Post-Training Ternarization of Qwen3 Language Models

Capability, Effective Bit Budget, Storage Compression, and Inference

CLOE V1.2 - OneBit AI

Qwen3-4B $\xrightarrow{\text{KOTMS}}$ $\xrightarrow{\text{E2M-ATQ}}$ $\xrightarrow{\text{GPTQ}}$ Ternary Qwen3-4B

A Capability, Representation, and Deployment Study of a 4B-Parameter Pretrained Language Model Under Aggressive Ternary Quantisation

Anirudh Malik | Poojith Devan | M Sparsh Mehra
OneBit AI | Technical Report | August 2026

Abstract

Ultra-low-bit weight representations promise large reductions in model memory and storage, but the practical behavior of aggressively ternarizing an already-trained language model is not fully captured by a nominal bit-width. In this work, we experimentally characterize an end-to-end post-training ternarization pipeline for Qwen3, combining **existing** KOTMS weight rotation, E2M-ATQ ternarization, and GPTQ-style error compensation. We explicitly separate these prior components from our contribution: **the empirical system study, the effective bit-budget accounting, the model-scale analysis, and the deployment measurements**.

Across Qwen3-0.6B and Qwen3-4B, we observe a strong scale dependence. The 0.6B model degrades sharply under the same aggressive post-training procedure, whereas the 4B model retains substantial downstream capability. For Qwen3-4B, 22.55% of weights receive a second ternary component, yielding an effective representation of approximately **1.94 bits/weight**, rather than the idealized single-plane ternary information content of $\log_2 3 \approx 1.585$ bits. The complete packed checkpoint falls from approximately **8.90 GB to 2.66 GB**, corresponding to **3.35× overall compression**.

We also measure the gap between **representation efficiency and execution efficiency**. Reconstruction-based inference achieves approximately 21.28 tokens/s versus 27.41 tokens/s for FP16, while a preliminary packed Triton GEMV kernel is approximately 4.6× slower than the corresponding cuBLAS microbenchmark. These results do not show that ternary execution is intrinsically slower than FP16; rather, they show that **compression alone does not provide a hardware-efficient inference path**. The resulting study provides a concrete empirical baseline for future work on native ternary kernels, hardware-aware packing, and scale-dependent ternarization robustness.

What is new in this report? We do *not* claim KOTMS, E2M-ATQ, GPTQ, ternary quantization, or the underlying 1.58-bit idea as inventions. Those components are prior art. Our contribution is to **build, run, and quantitatively characterize the complete post-training pipeline on Qwen3**, with measurements spanning capability retention, effective bits/weight, checkpoint compression, scale sensitivity, and inference behavior.

Central empirical finding. For Qwen3-4B, aggressive post-training ternarization reaches approximately **1.94 effective bits/weight** and **3.35× smaller total checkpoint** while retaining substantial benchmark capability. At the same time, the current execution path is slower than FP16, exposing a separation between **compressing a model** and **executing the compressed representation efficiently**.

1 Introduction

The rapid growth of large language models has created challenges in model storage, memory consumption, and inference cost. A modern language model may contain billions of parameters, making conventional FP16 storage expensive even before accounting for inference compute.

Weight quantisation provides an important route to reducing these costs, motivating extensive work on low-bit and post-training compression for LLMs [3, 4]. Conventional low-bit approaches represent weights using 8-bit, 4-bit, or other reduced-precision numerical formats. More aggressive approaches restrict weights to a small discrete set of values. Among these, ternary representations are particularly interesting because weights can be represented using compact discrete values, as explored by ternary architectures such as BitNet [2] and recent post-training ternary methods [6, 5].

$$w \in \{-1, 0, +1\}. \tag{1}$$

The theoretical information content of a ternary variable is

$$b_T = \log_2(3) \approx 1.585 \tag{2}$$

bits. This observation motivates research into approximately 1.58-bit language models.

However, the theoretical storage requirement of a ternary symbol does not automatically translate into an equivalent practical representation or faster inference. Scaling factors, auxiliary metadata, residual representations, packing overhead, and hardware execution all affect the final system.

In this work, we investigate these issues through an experimental study of Qwen3 language models. Rather than retraining an LLM from scratch, we begin with pretrained FP16 checkpoints and apply a post-training pipeline designed to make the weight distributions amenable to ternarization.

Our pipeline combines:

1. KOTMS-based weight rotation;
2. E2M-ATQ ternarization; and
3. GPTQ-style error compensation.

We investigate Qwen3-0.6B and Qwen3-4B, two dense models from the Qwen3 family [1], and evaluate the resulting models using perplexity and multiple-choice language-model benchmarks.

A central question is whether compact ternary representations actually result in faster inference. Our experiments show that they do not automatically do so: reconstruction-based inference is slower than FP16 inference, and a preliminary packed Triton kernel also underperforms conventional FP16 GEMV.

Our main contributions are:

1. **End-to-end empirical evaluation.** We run the complete post-training pipeline on Qwen3-0.6B and Qwen3-4B rather than presenting the method only at the algorithmic level.
2. **Effective representation accounting.** We quantify how often the adaptive second ternary component is invoked and convert that measurement into an effective bits/weight figure.
3. **Scale-dependent capability analysis.** We compare the same aggressive procedure across 0.6B and 4B models and show a large difference in robustness.
4. **Deployment-oriented measurement.** We report actual checkpoint sizes, reconstruction-based generation throughput, and a preliminary packed Triton kernel benchmark.
5. **A clearly scoped research baseline.** We identify the remaining systems bottleneck—native hardware-efficient ternary execution—without conflating representational compression with speedup.

> **Attribution boundary.** The rotation and ternarization machinery used here is based on prior work, particularly KOTMS and E2M-ATQ as described in TWLA [6], while GPTQ is an established second-order PTQ method [3]. The novelty claimed here is therefore **empirical and system-level**, not the invention of these component algorithms.

2 Background

2.1 Ternary Quantization

In FP16 inference, each weight is represented using a 16-bit floating-point value. Ternary quantization instead restricts weights to

$$T \in \{-1, 0, +1\}. \tag{3}$$

A simple ternary approximation can be written as

$$W \approx \alpha T, \tag{4}$$

where T is a ternary matrix and α is a scale.

The theoretical information content of each ternary symbol is

$$b_T = \log_2(3) \approx 1.585. \tag{5}$$

This is the origin of the commonly discussed "1.58-bit" representation. Practical systems, however, can require additional information beyond a single ternary symbol per weight.

2.2 Post-Training Quantization

Post-training quantization (PTQ) converts a pretrained model into a lower-precision representation without full model retraining. This is attractive for LLMs because retraining billions of parameters can be prohibitively expensive.

The basic objective is

$$\widehat{W} = Q(W), \tag{6}$$

while minimizing the resulting model error. At extremely low precision, the quantization error can be substantially larger than in 4-bit or 8-bit quantization, motivating weight transformations and error compensation.

3 Related Work

The work sits at the intersection of three lines of research. **Training-time ternary models** such as BitNet demonstrate that extreme low-bit representations can be viable when the model is designed or trained for them [2]. **Post-training quantization** methods such as GPTQ and AWQ instead seek to compress an already-trained model while controlling accuracy loss [3, 4]. Finally, recent research has pushed PTQ directly into the ternary regime, including structured ternary planes and KOTMS/E2M-ATQ-based approaches [5, 6, 7].

The distinction is important for interpreting our experiments. We do not retrain Qwen3 into a ternary-native architecture. Instead, we ask how far an existing pretrained model can be pushed through an aggressive *post-training* pipeline, and what practical costs remain once the resulting representation is measured end-to-end.

4 Method

The experimental pipeline is

$$\text{FP16 Qwen3} \rightarrow \text{KOTMS rotation} \rightarrow \text{E2M-ATQ} \rightarrow \text{GPTQ} \rightarrow \text{ternary model}. \tag{7}$$

4.1 KOTMS Weight Rotation

The first stage applies the KOTMS rotation described in prior ternary PTQ work [6]. Conceptually, an orthogonal transformation is applied to the weight matrices:

$$W' = WR, \tag{8}$$

where R satisfies

$$R^\top R = I. \tag{9}$$

The purpose of the rotation is to transform the weight space into a coordinate system that is more amenable to discrete ternary approximation.

4.2 E2M-ATQ Ternarization

Following rotation, the weight matrix is approximated using ternary values using the E2M-ATQ procedure from prior work [6]. For a row of weights, we center the weights:

$$W_c = W - \mu. \tag{10}$$

The threshold used in the experiment is

$$\Delta = 0.75 \cdot \text{mean}(|W_c|). \tag{11}$$

The ternary assignment is

$$T_i = \begin{cases} +1, & W_{c,i} > \Delta, \\ -1, & W_{c,i} < -\Delta, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

A scale is then estimated to minimize the approximation error between the original and ternary representations.

4.3 Adaptive Second Ternary Representation

A single ternary representation is not sufficient for every weight. The representation can therefore be written approximately as

$$W \approx \mu + \alpha_0 T_0 + \alpha_1 T_1, \tag{13}$$

where $T_0, T_1 \in \{-1, 0, +1\}$.

The second ternary component is applied selectively. In the Qwen3-4B experiment, approximately 22.55% of weights received the additional ternary representation.

Thus the average number of ternary digits per weight is

$$1 + 0.2255 = 1.2255. \tag{14}$$

The corresponding effective information budget is

$$B_{\text{eff}} = 1.2255 \log_2(3) \approx 1.94 \tag{15}$$

bits per weight.

This distinction is important: 1.58 bits describes the information content of one ternary symbol, whereas the adaptive representation used in this experiment has a higher effective weight-information budget because some weights receive an additional ternary symbol.

4.4 GPTQ Error Compensation

After ternarization, GPTQ-style compensation is used to reduce quantization error, following the error-compensation paradigm introduced by GPTQ [3]. The method accounts for the sensitivity of the network to weight perturbations rather than treating each weight independently.

Calibration in the experiment uses WikiText-2.

5 Experimental Setup

5.1 Research Questions

We organize the experiments around four concrete questions:

RQ1. Capability: How much downstream capability survives aggressive PTQ ternarization, and does robustness depend on model scale?

RQ2. Representation: What is the *actual* effective bits/weight once adaptive second-plane usage is accounted for?

RQ3. Storage: How much smaller is the complete checkpoint in practice, including tensors that are not ternarized in exactly the same way?

RQ4. Execution: Does a compact ternary representation automatically yield faster inference on commodity GPU hardware?

We evaluate:

- Qwen3-0.6B;
- Qwen3-4B.

Experiments were conducted on a single NVIDIA RTX 5070 with 12 GB VRAM. This is reported as the compute environment for the study, not as a claim that the hardware itself is a scientific contribution.

We evaluate language modeling using WikiText-2 perplexity and downstream capability using:

- BoolQ;
- PIQA;
- HellaSwag;
- WinoGrande;
- ARC-Challenge;
- MMLU; and
- GSM8K.

Where applicable, FP16 checkpoints are used as the reference baseline.

6 Results

6.1 Qwen3-0.6B

The initial Qwen3-0.6B experiment showed substantial degradation. The FP16 model obtained a WikiText-2 perplexity of approximately

$$PPL = 20.95, \tag{16}$$

while the ternarized model reached

$$PPL = 55.17. \tag{17}$$

Average task performance decreased from approximately

$$0.480 \rightarrow 0.336. \tag{18}$$

This is an absolute drop of approximately 0.144 score points, or about **30% relative to the FP16 score**. Three of the evaluated tasks reached approximately chance-level performance. This experiment demonstrated that the aggressive PTQ procedure can severely damage a small language model.

6.2 Iteration Study

Increasing KOTMS optimization iterations from 20 to 100 produced approximately a 15.7% reduction in perplexity. However, the corresponding QA improvement was only approximately

$$0.0012. \tag{19}$$

This suggests that improved perplexity does not necessarily translate into meaningful downstream capability recovery. Downstream benchmarks are therefore treated as an important complement to perplexity.

Table 1: Headline empirical findings of this study.

Dimension	Measured result	Interpretation
Capability, 0.6B	0.480 → 0.336 average score	Severe degradation under aggressive PTQ
Capability, 4B	0.652 → 0.565 average score	Substantial capability remains
Effective representation	1.94 bits/weight	Above the ideal single-plane ternary limit
Second ternary component	22.55% of weights	Selective adaptive capacity
Checkpoint size	8.90 → 2.66 GB	3.35× total compression
Generation throughput	27.41 → 21.28 tok/s	Compression does not imply speedup
Packed GEMV kernel	0.045 → 0.208 ms	Preliminary kernel is ∼ 4.6× slower

Table 2: Selected Qwen3-4B ternarization results.

Benchmark	Ternary	Retention/Context
MMLU	51.0% (0-shot)	55.6% (5-shot)
GSM8K	48.1% (5-shot)	–
BoolQ	80.76%	–
PIQA	68.9%	–
WinoGrande	61.56%	–
HellaSwag	56.0%	–
ARC-Challenge	38.57%	–

6.3 Qwen3-4B

The larger Qwen3-4B model behaved substantially better.

The FP16 model achieved an average QA score of approximately

$$0.652, \tag{20}$$

while the ternarized model achieved approximately

$$0.565. \tag{21}$$

The absolute average-score drop is approximately 0.087, corresponding to about **13.3% relative degradation** from the FP16 baseline. None of the evaluated tasks collapsed to chance level.

Perplexity increased from approximately

$$13.64 \rightarrow 18.06. \tag{22}$$

6.4 Capability Retention

To distinguish absolute accuracy from retained above-chance capability, we use

$$R = \frac{A_{\text{ternary}} - A_{\text{chance}}}{A_{\text{FP16}} - A_{\text{chance}}}. \tag{23}$$

For the Qwen3-4B experiment, the observed pattern indicates that contextual and commonsense tasks are generally more robust than knowledge-intensive examination tasks. For example, the measured BoolQ retention was approximately 84.6%, while ARC-Challenge retention was approximately 43.8% and MMLU retention was approximately 59.7%.

6.5 Few-Shot Evaluation

Few-shot prompting improves the measured performance of the ternary Qwen3-4B model. MMLU increases from approximately 51.0% in the 0-shot setting to 55.6% with five-shot prompting.

Table 3: Illustrative task-level capability retention for Qwen3-4B.

Task	Ternary Score	Approx. Retention
BoolQ	81.4%	84.6%
PIQA	68.9%	∼74%
HellaSwag	56.0%	∼81%
WinoGrande	61.6%	∼72%
ARC-Challenge	38.6%	43.8%
MMLU	51.0%	59.7%

The five-shot GSM8K result is approximately 48.1%.

These results indicate that the quantized model retains enough underlying capability to benefit from contextual examples, although few-shot prompting does not restore the full FP16 performance.

7 Storage Efficiency

The packed representation stores multiple ternary values per byte.

For the linear weights, storage decreased from approximately

$$7.27\,\text{GB} \tag{24}$$

to

$$1.02\,\text{GB}, \tag{25}$$

corresponding to approximately 7.11× compression for the linear weights.

For the complete checkpoint, storage decreased from approximately

$$8.90\,\text{GB} \tag{26}$$

to

$$2.66\,\text{GB}, \tag{27}$$

corresponding to approximately 3.35× overall checkpoint compression.

The difference between linear-layer compression and total-checkpoint compression arises because not all tensors are represented by the same ternary format.

> **What the storage result establishes.** The practical outcome is not a claim of a theoretical 1.58-bit checkpoint. Instead, the measured adaptive representation is approximately **1.94 bits/weight**, and the complete model artifact is **3.35× smaller**. This distinction matters for reproducible reporting because auxiliary tensors, scales, and non-ternarized parameters contribute to the final file size.

8 Inference Performance

Storage compression does not automatically imply faster inference.

Our initial inference implementation reconstructed FP16 weights before executing standard FP16 matrix multiplication:

$$\text{packed ternary} \rightarrow \text{FP16 reconstruction} \rightarrow \text{FP16 GEMM}. \tag{28}$$

Measured generation performance was approximately:

$$27.41\ \text{tok/s} \tag{29}$$

Table 4: Attribution boundary of the study.

Component	Status	How it is used here
Ternary weights / 1.58-bit information content	Prior concept [2]	Representation target and theoretical reference point
KOTMS rotation	Prior method [6]	Weight transformation before ternarization
E2M-ATQ	Prior method [6]	Ternary weight construction
GPTQ	Prior method [3]	Post-quantization error compensation
Adaptive second ternary component	**Measured in this study**	Effective-bit accounting for the deployed representation
Qwen3-0.6B vs. 4B scale study	**Measured in this study**	Empirical analysis of robustness to aggressive PTQ
Checkpoint compression	**Measured in this study**	End-to-end storage accounting
Reconstruction and Triton inference measurements	**Measured in this study**	Deployment characterization and identification of the execution bottleneck

for FP16 and

$$21.28 \text{ tok/s} \tag{30}$$

for the ternary implementation.

Corresponding token latency was approximately 36.49 ms/token for FP16 and 46.98 ms/token for the ternary implementation.

Prefill throughput was approximately 5919 tokens/s for FP16 and 5029 tokens/s for the ternary implementation.

> **Interpretation.** These results show that the current implementation does not yet convert representational compression into an inference-speed advantage. They should *not* be interpreted as proof that ternary arithmetic is intrinsically slower than FP16; the benchmark includes reconstruction and a preliminary kernel implementation.

8.1 Packed Triton Kernel

To investigate direct execution of the packed representation, we implemented a preliminary Triton packed-GEMV kernel.

The measured microbenchmark was:

$$t_{\text{cuBLAS}} = 0.045\,\text{ms}, \tag{31}$$

versus

$$t_{\text{Triton}} = 0.208\,\text{ms}. \tag{32}$$

Thus, the preliminary packed kernel was approximately 4.6× slower in this configuration.

This should not be interpreted as evidence that ternary arithmetic is fundamentally slower than FP16. Instead, it demonstrates that the present implementation does not yet provide a hardware-efficient execution path for packed ternary weights.

9 Novelty and Attribution

A central purpose of this report is to make the scientific boundary of the work explicit. Table 4 separates **prior methods we use or build upon** from the **measurements and analysis contributed by this study**.

Table 5: Recommended comparison set for future controlled evaluation. Results should be populated only after running all models through the same evaluation protocol.

Model	Parameters	Representation	Role	Status
Qwen3-4B	4B	FP16	Primary baseline	Evaluated
Qwen3-4B (Ours)	4B	∼1.94-bit	Proposed model	Evaluated
Qwen3-4B CAT-Q	4B	1.58-bit	Ternary baseline	Literature baseline
Qwen3-4B TWLA	4B	1.58-bit	PTQ baseline	Literature baseline
Qwen3-1.7B	1.7B	FP16	Smaller dense model	Future evaluation
Qwen3-0.6B	0.6B	FP16	Small-model baseline	Evaluated
Llama 3.2 3B	3B	FP16	Open-weight baseline	Future evaluation
Gemma 3 4B	4B	FP16	Same-scale baseline	Future evaluation
Phi-3.5-mini	3.8B	FP16	Small-model baseline	Future evaluation
SmolLM2 1.7B	1.7B	FP16	Edge-oriented baseline	Future evaluation

This positioning is deliberately conservative. Recent work such as CAT-Q has also advanced post-training ternarization directly, while TWLA explicitly combines KOTMS and E2M-ATQ in a broader W1.58A4 pipeline [7, 6]. Our claim is therefore not algorithmic novelty over these methods. The value of the present study is the **controlled end-to-end characterization of what happens when this family of techniques is applied to Qwen3, including the practical representation and deployment consequences that are easy to miss when reporting only a nominal bit-width**.

10 Comparative Evaluation

To contextualize the result, we list natural external comparison targets. These comparisons are **not used as primary evidence for our claims unless the same evaluation protocol is rerun end-to-end**.

The most scientifically controlled comparison is Qwen3-4B FP16 versus the proposed Qwen3-4B ternary model because architecture and pretrained weights remain fixed. Comparisons with other 4B-class models provide a broader deployment perspective, while smaller dense models enable analysis of the capability-to-memory trade-off.

10.1 Intelligence per Memory

For memory-constrained deployment, we propose reporting benchmark performance together with actual model footprint.

A simple empirical metric is

$$I_{\mathrm{GB}} = \frac{S}{M}, \tag{33}$$

where S is an aggregate benchmark score and M is the model memory footprint in GB.

This quantity is not intended as a universal measure of intelligence. Instead, it provides a practical deployment-oriented comparison: whether a larger but heavily quantized model can provide more useful capability per unit of memory than a smaller dense model.

11 Discussion

The experiments reveal three distinct dimensions of low-bit LLM compression:

1. **Model capability:** Qwen3-4B remains functional after aggressive ternarization, although performance decreases.

2. **Storage efficiency:** the packed representation substantially reduces checkpoint size.
3. **Inference efficiency:** the current implementation does not yet provide a speed advantage.

These dimensions should not be conflated:

$$\text{Representation Efficiency} \neq \text{Inference Efficiency}. \tag{34}$$

A model can be compact in storage while remaining inefficient to execute if the hardware and kernels are optimized for conventional dense floating-point operations.

11.1 Effect of Model Scale

The difference between Qwen3-0.6B and Qwen3-4B is one of the clearest findings of this study.

The 0.6B model experiences severe degradation, including multiple tasks approaching chance level. The 4B model, in contrast, retains substantial performance across all evaluated tasks.

A possible interpretation is that larger models may contain greater redundancy in their learned representations, making them more resilient to aggressive weight perturbations. However, this should remain an empirical hypothesis until more model scales and architectures are evaluated.

11.2 Theoretical versus Practical Bit Rate

A major lesson is the importance of distinguishing theoretical ternary information content from the effective representation.

A single ternary variable requires

$$\log_2(3) = 1.585 \tag{35}$$

bits theoretically.

The adaptive representation used here applies a second ternary component to approximately 22.55% of weights, producing

$$1.2255 \log_2(3) \approx 1.94 \tag{36}$$

bits per weight.

Additional metadata and non-quantized tensors further affect complete checkpoint size.

Consequently, “1.58-bit model” should be interpreted carefully unless the complete storage format and all auxiliary information are included in the calculation.

12 Limitations

This study has several limitations.

First, the evaluation focuses primarily on Qwen3-0.6B and Qwen3-4B. Additional model sizes and architectures are required before making broad claims about scaling behavior.

Second, the current inference implementation does not provide a fully optimized native ternary GPU execution path.

Third, the packed Triton kernel is a preliminary implementation and should not be considered a definitive benchmark of ternary hardware efficiency.

Fourth, the evaluation uses a limited collection of benchmarks compared with the full evaluation suites used in large-scale LLM studies.

Finally, the experiments evaluate post-training ternarization rather than full ternary-aware pretraining or extensive quantization-aware retraining.

13 Conclusion

We presented an empirical investigation of aggressive post-training ternarization for Qwen3 language models. The study is best viewed as a **capability–representation–deployment characterization**, rather than as a claim of a new ternarization algorithm.

The experimental pipeline combines KOTMS weight rotation, E2M-ATQ ternarization, and GPTQ error compensation. The results reveal a strong dependence on model scale: Qwen3-0.6B experiences severe degradation, whereas Qwen3-4B retains substantial language capabilities despite aggressive weight discretization.

For Qwen3-4B, the adaptive ternary representation corresponds to an effective weight information budget of approximately 1.94 bits per weight. The resulting packed representation provides substantial storage reduction, including approximately 3.35× reduction in total checkpoint size.

However, compression does not automatically result in faster inference. The reconstruction-based implementation is slower than FP16 inference, while the preliminary packed Triton kernel also underperforms conventional FP16 GEMV.

The results therefore suggest that future work should treat model compression and hardware-efficient inference as separate optimization problems. An important direction is the development of native ternary matrix multiplication kernels capable of directly exploiting packed ternary representations.

More broadly, these experiments support three conclusions:

1. **Scale matters:** the 4B model is materially more robust than the 0.6B model under the same aggressive PTQ procedure.
2. **The effective representation matters:** the deployed adaptive format is approximately 1.94 bits/weight, not simply the idealized 1.585-bit ternary symbol rate.
3. **Execution is a separate problem:** storage compression is already substantial, but native hardware-efficient ternary computation remains the key systems bottleneck.

The immediate research opportunity is therefore clear: preserve the capability and compression characteristics measured here while replacing reconstruction-based execution with kernels and hardware pathways that consume packed ternary weights directly.

Reproducibility Note

All quantitative results reported in this document correspond to the experiments described in Sections 6.1–6.3, using the stated Qwen3 checkpoints, a single NVIDIA RTX 5070 with 12 GB VRAM, WikiText-2 calibration, and the benchmark suite listed in the experimental setup. The reported storage figures distinguish linear-layer storage from complete-checkpoint storage, and the inference results distinguish reconstruction-based execution from the preliminary packed Triton kernel. We recommend preserving these distinctions in any subsequent reproduction or comparison.

References

[1] An Yang et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

[2] Hongyu Wang et al. BitNet: Scaling 1-bit Transformers for Large Language Models. *arXiv preprint arXiv:2310.11453*, 2023.

[3] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. In *International Conference on Learning Representations*, 2023.

[4] Ji Lin et al. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, 2024.

[5] He Xiao et al. PTQTP: Post-Training Quantization to Trit-Planes for Large Language Models. *arXiv preprint arXiv:2509.16989*, 2025.

[6] Zhixiong Zhao et al. TWLA: Achieving Ternary Weights and Low-Bit Activations for LLMs via Post-Training Quantization. *arXiv preprint arXiv:2606.13054*, 2026.

[7] Shigeng Wang, Chao Li, Yangyuxuan Kang, Jiawei Fan, and Anbang Yao. CAT-Q: Cost-efficient and Accurate Ternary Quantization for LLMs. In *International Conference on Machine Learning*, 2026.