the 4-bit Pareto points.

4.7 Trajectory Diagnostics

The step-count result is explained by different uncertainty profiles. Table 9 summarizes trajectory entropy, and Table 10 measures one-step BF16-vs-quantized divergence. Multiple-choice answers often settle early: the answer letter is selected before later denoising steps can repair an early perturbation. GSM8K is different because the generated solution is longer, uncertainty is high at the beginning, and later steps can repair some quantization-induced divergence. This evidence supports a task-conditioned step rule, not a general claim that more steps fix quantization.

4.8 Depth and Bit Placement

The storage-quality curve also hides where bits are spent. Table 11 reports a depth sweep that quan-

Method	Dataset	Mean H	First H	Mean conf.
BF16	MMLU	0.215	0.147	0.991
Turbo (4+1)	MMLU	0.217	0.161	0.990
Turbo (3+1)	MMLU	0.293	0.445	0.988
BF16	GSM8K	1.750	3.546	0.950
Turbo (3+1)	GSM8K	1.866	3.720	0.952

Table 9: Trajectory entropy summary. TurboLDM-(3+1) means 3 base bits plus a 1-bit residual code; it inflates MMLU first-step entropy but barely shifts the already-uncertain GSM8K entry state.

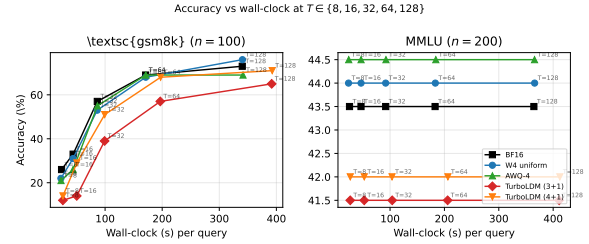


Figure 2: Step-count Pareto for GSM8K and MMLU. The figure visualizes the same task-conditioned pattern as Table 3.

tizes one contiguous eight-block range at 3-bit. Middle and late blocks are not uniformly fragile: blocks 16–23 are the worst range on both MMLU and GSM8K, but the final block range is relatively cheap on this GSM8K slice. We use this only as a diagnostic: transfer results show that family and depth rankings shift across architectures.

5 Artifacts and Reproducibility

Code, logs, scripts, and reproducibility details are included in the public repository linked in the abstract.

6 Statistical Scope and Negative Ablations

The $n=500$ evaluation changes the interpretation of several small-sample results. Table 12 reports paired-bootstrap confidence intervals on the gap to BF16. AWQ-4 is the closest tested quality-first 4-bit row overall: its MMLU, HellaSwag, and GSM8K intervals straddle zero, which supports near-parity only at the approximately 1.1–2.3 pp resolution of this experiment. Its ARC-C gap is statistically distinguishable. TurboLDM-(4+1) matches BF16 on GSM8K but loses on HellaSwag and ARC-C. TurboLDM-(3+1) is not a parity result: all four intervals are below zero. This is why we frame it as a compact checkpoint operating point rather than a replacement for AWQ-4 or

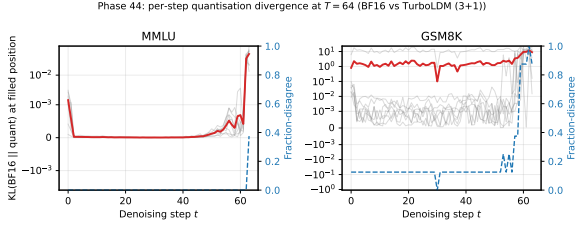


Figure 3: Trajectory lens for MMLU and GSM8K. MMLU decisions are sparse and early; GSM8K remains uncertain over more of the denoising path.

Task	Mean KL	Top-1 agree	Early KL	Late KL
MMLU	0.001	99.4%	0.000	0.006
GSM8K	2.171	80.5%	1.408	4.327

Table 10: One-step BF16-vs-quantized trajectory comparison. GSM8K has larger divergence and lower top-1 agreement than MMLU.

NF4.

The residual ablation in Table 13 gives the second reason for caution. The motivating analysis ranks layer-stage pairs by sensitivity, but the storage cost is layer-once: when any stage activates a residual, the whole layer pays for the stored residual tensor. Under that non-additive cost, sensitivity-ranked stage placement is only a heuristic. In the matched-budget evaluation, random, static-layer, and layer-once schedules match or beat stage sensitivity at similar average bits. The scheduler still identifies where perturbations appear, but the deployable knob is simpler: increase residual capacity when storage allows it.

7 Operational Recommendations

For deployment, choose by the binding resource in the evaluated stack: AWQ-4 is closest to BF16 among tested 4-bit methods, bitsandbytes NF4 reduces live VRAM, and TurboLDM-(3+1) provides the smallest serialized checkpoint. The live-memory ordering is runtime-specific; optimized AWQ/GPTQ kernels were not integrated into this stack.

Recipes should be re-measured on the target model. Individual attention projections are cheap, compound attention is harmful, and fragile MLP sublayers shift across families. Tune denoising steps by task: MCQ scoring saturates early, while GSM8K can recover with more steps.

Quantized range	Storage GiB	MMLU	GSM8K
Blocks 0–7	0.66	39.0	57.5
Blocks 8–15	0.66	42.0	62.5
Blocks 16–23	0.66	36.0	45.0
Blocks 24–31	0.66	36.0	70.0

Table 11: Per-depth sensitivity on LLaDA-8B-Instruct. Each row quantizes a contiguous transformer-block range at 3-bit and leaves the rest BF16.

Method	ARC-C	GSM8K	H.Swag	MMLU
W4	[-1.9,-0.1]	[-0.7,3.5]	[-3.7,-1.5]	[-1.4,1.3]
AWQ-4	[-2.7,-0.4]	[-1.3,3.3]	[-1.8,0.3]	[-0.8,2.1]
Q-Diff.-4	[-2.8,-0.8]	[-2.5,2.2]	[-3.5,-1.4]	[-1.1,1.4]
Turbo (4+1)	[-3.1,-0.9]	[-2.4,2.1]	[-5.2,-2.5]	[-2.3,0.4]
Turbo (3+1)	[-7.5,-4.3]	[-10.7,-5.6]	[-10.3,-6.6]	[-5.0,-1.5]

Table 12: Paired-bootstrap 95% confidence intervals on the gap to BF16 in percentage points. W4 is uniform 4-bit quantization, Q-Diff.-4 is the 4-bit Q-Diffusion-Text row, and Turbo ($b+1$) denotes TurboLDM with b base bits plus a 1-bit residual code. Intervals crossing zero are not distinguishable from BF16 at this sample size; they do not prove equivalence.

8 Runtime Diagnostic

The released TurboLDM runtime has no fused CUDA kernel. To diagnose the cost of BF16 reconstruction, the chunked proxy in Table 42 reconstructs weights in 512-row slices with UniformLinearChunked rather than keeping a persistent per-layer BF16 cache. On the reported MMLU slice, this lowers peak VRAM from 15.45 to 9.05 GiB but increases runtime from about 21 to 65 seconds; the chunked AWQ path also fails to preserve quality faithfully. We therefore treat it only as a diagnostic of reconstruction memory, not as a fused implementation or a deployable live-memory result.

9 Limitations

The main study uses batch size one, one GPU type, and one reference runtime, so its throughput measurements and live-memory ordering are not hardware-portable. TurboLDM stores compact codes but reconstructs BF16 weights; the released implementation has no fused CUDA kernel. NF4’s VRAM advantage is specific to the tested bitsandbytes integration, and optimized AWQ/GPTQ kernels could change that comparison. Several transfer and diagnostic experiments are single-seed. The LLaDA2.0-mini study lacks matched AWQ/NF4 baselines and establishes only memory

Schedule	Top frac.	Bits	GSM8K	ARC-C
Base only	0.00	3.25	60.0	77.0
Random	0.15	4.04	63.0	80.0
Static	0.15	3.51	61.0	79.0
Stage sens.	0.15	3.55	57.0	76.0
Layer-once	0.15	3.51	61.0	79.0
Stage sens.	0.30	3.79	67.0	80.0

Table 13: Matched-budget residual ablation. Sensitivity-ranked stage placement does not beat simpler schedules at similar bit budgets.

feasibility, not the cause of its quality collapse. Cross-family results are checks of where to measure, not universal module rankings. We also do not evaluate batched or concurrent serving.

Ethical Considerations

This work reduces the cost of deploying large language models. Lowering deployment cost can broaden access to capable systems, but it can also make harmful or low-quality generation cheaper. We do not introduce new training data, user data, or human-subject experiments. All evaluations use public benchmarks and public model checkpoints. We avoid releasing standalone quantized checkpoints; the artifact reproduces quantized states from upstream public checkpoints, preserving the original model licenses and access controls. Because quantization can alter behavior unevenly across tasks, we recommend task-specific evaluation before deployment, especially for safety-sensitive use cases.

Acknowledgments

This project was funded by the Qualcomm Faculty Award at IIIT, Hyderabad, India.

References

Marianne Arriola, Subham Sekhar Sahoo, Aaron Gokaslan, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Justin T Chiu, and Volodymyr Kuleshov. 2025. [Block diffusion: Interpolating between autoregressive and diffusion language models](#). In *The Thirteenth International Conference on Learning Representations*.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, and 12 others. 2025. [Llada2.0: Scaling up diffusion language models to 100b](#). ArXiv: 2512.15745.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. [Think you have solved question answering? try arc, the ai2 reasoning challenge](#). *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.

Tim Dettmers, Ruslan A. Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2024. [SpQR: A sparse-quantized representation for near-lossless LLM weight compression](#). In *The Twelfth International Conference on Learning Representations*.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. [OPTQ: Accurate quantization for generative pre-trained transformers](#). In *The Eleventh International Conference on Learning Representations*.

Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. 2023. [Ptqd: accurate post-training quantization for diffusion models](#). In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA. Curran Associates Inc.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). *Preprint*, arXiv:2009.03300.

Yushi Huang, Ruihao Gong, Jing Liu, Tianlong Chen, and Xianglong Liu. 2024. [Tfmq-dm: Temporal feature maintenance quantization for diffusion models](#). In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7362–7371.

Inception Labs. 2025. Introducing mercury, the world’s first commercial-scale diffusion large language model. <https://www.inceptionlabs.ai/blog/introducing-mercury>. Accessed 2026-06-17.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. 2025. [Mercury: Ultra-fast language models based on diffusion](#). *Preprint*, arXiv:2506.17298.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. 2023. [Q-diffusion: Quantizing diffusion models](#). In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17489–17499.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Guangxuan Xiao, and Song Han. 2025. [Awq: Activation-aware weight quantization for on-device llm compression and acceleration](#). *GetMobile: Mobile Comp. and Comm.*, 28(4):12–17.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025. [Scaling up masked diffusion models on text](#). In *The Thirteenth International Conference on Learning Representations*.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, JUN ZHOU, Yankai Lin, Jirong Wen, and Chongxuan Li. 2026. [Large language diffusion models](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS ’24*, Red Hook, NY, USA. Curran Associates Inc.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. 2023. Post-training quantization on diffusion models. In *CVPR*.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. Quip#: even better llm quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning, ICML’23*. JMLR.org.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. [Dream 7b: Diffusion large language models](#). ArXiv: 2508.15487.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. 2026. [Turboquant: Online vector quantization with near-optimal distortion rate](#). In *The Fourteenth International Conference on Learning Representations*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can](#)

[a machine really finish your sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

A Appendix Overview

The appendix gives the evidence behind the main claims: full main tables, confidence intervals, storage accounting, module-family maps, depth and step sweeps, stage and residual ablations, cross-family transfer, runtime diagnostics, trajectory analysis, adaptive-step policies, code-generation checks, and reproducibility notes. The tables come from archived result files and can be reproduced from the released scripts.

B Full Main Results and Storage Accounting

Tables 14 and 15 give the complete version of the compact main-body result. Table 14 reports the raw quality–storage trade-off for every baseline, including weak low-bit rows that are omitted from the main text. Table 15 adds three-seed means and paired-bootstrap intervals, which are the basis for calling AWQ-4 the closest tested quality-oriented 4-bit point at this resolution and TurboLDM-(3+1) a compact but non-parity point. Tables 16 and 17 separate quantized linears from the shared BF16 residual, making clear why serialized checkpoint size and live memory must be reported separately.

The full table fixes three accounting details. “Effective bits” includes block metadata, scales, zero-points, codebooks, and rotation metadata where applicable. “Total GiB” includes the shared BF16 residual outside quantized linears: embeddings, LM head, layer norms, and biases. Peak memory is measured for the live runtime, not inferred from serialized storage. This distinction is why TurboLDM is a compact-checkpoint result while NF4 is the live-memory result only in our tested runtime.

C Method and Algorithm Details

TurboLDM uses a base code for every quantized block and a residual code stored once per selected layer. The stage mask controls when the residual is used during denoising; it does not create a separate residual tensor per stage. Let $m_{\ell,s}$ denote residual activation. The scheduler ranks candidates by

$$\rho_{\ell,s} = \pi_s \alpha_{\ell,s} (\delta_{\ell}^{(0)} - \delta_{\ell}^{(1)}) / c_{\ell},$$

Table 14: Main quality–storage trade-off on LLaDA-8B-Instruct at $n=500$. Avg. bits is true compressed storage per quantized parameter; total GiB is the full checkpoint including the shared BF16 residual; peak mem is measured inference VRAM. W2/W3/W4 are uniform 2/3/4-bit group quantization rows. GPTQ-diag uses a diagonal-Hessian approximation, GPTQ-full uses the full-Hessian baseline, and Q-Diff. (4b) is the 4-bit Q-Diffusion-Text row. For TLDM/TurboLDM, $(b+1)$ means b base bits plus a 1-bit residual code. Rows with three-seed CIs are detailed in Table 15; additional low-bit baselines provide the broader Pareto context.

Method	Nom. bits	Eff. bits	Quant. GiB	Total GiB	Peak GiB	MMLU	ARC-C	H.Swag	GSM8K
BF16	16	16.00	13.00	14.93	15.6	61.8	86.0	69.2	68.6
W2	2	2.25	1.83	3.76	15.6	10.6	8.4	7.8	0.0
W3	3	3.25	2.64	4.57	15.6	53.0	77.6	58.8	54.8
W4	4	4.25	3.45	5.38	15.6	61.8	85.0	66.6	70.0
GPTQ-diag-3	3	3.25	2.64	4.57	15.6	44.4	62.2	56.0	13.4
GPTQ-diag-4	4	4.25	3.45	5.38	15.6	57.5	80.7	62.7	59.3
GPTQ-full-3	3	3.25	2.64	4.57	15.6	53.2	74.6	55.0	35.0
GPTQ-full-4	4	4.25	3.45	5.38	15.6	60.0	84.8	65.2	66.6
AWQ-3	3	3.25	2.64	4.57	15.6	53.6	76.0	61.8	39.6
AWQ-4	4	4.25	3.45	5.38	15.6	62.4	84.5	68.4	69.5
NF4	4	6.38	5.18	7.11	6.0	60.2	85.4	68.8	69.0
Q-Diff. (4b)	4	4.48	3.64	5.57	15.6	62.0	84.2	66.8	68.5
TLDM (3+1)	3	3.55	2.89	4.82	18.0	58.6	80.1	60.8	60.4
TLDM (4+1)	4	4.56	3.71	5.64	18.1	60.8	84.0	65.3	68.5

then activates candidates until the layer-once storage budget is reached. Because the true cost is $\sum_{\ell} c_{\ell} \mathbf{1}\{\exists s : m_{\ell,s} = 1\}$, this greedy rule is a heuristic rather than the exact optimizer of the non-additive objective.

Algorithm 1 TurboLDM calibration and encoding

- 1: Collect calibration states and assign each state to a masking-ratio stage.
- 2: **for** each quantized layer ℓ and stage s **do**
- 3: Quantize only layer ℓ with the base code.
- 4: Estimate $\alpha_{\ell,s}$ from masked-token KL.
- 5: Rank layer-stage pairs by $\rho_{\ell,s}$.
- 6: Activate residuals until the layer-once storage budget is reached.
- 7: **for** each quantized matrix **do**
- 8: Encode base blocks with Hadamard rotation and a Gaussian codebook.
- 9: Store one residual code if any selected stage uses that layer.

D Module-Family and Component Sensitivity

Tables 18, 19, and 20 refine the module-family map. The isolated-family and compound-family tables show that attention projections are cheap in isolation but harmful when quantized together. Tables 21, 22, and 23 test whether the unquantized embedding/head portion is merely accounting overhead. These checks show that head and embedding decisions can change quality, so the

BF16 residual should be treated as part of the deployed model rather than ignored.

These tables separate isolated-family quantization from compound quantization. Individual attention projections are cheap to quantize, whereas quantizing the full attention block is much more damaging. Head and embedding checks show that the shared BF16 residual is part of the model, not bookkeeping: the output head and embeddings remain important for quality and explain part of the fixed storage floor.

E Depth, Step Count, and Trajectory Diagnostics

Table 24 quantizes contiguous block ranges and shows that mid-late blocks are more fragile than early blocks. Tables 25, 26, 27, and 28 then vary denoising steps and bit budgets; together they establish that GSM8K can recover with additional steps while MMLU is largely step-invariant. Table 29 measures entropy and confidence along the trajectory, while Table 30 measures BF16-vs-quantized KL and top-1 agreement. These diagnostics are the evidence for the main text’s task-conditioned step-count rule.

F Calibration, Stages, Residuals, and Bit Budgets

Tables 31, 32, and 33 cover calibration size and stage-bin choices. They show why the apparent stage-stratification gain in small samples is not a

Table 15: Main quality–storage trade-off on LLaDA-8B-Instruct with 3 seeds and paired bootstrap 95% CIs. Each cell is the across-seed mean accuracy $\pm$ the seed standard deviation; the trailing [lo, hi] is the paired bootstrap 95% CI on the accuracy gap to BF16 in percentage points (negative = method is worse than BF16). W3/W4 are uniform 3/4-bit group quantization, GPTQ-d4 is diagonal-Hessian 4-bit GPTQ, QDiff-4 is 4-bit Q-Diffusion-Text, and TLDM-($b+1$) is TurboLDM with b base bits plus a 1-bit residual code. A gap whose 95% CI straddles zero cannot be distinguished from BF16 at $n=500$; it indicates near-parity at this resolution, not proven equivalence.

Method	Bits	Total GiB	ARC-C	CI vs BF16	GSM8K	CI vs BF16	H.Swag	CI vs BF16	MMLU	CI vs BF16
BF16	16.00	14.93	86.0 $\pm$ 0.0	ref	68.6 $\pm$ 0.0	ref	69.2 $\pm$ 0.0	ref	61.8 $\pm$ 0.0	ref
W3	3.25	2.64	77.6 $\pm$ 0.0	[-10.3,-6.6]	54.8 $\pm$ 0.0	[-16.6,-11.2]	58.8 $\pm$ 0.0	[-12.3,-8.4]	53.0 $\pm$ 0.0	[-10.7,-6.8]
W4	4.25	3.45	85.0 $\pm$ 0.0	[-1.9,-0.1]	70.0 $\pm$ 0.0	[-0.7,3.5]	66.6 $\pm$ 0.0	[-3.7,-1.5]	61.8 $\pm$ 0.0	[-1.4,1.3]
GPTQ-d4	4.25	3.45	80.7 $\pm$ 0.9	[-6.7,-3.8]	59.3 $\pm$ 0.6	[-11.8,-6.7]	62.7 $\pm$ 0.9	[-8.2,-4.8]	57.5 $\pm$ 0.6	[-6.0,-2.5]
AWQ-4	4.25	3.45	84.5 $\pm$ 0.6	[-2.7,-0.4]	69.5 $\pm$ 1.1	[-1.3,3.3]	68.4 $\pm$ 0.7	[-1.8,0.3]	62.4 $\pm$ 0.9	[-0.8,2.1]
QDiff-4	4.48	3.64	84.2 $\pm$ 0.0	[-2.8,-0.8]	68.5 $\pm$ 2.8	[-2.5,2.2]	66.8 $\pm$ 0.2	[-3.5,-1.4]	62.0 $\pm$ 0.3	[-1.1,1.4]
TLDM-3+1	3.55	2.89	80.1 $\pm$ 1.1	[-7.5,-4.3]	60.4 $\pm$ 2.6	[-10.7,-5.6]	60.8 $\pm$ 1.8	[-10.3,-6.6]	58.6 $\pm$ 0.7	[-5.0,-1.5]
TLDM-4+1	4.56	3.71	84.0 $\pm$ 1.2	[-3.1,-0.9]	68.5 $\pm$ 0.7	[-2.4,2.1]	65.3 $\pm$ 1.2	[-5.2,-2.5]	60.8 $\pm$ 0.2	[-2.3,0.4]

Table 16: Per-parameter storage audit on LLaDA-8B-Instruct. “Quant. linears” sums the bytes that the quantizer actually rewrote (indices, scales, signs, residual, codebook); “BF16 residual” is the bytes that every row keeps at BF16 (embeddings, LM head, layer norms, biases). The full checkpoint is the sum. The 4.25-bit uniform and AWQ rows collapse onto the same storage (same q /scales layout); the difference between them is entirely in weight values, not bit width. Every row carries the *same* BF16 residual, so the method ordering in Table 14 does not depend on how one reports compression. Source: runs/overnight/storage_audit_summary.csv.

Method	Quant. bits	Quant. linears	BF16 residual	Total checkpoint	Compression vs BF16
BF16 baseline	16.00	13.00 GiB	1.93 GiB	14.93 GiB	1.00 $\times$
W4 uniform	4.25	3.45 GiB	1.93 GiB	5.38 GiB	2.77 $\times$
AWQ-4	4.25	3.45 GiB	1.93 GiB	5.38 GiB	2.77 $\times$
TurboLDM (4+1)	4.56	3.71 GiB	1.93 GiB	5.64 GiB	2.65 $\times$
TurboLDM (3+1)	3.55	2.89 GiB	1.93 GiB	4.82 GiB	3.10 $\times$

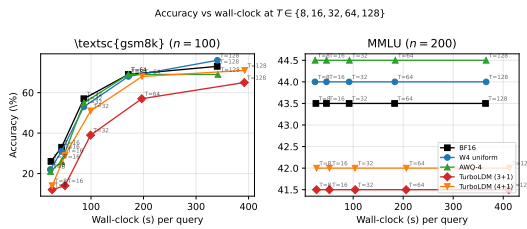


Figure 4: Accuracy vs. denoising-step budget for GSM8K and MMLU. GSM8K recovers with more steps; MMLU is flat in this grid.

reliable deployment rule. Table 34 is the matched-budget residual-scheduler ablation: it compares random, static, stage-sensitive, entropy-gated, and layer-once schedules at similar bit budgets. The ablation grid below collects the smaller-sample variants and identifies residual capacity, not exact stage placement, as the repeatable lever.

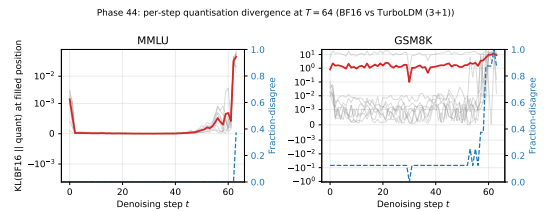


Figure 5: Trajectory lens for MMLU and GSM8K. MMLU decisions are sparse and early, while GSM8K remains uncertain over more of the denoising path and can recover from additional steps.

Stage-stratified vs pooled calibration.

Variant	Avg. bits	MMLU	GSM8K
Stage-stratified (S=4)	3.53	37.0	52.5
Pooled (S=1)	3.50	37.0	47.5

Residual schedule.

Table 17: Storage audit on LLaDA-8B-Instruct. *Total* is the full model footprint (quantized linears + BF16 embeddings/head/norms). *BF16 residual* is the unquantized portion (embeddings + LM head + layer norms + biases). Compression ratios are reported *against the full BF16 model*, rather than only the quantized linears, so they reflect what a deployed checkpoint actually weighs.

Method	Total GiB	BF16 residual GiB	Quantized part GiB	Compression vs BF16
BF16 baseline	14.93	14.93	0.00	1.00×
W4 uniform	5.38	1.93	3.45	2.77×
AWQ-4	5.38	1.93	3.45	2.77×
TurboLDM (4+1)	5.64	1.93	3.71	2.65×
TurboLDM (3+1)	4.82	1.93	2.89	3.10×

Table 18: Pilot isolated-projection sensitivity on LLaDA-8B (MMLU-100). Each row quantizes only the named projection or sublayer with 3-bit TurboQuant (no residual). “Storage GB” is the compressed storage of that named quantized projection or sublayer only; the BF16 baseline quantizes none, so this quantity is not applicable and is shown as “–”. Compound-family conclusions use the corrected $n=500$ rerun in Table 20. Δ is change vs. BF16 baseline.

Module family	Storage GB	MMLU	Δ vs. BF16
mlp_down	0.62	35.0	-3.0
v_proj	0.21	38.0	+0.0
k_proj	0.21	39.0	+1.0
mlp_gate	0.62	40.0	+2.0
mlp_up	0.62	40.0	+2.0
q_proj	0.21	41.0	+3.0
attn_out	0.21	41.0	+3.0
BF16 baseline	–	38.0	0.0

Variant	Avg. bits	MMLU	GSM8K
No residual	3.26	36.0	62.5
Static mixed (Q-Diff.)	4.48	41.0	52.5
Stage residual	3.53	37.0	52.5
Stage + entropy gating	3.54	38.0	55.0

Sensitivity proxy.

Variant	Avg. bits	MMLU	GSM8K
hidden MSE	3.53	41.0	45.0
attention MSE	3.51	37.0	62.5
activation max	3.60	36.0	50.0

Block size.

Variant	Avg. bits	MMLU	GSM8K
block=64	3.83	37.0	62.5
block=128	3.53	33.0	55.0

Base/residual bit configuration.

Variant	Avg. bits	MMLU	GSM8K
base=2 res=1	2.54	26.0	20.0
base=3 res=0	3.26	38.0	52.5
base=3 res=1	3.53	42.0	62.5
base=3 res=2	3.71	34.0	60.0
base=4 res=0	4.26	34.0	60.0
base=4 res=1	4.54	35.0	57.5

The calibration appendix is cautious. Stage-stratified calibration appears promising in the smallest sample but does not retain its margin on the larger evaluation. Matched-budget residual placement also does not favor the sensitivity-ranked stage schedule. The most repeatable knob is residual capacity, especially the transition from weaker low-bit settings to the (3+1) operating point.

G Mixed-Bit Recipes

Table 35 reports a mixed-bit recipe comparison, with recipes selected from the module-family map. Table 36 evaluates the best recipes with larger n and three seeds. The larger table supports the final interpretation: mixed-bit recipes can save storage, but they do not replace the 4-bit Pareto points at quality parity.

The mixed-bit tables support a storage-first interpretation. The best hand recipe saves storage relative to uniform 4-bit linears, but the larger evaluation does not make it statistically interchangeable with the best 4-bit baselines. More aggressive attention compression collapses, consistent with the compound-attention sensitivity result.

H Extra Tasks and Baseline Variants

Table 37 adds WinoGrande, OpenBookQA, and BoolQ to check that the main-task conclusions are not specific to MMLU/ARC/GSM8K. Table 38 distinguishes GPTQ-full from the diagonal approximation used in earlier rows. Table 39 reports HumanEval/MBPP checks. LLaDA-8B is not code-tuned, so these numbers are an auxiliary robustness check rather than a code-model deployment claim.

The extra MCQ datasets extend task coverage beyond the four main benchmarks. GPTQ-full closes much of the gap left by the diagonal-Hessian variant, confirming that GPTQ-diag

Table 19: Isolated-projection sensitivity across four tasks with three seeds. Each row quantises *only* the named projection or sublayer with 3-bit TurboQuant (no residual); everything else stays BF16. Reported as mean $\pm$ std percentage accuracy over seeds {42, 43, 44}. The erroneous duplicate compound rows from the review version are removed; Table 20 gives the corrected joint-family rerun.

Family	MMLU	ARC-C	HellaSwag	GSM8K
attn_out	41.7 $\pm$ 0.6	83.0 $\pm$ 1.3	66.7 $\pm$ 0.3	68.7 $\pm$ 3.5
q_proj	42.3 $\pm$ 1.0	84.3 $\pm$ 0.3	66.4 $\pm$ 1.0	69.7 $\pm$ 1.5
k_proj	42.0 $\pm$ 1.3	83.5 $\pm$ 0.5	66.3 $\pm$ 0.7	68.0 $\pm$ 0.0
v_proj	43.2 $\pm$ 1.6	82.7 $\pm$ 1.0	65.1 $\pm$ 0.5	65.3 $\pm$ 1.5
ff_proj	43.0 $\pm$ 1.0	82.0 $\pm$ 1.5	65.9 $\pm$ 0.5	67.0 $\pm$ 5.3
ff_out	44.5 $\pm$ 1.8	82.2 $\pm$ 1.2	66.1 $\pm$ 0.7	66.0 $\pm$ 3.6

Table 20: Corrected compound-family rerun on LLaDA-8B-Instruct, also summarized in Table 2: full compound families (attn_all = {q, k, v, attn_out}, mlp_all = {ff_proj, ff_out}) are quantised together at 3-bit (no residual). “GB” is the compressed storage of the named quantized family only. The BF16 reference quantizes no family, so this family-only storage quantity is not applicable and is shown as “-”. Single seed, $n=500$. At the compound level attention is the more sensitive family.

Family	GB	MMLU	ARC-C	H.Swag	GSM8K
BF16 reference	-	61.8	86.0	69.2	68.6
attn_all	0.82	55.6	77.6	60.0	55.2
mlp_all	1.23	60.8	83.6	66.0	67.6

is a lower-bound implementation. The code-generation check is outside the main benchmark family. NF4 is the highest-scoring tested row on these two slices, while TurboLDM-(3+1) gives up accuracy on both HumanEval and MBPP.

I Runtime, Memory, NF4, and Seed Variance

Tables 40 and 41 report wall-clock throughput and active VRAM. They separate the reference TurboLDM implementation, which dequantizes into BF16 buffers, from NF4, which is integrated with a low-bit kernel in the tested stack. Table 42 removes the persistent BF16 dequantization cache by reconstructing weights in row chunks during the forward pass. This reduces peak VRAM but slows evaluation, so it is a reconstruction-memory diagnostic, not a fused kernel. Table 43 extends the NF4 check across model families. Tables 44 and 45 quantify seed variance, especially on GSM8K. The larger paired-bootstrap evaluation is the basis for the main claims.

In the end-to-end decoder study, NF4 is the integrated low-bit kernel path in the tested stack.

Table 21: Embedding/head/module-family ablation on LLaDA-8B-Instruct, all using the TurboLDM (3+1) recipe, i.e., 3 base bits plus a 1-bit residual code. “Quant. GiB” is the size of the low-bit linears only; the BF16 residual varies across rows because each ablation leaves a different subset untouched. “With head” quantizes the $126,464 \times 4096$ LM head (which the default recipe leaves in BF16) by raising the iter_named_linears cap from 5×10^4 to 2×10^5 ; quantizing the embedding table would require a QuantizedEmbedding module and is out of scope. All scores are MMLU with $n=75$ examples (std ~ 5.7 pp at this n); differences inside that envelope are noise, differences outside it are signal.

Ablation	Avg. bits	Quant. GiB	MMLU (n=75)
Default (attn + MLP; embed, head, norms BF16)	3.55	2.89	0.360
+ LM head quantized (3+1, cap raised)	3.61	2.94	0.453
Attn family only (MLP left BF16)	3.68	0.92	0.347
MLP family only (attn left BF16)	3.46	1.95	0.387

It is slower in this measurement because dequantization occurs inside the matmul, but it materially lowers active inference VRAM. The chunked-dequant diagnostic shows that removing the cached BF16 buffer can recover part of the memory gap, but the released TurboLDM implementation remains a storage result rather than a deployment-speed runtime. Seed tables show that small-sample GSM8K differences can fall within seed variance, which is why we emphasize paired-bootstrap intervals for the main conclusions.

J Cross-Family Transfer

Table 46 evaluates Dream-7B with its native decoder and shows that Dream is more fragile under the same PTQ recipes than LLaDA. Table 47 re-

Table 22: LM-head quantisation across four tasks with three seeds. For TurboLDM, $(b+1)$ means b base bits plus a 1-bit residual code; we compare leaving the LM head at BF16 with quantising it using the same recipe. Scores are accuracy percentage: across-seed mean $\pm$ seed standard deviation. MMLU $n=200$, ARC-C $n=200$, HellaSwag $n=300$, GSM8K $n=100$.

Method	LM head	MMLU	ARC-C	HellaSwag	GSM8K
TurboLDM (3+1)	BF16	42.0 $\pm$ 1.3	75.5 $\pm$ 0.9	59.6 $\pm$ 1.5	61.3 $\pm$ 4.0
TurboLDM (3+1)	quantised	42.3 $\pm$ 0.6	74.8 $\pm$ 1.3	59.3 $\pm$ 1.2	60.7 $\pm$ 2.5
TurboLDM (4+1)	BF16	42.0 $\pm$ 1.0	82.0 $\pm$ 1.0	62.9 $\pm$ 1.5	66.3 $\pm$ 2.9
TurboLDM (4+1)	quantised	43.3 $\pm$ 1.5	80.5 $\pm$ 0.9	63.0 $\pm$ 0.7	63.7 $\pm$ 3.1
AWQ-4	BF16	44.7 $\pm$ 1.3	82.8 $\pm$ 1.3	67.1 $\pm$ 0.5	69.7 $\pm$ 1.2

Table 23: 4-bit groupwise QuantizedEmbedding for the input embedding table on top of TurboLDM-(3+1) (3 base bits plus a 1-bit residual code). This saves ~ 780 MiB of BF16 residual at the cost of an in-flight dequantisation per gathered row. Compare the rows below to the TurboLDM-(3+1) row of Table 14.

Dataset	Avg bits	Storage GB	Score	Wall (s)
MMLU	3.55	2.89	58.2	131.9
ARC-C	3.55	2.89	79.6	132.3
HellaSwag	3.55	2.89	60.8	150.5
GSM8K	3.55	2.89	58.6	981.0

Table 24: Per-depth sensitivity on LLaDA-8B-Instruct. Each row quantizes a contiguous transformer-block range at 3-bit uniform and leaves the rest BF16.

Block range	Storage GB	MMLU	GSM8K
blocks 0-7	0.66	39.0	57.5
blocks 8-15	0.66	42.0	62.5
blocks 16-23	0.66	36.0	45.0
blocks 24-31	0.66	36.0	70.0

peats the per-family map on Dream and preserves the compound-attention warning. Table 48 moves to MDLM-OWT and shows that the fragile MLP sublayer can change by architecture. Tables 49, 50, 51, and 52 test encoder-only masked LMs and show that encoder transfer is qualitatively different from generative diffusion-LM transfer. Tables 53 and 54 cover sparse/MoE scaling and lazy reconstruction; they demonstrate memory feasibility but not quality parity at 16B.

Transfer is uneven. The compound-attention pattern transfers to Dream-7B, but Dream is more PTQ-fragile overall. MDLM-OWT changes which MLP sublayer is most fragile. Encoder-only masked LMs are easier under 4-bit quantization than the generative diffusion LMs in these slices. The 16B lazy-reconstruction experiment establishes memory feasibility, but its quality collapse and missing AWQ/NF4 controls prevent

treating it as a deployment solution or assigning the failure to lazy reconstruction alone.

K Adaptive Denoising and Weight Fingerprints

Table 55 evaluates an adaptive denoising-step policy that escalates when an answer is not parseable or low-margin. It matches the step-count tables: adaptive compute helps GSM8K-like generation in some rows but does little for MMLU-like MCQ scoring. Table 56 reports correlations between simple weight statistics and observed fragility. These fingerprints do not replace direct calibration, but they identify cheap statistics worth testing for sensitivity prediction.

Adaptive- T policies matter because step count is one of the few deployment knobs that interacts directly with diffusion-LM quality. The fingerprint table records correlations between layer statistics and observed family fragility. These diagnostics are not standalone selection rules, but they point to cheaper sensitivity predictors.

L Figures

This appendix collects the full-size plots used in the main analysis. Figure 6 expands the stage-by-layer sensitivity view behind the compound-attention result. Figure 7 shows the quality-storage frontier that separates AWQ-4, NF4, and the lower-storage TurboLDM rows. The step-count and trajectory plots are shown earlier in Appendix E: Figure 4 shows that GSM8K benefits from more denoising steps while MMLU does not, and Figure 5 explains that difference through the timing of token decisions along the denoising path.

M Implementation Notes

All quantized linears preserve the same `forward(x)` interface as `nn.Linear`, which

Table 25: Step-count $\times$ quantisation Pareto. Each cell is accuracy% (wall seconds). Five methods $\times$ five T values on MMLU $n=200$ and GSM8K $n=100$, single seed.

Method / dataset	$T=8$	$T=16$	$T=32$	$T=64$	$T=128$
BF16 / MMLU	43.5 (25s)	43.5 (47s)	43.5 (92s)	43.5 (183s)	43.5 (365s)
BF16 / GSM8K	26.0 (24s)	33.0 (44s)	57.0 (86s)	69.0 (171s)	73.0 (340s)
W4 / MMLU	44.0 (24s)	44.0 (47s)	44.0 (92s)	44.0 (183s)	44.0 (366s)
W4 / GSM8K	22.0 (23s)	31.0 (44s)	53.0 (86s)	68.0 (171s)	76.0 (341s)
AWQ-4 / MMLU	44.5 (24s)	44.5 (47s)	44.5 (92s)	44.5 (184s)	44.5 (366s)
AWQ-4 / GSM8K	21.0 (23s)	26.0 (44s)	55.0 (87s)	69.0 (171s)	69.0 (342s)
TurboLDM (3+1) / MMLU	41.5 (27s)	41.5 (52s)	41.5 (104s)	41.5 (206s)	41.5 (411s)
TurboLDM (3+1) / GSM8K	12.0 (26s)	14.0 (50s)	39.0 (99s)	57.0 (197s)	65.0 (392s)
TurboLDM (4+1) / MMLU	42.0 (27s)	42.0 (53s)	42.0 (104s)	42.0 (206s)	42.0 (412s)
TurboLDM (4+1) / GSM8K	14.0 (26s)	29.0 (51s)	51.0 (99s)	68.0 (197s)	71.0 (393s)

Table 26: Step-count $\times$ quantization interaction on LLaDA-8B.

Method	Dataset	$T=16$	$T=32$	$T=64$
FP16/BF16	MMLU	38.0	38.0	38.0
FP16/BF16	GSM8K	25.0	60.0	62.5
AWQ-4	MMLU	45.0	45.0	45.0
AWQ-4	GSM8K	22.5	45.0	70.0
TurboLDM (3+1)	MMLU	36.0	36.0	36.0
TurboLDM (3+1)	GSM8K	15.0	37.5	52.5

Table 27: Bit-width versus denoising-step trade-off on LLaDA-8B-Instruct ($n=150$ GSM8K, $n=200$ MMLU). In TurboLDM (3+1), 3 is the base-bit code and +1 is a 1-bit residual code. More denoising steps recover GSM8K accuracy but leave MMLU unchanged in this grid.

Method	Bits	GSM8K (steps)			MMLU (steps)		
		16	32	64	16	32	64
BF16	16.00	38.0	55.3	68.0	44.0	44.0	44.0
TurboLDM (4+1)	4.55	26.0	46.7	66.7	43.0	43.0	43.0
TurboLDM (3+1)	3.55	18.0	40.0	57.3	40.5	40.5	40.5

is why the same decoder code can be used for BF16 and quantized rows. The reference TurboLDM runtime favors simple verification over deployment efficiency: it stores compact codes on disk but dequantizes to BF16 buffers for forward passes. NF4 uses `bnb.Linear4bit`; its integration is the source of the observed live-memory advantage in this stack, not an inherent property of the NF4 format. Optimized AWQ/GPTQ kernels exist in other runtimes. The GPTQ-dia rows use a diagonal Hessian approximation and are not claimed to represent the strongest possible GPTQ implementation; the GPTQ-full rows are included to make this distinction explicit. The AWQ rows use an activation-aware scaling fallback inside the same wrapper stack, not an AutoAWQ deployment kernel; accordingly they should be read as quality baselines rather than low-memory

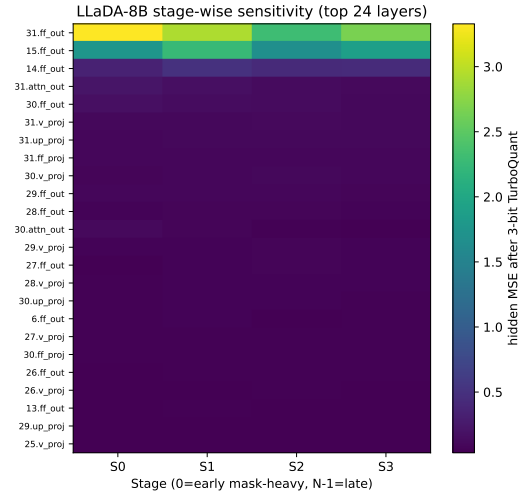


Figure 6: Stage-wise sensitivity heatmap for the most sensitive LLaDA-8B linear layers. Sensitivity is concentrated in specific module families rather than uniformly spread across depth and denoising stage.

runtime claims.

N Compute and Reproducibility

Experiments were run on a single NVIDIA RTX 5090 with 32 GB VRAM, PyTorch 2.11, CUDA 13.0, and transformers 4.55. Calibration uses WikiText-103 snippets capped at 256 tokens. The main comparison uses $n=500$ examples and three seeds for non-degenerate rows; many diagnostic and transfer checks are single-seed and labeled accordingly. The supplementary artifact includes the generated tables, figures, and logs needed to verify the reported numbers. Two raw trajectory dumps are too large for ordinary Git storage and are omitted; the repository includes their derived tables and summaries. Camera-ready source and the public artifact are available at <https://github.com/misterpaw>

Table 28: Step-budget ablation under quantisation on LLaDA-8B. TurboLDM-(3+1) means 3 base bits plus a 1-bit residual code. MMLU is T -invariant for both TurboLDM and NF4, while GSM8K improves monotonically across the tested range.

Method / Task	T					
	2	4	8	16	32	64
NF4 / MMLU	0.445	0.420	0.410	0.410	0.410	0.410
TurboLDM-(3+1) / MMLU	0.400	0.415	0.410	0.410	0.410	0.410

Method / Task	T			
	16	32	64	128
NF4 / GSM8K	0.34	0.52	0.69	0.76
TurboLDM-(3+1) / GSM8K	0.14	0.39	0.57	0.65

Table 29: Trajectory entropy and per-step confidence on LLaDA-8B over 32 prompts (16 MMLU + 16 GSM8K) at $T=64$. TLDM abbreviates TurboLDM, and $(b+1)$ means b base bits plus a 1-bit residual code. TLDM (3+1) raises MMLU early-step uncertainty, while TLDM (4+1) is entropy-equivalent to BF16 on GSM8K.

Method	Dataset	Mean H	First H	Final H	Mean conf.
BF16	MMLU	0.215	0.147	1.291	0.991
BF16	GSM8K	1.750	3.546	1.116	0.950
TLDM	MMLU	0.217	0.161	1.284	0.990
(4+1)					
TLDM	GSM8K	1.727	3.637	1.013	0.951
(4+1)					
TLDM	MMLU	0.293	0.445	1.384	0.988
(3+1)					
TLDM	GSM8K	1.866	3.720	1.120	0.952
(3+1)					

Table 30: Trajectory quantisation lens. TurboLDM-(3+1) means 3 base bits plus a 1-bit residual code. Each row reports BF16-vs-TurboLDM KL at the filled position plus top-1 agreement; on GSM8K late-step KL is much smaller than early, while MCQ changes little.

Task	mean KL (all steps)	mean top-1 agree	mean KL early	mean KL late
MMLU	0.001	99.4%	0.000	0.006
GSM8K	2.171	80.5%	1.408	4.327

[an/turbo-quant-paper.git](https://github.com/AnTao/turbo-quant-paper).

Use of LLMs

Large language models were used to improve writing and editing. The research ideas, experimental design, and experimental plan were developed by the authors. A coding assistant was used for implementation support.

Table 31: Calibration-size sweep for TurboLDM (3+1), meaning 3 base bits plus a 1-bit residual code, on LLaDA-8B-Instruct. All five calibration sizes sit inside the MMLU $n=75$ noise envelope, so this slice does not show a reliable calibration-size effect.

N_{cal}	Avg. bits	MMLU ($n=75$)	Wall (s)
8	3.54	0.347	20.9
24 (default)	3.55	0.360	20.2
64	3.57	0.360	20.5
128	3.57	0.360	20.3
256	3.57	0.320	20.0

Table 32: Stage-bin ablation for TurboLDM (3+1), meaning 3 base bits plus a 1-bit residual code, on LLaDA-8B-Instruct. All four S values fall inside the MMLU $n=75$ noise envelope; residual capacity, rather than stage placement, is the stable signal.

S	Avg. bits	MMLU ($n=75$)	Wall (s)
1 (pooled)	3.50	0.413	18.9
2	3.53	0.333	18.2
4 (default)	3.55	0.360	20.3
8	3.55	0.360	20.2

Table 33: Stage-stratified calibration on GSM8K and MMLU. TurboLDM (3+1) means 3 base bits plus a 1-bit residual code. Pooled $S=1$ stays within $\pm 5\text{pp}$ of $S=4$ on GSM8K, and MMLU is flat, so stage placement is not reliable here.

Dataset	S	$N_{\text{cal}} = 24$	$N_{\text{cal}} = 64$
GSM8K	1	61.0	60.0
GSM8K	2	56.0	61.0
GSM8K	4	57.0	57.0
GSM8K	8	57.0	57.0
MMLU	1	44.0	39.5
MMLU	2	39.0	42.5
MMLU	4	41.5	41.0
MMLU	8	41.5	41.0

Table 34: Matched-budget residual-schedule ablation. Every row uses base_bits= 3, residual_bits= 1, $S=4$; we vary only the schedule mode and residual top-fraction. Three coverages compare the sensitivity-ranked “stage” mode at matched average bits against unscheduled baselines. Score is accuracy percentage on a single seed; the headline (3+1) operating point in the main table corresponds to the “stage, 0.15” row.

Schedule mode	top-frac	Avg bits	GSM8K	MMLU	ARC-C
No residual (base only)	0.00	3.25	60.0	41.5	77.0
Random (top-frac pairs)	0.15	4.04	63.0	42.5	80.0
Static layer (all stages)	0.15	3.51	61.0	40.0	79.0
Stage sensitivity (default)	0.15	3.55	57.0	41.5	76.0
Stage + entropy gating	0.15	3.55	57.0	41.5	76.0
Layer-once knapsack	0.15	3.51	61.0	40.0	79.0
Stage sensitivity (default)	0.30	3.79	67.0	42.5	80.0
Stage sensitivity (default)	0.60	4.23	66.0	41.5	78.5

Table 35: Mixed-bit recipes on LLaDA-8B-Instruct. Each row applies different bit-widths to different module families in the same model; @3 and @4 denote the bit width assigned to that family, and up, gate, and down are the MLP sublayers.

Recipe	Avg. bits	Storage GB	MMLU	ARC-C	H.Swag	GSM8K	Mean
attn@3 + MLP@4	3.94	3.20	39.3	79.0	59.5	60.0	59.5
attn@4 + MLP@3	3.56	2.89	39.3	78.0	63.5	54.7	58.9
attn@3 + up&gate@3 + down@4	3.48	2.83	39.3	76.0	57.5	53.3	56.5
attn@2 + MLP@4	3.63	2.95	17.3	27.0	29.5	0.0	18.5

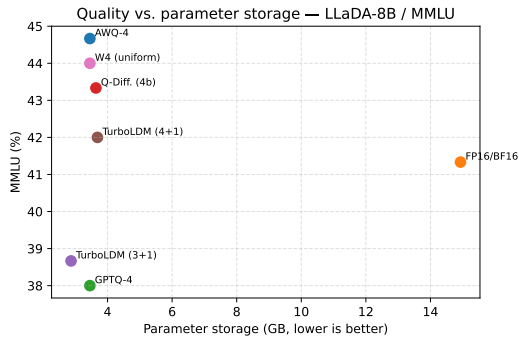


Figure 7: Quality–storage Pareto curve. AWQ-4 occupies the highest-quality end of the curve, while lower-bit TurboLDM rows move toward smaller checkpoints with larger quality loss.

Table 36: Mixed-bit recipes at $n=500$ with three seeds. Mean $\pm$ std accuracy. In recipe names, @3 and @4 denote the bit width assigned to that module family; up, gate, and down are MLP sublayers. Compare against the corresponding W4 / AWQ-4 / TurboLDM rows of Table 14.

Recipe	MMLU	ARC-C	H.Swag	GSM8K
attn@3 + MLP@4	58.2 $\pm$ 0.0	82.4 $\pm$ 0.0	61.0 $\pm$ 0.0	67.0 $\pm$ 0.0
attn@4 + MLP@3	56.2 $\pm$ 0.0	81.4 $\pm$ 0.0	65.2 $\pm$ 0.0	62.8 $\pm$ 0.0
attn@3 + up&gate@3 + down@4	56.4 $\pm$ 0.0	79.8 $\pm$ 0.0	59.6 $\pm$ 0.0	59.8 $\pm$ 0.0
attn@2 + MLP@4	20.0 $\pm$ 0.0	27.8 $\pm$ 0.0	24.6 $\pm$ 0.0	1.0 $\pm$ 0.0

Table 37: Additional MCQ benchmarks on LLaDA-8B-Instruct. TLDM abbreviates TurboLDM; $(b+1)$ means b base bits plus a 1-bit residual code.

Method	Bits	WinoG	OBQA	BoolQ
FP16/BF16	16.00	80.0	84.5	89.0
AWQ-4	4.25	80.5	85.5	90.0
TLDM (4+1)	4.58	78.5	83.5	89.0
TLDM (3+1)	3.57	74.5	79.0	87.5

Table 38: Full-Hessian GPTQ baseline at $n=500$. For each method the value is single-seed accuracy.

Method	Avg bits	MMLU	ARC-C	H.Swag	GSM8K
gptq_full_3	3.25	53.2	74.6	55.0	35.0
gptq_full_4	4.25	60.0	84.8	65.2	66.6

Table 39: Code-generation check on LLaDA-8B. TurboLDM-(3+1) means 3 base bits plus a 1-bit residual code. We report single-sample pass@1 with a sandboxed Python executor on HumanEval ($n=80$) and MBPP ($n=150$, full config). LLaDA-8B-Instruct is not code-tuned, so the absolute MBPP scores are low; the table is an auxiliary robustness check rather than a code-model claim. NF4 is the highest-scoring tested row on both slices, while TurboLDM-(3+1) trails BF16 on HumanEval and MBPP.

Method	HumanEval pass@1	MBPP pass@1
BF16	0.625	0.040
NF4	0.663	0.047
TurboLDM-(3+1)	0.588	0.020

Table 40: Wall-clock throughput (tokens/sec) on a single NVIDIA RTX 5090. Config values gen64 and gen128 denote generation lengths of 64 and 128 tokens. Peak mem is GPU peak memory during generation; our reference runtime caches the dequantized weight, so the storage compression in Table 14 does not translate to lower peak inference VRAM.

Method	Avg. bits	Config	Peak mem. GB	Tokens/sec	Wall (s)
FP16/BF16	16.00	gen64	15.2	197.0	0.6
FP16/BF16	16.00	gen128	15.5	137.9	1.9
W4 (uniform)	4.25	gen64	15.2	196.8	0.7
W4 (uniform)	4.25	gen128	15.5	137.9	1.9
AWQ-4	4.25	gen64	15.2	196.8	0.7
AWQ-4	4.25	gen128	15.5	138.3	1.9
TurboLDM (4+1)	4.57	gen64	18.1	165.1	0.8
TurboLDM (4+1)	4.57	gen128	18.3	122.0	2.1
TurboLDM (3+1)	3.56	gen64	17.9	163.4	0.8
TurboLDM (3+1)	3.56	gen128	18.1	121.8	2.1

Table 41: Decoder throughput on LLaDA-8B at three generation lengths, batch 1, single warm-up plus one timed pass averaged across two prompts. Among these end-to-end decoder rows, NF4 is the one integrated with a low-bit kernel in our stack, and the peak VRAM column reflects that: NF4 cuts active memory from ~ 15 GiB to ~ 6 GiB. The reference TurboLDM implementation caches a dequantised BF16 buffer per linear, so its peak VRAM is intentionally the bf16 footprint plus the small dequantised overhead. These measurements characterize the tested implementation, not an inherent NF4-format advantage; optimized AWQ/GPTQ kernels also exist.

Method	gen_len=64, $T=32$		gen_len=128, $T=64$		gen_len=256, $T=128$	
	tok/s	VRAM (GiB)	tok/s	VRAM (GiB)	tok/s	VRAM (GiB)
BF16	195.3	15.24	138.6	15.45	76.3	15.88
TurboLDM-(3+1) (<i>deq buffer</i>)	164.3	17.90	122.0	18.11	70.7	18.53
NF4	65.4	5.72	58.3	5.83	43.6	6.23

Table 42: Cached-dequant versus chunked low-bit forward on LLaDA-8B-Instruct (MMLU, $n=75$). Implemented by UniformLinearChunked in codes/TurboLDM_experiments_helper.py, the chunked proxy reconstructs each BF16 weight in 512-row slices during the forward pass and removes the persistent per-layer BF16 cache. It cuts peak VRAM from 15.45 to 9.05 GiB but slows the forward pass and is explicitly not a fused CUDA kernel.

Method	Mode	Storage GB	Peak VRAM GB	MMLU	Wall (s)
AWQ-4	cached	3.45	15.45	36.0	21.5
AWQ-4	chunked	3.45	9.05	8.0	65.3
W4	cached	3.45	15.45	42.7	21.1
W4	chunked	3.45	9.05	40.0	65.7

Table 43: NF4 (bnb.Linear4bit, weight-only) across model families. The LLaDA-8B scores were reproduced exactly with seeds 17/42/101 under deterministic decoding; Dream-7B and the other transfer checks are single-seed unless stated otherwise. Peak inference VRAM is the runtime memory of the live forward pass at gen_length=128, T=64, batch 1. NF4 is integrated with a low-bit kernel in this tested runtime; this does not imply an inherent advantage over optimized implementations of other formats. LLaDA-MoE was evaluated at n=300 to keep runtime bounded; all other rows are n=500. Encoder rows (BERT/RoBERTa/MDLM-OWT) report masked-token NLL on a 300-paragraph WikiText-103 slice at 0.15 masking ratio; lower is better. A “—” in the Peak column means that a comparable end-to-end peak-VRAM measurement was not collected for that transfer run. The generative MMLU/ARC-C/HellaSwag/GSM8K columns are not applicable to the encoder rows and are therefore replaced by their masked-token NLL and perplexity.

Model	MMLU	ARC-C	HellaSwag	GSM8K	Peak (GiB)
<i>Decoders, generative MCQ exact-match accuracy / GSM8K final-number match</i>					
LLaDA-8B-Instruct (NF4)	60.2	85.4	68.8	69.0	6.07
Dream-7B-Instruct (NF4)	59.8	76.0	48.6	13.0	—
LLaDA-MoE-7B (NF4)	51.7	83.0	59.7	64.7	5.42
<i>Encoders, masked-NLL on WikiText-103 (lower is better; PPL in parens)</i>					
BERT-base (NF4)	NLL 2.40	PPL 11.0			—
RoBERTa-base (NF4)	NLL 1.90	PPL 6.7			—
MDLM-OWT (NF4)	NLL 2.28	PPL 9.8			—

Table 44: GSM8K seed variation on LLaDA-8B-Instruct (n=200, low-confidence remasking). Quantized rows use seeds {17, 42, 101}; BF16 is the seed-42 reference. In TurboLDM (3+1), 3 is the base-bit code and +1 is a 1-bit residual code; the row is compact but not quality-equivalent.

Method	Avg. bits	Seeds	Mean	Std	Min-Max
BF16	16.00	1	68.00	0.00	68.0
Uniform-int4	4.25	3	67.00	0.00	67.0–67.0
GPTQ-4	4.25	3	58.83	2.02	57.0–61.0
AWQ-4	4.25	3	68.00	2.29	65.5–70.0
TurboLDM (4+1)	4.60	3	63.83	2.57	61.0–66.0
TurboLDM (3+1)	3.60	3	59.33	1.89	58.0–61.5

Table 45: Three-seed variance on LLaDA-8B-Instruct (seeds 11, 22, 33). Reported as mean $\pm$ std over seeds.

Method	MMLU	GSM8K
FP16/BF16	38.0 $\pm$ 0.0	62.5 $\pm$ 0.0
AWQ-4	44.0 $\pm$ 1.7	62.5 $\pm$ 7.5
TurboLDM (3+1)	36.3 $\pm$ 1.5	55.0 $\pm$ 6.6

Table 46: Cross-family transfer to Dream-7B-Instruct (Ye et al., 2025) on a matched $n=500$ grid. Evaluation uses Dream’s native `diffusion_generate` decoder. TLDM abbreviates TurboLDM; $(b+1)$ means b base bits plus a 1-bit residual code. Dream is more sensitive to PTQ, especially for TLDM- $(3+1)$ and GSM8K.

Method	Avg. bits	Storage GB	MMLU	ARC-C	H.Swag	GSM8K
BF16	16.00	14.19	59.2	77.4	52.6	17.4
NF4	6.38	5.18	59.8	76.0	48.6	13.0
AWQ-4	4.25	3.23	52.0	73.6	52.8	8.4
TLDM (3+1)	3.56	2.71	43.6	58.4	38.0	5.8

Table 47: Dream-7B per-family isolated 3-bit quantisation, counterpart of the LLaDA-8B map (Appendix D). Each row quantises only one family of linears at base 3 bits, residual 0, with the rest of the model held in BF16. Compound entries `ATTN_ALL` and `MLP_ALL` apply the same 3-bit quantiser to every linear in the named block. Single-seed, $n=200$ MCQ / $n=100$ GSM8K. The compound-attention drop reproduces the LLaDA-8B finding that individual attention projections are cheap to quantise but compound-quantising attention is much more expensive.

Family quantised	MMLU	ARC-C	HellaSwag	GSM8K
Q_PROJ	0.420	0.735	0.430	0.130
K_PROJ	0.410	0.750	0.525	0.200
V_PROJ	0.440	0.770	0.510	0.130
O_PROJ	0.415	0.735	0.485	0.190
ATTN_ALL	0.360	0.610	0.345	0.060
GATE_PROJ	0.400	0.760	0.515	0.160
UP_PROJ	0.400	0.750	0.530	0.140
DOWN_PROJ	0.390	0.795	0.475	0.160
MLP_ALL	0.380	0.695	0.460	0.060

Table 48: Cross-family transfer to MDLM-OWT (170M, Kuleshov group). Masked-token NLL / perplexity on a 128-paragraph WikiText-103 validation slice at 15% masking ratio. Lower is better.

Method	Avg. bits	Storage GB	NLL	PPL
FP32 baseline	32.00	0.63	2.236	9.35
W4 (uniform)	4.29	0.05	2.492	12.09
AWQ-4	4.29	0.05	2.248	9.47
TurboLDM (4+1)	4.48	0.05	2.310	10.07
TurboLDM (3+1)	3.45	0.04	2.599	13.45
<i>Module-family sensitivity (3-bit TurboQuant on only that family):</i>				
MLP up (mlp.0)	3.29	0.01	2.499	12.17
MLP down (mlp.2)	3.37	0.01	2.248	9.47
attn QKV (packed)	3.46	0.01	2.278	9.75
attn output	3.46	0.00	2.059	7.84

Table 49: Encoder-only masked-LM transfer: BERT-base, RoBERTa-base, DeBERTa-v3-base. Masked-token NLL at 15% masking on a WikiText-103 validation slice (lower is better).

Model	Method	Avg. bits	Storage GB	NLL	PPL
bert-base	FP16/BF16	32.00	0.41	2.289	9.86
	W4 (uniform)	4.28	0.04	2.360	10.59
	AWQ-4	4.28	0.04	2.258	9.57
	TurboLDM (4+1)	4.65	0.05	2.371	10.71
	TurboLDM (3+1)	3.66	0.04	2.655	14.22
roberta-base	FP16/BF16	32.00	0.46	1.555	4.73
	W4 (uniform)	4.28	0.04	2.518	12.40
	AWQ-4	4.28	0.04	2.021	7.55
	TurboLDM (4+1)	4.43	0.04	2.441	11.49
	TurboLDM (3+1)	3.45	0.03	5.138	170.29

Table 50: Encoder-only masked-LM transfer: BERT-base, RoBERTa-base, and DeBERTa-v3-base on a 512-paragraph WikiText-103 slice. Masked-token NLL at 15% masking (lower is better). Δ NLL is the increase over the FP32 baseline.

Model	Method	Avg bits	NLL	Δ NLL	PPL
bert-base	bf16	32.00	2.349	+0.000	10.47
bert-base	uniform-int4	4.28	2.431	+0.082	11.37
bert-base	awq-4	4.28	2.469	+0.121	11.82
bert-base	turboldm-4p1	4.62	2.341	-0.007	10.40
bert-base	turboldm-3p1	3.68	2.828	+0.480	16.92
roberta-base	bf16	32.00	1.695	+0.000	5.45
roberta-base	uniform-int4	4.28	2.631	+0.935	13.88
roberta-base	awq-4	4.28	2.126	+0.430	8.38
roberta-base	turboldm-4p1	4.45	2.487	+0.792	12.03
roberta-base	turboldm-3p1	3.47	5.025	+3.330	152.24

Table 51: DeBERTa-v3-base masked-LM transfer. Masked-token NLL at 15% masking on the same 512-paragraph WikiText-103 slice as the BERT/RoBERTa encoder checks; lower is better; Δ NLL is increase over the FP32 baseline.

Method	Avg bits	NLL	Δ NLL	PPL
FP32 baseline	32.00	15.862	+0.000	7738557.00
W4 (uniform)	4.28	16.486	+0.624	14440152.00
AWQ-4	4.28	16.171	+0.310	10546421.00
TurboLDM (4+1)	4.73	15.747	-0.115	6897126.50
TurboLDM (3+1)	3.73	15.670	-0.191	6391318.50

Table 52: DeBERTa v1 / v2 masked-LM transfer on non-RTD checkpoints. Masked-token NLL at 15% masking on 512-paragraph WikiText-103. Lower is better; Δ NLL is increase over the FP32 baseline.

Model	Method	Avg bits	NLL	PPL
deberta-v1-base	FP32 baseline	32.00	11.558	104617.43
deberta-v1-base	W4 (uniform)	4.27	11.520	100673.71
deberta-v1-base	AWQ-4	4.27	11.550	103729.56
deberta-v1-base	TurboLDM (4+1)	4.61	11.608	109943.86
deberta-v1-base	TurboLDM (3+1)	3.58	11.568	105631.19
deberta-v2-xlarge	FP32 baseline	32.00	12.935	414506.53
deberta-v2-xlarge	W4 (uniform)	4.27	12.997	441119.56
deberta-v2-xlarge	AWQ-4	4.27	13.011	447290.88
deberta-v2-xlarge	TurboLDM (4+1)	4.74	13.073	475827.25
deberta-v2-xlarge	TurboLDM (3+1)	3.76	13.127	502544.03

Table 53: Scaling study: LLaDA-MoE-7B-A1B-Instruct (7B, 1B-active MoE) and LLaDA2.0-mini (16B, A1B MoE). TurboLDM-(3+1) means 3 base bits plus a 1-bit residual code and trades substantial quality for storage. TLDM-lazy reconstructs compact TurboLDM weights on the fly rather than caching a BF16 copy. The 16B rows test memory feasibility, not quality parity; without matched AWQ/NF4 baselines, they do not isolate the cause of the quality collapse.

Model	Method	Avg. bits	Storage GB	Peak mem. GB	MMLU
LLaDA-MoE-7B	FP16/BF16	16.00	13.70	14.4	50.0
LLaDA-MoE-7B	NF4	4.00	4.33	5.4	51.7
LLaDA-MoE-7B	AWQ-4	4.25	3.32	14.6	51.7
LLaDA-MoE-7B	TLDM (3+1)	3.45	2.70	16.6	45.3
LLaDA2.0-mini (16B)	FP16/BF16	16.00	30.28	26.2	64.0
LLaDA2.0-mini (16B)	TLDM-lazy (4-bit)	4.25	7.72	17.1	22.7
LLaDA2.0-mini (16B)	TLDM-lazy (3-bit)	3.25	5.91	17.1	17.3

Table 54: LLaDA2.0-mini (16B) lazy-quantisation diagnostic. Lazy-4 MLP only applies lazy 4-bit TurboLDM reconstruction to MLP layers; Lazy-(4+2) full model uses 4 base bits plus a 2-bit residual code across the full model. Each row is MMLU $n=75$ and the BF16 baseline is 64.0. These rows establish memory feasibility only. Because matched AWQ/NF4 baselines are absent, the quality loss cannot be attributed specifically to lazy reconstruction.

Config	Avg bits	Storage GB	Peak mem GB	MMLU
Lazy-4 MLP only	4.25	7.62	17.2	29.3
Lazy-(4+2) full model	4.92	8.94	21.5	28.0

Table 55: Adaptive denoising-step controller (start at $T=32$, escalate to 64 then 128 when the answer is not parseable or its margin is too low). NFE is the cumulative number of denoiser evaluations across the slice. On LLaDA-8B BF16 the adaptive policy reaches fixed- $T=64$ accuracy on GSM8K at roughly half the NFE; on NF4 the policy collapses (fixed- $T=128$ wins outright); on Dream-7B under TurboLDM-(3+1), meaning 3 base bits plus a 1-bit residual code, the policy matches fixed- $T=32$ but cannot recover the BF16 score because the underlying quantised Dream is already saturated. MMLU is policy-invariant on every (model, method) pair we tested, consistent with the observation that the answer letter settles within the first few denoising steps.

Model	Method	Policy	MMLU ($n=100$)		GSM8K ($n=50$)	
			Acc	NFE	Acc	NFE
LLaDA-8B	BF16	fixed $T=32$	0.38	3200	0.58	1600
		fixed $T=64$	0.38	6400	0.64	3200
		fixed $T=128$	0.38	12800	0.64	6400
		adaptive	0.38	3200	0.62	1728
LLaDA-8B	TurboLDM-(3+1)	fixed $T=32$	0.35	3200	0.38	1600
		fixed $T=64$	0.35	6400	0.52	3200
		fixed $T=128$	0.35	12800	0.62	6400
		adaptive	0.35	3200	0.40	1856
LLaDA-8B	NF4	fixed $T=32$	0.37	3200	0.52	1600
		fixed $T=64$	0.37	6400	0.58	3200
		fixed $T=128$	0.37	12800	0.68	6400
		adaptive	0.37	3200	0.52	1600
Dream-7B	TurboLDM-(3+1)	fixed $T=32$	0.31	3200	0.04	1600
		fixed $T=64$	0.22	6400	0.08	3200
		fixed $T=128$	0.30	12800	0.04	6400
		adaptive	0.31	3200	0.04	2368

Table 56: Per-(model, family) weight fingerprint and its correlation with isolated-family quantisation loss. Top block: mean of the 99.9th-percentile-to-RMS ratio of $|w|$ within each family, averaged across all architecture-native layers matched to that family. A dash means that no separately named family matches the column, not that the statistic is zero: LLaDA-8B uses ff_out rather than down_proj, while LLaDA2.0-mini packs Q/K/V into query_key_value, so separate q_proj and v_proj entries are not applicable. Bottom block: Spearman ρ between the per-family weight stat and the BF16-relative accuracy drop when only that family is quantised at 3-bit (uses the sensitivity numbers from Appendix D for LLaDA-8B and Dream-7B). The $p_{99.9}$ /RMS ratio predicts family fragility on competence-saturated tasks ($\rho=+0.77$ on LLaDA-8B/MMLU, $+0.53$ on Dream-7B/ARC-C); max/RMS alone (i.e. a single outlier) does not. Lower ρ on GSM8K reflects that the underlying BF16 reference is itself noisy on that task for Dream-7B.

Model	Mean $p_{99.9}$ /RMS per family			
	q_proj	v_proj	up_proj	down_proj
LLaDA-8B-Instruct	3.88	3.49	3.41	—
Dream-7B-Instruct	4.24	3.98	3.65	3.67
LLaDA-MoE-7B	4.57	3.68	3.54	3.51
LLaDA2.0-mini (16 B)	—	—	3.44	3.44

Stat	Spearman ρ vs. family acc. drop			
	LLaDA-8B / MMLU	Dream / MMLU	Dream / ARC-C	Dream / GSM8K
max / RMS	-0.03	-0.30	-0.29	-0.20
$p_{99.9}$ /RMS	+0.77	+0.53	+0.53	-0.27
mean $ w $ /RMS	-0.77	-0.53	-0.53	+0.27
frac. $ w > 5$ RMS	+0.66	+0.53	+0.53	-0.44
stable rank	-0.43	-0.36	-0.25	+0.35