TTQ: ACTIVATION-AWARE TEST-TIME QUANTIZATION TO ACCELERATE LLM INFERENCE ON THE FLY

Toshiaki Koike-Akino, Jing Liu, Ye Wang
Mitsubishi Electric Research Laboratories (MERL)
201 Broadway, Cambridge, MA 02139, USA

ABSTRACT

To tackle the huge computational demand of large foundation models, activation-aware compression techniques without retraining have been introduced. However, since these methods highly rely on calibration data, domain shift issues may arise for unseen downstream tasks. We propose a test-time quantization (TTQ) framework which compresses large models on the fly at inference time to resolve this issue. With an efficient online calibration, instant activation-aware quantization can adapt every prompt regardless of the downstream tasks, yet achieving inference speedup. Several experiments demonstrate that TTQ can improve the quantization performance over state-of-the-art baselines.

1 INTRODUCTION

Large foundation models Touvron et al. (2023); Achiam et al. (2023); Liu et al. (2023b) have shown excellent performance across a variety of tasks (Wei et al., 2022; Katz et al., 2024; Bubeck et al., 2023). Nonetheless, these models, with billions of parameters, demand significant computational resources (Schwartz et al., 2020). Towards increasing the accessibility of large language models (LLMs), a number of compression methods Xu & McAuley (2023); Zhu et al. (2024); Bai et al. (2024a) have been introduced: e.g., partial activation (Jiang et al., 2024; Lin et al., 2024a), weight pruning (Frantar & Alistarh, 2023b; Sun et al., 2023; Bai et al., 2024b; Hassibi et al., 1993), quantization (Frantar et al., 2022; Lin et al., 2024b; Wang et al., 2024), knowledge distillation (Hsieh et al., 2023; Hwang et al., 2024), and rank reduction (Yuan et al., 2023; Liu et al., 2024; Hwang et al., 2024; Saxena et al., 2024). Related literature is further discussed in Appendix A.

Test-time scaling (Chen et al., 2024; Muennighoff et al., 2025) is a new paradigm to improve LLM performance by increasing inference computation. Instead of increasing the complexity, *test-time compression* is aimed at reducing the total cost of inference on the fly. For instance, test-time pruning has been used as a mixture-of-expert (MoE) framework, which dynamically selects important modules depending on each prompt. Recently, micro-grained MoE (Koike-Akino et al., 2025b) exploits a fast activation-aware weight pruning method at test-time to accelerate LLMs. However, unstructured pruning generally does not improve the hardware efficiency unlike quantization or rank reduction. We hence propose a new **test-time quantization (TTQ)** framework, enabled by a fast activation-aware quantization method. We make the following contributions: (1) Our TTQ accelerates LLMs at inference time, (2) introducing low-complexity activation-aware quantization to compress LLMs on the fly, with negligible overhead, and (3) low-rank decomposition integrated into TTQ, (4) while avoiding domain shift inherent to offline calibration of baseline static quantization. (5) We demonstrate the benefit of TTQ over state-of-the-art methods for several LLM benchmarks.

2 TEST-TIME QUANTIZATION: TTQ

Groupwise Quantization Round-to-nearest (RTN) is the simplest quantization method to minimize approximation error: $\mathcal{L}_0 = \|W - \hat{W}\|^2$, where $W \in \mathbb{R}^{d' \times d}$ is a weight matrix and $\hat{W}$ is its quantized version. RTN uses the groupwise quantization-dequantization (QDQ) operation: $\hat{W} = \mathcal{Q}[W] \triangleq \mathcal{G}^-[\mathcal{G}[W]]$. The quantization $\mathcal{G}[\cdot]$ and dequantization $\mathcal{G}^-[\cdot]$ are defined as:

$$W_{\rm int} = \mathcal{G}(W) \triangleq \mathsf{round}[\mathsf{clamp}_q[(W - Z) \oslash S]], \qquad \hat{W} = \mathcal{G}^-(W_{\rm int}) \triangleq W_{\rm int} \circ S + Z, \qquad (1)$$

where $S \in \mathbb{R}^{d' \times d}$ and $Z \in \mathbb{R}^{d' \times d}$ are scale and zero-point parameters. Here, $\oslash$ is element-wise division, $\circ$ is element-wise product, $\mathsf{round}[x]$ gives the closest integer to x, and $\mathsf{clamp}_q[x] = \min(\max(x, 0), 2^q - 1)$ is q-bit limiting operator. See Appendix B for more details. Pseudo-code of RTN with a groupsize g is as follows:

```
def rtn(W, q, g): # q: bits, g: groupsize
    ddash, d = W.shape # W: (d', d)
    W = W.reshape(-1, g) # grouping (d'*d/g, g)
    Wmax, Wmin = W.amax(axis=1), W.amin(axis=1) # (d'*d/g,)
    S, Z = (Wmax - Wmin) / (2**q - 1), Wmin # scale, zero-point
    Wint = ((W - Z[:, None]) / S[:, None]).round().clamp(0, 2**q - 1)
    What = Wint * S[:, None] + Z[:, None] # dequantization
    return What.reshape(ddash, d) # reshaping back
```

The recent NVFP format (Egiazarian et al., 2025) uses a microscaling groupsize of 16 to accelerate GPU computing. Most literature claims 2 to 4-fold speedup with 4-bit LLMs Frantar et al. (2025).

AWQ: Activation-Aware Quantization To improve over naïve RTN quantization, the activation-aware framework (Lin et al., 2023; Frantar et al., 2022; Liu et al., 2025) leverages activation statistics. Let $X \in \mathbb{R}^{d \times T}$ be an input activation with embedding dimension d and token length T. The aim is to minimize the approximation loss:

$$\mathcal{L} \triangleq \mathbb{E}_X\left[\left\|(W - \hat{W})X\right\|^2\right] = \mathrm{tr}[(W - \hat{W})\mathbb{E}_X[XX^\top](W - \hat{W})^\top] = \|(W - \hat{W})C^{1/2}\|^2, \quad (2)$$

where $C \triangleq \mathbb{E}_X[XX^\top] \in \mathbb{R}^{d \times d}$ is the auto-correlation statistics of input X. Since C is not exactly known at test time, we estimate it with small amount of calibration data, e.g., via shrunk estimator: $C_\lambda = (1 - \lambda)XX^\top + \lambda\eta I$, where λ is a shrinkage parameter (Ledoit & Wolf, 2004), and $\eta = \|X\|^2/d$. GPTQ (Frantar et al., 2022) uses the greedy method inspired by optimal brain surgeon (Hassibi et al., 1993). It requires the Cholesky factorization, whose complexity is at least of cubic order: $\mathcal{O}[d^3 + dd'T]$. Whereas, AWQ (Lin et al., 2023) greatly simplifies the problem by approximating with a diagonal correlation: $C_\lambda \simeq D \triangleq \mathsf{diag}[XX^\top + \lambda I]^\alpha$, where $\mathsf{diag}[C] \triangleq C \circ I$ offers a diagonal matrix, and α is an auxiliary parameter. Note that the above expression gives the ℓ_2-norm diagonal $D_{i,i} = (\|X_{i,:}\|_2^2 + \lambda)^\alpha$, while the original AWQ uses ℓ_1-norm $D_{i,i} = (\|X_{i,:}\|_1^2 + \lambda)^\alpha$. Given a diagonal correlation matrix, the closed-form solution for (2) is given by the scaled QDQ operation: $\hat{W} = \mathcal{Q}[WD^{1/2}]D^{-1/2}$. The pseudo-code of the AWQ concept is given as follows:

```
def awq(X, W, q, g, p, lam, alpha): # q: bits, g: groupsize, p: lp-norm
    D = (X.norm(p=p, axis=1) + lam) ** alpha # X: (d, T); D: (d,)
    What = rtn(W * D[None, :], q, g) # scaled QDQ
    return What * D.reciprocal()[None, :] # scaling back
```

Here, we generalized to any ℓ_p-norm with arbitrary p. See Appendix C for more details.

TTQ: Test-Time Quantization with Online AWQ Activation-aware post-training zero-short quantization like GPTQ/AWQ has several drawbacks: (1) calibration and extra data are required before deployment; (2) a severe domain shift issue may arise at test time when the calibration data were irrelevant; and (3) the original full-precision weights are not recoverable for re-calibrating new domain once the quantized model is deployed. To solve these issues, we propose a test-time quantization (TTQ) framework shown in Fig. 1, which employs light-weight online AWQ at inference time. Specifically, we dynamically calculate the diagonal correlation D for incoming tokens X on the fly to adapt scale S and zero-point Z. It thus requires **zero** offline calibration data before deployment. Once the weight is quantized, `int_matmul` kernels [1] can significantly accelerate the linear projection at GPU devices because of reduced caching overhead.

To keep AWQ fast enough at test time, we keep constant hyperparameters (α, λ, p) without exhaustively searching the best values. Appendix F discusses the hyperparameter selection. Importantly, the extra computation for online quantization is negligible: specifically, a fraction of overhead complexity over the original un-quantized projection is:

$$\rho = \frac{\mathcal{O}[dT + 3d'd]}{\mathcal{O}[d'dT]} = \mathcal{O}\left[\frac{1}{d'} + \frac{3}{T}\right] \underset{d', T \gg 1}{\longrightarrow} 0. \quad (3)$$

[1] e.g., `awq_gemm` CUDA kernel in `https://github.com/vllm-project/vllm`

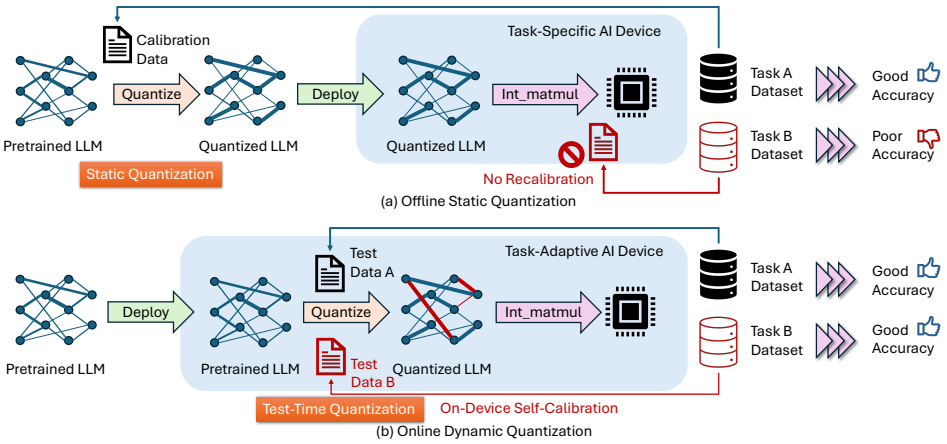


Figure 1: (a) Offline static quantization (e.g., AWQ/GPTQ) requires calibration data, incurs domain shift risk, and cannot be recalibrated after deployment. (b) Our TTQ is online dynamic quantization, with zero offline calibration, and capable of on-device self-calibration at inference time.

Table 1: Calibration length impact at 3-bit quantization with $g = 32$ for OPT-350M model.

	TTQ		AWQ (C4 Calib)						
Calib Tokens T	$\mathbf{0}_{(r=0)}$	$\mathbf{0}_{(r=16)}$	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
WT2 Perplexity (↓)	<u>24.93</u>	**24.17**	25.73	25.56	25.62	25.55	25.42	25.07	25.02

Here, the original projection for WX requires $\mathcal{O}[d'dT]$, the norm calculation for $D_{ii} = (\|X_{i,:}\|^2 + \lambda)^{\alpha}$ requires $\mathcal{O}[dT]$, scaling as $WD^{1/2}$ requires $\mathcal{O}[d'd]$, quantization $\mathcal{G}[\cdot]$ in (1) needs $\mathcal{O}[d'd]$, and re-scaling back $\mathcal{Q}[WD^{1/2}]D^{-1/2}$ uses another $\mathcal{O}[d'd]$. Hence, with sufficiently large output dimension $d' \gg 1$ and token length $T \gg 1$, online AWQ introduces negligible extra-complexity.

TTQ with Low-Rank Decomposition Nonetheless, extreme bit quantization often causes a severe degradation, and QLoRA (Dettmers et al., 2023) could compensate for quantized LLM performance by adapting low-rank factors. It motivates us to integrate TTQ with low-rank decomposition. We now have low-rank factors with quantized weights: $\hat{W} = W_{\text{q}} + BA$, where $W_{\text{q}} \in \mathbb{R}^{d' \times d}$ is quantized residual weights from un-quantized low-rank factors $B \in \mathbb{R}^{d' \times r}$ and $A \in \mathbb{R}^{r \times d}$ with a rank r. When $r \ll \min(d, d')$, it can accelerate the inference as the complexity for low-rank projection is reduced from $\mathcal{O}[d'dT]$ for WX to $\mathcal{O}[r(d'+d)T]$ for BAX — hence the relative extra-complexity is negligible: $\mathcal{O}[r/d + r/d'] \to 0$. A key difference from QLoRA is that TTQ **dynamically** adapts W_{q} depending on X, whereas QLoRA uses a **static** W_{q} but adapting only B and A. We may initialize low-rank factors B and A by top-r principal components, and W_{q} is obtained on the fly to quantize the residual weights through scaled QDQ: $W_{\text{q}} = \mathcal{Q}[(W - BA)D^{1/2}]D^{-1/2}$. Appendix E discusses some alternative initializations. Optionally, we may dynamically adapt B and A, e.g., through online PCA (Feng et al., 2013). Nevertheless, our primary objective is to accelerate LLMs at the inference time through the use of dynamic weight quantization, and thus we focus on static low-rank factors. Appendix H shows up-to **5-fold** speedup even with an overhead of low-rank projection.

Experiments We evaluate the OPT (Zhang et al., 2022), Qwen3 (Yang et al., 2025), and Gemma3 (Gemma Team et al., 2025) LLM models on the witkitext-2 (WT2) (Merity et al., 2016), PTB (Marcus et al., 1994), and C4 (Raffel et al., 2020) benchmarks. See details of experimental setup in Appendix G. We use a groupsize of $g = 32$ for all experiments unless specified. We first show the benefit of our TTQ, which needs **zero** calibration data, whereas AWQ is sensitive to the size of calibration data, as shown in Table 1. We show the perplexity score for WT2 on OPT-350M, with C4 calibration for AWQ. TTQ achieves the best over AWQ baselines, where AWQ degrades the performance when fewer calibration tokens are available. Table 2 shows the impact of group-size g for 3-bit quantized Qwen3-1.7B model. As expected, micro-scaling ($g < 32$) achieves good

Table 2: Groupsize impact on WT2 perplexity at 3-bit quantization for Qwen3-1.7B model.

Groupsize g	8	16	32	64	128	256	512	1024
RTN	33.39	66.61	120.89	114.97	1450.28	15503.10	949478.44	57443.80
AWQ (WT2 Calib)	17.64	19.00	20.35	20.69	24.96	31.41	36.14	68.96
TTQ ($r = 16$)	**17.54**	**17.81**	**18.29**	**19.46**	**22.33**	**24.81**	**31.43**	**48.70**

Table 3: Perplexity (↓) of OPT/Qwen3/Gemma3 models with different quantization methods. It shows macro average across WT2/PTB/C4 datasets. Groupsize is $g = 32$ for all cases. Calibration token length is $T = 2^{17}$ for AWQ. **Bold** and underline denote the best and second best, respectively. Asterisk "*" indicates reaching competitive performance to the original un-compressed LLM.

Quantization q	2 bits	3 bits	4 bits	5 bits
OPT-125M (WT2: 27.7, PTB: 39.0, C4: 26.6, Avg: **31.1**)				
RTN	5058.5	56.3	33.5	31.8
AWQ (WT2 Calib)	381.7	37.4	32.3	31.4
AWQ (PTB Calib)	375.3	37.3	32.2	31.3
AWQ (C4 Calib)	451.7	37.7	32.6	31.3
TTQ ($r = 0$)	257.4	36.6	31.9	31.2
TTQ ($r = 16$)	**141.7**	**35.8**	**31.8**	***31.1**
OPT-1.3B (WT2: 14.6, PTB: 20.3, C4: 16.1, Avg: **17.0**)				
RTN	11514.4	27.2	18.1	17.2
AWQ (WT2 Calib)	32.4	18.0	17.3	***17.0**
AWQ (PTB Calib)	32.6	18.1	17.3	***17.0**
AWQ (C4 Calib)	31.7	18.0	17.2	***17.0**
TTQ ($r = 0$)	32.2	18.2	17.2	***17.0**
TTQ ($r = 16$)	**27.2**	**17.9**	**17.1**	***17.0**
OPT-2.7B (WT2: 12.5, PTB: 18.0, C4: 14.3, Avg: **14.9**)				
RTN	6274.5	36.0	15.7	15.0
AWQ (WT2 Calib)	23.1	15.7	15.1	15.0
AWQ (PTB Calib)	23.2	15.7	**15.0**	15.0
AWQ (C4 Calib)	22.9	15.7	**15.0**	15.0
TTQ ($r = 0$)	23.7	15.7	**15.0**	***14.9**
TTQ ($r = 16$)	**21.2**	**15.5**	**15.0**	***14.9**
OPT-6.7B (WT2: 10.9, PTB: 15.8, C4: 12.7, Avg: **13.1**)				
RTN	5716.5	26.2	13.7	13.2
AWQ (WT2 Calib)	17.2	13.5	13.2	***13.1**
AWQ (PTB Calib)	17.1	13.6	13.2	***13.1**
AWQ (C4 Calib)	16.9	13.6	13.2	***13.1**
TTQ ($r = 0$)	17.2	13.6	13.2	***13.1**
TTQ ($r = 16$)	**16.3**	**13.4**	***13.1**	***13.1**

Quantization q	2 bits	3 bits	4 bits	5 bits
Qwen3-0.6B (WT2: 21.0, PTB: 43.8, C4: 30.3, Avg: **31.7**)				
RTN	8.2e6	127.3	38.2	33.5
AWQ (WT2 Calib)	9739.1	49.4	33.5	32.4
AWQ (PTB Calib)	17344.3	47.9	34.1	32.0
AWQ (C4 Calib)	5388.1	48.3	33.0	32.1
TTQ ($r = 0$)	2827.8	44.7	33.0	**31.9**
TTQ ($r = 16$)	**1552.6**	**42.0**	**32.9**	**31.9**
Qwen3-1.7B (WT2: 16.7, PTB: 33.8, C4: 16.1, Avg: **24.2**)				
RTN	1.4e6	162.8	30.6	26.1
AWQ (WT2 Calib)	1864.7	30.0	24.5	24.4
AWQ (PTB Calib)	2309.6	29.8	24.7	24.5
AWQ (C4 Calib)	2364.7	28.2	24.9	24.4
TTQ ($r = 0$)	522.6	27.3	24.4	24.3
TTQ ($r = 16$)	**264.6**	**26.4**	**24.3**	***24.1**
Gemma3-270M (WT2: 70.1, PTB: 698.7, C4: 68.1, Avg: **279.0**)				
RTN	2.6e11	1795.0	391.9	315.0
AWQ (WT2 Calib)	89188.4	517.4	279.6	306.0
AWQ (PTB Calib)	1.9e5	537.7	310.0	293.4
AWQ (C4 Calib)	1.3e5	598.1	326.4	*276.8
TTQ ($r = 0$)	30199.0	387.7	*277.0	*262.5
TTQ ($r = 16$)	**19657.0**	**382.8**	***275.7**	***261.5**
Gemma3-1B (WT2: 27.7, PTB: 212.4, C4: 33.2, Avg: **91.1**)				
RTN	8.6e5	209.4	111.1	96.6
AWQ (WT2 Calib)	4734.7	134.7	91.9	95.9
AWQ (PTB Calib)	9326.6	138.9	99.2	95.5
AWQ (C4 Calib)	5486.9	150.8	93.5	93.6
TTQ ($r = 0$)	1928.5	127.3	***89.9**	***90.2**
TTQ ($r = 16$)	**1804.9**	**114.5**	91.7	*90.3

performance, while micro-scaling alone for RTN cannot compete with AWQ/TTQ. TTQ can tolerate roughly twice larger groupsizes than AWQ, that can halve the required memory for scale S and zero-point Z. Next we show the performance across different models in Table 3, where we show the macro average of perplexity over WT2, PTB, and C4 benchmarks. We observe that few-bit quantization like $q = 2$ is more feasible for larger LLMs, and 5-bit quantization achieves nearly un-quantized performance for most cases. TTQ achieves the best over all cases, even though AWQ used a long calibration of $T = 2^{17}$. More importantly, AWQ shows fluctuation across different calibration datasets, indicating the potential risk of such **static** quantization relying heavily on calibration. We notice that the original un-compressed Gemma3 is significantly poorer for the PTB dataset, and the macro average perplexity can be slightly better with compression occasionally. See Appendices I, J and K for full results in detail including other VLM and VLA benchmarks.

3 Conclusion

We proposed a new TTQ framework, which employs a light-weight activation-aware quantization at inference time to accelerate LLMs, without requiring offline calibration or fine-tuning. We demonstrated a consistent advantage over state-of-the-art baselines on several LLM models. Further benchmark evaluations will be reported in the future, and we plan to integrate test-time pruning and decomposition into TTQ. How to dynamically adjust hyperparameters is interesting to explore.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Zeyuan Allen-Zhu and Yuanzhi Li. First efficient convergence for streaming k-PCA: a global, gap-free, and near-optimal rate. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 487–492. IEEE, 2017.

Yamato Arai and Yuma Ichikawa. Quantization error propagation: Revisiting layer-wise post-training quantization. *arXiv preprint arXiv:2504.09629*, 2025.

Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*, 2024a.

Guangji Bai, Yijiang Li, Chen Ling, Kibaek Kim, and Liang Zhao. SparseLLM: Towards global pruning for pre-trained language models. *arXiv preprint arXiv:2402.17946*, 2024b.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *European Conference on Computer Vision*, pp. 707–720. Springer, 2002.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. QuIP: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36:4396–4429, 2023.

Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 295–305, 2022.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. HAWQ: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 293–302, 2019.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. HAWQ-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*, 2024.

Vage Egiazarian, Roberto L Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, et al. Bridging the gap between promise and performance for microscaling FP4 quantization. *arXiv preprint arXiv:2509.23202*, 2025.

Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. DeepShift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2359–2368, 2021.

Jiashi Feng, Huan Xu, and Shuicheng Yan. Online robust PCA via stochastic optimization. *Advances in neural information processing systems*, 26, 2013.

Elias Frantar and Dan Alistarh. QMoE: Practical sub-1-bit compression of trillion-parameter models. *arXiv preprint arXiv:2310.16795*, 2023a.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023b.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. Marlin: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pp. 239–251, 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Peter M Hall, A David Marshall, and Ralph R Martin. Incremental eigenanalysis for classification. In *BMVC*, volume 98, pp. 286–295, 1998.

Babak Hassibi, David Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. *Advances in neural information processing systems*, 6, 1993.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

Injoon Hwang, Haewon Park, Youngwan Lee, Jooyoung Yang, and SunJae Maeng. PC-LoRA: Low-rank adaptation for progressive model compression with knowledge distillation. *arXiv preprint arXiv:2406.09117*, 2024.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. GPT-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, 382(2270):20230254, 2024.

Minje Kim and Paris Smaragdis. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016.

Toshiaki Koike-Akino, Chang Meng, Volkan Cevher, and Giovanni De Micheli. Hardware-efficient quantization for green custom foundation models. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*, 2024.

Toshiaki Koike-Akino, Xiangyu Chen, Jing Liu, Ye Wang, Matthew Brand, et al. LatentLLM: Attention-aware joint tensor compression. *arXiv preprint arXiv:2505.18413*, 2025a.

Toshiaki Koike-Akino, Jing Liu, and Ye Wang. μ-moe: Test-time pruning as micro-grained mixture-of-experts. *arXiv preprint arXiv:2505.18451*, 2025b.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.

Bin Lin, Zhenyu Tang, Yang Ye, Jiaxi Cui, Bin Zhu, Peng Jin, Junwu Zhang, Munan Ning, and Li Yuan. MoE-LlaVa: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

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 on-device LLM compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024b.

Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. *Advances in neural information processing systems*, 33:7474–7485, 2020.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *Advances in neural information processing systems*, 30, 2017.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023b.

Jing Liu, Toshiaki Koike-Akino, Ye Wang, Hassan Mansour, and Matthew Brand. AWP: Activation-aware weight pruning and quantization with projected gradient descent. *arXiv preprint arXiv:2506.10205*, 2025.

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-Real Net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 722–737, 2018.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. OK-VQA: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pp. 3195–3204, 2019.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Augustus Odena, Dieterich Lawson, and Christopher Olah. Changing model behavior at test-time using reinforcement learning. *arXiv preprint arXiv:1702.07780*, 2017.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2250–2259, 2020.

Haotong Qin, Mingyuan Zhang, Yifu Ding, Aoyu Li, Zhongang Cai, Ziwei Liu, Fisher Yu, and Xianglong Liu. BiBench: Benchmarking and analyzing network binarization. In *International Conference on Machine Learning*, pp. 28351–28388. PMLR, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. *Advances in Neural Information Processing Systems*, 37:88981–89018, 2024.

Gabriele Santini, Francesco Paissan, and Elisabetta Farella. A probabilistic framework for dynamic quantization. *arXiv preprint arXiv:2505.10689*, 2025.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for KV cache compression. *arXiv preprint arXiv:2408.05646*, 2024.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020.

Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Changyuan Wang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Q-VLM: Post-training quantization for large vision-language models. *arXiv preprint arXiv:2410.08119*, 2024.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

T Wolf. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Canwen Xu and Julian McAuley. A survey on model compression and acceleration for pretrained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10566–10575, 2023.

Yuhui Xu, Shuai Zhang, Yingyong Qi, Jiaxian Guo, Weiyao Lin, and Hongkai Xiong. DNQ: Dynamic network quantization. *arXiv preprint arXiv:1812.02375*, 2018.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Bin Yang. Projection approximation subspace tracking. *IEEE Transactions on Signal processing*, 43(1):95–107, 1995.

Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. HAWQ-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pp. 11875–11886. PMLR, 2021.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, et al. LLM inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*, 2024.

Jinjie Zhang, Yixuan Zhou, and Rayan Saab. Post-training quantization for neural networks with provable guarantees. *SIAM journal on mathematics of data science*, 5(2):373–399, 2023.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Yichi Zhang, Junhao Pan, Xinheng Liu, Hongzheng Chen, Deming Chen, and Zhiru Zhang. FracBNN: Accurate and FPGA-efficient binary neural networks with fractional activations. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 171–182, 2021.

Ritchie Zhao, Weinan Song, Wentao Zhang, Tianwei Xing, Jeng-Hau Lin, Mani Srivastava, Rajesh Gupta, and Zhiru Zhang. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 15–24, 2017.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12: 1556–1577, 2024.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. TTRL: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A Related Work

A.1 Model Compression

The field of model compression for LLMs has aimed at mitigating the substantial computation and memory requirements Zhu et al. (2024); Yuan et al. (2024). Such methods primarily fall into four categories: weight quantization Lin et al. (2024b); Frantar et al. (2022); Wang et al. (2024), network pruning LeCun et al. (1989); Hassibi et al. (1993); Frantar & Alistarh (2023b); Bai et al. (2024b), knowledge distillation Hsieh et al. (2023); Hwang et al. (2024), and rank reduction Yuan et al. (2023); Liu et al. (2024); Hwang et al. (2024); Saxena et al. (2024); Saha et al. (2024).

A.2 Dynamic Quantization

Dynamic quantization is a strategy for adapting precision at runtime rather than using fixed bit-widths across all layers and inputs. DNQ Xu et al. (2018) uses a controller that learns per-layer bit-width policies to balance accuracy and compression. Recent research Santini et al. (2025) uses probabilistic frameworks for input-adaptive quantization, where quantization parameters are conditioned on each input, demonstrating negligible performance loss with adaptive schemes compared to static counterparts.

A.3 Few-Bit Quantization

Extreme 1-bit quantization Qin et al. (2023) has shown relatively good performance even with 1-bit quantization, e.g., BNN Courbariaux et al. (2016); Kim & Smaragdis (2016), XNOR-Net Rastegari et al. (2016), DoReFa-Net Zhou et al. (2016), Bi-Real Liu et al. (2018), IR-Net Qin et al. (2020), RBNN Lin et al. (2020), and ABC-Net Lin et al. (2017). They proposed a variety of gradient approximation methods for non-differentiable binarization operation. For example, BNN uses STE trick for non-differentiable sign function.

Binarized networks have many papers on hardware implementation, mostly on generic field-programmable gate-array (FPGA) platform, e.g., Zhao et al. (2017); Zhang et al. (2021). Nevertheless, binarization often causes a substantial loss, and most quantization papers Frantar et al. (2022); Lin et al. (2023) for foundation models consider at least 3 bits to achieve an acceptable performance. Whereas, 2-bit quantization was shown feasible for LLM compression with vector quantization Chee et al. (2023); Egiazarian et al. (2024).

A.4 Activation/Hardware-Aware Quantization

DeepShift Elhoushi et al. (2021) uses power-of-two weights to eliminate multiplication operations. HAWQ Dong et al. (2019) uses layer-wise quantization based on optimal brain pruning LeCun et al. (1989). HAWQv2 Dong et al. (2020) considers mixed-precision weight and activation. HAWQv3 Yao et al. (2021) uses integer weight in dyadic format. HEQ Koike-Akino et al. (2024) jointly minimizes quantization loss and computing energy on custom hardware. Then, GPTQ Frantar et al. (2022) extends HAWQ using zero-shot calibration, while activation-aware quantization (AWQ) Lin et al. (2023) uses activation-dependent scaling. GPFQ Zhang et al. (2023) uses a greedy path-following mechanism to quantize weights with theoretical guarantees. AWP Liu et al. (2025) uses projected gradient descent for activation aware quantization and pruning based on compressed sensing framework. QEP Arai & Ichikawa (2025) mitigates error propagation across layers. QLoRA Dettmers et al. (2023) uses low-rank adapter to compensate for the loss of quantized LLMs.

A.5 TEST-TIME COMPUTING

Prior work has explored improving performance at inference time by increasing computation, adaptation, or interaction beyond a single forward pass. Test-time scaling Chen et al. (2024); Muennighoff et al. (2025); Snell et al. (2024) allocate additional inference-time compute via sampling, search, and reasoning while keeping model parameters fixed, whereas test-time computing dynamically structures inference through adaptive computation, tool use, and external memory. In contrast, test-time update/adaptation Liang et al. (2025); Chen et al. (2022) performs unsupervised or self-supervised parameter updates during deployment, often under distribution shift, with meta-learning explicitly training models for rapid test-time adaptation. Test-time reinforcement learning Zuo et al. (2025); Odena et al. (2017) further frames inference as an interactive process with online policy improvement driven by reward signals. μ-MoE Koike-Akino et al. (2025b) is a new test-time pruning framework, using activation-aware pruning on the fly at the inference time.

B RTN: GROUPWISE ROUND-TO-NEAREST QUANTIZATION

Round-to-nearest (RTN) is the simplest quantization method to minimize approximation error:

$$\mathcal{L}_0 = \|W - \hat{W}\|^2, \tag{4}$$

where $W \in \mathbb{R}^{d' \times d}$ is a weight matrix and $\hat{W}$ is its quantized version. RTN uses the groupwise quantization-dequantization (QDQ) operation:

$$\hat{W} = \mathcal{Q}[W] \triangleq \mathcal{G}^-[\mathcal{G}[W]], \tag{5}$$

Here, the quantization $\mathcal{G}[\cdot]$ and dequantization $\mathcal{G}^-[\cdot]$ are defined as:

$$W_{\rm int} = \mathcal{G}(W) \triangleq \mathsf{round}[\mathsf{clamp}_q[(W - Z) \oslash S]], \tag{6}$$

$$\hat{W} = \mathcal{G}^-(W_{\rm int}) \triangleq W_{\rm int} \circ S + Z, \tag{7}$$

where $S \in \mathbb{R}^{d' \times d}$ and $Z \in \mathbb{R}^{d' \times d}$ are scale and zero-point parameters. Here, $\oslash$ is element-wise division, $\circ$ is element-wise product, $\mathsf{round}[x]$ gives the closest integer to x, and $\mathsf{clamp}_q[x] = \min(\max(x, 0), 2^q - 1)$ is q-bit limiting operator. The straightforward choice of scale and zero-point is

$$S = (W_{\max} - W_{\min})/(2^q - 1), \tag{8}$$

$$Z = W_{\min}, \tag{9}$$

while they can be adjusted to improve accuracy slightly. The QDQ has several variants and non-uniform formats, such as NF4 (Dettmers et al., 2023), see Appendix D. Pseudo-code of RTN with a groupsize g is as follows:

```
def rtn(W, q, g): # q: bits, g: groupsize
    ddash, d = W.shape # W: (d', d)
    W = W.reshape(-1, g) # grouping (d'*d/g, g)
    Wmax = W.amax(axis=1, keepdim=True) # (d'*d/g, 1)
    Wmin = W.amin(axis=1, keepdim=True) # (d'*d/g, 1)
    S, Z = (Wmax - Wmin) / (2**q - 1), Wmin # scale, zero-point
    Wint = ((W - Z) / S).round().clamp(0, 2**q - 1)
    What = Wint * S + Z # dequantization
    return What.reshape(ddash, d) # reshaping back
```

The total memory will be $qd'd$ bits for $W_{\rm int}$ and $d'd/g$ parameters for S and Z. Larger groupsize reduces the total required memory to represent $\hat{W}$, while smaller groupsize can reduce the quantization error. The recent NVFP format (Egiazarian et al., 2025) uses a microscaling groupsize of 16 to accelerate GPU computing. Because most GPUs are not fully compatible to arbitrary bitwidth integer operations, the practical advantage for the quantization comes with the reduction of required memory, which also leads to GPU acceleration due to the reduction of caching bottleneck. Most literature Lin et al. (2023); Frantar et al. (2022; 2025) claims 2 to 4-fold speedup with quantized LLMs.

C AWQ: ACTIVATION-AWARE QUANTIZATION

To improve over naïve RTN quantization, the activation-aware framework (Lin et al., 2023; Frantar et al., 2022; Liu et al., 2025) leverages activation statistics. Let $X \in \mathbb{R}^{d\times T}$ be an input activation with embedding dimension d and token length T. The aim is to minimize the approximation loss:

$$\mathcal{L} \triangleq \mathbb{E}_X\left[\left\|(W-\hat{W})X\right\|^2\right] \tag{10}$$

$$= \mathrm{tr}[(W-\hat{W})\underbrace{\mathbb{E}_X[XX^\top]}_{C\in\mathbb{R}^{d\times d}}(W-\hat{W})^\top] \tag{11}$$

$$= \|(W-\hat{W})C^{1/2}\|^2, \tag{12}$$

where $C \triangleq \mathbb{E}_X[XX^\top] \in \mathbb{R}^{d\times d}$ is the auto-correlation statistics of input X. Since C is not exactly known at test time, we estimate it with small amount of calibration data, e.g., via shrunk estimator:

$$C_\lambda = (1-\lambda)XX^\top + \lambda\eta I, \tag{13}$$

where λ is a shrinkage parameter, and $\eta = \|X\|^2/d$. For large dimension d, the Ledoit–Wolf shrinkage (Ledoit & Wolf, 2004) is reliable. Note that this is equivalent to minimize the weighted errors of activation-aware and weight-only quantization objectives:

$$\mathcal{L}' \triangleq (1-\lambda)\|(W-\hat{W})X\|^2 + \lambda\eta\|W-\hat{W}\|^2 \tag{14}$$

$$= \|(W-\hat{W})C_\lambda^{1/2}\|^2. \tag{15}$$

Also note that the solution is invariant to any scaling factor to the correlation, e.g., many literature use

$$C'_{\lambda'} \triangleq C_\lambda/(1-\lambda) \tag{16}$$

$$= XX^\top + \lambda' I, \tag{17}$$

where $\lambda' = \lambda\eta/(1-\lambda)$ is a so-called damping factor. We treat any scaled correlation matrix estimate as C_λ.

When there is no structure in correlation matrix C_λ, it has no closed-form solution to minimize $\mathcal{L}'$, whereas GPTQ (Frantar et al., 2022) uses the greedy method based on the inverse Hessian inspired by optimal brain surgeon (Hassibi et al., 1993). GPTQ requires the Cholesky factorization, whose complexity is at least of cubic order: $\mathcal{O}[d^3 + dd'T]$. In addition, it needs extra computations for off-diagonal error cancellation with the Gaussian elimination. Alternatively, AWP (Liu et al., 2025) uses projected gradient descent to solve it iteratively, i.e., at the kth step:

$$\hat{W}^{(k+1)} = \mathcal{Q}[\hat{W}^{(k)} + \mu(W-\hat{W}^{(k)})C_\lambda], \tag{18}$$

where μ is the stepsize. It requires a cubic complexity of $\mathcal{O}[d'd^2K]$ for K total iterations.

AWQ (Lin et al., 2023) greatly simplifies the problem by approximating with a diagonal correlation:

$$C_\lambda \simeq D \triangleq \mathsf{diag}[XX^\top + \lambda I]^\alpha, \tag{19}$$

where $\mathsf{diag}[C] \triangleq C \circ I$ offers a diagonal matrix zeroing out off-diagonal elements of an argument matrix C, and α is an auxiliary parameter to compensate for the diagonal approximation loss. Note that the above expression gives the ℓ_2-norm diagonal $D_{i,i} = (\|X_{i,:}\|_2^2 + \lambda)^\alpha$, while the original AWQ uses ℓ_1-norm $D_{i,i} = (\|X_{i,:}\|_1^2 + \lambda)^\alpha$. Given a diagonal correlation matrix, the closed-form solution for (15) is given by the scaled QDQ operation:

$$\hat{W} = \mathcal{Q}[WD^{1/2}]D^{-1/2}. \tag{20}$$

Importantly, the diagonal correlation also reduces the computation for D from $\mathcal{O}[d^2T']$ to $\mathcal{O}[dT']$. AWQ then requires only quadratic extra-complexity of $\mathcal{O}[2dd' + dT']$, yet achieves performance competitive with GPTQ. The pseudo-code of the AWQ concept is given as follows:

```
def awq(X, W, q, g, p, lam, alpha): # q: bits, g: groupsize, p: lp-norm
    D = X.norm(p=p, axis=1) # X: (d, T)
    D = (D + lam) ** alpha  # D: (d,)
    What = rtn(W * D[None, :], q, g) # scaled QDQ
    return What * D.reciprocal()[None, :] # scaling back
```

Here, we generalized to any ℓ_p-norm with arbitrary p. The hyper-parameters such as α and p can be adjusted by searching for the best values to minimize (15).

D Quantization-Dequantization (QDQ) Formats

There are many variants for QDQ operations, where we consider one example:

$$\mathcal{G}(W) \triangleq \mathsf{round}[\mathsf{clamp}_q[(W - Z) \oslash S]], \tag{21}$$

$$\mathcal{G}^{-}(W_{\text{int}}) \triangleq W_{\text{int}} \circ S + Z, \tag{22}$$

where $\mathcal{G}[\cdot]$ and $\mathcal{G}^{-}[\cdot]$ denote quantization and dequantization operations, respectively. Instead, many literature use an alternative representation:

$$\mathcal{G}'(W) \triangleq \mathsf{clamp}_q[\mathsf{round}[W \oslash S'] + Z'], \tag{23}$$

$$\mathcal{G}'^{-}(W_{\text{int}}) \triangleq (W_{\text{int}} - Z') \circ S'. \tag{24}$$

Both representations have no significant difference, and thus we focus on the first one. Yet another format includes nonuniform scaling like the NF4 format, which is based on the normal distribution.

We consider the asymmetric format for the scale and zero-point:

$$S = (W_{\max} - W_{\min})/(2^q - 1), \tag{25}$$

$$Z = W_{\min}. \tag{26}$$

Some literature further adjust the scale and zero-point factors to minimize the quantization error. For example, we may use expanded $W'_{\max}$ and $W'_{\min}$ instead of $W_{\max}$ and $W_{\min}$:

$$W'_{\max} \triangleq \frac{1+\nu}{2} W_{\max} + \frac{1-\nu}{2} W_{\min} \tag{27}$$

$$W'_{\min} \triangleq \frac{1-\nu}{2} W_{\max} + \frac{1+\nu}{2} W_{\min} \tag{28}$$

where $\nu \in \mathbb{R}$ is an expansion factor. Note that $\nu = 1$ reduces to the standard scaling. The best expansion factor is often around $\nu \simeq 0.95$.

Alternatively, the symmetric format is often used:

$$S = 2|W|_{\max}/(2^q - 1), \tag{29}$$

$$Z = -|W|_{\max}. \tag{30}$$

As it has fewer degrees of freedom (i.e., Z or S is redundant as $S = -Z/(2^q - 1)$), it offers a degraded accuracy in general, while it gains fewer memory requirement.

Yet another option is to extend test-time activation-aware quantization towards vector quantization Egiazarian et al. (2024) and compression Frantar & Alistarh (2023a).

E Quantization-Aware Low-Rank Factorization

Initialization of low-rank factors may impact the quantization performance. What is the best low-rank factors B and A, to reduce the quantization error of the residual? The most naïve way is to use the top-r principal components of W:

$$U_r \Lambda_r V_r = \mathsf{svd}_r[W], \tag{31}$$

$$B = U_r \Lambda_r^{1/2}, \tag{32}$$

$$A = \Lambda_r^{-1/2} V_r, \tag{33}$$

where $\mathsf{svd}_r[\cdot]$ is rank-r truncated singular-value decomposition (SVD), $U_r \in \mathbb{R}^{d' \times r}$, $\Lambda_r \in \mathbb{R}^{r \times r}$, $V_r \in \mathbb{R}^{r \times d}$ are left singular vectors, diagonal singular-values matrix, and right singular vectors, respectively. If we have a calibration data, we can use ASVD Yuan et al. (2023); Koike-Akino et al. (2025a) instead: $\mathsf{svd}_r[W C_\lambda^{1/2}] C_\lambda^{-1/2}$.

However, it does not guarantee that the residual $W - BA$ can reduce the quantization error in the end. To consider quantization-aware low-rank factorization, we may use alternating method, e.g., at the kth step:

$$B^{(k)} A^{(k)} = \mathsf{svd}_r[W - W_{\text{q}}^{(k)}], \tag{34}$$

$$W_{\text{q}}^{(k+1)} = \mathcal{Q}[W - B^{(k)} A^{(k)}], \tag{35}$$

with a proper initialization. This quantization-aware low-rank factorization may reduce the quantization error, while it still has no guarantee that the loss is small at inference time. We found that the alternating solution had almost no gain, and we focus on the straightforward principal components for static low-rank factor initialization.

Low-Rank Factor Quantization Alternatively, the low-rank factors A and/or B can be also quantized online or offline, to further accelerate inference. For example, we may consider: (1) A is quantized while B is un-quantized; (2) B is quantized while A is un-quantized; and (3) both A and B are quantized. The cases (1) and (2) may be more advantageous because the product of BA is no longer quantized.

Low-Rank Factor Pruning In addition, we can integrate with pruning with quantization and low-rank factorizations. For example, we can use test-time pruning used in μ-MoE Koike-Akino et al. (2025b) in conjunction with TTQ. Because both uses similar diagonal correlation matrix, we do not need extra computation for D. Similarly, we can consider pruning for low-rank factors as well: (1) A is pruned while B is un-pruned; (2) B is pruned while A is un-pruned; and (3) both A and B are pruned. Because the product of BA is no longer sparse, the cases (1) and (2) may be more advantageous.

Test-Time Decomposition Another option is to use test-time low-rank decomposition for adaptive A and B at inference time. We may consider using efficient online PCA algorithms:

- **Stochastic gradient methods.** Oja's rule (Oja, 1982) performs stochastic gradient ascent on the Rayleigh quotient to estimate the top principal components from streaming data.
- **Incremental SVD methods.** Deterministic approaches to update low-rank factorizations include incremental eigen-analysis (Hall et al., 1998) and incremental SVD techniques (Brand, 2002).
- **Subspace tracking algorithms.** Recursive least-squares formulations such as PAST (Yang, 1995) provide fast convergence and strong tracking capability under non-stationary data.
- **Online optimization perspectives.** Analyzing online PCA via stochastic optimization frameworks establishes finite-sample convergence and performance guarantees (Allen-Zhu & Li, 2017).

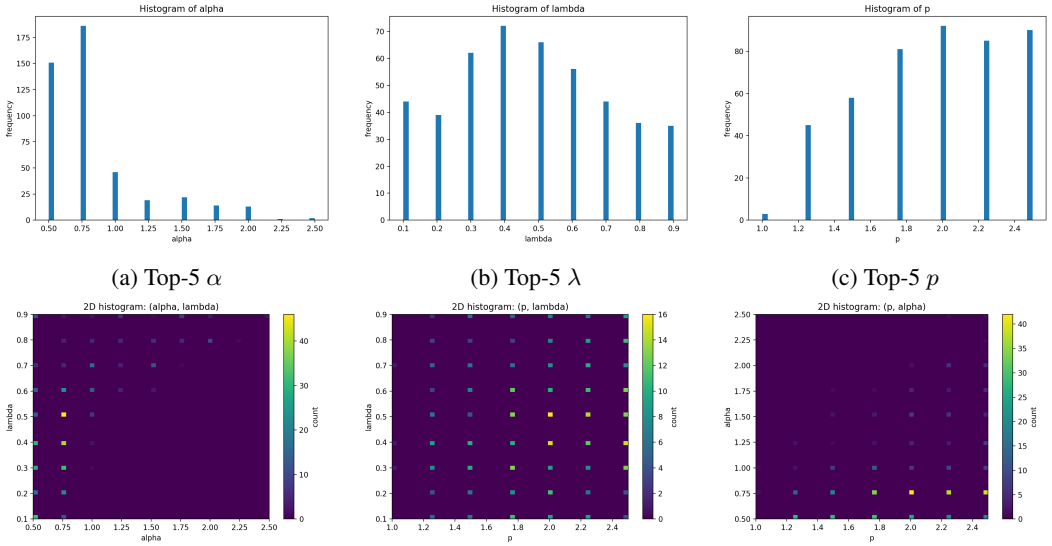
F Hyperparameter Selection for TTQ

In static/original AWQ, quantization hyperparameters like α can be adjusted at the offline calibration phase. However, TTQ needs to re-compute scale and zero-points S and Z based on diagonal correlation C_λ, dynamically given every incoming tokens X. Hence, searching those hyperparameters on the fly should be restricted.

Nevertheless, if we still can use post-training calibration data, then we can find the best possible hyperparameters and prior correlation matrix in more exhaustive way before test time. And, online update of correlation matrix is carried out at inference time to improve the correlation estimation accuracy. In our work, we use constant hyperparameters manually selected. How to choose the hyperparameters?

We evaluated the perplexity performance for OPT family when using uniformly sampled grids of α, λ, and p. We then selected 5 best combinations per OPT model and quantization bits q. Then, we analyzed the histogram of those best parameter sets, as shown in Figure 2. Those histogram plots may suggest some general trends:

- The norm exponent α should be around 0.5 or 0.75. It suggests that the diagonal correlation approximation should be cooled down $\alpha < 1$ to be more uniform than amplified with $\alpha > 1$.
- The damping factor λ should be around 0.4, which is way larger than the nominal choice like $\lambda = 0.01$ in most literature. It may be because the dimension d is very large compared



(a) Top-5 α (b) Top-5 λ (c) Top-5 p

(d) Top-5 (α, λ) (e) Top-5 (λ, p) (f) Top-5 (p, α)

Figure 2: Histogram of top-5 hyperparameter selections (α, λ, p) for different OPT models with quantization bits of $q \in \{2, 3, 4, 5\}$.

to available token length T, leading to a larger shrinkage factor to be reliable under certain criterion Ledoit & Wolf (2004). Also, it indicates that selecting around 50% damping factor offers a good trade-off between activation-aware loss and activation-unaware loss in (15).

- The best ℓ_p-norm is around $p = 2$, which can be well-justified from the derivation in (19). More importantly, ℓ_1-norm was found to be a terrible choice. The original AWQ relied on a heuristic metric based on ℓ_1, while some literature pointed out such an approximation may be suboptimal: e.g., LatentLLM Koike-Akino et al. (2025a) made a thorough comparison to other variants of preconditioning; and Wanda Sun et al. (2023) derives a proper diagonal approximation with ℓ_2 correlation.

In our experiments, the results of AWQ are based on ℓ_2-norm not the original ℓ_1-norm, while α is optimized with line search.

G Experiments Setup

We conduct experiments for LLM benchmarks to evaluate the effectiveness of our method. Our experiments are based on the same setting of SparseLLM Bai et al. (2024b) and their code base[2]. Following existing work Sun et al. (2023), we quantize all linear layers in LLM transformers. We implemented TTQ in PyTorch Paszke et al. (2019) and used the HuggingFace Transformers library Wolf (2019) for handling models and datasets. All experiments are conducted on NVIDIA A40 or A100 GPUs.

For LLM experiments, we first consider the OPT model family Zhang et al. (2022) as it provides a wide range of model scales from 125M to 175B. We then evaluate more recent LLMs: Qwen3 Yang et al. (2025) and Gemma3 Gemma Team et al. (2025). We show results on different sizes of models to provide a broader picture for the performance of TTQ. We measure perplexity score for three popular benchmarks: raw-WikiText2 (WT2) Merity et al. (2016); the Penn Treebank (PTB) Marcus et al. (1994); and C4 Raffel et al. (2020). Details of LLMs and datasets we used are found in Appendices L and M.

[2] https://github.com/BaiTheBest/SparseLLM

H RUNTIME BENCHMARK

We show TTQ can accelerate inference at test time over un-quantized LLMs, even with the extra overhead of prescaling and low-rank factors. To evaluate the TTQ runtime speed, we use Marlin kernel Frantar et al. (2025) which is one of best-known `int_matmul` CUDA kernels. Specifically, we use `vllm._custom_ops.marlin_gemm` in vllm library[3]. It often outperforms `vllm._custom_ops.awq_gemm`. We use CUDA 12.8, PyTorch 2.9.1, and vllm 0.15.1. We include prescaling with D diagonal matrix and low-rank projections with B and A on top of the Marlin kernel for TTQ like below:

```
def forward(self, x):
    # quantization params on the fly
    W, S, Z, D = self.find_params(x)
    # low-rank projection
    y0 = (x @ self.A.t().contiguous()) @ self.B.t().contiguous()
    # pre-scaling
    x = x / D
    # scaled int_matmul
    return marlin_gemm(x, y0, W, S, Z)

def find_params(self, x):
    # scale
    D = (x.norm(p=self.p, axis=0) + self.lam) ** self.alpha
    # prescale weight
    W = self.weight * D
    W = W.reshape(-1, self.groupsize)
    # scale, zeros
    Z = W.amin(axis=1, keepdim=True)
    S = (W.amax(axis=1, keepdim=True) - Z) / self.qmax
    # quantize
    W = ((W - Z) / S).round().clamp(0. self.qmax).to(torch.int32)
    return W, S, Z, D
```

To reduce the overhead of multi-kernel launching, we use `torch.compile` and CUDA graph replay[4] for all cases. We consider a decode phase of LLM projecting a single token.

Tables 4, 5, 6, 7 and 8 show the runtime benchmark for Qwen3 models on NVIDIA A40, A100, L40, RTX3090 and RTX4090 GPUs, respectively. To highlight the speedup, we focus on linear query projection module for benchmarking the original linear layer, offline 4-bit quanitzed AWQ, and online 4-bit quantized TTQ. We can observe:

- The throughput generally degrades with larger LLMs.
- AWQ can accelerate LLMs, especially with the Marlin kernel, up to 6.7 folds at 32B model on RTX4090.
- TTQ ($r = 0$) has no significant loss in speed over AWQ, even with extra operations.
- Even without custom TTQ kernel in the presence of $r = 16$ low-rank projections, TTQ can still accelerate the inference up to 4.9 folds at 32B model on RTX4090.
- TTQ shows more advantage on larger LLMs because of dominating weight traffic (e.g., Qwen3-32B model needs to transfer $5{,}120 \times 8{,}192$ weights of 168MB from global memory to shared memory on GPUs for FP16 query projection).

AWQ fuses the prescaling D into previous modules, while TTQ needs to fuse into `int_matmul`. Most kernels are not yet compatible for such prologue fusion, and thus designing custom CUDA kernels for TTQ should further accelerate the inference on GPU. In addition, most kernels are not designed for arbitrary bit width yet, and custom kernel design for 2-bit TTQ should speedup more, theoretically doubling due to traffic reduction with weight packing.

[3] `https://github.com/vllm-project/vllm`

[4] `https://docs.vllm.ai/en/stable/design/cuda_graphs/`

Table 4: Runtime Speed (k tokens/sec) of Query Projection Module for Qwen3 Models with 4-bit AWQ and TTQ, on NVIDIA A40 GPU.

Qwen3	0.6B	1.7B	4B	8B	14B	32B
FP16	57.58	41.29	21.38	14.10	10.43	6.86
AWQ (`awq_gemm`)	75.76	71.60	54.61	32.68	25.28	18.28
AWQ (`marlin_gemm`)	73.32	73.70	**73.98**	**38.91**	**37.52**	**30.32**
TTQ ($r = 0$)	**80.63**	**78.68**	69.11	36.52	24.94	19.34
TTQ ($r = 16$)	66.67	59.08	49.67	30.32	23.78	17.43

×2.5

Table 5: Runtime Speed (k tokens/sec) of Query Projection Module for Qwen3 Models with 4-bit AWQ and TTQ, on NVIDIA A100 GPU.

Qwen3	0.6B	1.7B	4B	8B	14B	32B
FP16	68.43	**68.91**	42.76	30.69	20.16	15.90
AWQ (`awq_gemm`)	68.10	56.09	48.20	38.28	32.25	24.35
AWQ (`marlin_gemm`)	**69.41**	64.93	**58.41**	**53.13**	**49.07**	**42.28**
TTQ ($r = 0$)	61.85	57.72	52.67	48.58	45.17	39.40
TTQ ($r = 16$)	46.04	43.11	40.01	37.19	34.32	30.46

×1.9

I LLM Benchmark Results

Tables 9, 10, and 11 show full performance results including standard deviation and more model variants for OPT, Qwen3, and Gemma3 LLMs, respectively.

J VLM Benchmark Results

We show the quantization results for Qwen3-VL VLM models on TextVQA benchmark. We quantize all linear modules except LM head and vision encoder. Table 12 shows VQA soft accuracy performance with different quantization methods. For AWQ, we use different calibration dataset: COCO-Caption; OK-VQA; ChartQA; and TextVQA. We confirm that our TTQ achieves best overall performance. It is interesting to see that some quantized models can slightly outperform unquantized Qwen-VL models.

K VLA Benchmark Results

We show the quantization results for $\pi_{0.5}$ VLA model on LIBERO robot manipulation benchmark in Table 13. We quantize all linear modules for VLM backbone of $\pi_{0.5}$ model except LM head and vision encoder. Here, we use 200 episodes per benchmark. For AWQ, we compare with different calibration dataset from LIBERO Spatial, Object, Goal, and 10 task suites. Although AWQ performs much better than RTN, its performance highly depends on offline calibration data. TTQ achieves best in average success rates. The advantage is clearer at the long-horizon tasks of LIBERO 10. It is interesting to see that our TTQ can allow 2-bit quantization for $\pi_{0.5}$ without causing much performance loss.

L LLM Models

OPT Family The Open Pre-trained Transformers (OPT) (Zhang et al., 2022) is a suite of decoder-only pre-trained transformers ranging from 125M to 175B parameters. It was claimed that OPT-175B is comparable to GPT-3, while requiring only 1/7th the carbon footprint to develop. Table 14 shows model parameters for the OPT open LLM family.

Table 6: Runtime Speed (k tokens/sec) of Query Projection Module for Qwen3 Models with 4-bit AWQ and TTQ, on NVIDIA L40 GPU.

Qwen3	0.6B	1.7B	4B	8B	14B	32B
FP16	**91.35**	75.18	74.60	53.31	44.45	34.74
AWQ (`awq_gemm`)	91.01	84.70	76.00	59.99	48.95	47.87
AWQ (`marlin_gemm`)	86.57	91.35	91.28	**90.93**	**77.81**	**68.15**
TTQ ($r = 0$)	90.59	**92.57**	**91.86**	84.19	71.28	63.37
TTQ ($r = 16$)	72.82	69.11	69.84	63.37	55.87	49.62

Table 7: Runtime Speed (k tokens/sec) of Query Projection Module for Qwen3 Models with 4-bit AWQ and TTQ, on NVIDIA RTX3090 GPU.

Qwen3	0.6B	1.7B	4B	8B	14B	32B
FP16	59.75	54.38	29.53	20.37	14.15	9.77
AWQ (`awq_gemm`)	62.89	60.54	48.39	33.48	27.92	21.62
AWQ (`marlin_gemm`)	63.13	**67.86**	**65.88**	**46.97**	**36.30**	**26.17**
TTQ ($r = 0$)	**68.07**	67.34	64.43	43.10	34.25	24.96
TTQ ($r = 16$)	57.57	58.72	44.73	33.20	28.45	21.85

Qwen3 Family We use Qwen3 (Yang et al., 2025) dense models, which are decoder-only transformers spanning 270M to 30B parameters, built on a consistent architecture with RMSNorm, SwiGLU feed-forward layers, and rotary positional embeddings. All variants employ grouped-query attention (GQA) with a fixed small number of key–value heads while scaling the number of query heads with model width, reducing KV-cache cost. Importantly, the hidden size is decoupled from the attention projection width, providing additional flexibility. Parameters are listed in Table 15.

Gemma3 Family Gemma3 models (Gemma Team et al., 2025) are decoder-only transformer architectures released across a wide range of scales, from 270M to 27B parameters, and include both text-only and multimodal variants. Similar to Qwen3, all Gemma3 models adopt RMSNorm, SwiGLU feed-forward networks, and rotary positional embeddings, as well as grouped-query attention (GQA). The per-head dimension is fixed at 256 across model sizes. Parameters are listed in Table 16.

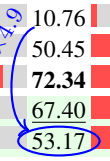
LLaVA Family LLaVA Liu et al. (2023b) integrates a vision encoder with LLMs to enable processing both language and visual context modalities through instruction-tuning. The vision encoder typically uses contrastive language-image pretraining (CLIP) vision transformer (ViT), while the LLMs are based on LLaMA or Vicuna. It was claimed that LLaVA-1.6-34B outperforms Gemini Pro on some benchmarks including MMMU and MathVista.

Qwen3-VL Family Qwen3-VL Bai et al. (2025) is a family of multimodal LLMs that extend the Qwen3 transformer with vision–language capabilities. The models integrate a SigLIP-based vision encoder with the Qwen3 language backbone through a projection module, enabling joint reasoning over images, videos, and text for tasks such as visual question answering, captioning, and document understanding. Parameters are listed in Table 17.

$\pi_{0.5}$ Family The $\pi_{0.5}$ model Intelligence et al. (2025) is a PaliGemma-based state-of-the-art VLA transformer having 2.3B parameters, that maps visual observations and language instructions directly to continuous robot actions through flow-matching diffusion policy. Trained via distillation from a larger foundation model on diverse robot interaction data, it achieves strong zero-shot generalization while remaining lightweight and deployable for real-world manipulation tasks. Parameters are listed in Table 18.

Table 8: Runtime Speed (k tokens/sec) of Query Projection Module for Qwen3 Models with 4-bit AWQ and TTQ, on NVIDIA RTX4090 GPU.

Qwen3	0.6B	1.7B	4B	8B	14B	32B
FP16	116.82	77.04	72.45	58.44	46.78	10.76
AWQ (`awq_gemm`)	113.84	90.42	80.19	62.94	51.32	50.45
AWQ (`marlin_gemm`)	**120.62**	**115.20**	**112.61**	**101.00**	**80.45**	**72.34**
TTQ ($r = 0$)	108.00	104.11	103.45	92.23	74.72	67.40
TTQ ($r = 16$)	77.88	73.47	72.55	66.62	58.65	53.17



M DATASETS

Wikitext-2 (WT2) The WikiText language modeling dataset (Merity et al., 2016) is a collection of over 100 million tokens extracted from the set of verified good and featured articles on Wikipedia. The dataset is available under the CC BY-SA-4.0 license. The wikitext-2-raw-v1 contains 36,718, 3,760, and 4,358 samples for train, validation, and test splits, respectively. We use `https://huggingface.co/datasets/mindchain/wikitext2`.

Penn Treebank (PTB) The English Penn Treebank (PTB) corpus (Marcus et al., 1994) is one of the most known and used corpus for the evaluation of models for sequence labeling. The dataset features a million words of 1989 Wall Street Journal material. We use `https://huggingface.co/datasets/ptb-text-only/ptb_text_only`.

C4 C4 (Raffel et al., 2020) is based on a colossal, cleaned version of Common Crawl's web crawl corpus. This is release under the OCD-By license. We consider a subset "en", containing 364,868,892 and 364,608 samples for train and validation splits, respectively, while we use the first shard for each split in `https://huggingface.co/datasets/allenai/c4`.

ScienceQA ScienceQA Lu et al. (2022) is collected from elementary and high school science curricula (i.e., grades 1 through 12), and contains 21,208 multimodal multiple-choice science questions. Out of the questions in ScienceQA, 10,332 (48.7%) have an image context, 10,220 (48.2%) have a text context, and 6,532 (30.8%) have both. Most questions are annotated with grounded lectures (83.9%) and detailed explanations (90.5%). The lecture and explanation provide general external knowledge and specific reasons, respectively, for arriving at the correct answer. ScienceQA has rich domain diversity from three subjects: natural science, language science, and social science. ScienceQA features 26 topics, 127 categories, and 379 skills that cover a wide range of domains. We use `https://huggingface.co/datasets/derek-thomas/ScienceQA`, released under the CC BY-NC-SA 4.0 license.

TextVQA TextVQA Singh et al. (2019) requires VLM models to read and reason about text in images to answer questions about them. Specifically, models need to incorporate the new modality of text present in the images and reason over it to answer TextVQA questions. TextVQA dataset contains 45,336 questions over 28,408 images from the OpenImages dataset. We use `https://huggingface.co/datasets/llms-eval/textvqa`, licensed under CC-BY-4.0.

COCO-Caption The COCO-Caption dataset is a standard benchmark for image captioning built on the Microsoft COCO dataset Lin et al. (2014). It contains over 120k images, each annotated with five human-written captions describing the visual scene. The images cover diverse everyday environments with multiple objects and interactions for evaluating VLMs. We use `https://huggingface.co/datasets/llms-lab/COCO-Caption2017`, licensed under CC-BY-4.0.

OK-VQA The OK-VQA dataset Marino et al. (2019) is a benchmark for knowledge-based visual question answering, where answering questions requires external world knowledge beyond the visual content of the image. It contains over 14k questions paired with images from the MS COCO

Table 9: Perplexity (↓) of OPT models with different quantization methods. It shows macro average and standard deviation across WT2/PTB/C4 datasets. Groupsize is $g = 32$ for all cases. Calibration token length is $T = 2^{17}$ for AWQ. **Bold** and underline denote the best and second best, respectively. Asterisk "*" indicates reaching competitive performance to the original un-compressed LLM.

Quantization q	2 bits	3 bits	4 bits	5 bits
OPT-125M (WT2: 27.7, PTB: 39.0, C4: 26.6, Avg: **31.1**$_{\pm 6.8}$)				
RTN	5058.5$_{\pm 833.7}$	56.3$_{\pm 10.5}$	33.5$_{\pm 7.0}$	31.8$_{\pm 7.0}$
AWQ (WT2 Calib)	381.7$_{\pm 159.8}$	37.4$_{\pm 7.4}$	32.3$_{\pm 6.5}$	31.4$_{\pm 6.1}$
AWQ (PTB Calib)	375.3$_{\pm 84.9}$	37.3$_{\pm 6.9}$	32.2$_{\pm 6.3}$	31.3$_{\pm 6.0}$
AWQ (C4 Calib)	451.7$_{\pm 181.7}$	37.7$_{\pm 7.5}$	32.6$_{\pm 6.6}$	31.3$_{\pm 6.0}$
TTQ ($r = 0$)	257.4$_{\pm 65.3}$	36.6$_{\pm 8.2}$	31.9$_{\pm 7.0}$	31.2$_{\pm 6.8}$
TTQ ($r = 16$)	**141.7**$_{\pm 36.9}$	**35.8**$_{\pm 8.3}$	**31.8**$_{\pm 7.0}$	***31.1**$_{\pm 6.8}$
OPT-350M (WT2: 22.0, PTB: 31.1, C4: 22.6, Avg: **25.2**$_{\pm 5.1}$)				
RTN	25808.8$_{\pm 9601.0}$	55.3$_{\pm 13.8}$	27.6$_{\pm 5.5}$	25.6$_{\pm 5.3}$
AWQ (WT2 Calib)	293.2$_{\pm 103.5}$	28.8$_{\pm 5.1}$	25.8$_{\pm 4.5}$	25.4$_{\pm 4.4}$
AWQ (PTB Calib)	350.7$_{\pm 107.0}$	28.9$_{\pm 5.1}$	25.8$_{\pm 4.5}$	25.4$_{\pm 4.4}$
AWQ (C4 Calib)	322.8$_{\pm 98.6}$	29.0$_{\pm 5.6}$	25.9$_{\pm 4.5}$	25.4$_{\pm 4.4}$
TTQ ($r = 0$)	358.3$_{\pm 123.9}$	28.6$_{\pm 6.0}$	**25.7**$_{\pm 5.3}$	**25.3**$_{\pm 5.1}$
TTQ ($r = 16$)	**129.8**$_{\pm 46.8}$	**27.9**$_{\pm 5.9}$	**25.7**$_{\pm 5.2}$	**25.3**$_{\pm 5.1}$
OPT-1.3B (WT2: 14.6, PTB: 20.3, C4: 16.1, Avg: **17.0**$_{\pm 3.0}$)				
RTN	11514.4$_{\pm 2621.8}$	27.2$_{\pm 6.8}$	18.1$_{\pm 3.4}$	17.2$_{\pm 3.1}$
AWQ (WT2 Calib)	32.4$_{\pm 8.6}$	18.0$_{\pm 2.8}$	17.3$_{\pm 2.6}$	***17.0**$_{\pm 2.5}$
AWQ (PTB Calib)	32.6$_{\pm 6.6}$	18.1$_{\pm 2.7}$	17.3$_{\pm 2.6}$	***17.0**$_{\pm 2.5}$
AWQ (C4 Calib)	31.7$_{\pm 8.1}$	18.0$_{\pm 2.8}$	17.2$_{\pm 2.6}$	***17.0**$_{\pm 2.5}$
TTQ ($r = 0$)	32.2$_{\pm 8.3}$	18.2$_{\pm 3.4}$	17.2$_{\pm 3.0}$	***17.0**$_{\pm 3.0}$
TTQ ($r = 16$)	**27.2**$_{\pm 6.1}$	**17.9**$_{\pm 3.1}$	**17.1**$_{\pm 3.0}$	***17.0**$_{\pm 3.0}$
OPT-2.7B (WT2: 12.5, PTB: 18.0, C4: 14.3, Avg: **14.9**$_{\pm 2.8}$)				
RTN	6274.5$_{\pm 1350.7}$	36.0$_{\pm 11.5}$	15.7$_{\pm 3.1}$	15.0$_{\pm 2.8}$
AWQ (WT2 Calib)	23.1$_{\pm 5.3}$	15.7$_{\pm 3.0}$	15.1$_{\pm 2.8}$	15.0$_{\pm 2.8}$
AWQ (PTB Calib)	23.2$_{\pm 4.0}$	15.7$_{\pm 3.0}$	**15.0**$_{\pm 2.9}$	15.0$_{\pm 2.9}$
AWQ (C4 Calib)	22.9$_{\pm 5.1}$	15.7$_{\pm 3.0}$	**15.0**$_{\pm 2.8}$	15.0$_{\pm 2.8}$
TTQ ($r = 0$)	23.7$_{\pm 5.1}$	15.7$_{\pm 3.0}$	**15.0**$_{\pm 2.8}$	***14.9**$_{\pm 2.8}$
TTQ ($r = 16$)	**21.2**$_{\pm 4.5}$	**15.5**$_{\pm 2.9}$	**15.0**$_{\pm 2.8}$	***14.9**$_{\pm 2.8}$
OPT-6.7B (WT2: 10.9, PTB: 15.8, C4: 12.7, Avg: **13.1**$_{\pm 2.5}$)				
RTN	5716.5$_{\pm 664.0}$	26.2$_{\pm 8.9}$	13.7$_{\pm 2.8}$	13.2$_{\pm 2.5}$
AWQ (WT2 Calib)	17.2$_{\pm 3.6}$	13.5$_{\pm 2.6}$	13.2$_{\pm 2.5}$	***13.1**$_{\pm 2.5}$
AWQ (PTB Calib)	17.1$_{\pm 3.0}$	13.6$_{\pm 2.6}$	13.2$_{\pm 2.5}$	***13.1**$_{\pm 2.5}$
AWQ (C4 Calib)	16.9$_{\pm 3.5}$	13.6$_{\pm 2.6}$	13.2$_{\pm 2.6}$	***13.1**$_{\pm 2.5}$
TTQ ($r = 0$)	17.2$_{\pm 3.5}$	13.6$_{\pm 2.6}$	13.2$_{\pm 2.5}$	***13.1**$_{\pm 2.5}$
TTQ ($r = 16$)	**16.3**$_{\pm 3.4}$	**13.4**$_{\pm 2.6}$	***13.1**$_{\pm 2.5}$	***13.1**$_{\pm 2.5}$
OPT-13B (WT2: 10.1, PTB: 14.5, C4: 12.1, Avg: **12.2**$_{\pm 2.2}$)				
RTN	6413.1$_{\pm 1051.0}$	15.5$_{\pm 3.1}$	12.5$_{\pm 2.4}$	12.3$_{\pm 2.3}$
AWQ (WT2 Calib)	15.3$_{\pm 2.9}$	12.6$_{\pm 2.3}$	**12.3**$_{\pm 2.2}$	***12.2**$_{\pm 2.2}$
AWQ (PTB Calib)	15.4$_{\pm 2.7}$	12.6$_{\pm 2.3}$	**12.3**$_{\pm 2.2}$	***12.2**$_{\pm 2.2}$
AWQ (C4 Calib)	15.2$_{\pm 2.9}$	12.6$_{\pm 2.3}$	**12.3**$_{\pm 2.2}$	***12.2**$_{\pm 2.2}$
TTQ ($r = 0$)	15.0$_{\pm 2.9}$	**12.5**$_{\pm 2.7}$	**12.3**$_{\pm 2.2}$	***12.2**$_{\pm 2.2}$
TTQ ($r = 16$)	**14.8**$_{\pm 2.8}$	**12.5**$_{\pm 2.2}$	**12.3**$_{\pm 2.2}$	***12.2**$_{\pm 2.2}$

dataset, along with multiple human-provided answers for evaluation. The questions cover diverse topics such as science, history, and common knowledge, making the task more challenging than standard VQA benchmarks. We use `https://huggingface.co/datasets/llms-lab/COCO-Caption2017`, inheriting the COCO image license of CC-BY-4.0, while the annotations are released for research use.

Table 10: Perplexity (↓) of Qwen3 models with different quantization methods. It shows macro average and standard deviation across WT2/PTB/C4 datasets. Groupsize is $g = 32$ for all cases. Calibration token length is $T = 2^{17}$ for AWQ. **Bold** and underline denote the best and second best, respectively. Asterisk "*" indicates reaching competitive performance to the original un-compressed LLM.

Quantization q	2 bits	3 bits	4 bits	5 bits
Qwen3-0.6B (WT2: 21.0, PTB: 43.8, C4: 30.3, Avg: **31.7**$_{\pm 11.5}$)				
RTN	8.2e6$_{\pm 4.4e6}$	127.3$_{\pm 54.1}$	38.2$_{\pm 14.7}$	33.5$_{\pm 12.4}$
AWQ (WT2 Calib)	9739.1$_{\pm 2765.7}$	49.4$_{\pm 20.2}$	33.5$_{\pm 11.7}$	32.4$_{\pm 11.8}$
AWQ (PTB Calib)	17344.3$_{\pm 1962.7}$	47.9$_{\pm 17.3}$	34.1$_{\pm 12.5}$	32.0$_{\pm 11.5}$
AWQ (C4 Calib)	5388.1$_{\pm 1862.6}$	48.3$_{\pm 17.3}$	33.0$_{\pm 11.6}$	32.1$_{\pm 11.8}$
TTQ ($r = 0$)	2827.8$_{\pm 883.8}$	44.7$_{\pm 16.9}$	33.0$_{\pm 12.1}$	**31.9**$_{\pm 11.3}$
TTQ ($r = 16$)	**1552.6**$_{\pm 386.3}$	**42.0**$_{\pm 16.0}$	**32.9**$_{\pm 12.2}$	**31.9**$_{\pm 11.3}$
Qwen3-1.7B (WT2: 16.7, PTB: 33.8, C4: 16.1, Avg: **24.2**$_{\pm 8.7}$)				
RTN	1.4e6$_{\pm 2.2e5}$	162.8$_{\pm 107.4}$	30.6$_{\pm 12.2}$	26.1$_{\pm 9.6}$
AWQ (WT2 Calib)	1864.7$_{\pm 2051.9}$	30.0$_{\pm 12.0}$	24.5$_{\pm 8.7}$	24.4$_{\pm 8.6}$
AWQ (PTB Calib)	2309.6$_{\pm 2524.0}$	29.8$_{\pm 11.6}$	24.7$_{\pm 8.3}$	24.5$_{\pm 8.9}$
AWQ (C4 Calib)	2364.7$_{\pm 2878.7}$	28.2$_{\pm 9.9}$	24.9$_{\pm 9.1}$	24.4$_{\pm 8.8}$
TTQ ($r = 0$)	522.6$_{\pm 259.4}$	27.3$_{\pm 9.6}$	24.4$_{\pm 8.5}$	24.3$_{\pm 8.5}$
TTQ ($r = 16$)	**264.6**$_{\pm 90.4}$	**26.4**$_{\pm 9.3}$	**24.3**$_{\pm 8.8}$	***24.1**$_{\pm 8.5}$
Qwen3-4B (WT2: 13.7, PTB: 24.8, C4: 19.9, Avg: **19.4**$_{\pm 5.6}$)				
RTN	5803.3$_{\pm 2179.1}$	28.6$_{\pm 10.1}$	21.4$_{\pm 6.4}$	19.6$_{\pm 5.5}$
AWQ (WT2 Calib)	180.8$_{\pm 65.3}$	21.2$_{\pm 6.4}$	20.1$_{\pm 5.7}$	***19.4**$_{\pm 5.6}$
AWQ (PTB Calib)	199.2$_{\pm 29.7}$	26.6$_{\pm 9.0}$	19.8$_{\pm 5.9}$	***19.4**$_{\pm 5.6}$
AWQ (C4 Calib)	246.9$_{\pm 85.2}$	21.7$_{\pm 6.3}$	19.9$_{\pm 5.5}$	19.6$_{\pm 5.6}$
TTQ ($r = 0$)	120.4$_{\pm 34.3}$	**20.9**$_{\pm 6.0}$	**19.6**$_{\pm 5.5}$	***19.4**$_{\pm 5.5}$
TTQ ($r = 16$)	**78.1**$_{\pm 19.5}$	21.1$_{\pm 6.2}$	**19.6**$_{\pm 5.6}$	***19.4**$_{\pm 5.5}$
Qwen3-8B (WT2: 9.7, PTB: 17.2, C4: 15.4, Avg: **14.1**$_{\pm 3.9}$)				
RTN	5366.8$_{\pm 1985.5}$	18.4$_{\pm 5.7}$	14.9$_{\pm 4.1}$	14.4$_{\pm 3.9}$
AWQ (WT2 Calib)	39.5$_{\pm 12.9}$	15.3$_{\pm 4.2}$	14.4$_{\pm 4.0}$	14.2$_{\pm 4.0}$
AWQ (PTB Calib)	39.4$_{\pm 9.3}$	15.4$_{\pm 4.2}$	14.4$_{\pm 4.0}$	14.2$_{\pm 3.9}$
AWQ (C4 Calib)	37.6$_{\pm 10.9}$	15.2$_{\pm 4.1}$	14.4$_{\pm 4.0}$	***14.1**$_{\pm 3.9}$
TTQ ($r = 0$)	32.3$_{\pm 8.8}$	**15.0**$_{\pm 4.1}$	**14.2**$_{\pm 3.9}$	***14.1**$_{\pm 3.9}$
TTQ ($r = 16$)	**30.2**$_{\pm 8.0}$	**15.0**$_{\pm 4.1}$	14.3$_{\pm 3.9}$	***14.1**$_{\pm 3.9}$
Qwen3-14B (WT2: 8.6, PTB: 15.2, C4: 13.8, Avg: **12.6**$_{\pm 3.5}$)				
RTN	366.3$_{\pm 107.7}$	15.6$_{\pm 4.6}$	12.9$_{\pm 3.7}$	12.8$_{\pm 3.6}$
AWQ (WT2 Calib)	23.0$_{\pm 7.4}$	13.4$_{\pm 3.7}$	12.7$_{\pm 3.5}$	***12.6**$_{\pm 3.4}$
AWQ (PTB Calib)	23.0$_{\pm 6.7}$	13.3$_{\pm 3.7}$	12.7$_{\pm 3.5}$	***12.6**$_{\pm 3.5}$
AWQ (C4 Calib)	23.3$_{\pm 7.5}$	13.4$_{\pm 3.7}$	12.7$_{\pm 3.5}$	***12.6**$_{\pm 3.4}$
TTQ ($r = 0$)	21.1$_{\pm 6.5}$	13.2$_{\pm 3.6}$	12.7$_{\pm 3.5}$	***12.6**$_{\pm 3.5}$
TTQ ($r = 16$)	**20.3**$_{\pm 6.4}$	**13.1**$_{\pm 3.6}$	***12.6**$_{\pm 3.5}$	***12.6**$_{\pm 3.5}$

Table 11: Perplexity (↓) of Gemma3 models with different quantization methods. It shows macro average and standard deviation across WT2/PTB/C4 datasets. Groupsize is $g = 32$ for all cases. Calibration token length is $T = 2^{17}$ for AWQ. **Bold** and underline denote the best and second best, respectively. Asterisk "*" indicates reaching competitive performance to the original un-compressed LLM.

Quantization q	2 bits	3 bits	4 bits	5 bits
Gemma3-270M (WT2: 70.1, PTB: 698.7, C4: 68.1, Avg: **279.0**$_{\pm 353.5}$)				
RTN	2.6e11$_{\pm 1.2e11}$	1795.0$_{\pm 2207.5}$	391.9$_{\pm 505.5}$	315.0$_{\pm 410.9}$
AWQ (WT2 Calib)	89188.4$_{\pm 58307.2}$	517.4$_{\pm 690.0}$	279.6$_{\pm 348.6}$	306.0$_{\pm 405.2}$
AWQ (PTB Calib)	1.9e5$_{\pm 2.0e9}$	537.7$_{\pm 711.8}$	310.0$_{\pm 408.2}$	293.4$_{\pm 386.4}$
AWQ (C4 Calib)	1.3e5$_{\pm 1.6e5}$	598.1$_{\pm 832.9}$	326.4$_{\pm 420.5}$	*276.8$_{\pm 358.0}$
TTQ ($r = 0$)	30199.0$_{\pm 23705.9}$	387.7$_{\pm 489.0}$	*277.0$_{\pm 357.1}$	*262.5$_{\pm 335.2}$
TTQ ($r = 16$)	**19657.0**$_{\pm 17934.2}$	**382.8**$_{\pm 486.2}$	***275.7**$_{\pm 353.6}$	***261.5**$_{\pm 343.1}$
Gemma3-1B (WT2: 27.7, PTB: 212.4, C4: 33.2, Avg: **91.1**$_{\pm 105.1}$)				
RTN	8.6e5$_{\pm 7.7e5}$	209.4$_{\pm 253.2}$	111.1$_{\pm 130.0}$	96.6$_{\pm 112.0}$
AWQ (WT2 Calib)	4734.7$_{\pm 5394.7}$	134.7$_{\pm 159.3}$	91.9$_{\pm 104.1}$	95.9$_{\pm 112.2}$
AWQ (PTB Calib)	9326.6$_{\pm 11932.8}$	138.9$_{\pm 167.5}$	99.2$_{\pm 115.1}$	95.5$_{\pm 112.0}$
AWQ (C4 Calib)	5486.9$_{\pm 6662.0}$	150.8$_{\pm 186.3}$	93.5$_{\pm 106.0}$	93.6$_{\pm 108.4}$
TTQ ($r = 0$)	1928.5$_{\pm 2313.2}$	127.3$_{\pm 150.5}$	***89.9**$_{\pm 100.8}$	***90.2**$_{\pm 103.3}$
TTQ ($r = 16$)	**1804.9**$_{\pm 1850.3}$	**114.5**$_{\pm 130.8}$	91.7$_{\pm 107.4}$	*90.3$_{\pm 103.6}$
Gemma3-4B (WT2: 17.4, PTB: 641.5, C4: 23.3, Avg: **227.4**$_{\pm 358.6}$)				
RTN	8.1e6$_{\pm 1.4e7}$	606.4$_{\pm 997.2}$	251.0$_{\pm 395.1}$	257.8$_{\pm 410.5}$
AWQ (WT2 Calib)	5667.3$_{\pm 9506.0}$	351.7$_{\pm 565.7}$	*225.3$_{\pm 353.5}$	*221.9$_{\pm 348.7}$
AWQ (PTB Calib)	4444.0$_{\pm 7433.4}$	352.9$_{\pm 571.5}$	240.4$_{\pm 380.6}$	240.5$_{\pm 380.9}$
AWQ (C4 Calib)	4938.5$_{\pm 8259.2}$	366.3$_{\pm 593.1}$	245.6$_{\pm 388.9}$	245.8$_{\pm 389.7}$
TTQ ($r = 0$)	**3561.6**$_{\pm 6003.4}$	**239.5**$_{\pm 375.8}$	*222.2$_{\pm 349.4}$	***221.7**$_{\pm 349.1}$
TTQ ($r = 16$)	3974.9$_{\pm 6696.8}$	251.7$_{\pm 396.6}$	***222.0**$_{\pm 349.1}$	*221.9$_{\pm 353.6}$
Gemma3-12B (WT2: 25.1, PTB: 283.8, C4: 35.0, Avg: **114.6**$_{\pm 146.6}$)				
RTN	15467.0$_{\pm 13512.6}$	294.9$_{\pm 390.2}$	146.6$_{\pm 196.1}$	*108.3$_{\pm 136.5}$
AWQ (WT2 Calib)	1700.6$_{\pm 2328.9}$	147.4$_{\pm 196.4}$	125.9$_{\pm 166.1}$	114.7$_{\pm 149.9}$
AWQ (PTB Calib)	4193.8$_{\pm 6142.5}$	128.3$_{\pm 164.6}$	117.3$_{\pm 146.2}$	122.3$_{\pm 157.6}$
AWQ (C4 Calib)	1464.4$_{\pm 2003.8}$	*100.8$_{\pm 122.2}$	146.0$_{\pm 197.0}$	132.0$_{\pm 173.5}$
TTQ ($r = 0$)	**1295.6**$_{\pm 1743.4}$	***84.6**$_{\pm 94.1}$	***86.6**$_{\pm 102.3}$	***92.9**$_{\pm 112.4}$
TTQ ($r = 16$)	1620.9$_{\pm 2248.4}$	*100.7$_{\pm 116.9}$	*91.2$_{\pm 109.4}$	*98.3$_{\pm 121.1}$

Table 12: Accuracy (↑) of Qwen3-VL models with different quantization methods on TextVQA benchmark. Groupsize is g = 32 for all cases. **Bold** and underline denote the best and second best, respectively. Asterisk "*" indicates reaching competitive performance to the original un-compressed VLM.

Quantization q	2 bits	3 bits	4 bits	5 bits
Qwen3-VL-2B: Acc 80.35%				
RTN	0.00%	9.23%	73.27%	80.22%
AWQ (COCO-Cap Calib)	1.41%	75.67%	79.04%	79.89%
AWQ (OK-VQA Calib)	1.37%	74.68%	79.47%	79.80%
AWQ (ChartQA Calib)	0.67%	75.52%	79.29%	79.80%
AWQ (TextVQA Calib)	0.82%	72.15%	78.85%	80.21%
TTQ ($r = 0$)	1.42%	**76.59%**	79.59%	80.34%
TTQ ($r = 16$)	**1.93%**	76.01%	**79.67%**	***80.38%**
Qwen3-VL-4B: Acc 81.44%				
RTN	0.03%	74.79%	80.77%	*81.47%
AWQ (COCO-Cap Calib)	0.13%	78.68%	80.49%	81.20%
AWQ (OK-VQA Calib)	0.18%	77.89%	80.58%	80.75%
AWQ (ChartQA Calib)	0.42%	77.37%	80.67%	*81.47%
AWQ (TextVQA Calib)	0.25%	77.49%	80.28%	81.01%
TTQ ($r = 0$)	1.51%	**79.93%**	**81.29%**	*81.48%
TTQ ($r = 16$)	**7.47%**	79.45%	81.22%	***81.49%**
Qwen3-VL-8B: Acc 81.72%				
RTN	0.17%	78.51%	80.63%	*81.73%
AWQ (COCO-Cap Calib)	41.39%	80.37%	81.52%	*81.79%
AWQ (OK-VQA Calib)	31.62%	79.51%	81.01%	81.55%
AWQ (ChartQA Calib)	41.17%	78.65%	81.23%	81.39%
AWQ (TextVQA Calib)	39.60%	78.43%	81.25%	81.69%
TTQ ($r = 0$)	42.20%	80.77%	81.57%	*81.81%
TTQ ($r = 16$)	**47.22%**	**81.04%**	**81.71%**	***81.85%**

Table 13: Success rate (↑) of $\pi_{0.5}$ VLA model with different quantization methods on LIBERO robot manipulation benchmark. We use $q = 2$ bits and $g = 64$ groupsize. **Bold** and underline denote the best and second best, respectively.

Benthmark	Libero Spatial	Libero Object	Libero Goal	Libero 10	Avg
BF16	97.5%	100.0%	97.0%	96.5%	97.75%
RTN	57.0%	65.0%	27.5%	2.0%	37.88%
AWQ (Spatial Calib)	90.5%	**100.0%**	85.0%	82.0%	89.34%
AWQ (Object Calib)	91.5%	98.5%	92.0%	78.0%	90.00%
AWQ (Goal Calib)	92.5%	**100.0%**	**93.5%**	84.5%	92.63%
AWQ (10 Calib)	94.0%	99.5%	92.5%	76.5%	90.63%
TTQ ($r = 0$)	93.0%	99.5%	**93.5%**	87.0%	93.25%
TTQ ($r = 16$)	**94.5%**	**100.0%**	**93.5%**	**87.5%**	**93.88%**

ChartQA The ChartQA dataset Masry et al. (2022) is a benchmark for chart-based visual question answering, designed to evaluate models' ability to interpret charts and perform numerical and logical reasoning. It contains 33k question–answer pairs over 21k charts collected from multiple real-world sources. The dataset includes both human-written and automatically generated questions, requiring models to extract information or compute values from chart data. We use `https://huggingface.co/datasets/llms-lab/ChartQA`, released under the GPL-3.0 license.

Table 14: OPT model parameters Zhang et al. (2022)

Model	layers	heads	hidden size	head dim	MLP dim	Huggingface ID
125M	12	12	768	64	3072	facebook/opt-125m
350M	24	16	1024	64	4096	facebook/opt-350m
1.3B	24	32	2048	64	8192	facebook/opt-1.3b
2.7B	32	32	2560	80	10240	facebook/opt-2.7b
6.7B	32	32	4096	128	16384	facebook/opt-6.7b
13B	40	40	5120	128	20480	facebook/opt-13b
30B	48	56	7168	128	28672	facebook/opt-30b
66B	64	72	9216	128	36864	facebook/opt-66b

Table 15: Architecture parameters of Qwen3 dense models Yang et al. (2025)

Model	layers	heads	KV heads	hidden size	head dim	MLP dim	Huggingface ID
0.6B	28	16	8	1024	128	3072	Qwen/Qwen3-0.6B
1.7B	28	16	8	2048	128	6144	Qwen/Qwen3-1.7B
4B	36	32	8	2560	128	9728	Qwen/Qwen3-4B
8B	36	32	8	4096	128	12288	Qwen/Qwen3-8B
14B	40	40	8	5120	128	17408	Qwen/Qwen3-14B
32B	64	64	8	5120	128	25600	Qwen/Qwen3-32B

LIBERO The LIBERO dataset Liu et al. (2023a) is a benchmark for robotic vision-language-action (VLA) learning that evaluates long-horizon, compositional manipulation in simulated household environments. It provides diverse task suites with structured train/test splits designed to measure cross-task generalization, skill composition, and transfer to novel object and scene configurations. Each task includes multimodal data—video observations, language instructions, and low-level control trajectories—enabling end-to-end learning from vision and language to robot actions. We use `https://huggingface.co/datasets/lerobot/libero`, licensed under Apache-2.0.

Table 16: Gemma3 instruction-tuned text transformer parameters Gemma Team et al. (2025)

Model	layers	heads	KV heads	hidden size	head dim	MLP dim	Huggingface ID
270M	18	4	1	640	256	2048	google/gemma-3-270m-it
1B	26	4	1	1152	256	6912	google/gemma-3-1b-it
4B	34	8	4	2560	256	10240	google/gemma-3-4b-it
12B	48	16	8	3840	256	15360	google/gemma-3-12b-it
27B	62	32	16	5376	256	21504	google/gemma-3-27b-it

Table 17: Qwen3-VL transformer parameters Bai et al. (2025)

Model	layers	heads	KV heads	hidden size	head dim	MLP dim	Hugginface ID
2B	24	16	8	2048	128	11008	Qwen/Qwen3-VL-2B-Instruct
4B	32	32	8	4096	128	22016	Qwen/Qwen3-VL-4B-Instruct
8B	36	32	8	4096	128	22016	Qwen/Qwen3-VL-8B-Instruct
32B	64	40	8	5120	128	27392	Qwen/Qwen3-VL-32B-Instruct

Table 18: $\pi_{0.5}$ VLA transformer parameters Intelligence et al. (2025): Huggingface ID lerobot/pi05_libero_finetuned

Module	layers	heads	KV heads	hidden size	head dim	MLP dim
LLM: Gemma-2B	18	8	1	2048	256	16384
Vision Encoder: SigLIP ViT-L	24	16	16	1024	64	4096
Diffusion Policy: Gemma-300M	18	8	1	1024	128	4096