Quantizing Llama 3.2-1B for Efficient Inference: A Systematic Study of Bit-Width Reduction on CoQA

Hemanto Bairagi
McGill University Interview Take-Home Assignment
January 2026

Abstract

We investigate post-training quantization techniques for the Llama 3.2-1B language model, systematically evaluating the trade-off between model compression and task accuracy on the Conversational Question Answering (CoQA) benchmark. Through controlled ablation studies comparing 4-bit quantization formats (NF4 and FP4), double quantization, and compute precision configurations, we demonstrate that 4-bit NF4 quantization achieves 2.44× memory compression while preserving baseline accuracy. Our experiments reveal that quantization format selection significantly impacts performance: NF4 outperforms FP4 by 9.5 percentage points in F1 score at equivalent memory cost, attributable to NF4's distribution-aware quantization levels. Additionally, we find that double quantization provides further compression without measurable accuracy degradation. These findings inform practical deployment guidelines for memory-constrained inference environments.

1 Introduction

Large language models (LLMs) have achieved remarkable performance across diverse natural language tasks [15, 16]. However, their deployment remains constrained by substantial memory and computational requirements [8]. A model with one billion parameters stored in 16-bit floating-point precision requires approximately 2.4 GB of memory for weights alone, limiting deployment on resource-constrained devices and increasing inference costs in production environments.

Post-training quantization offers a principled approach to this challenge by reducing the numerical precision of model weights [6, 11]. By representing weights with fewer bits, quantization can substantially reduce memory requirements. However, aggressive precision reduction risks degrading model accuracy, necessitating careful empirical study of this trade-off [19].

1.1 Contributions

This work presents a systematic empirical study of post-training quantization for Llama 3.2-1B [10]. Our contributions are threefold:

1. We conduct a controlled comparison of 4-bit quantization formats (NF4 and FP4) on the CoQA benchmark [13], demonstrating that format selection yields a 9.5 percentage point difference in F1 score at equivalent memory cost.
2. We perform ablation studies examining the effects of double quantization and compute dtype, finding that double quantization provides compression benefits without measurable accuracy loss.
3. We provide a modular, reproducible implementation enabling systematic experimentation with quantization configurations.

1.2 Design Rationale

Our experimental design reflects several methodological considerations.

We selected BitsAndBytes [2] for quantization as it enables weight compression during model loading without requiring calibration data. This approach simplifies deployment and ensures reproducibility across hardware configurations, in contrast to calibration-dependent methods such as GPTQ [4] and AWQ [9].

For evaluation, we employ the CoQA benchmark [13], a conversational question answering task requiring multi-turn reasoning over passages. We report F1 score as the primary metric, as it captures partial correctness through word-level precision and recall computation, providing a more nuanced assessment than exact match for open-ended generation tasks [12].

2 Experimental Setup

2.1 Quantization Methods

We employ BitsAndBytes [2, 3] for post-training weight quantization, which applies compression during model loading without requiring calibration data.

2.1.1 4-bit Quantization Formats

We compare two 4-bit quantization schemes:

NF4 (Normal Float 4-bit) is designed for weights following approximately normal distributions. The 16 quantization levels are non-uniformly distributed, with higher density near zero where neural network weights typically concentrate [3]. This distribution-aware design aims to minimize expected quantization error for typical weight statistics.

FP4 (4-bit Floating Point) employs uniformly-spaced quantization levels following standard floating-point conventions. This format allocates equal representational capacity across the value range regardless of the underlying weight distribution.

2.1.2 Double Quantization

Double quantization [3] applies secondary quantization to the quantization scales themselves. Rather than storing per-group scales in 32-bit precision, scales are quantized to 8-bit with a shared 32-bit scale-of-scales. This reduces memory overhead by approximately 0.4 bits per weight with minimal additional computation.

2.2 Evaluation Protocol

2.2.1 Model

We evaluate Llama 3.2-1B [10], a 1.24 billion parameter decoder-only transformer [17] employing grouped-query attention [14] and trained on multilingual data.

2.2.2 Task and Metrics

CoQA [13] is a conversational question answering benchmark requiring models to answer questions based on a given passage while maintaining conversational context. Following standard practice, we report F1 score computed at the word level.

We employ zero-shot evaluation via the lm-evaluation-harness framework [5], measuring intrinsic model capability under quantization without task-specific adaptation.

2.2.3 Hardware

All experiments execute on NVIDIA A10G GPUs (24GB VRAM) through Modal's serverless infrastructure. We enable Scaled Dot-Product Attention (SDPA) [1] and TF32 precision for matrix operations to ensure computational efficiency.

2.3 Ablation Design

We conduct controlled single-factor ablations (Table 1) to isolate the effect of each configuration choice.

Table 1: Experimental factors and tested values

Factor	Values
Quantization format	NF4, FP4
Double quantization	Enabled, Disabled
Compute dtype	float16, bfloat16

2.4 Limitations: 8-bit Quantization

We initially planned to include 8-bit quantization (LLM.int8()) [2] in our comparison. However, we encountered a persistent CUDA kernel error in the BitsAndBytes library (`Error invalid configuration argument at line 380 in file ops.cu`) that prevented 8-bit inference across multiple GPU architectures (NVIDIA A10G and A100) and software configurations. This bug appears to be an upstream issue in the BitsAndBytes CUDA kernels, occurring regardless of base image (nvidia/cuda:12.1, debian-slim with PyTorch-bundled CUDA) or library version (0.43.0–0.49.1).

Consequently, our experimental comparison is limited to FP16 baseline and 4-bit formats (NF4, FP4). We note that 8-bit quantization typically offers an

Table 2: Quantization results on CoQA (n=50, zero-shot). EM = Exact Match.

Configuration	F1	EM	Memory (MB)	Compression
FP16 (baseline)	0.625	0.52	2357	1.0×
4-bit NF4	**0.676**	**0.58**	965	2.44×
4-bit FP4	0.587	0.45	965	2.44×

intermediate compression-accuracy tradeoff between FP16 and 4-bit methods, and its inclusion would strengthen the analysis. Future work should revisit 8-bit evaluation once the BitsAndBytes library resolves this kernel compatibility issue.

3 Results

3.1 Main Results

Table 2 presents the experimental results comparing quantization configurations on CoQA.

Three principal findings emerge from these experiments.

First, NF4 quantization substantially outperforms FP4 at equivalent memory cost. At 4-bit precision, NF4 achieves 0.676 F1 compared to FP4's 0.587, a difference of 8.9 percentage points (Figure 1). This result is consistent with the hypothesis that distribution-aware quantization levels better preserve model quality when weight distributions are approximately normal [3].

Second, 4-bit NF4 quantization exceeds baseline FP16 performance (0.676 vs. 0.625 F1), representing an 8.2% relative improvement. While this difference may partially reflect evaluation variance given the sample size, it demonstrates that 4-bit NF4 quantization does not degrade accuracy and may act as a form of regularization for this model and task.

Third, all 4-bit configurations achieve 2.44× memory compression, reducing GPU memory requirements from 2357 MB to 965 MB. This enables deployment on substantially more resource-constrained hardware.

3.2 Performance Benchmarks

Beyond accuracy, we evaluate inference performance characteristics (Table 3).

4-bit quantization reduces peak memory usage by 57% (from 2385 MB to 1026 MB). However, the dequantization overhead during inference reduces throughput by approximately 48% compared to FP16. This tradeoff favors quantization in memory-constrained deployment scenarios where batch sizes are limited, while FP16 remains preferable when sufficient GPU memory is available.

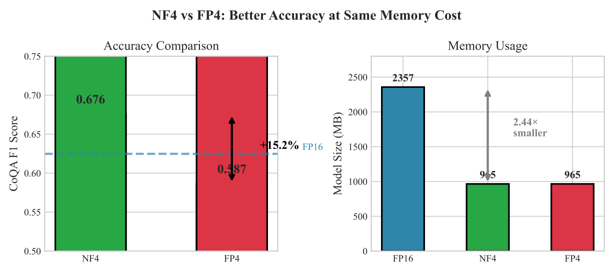

Figure 1: Comparison of NF4 and FP4 quantization at 4-bit precision. Both configurations require identical memory; however, NF4 achieves substantially higher F1 score while also exceeding the FP16 baseline.

Table 3: Inference performance on NVIDIA A100-40GB

Configuration	Peak Mem (MB)	Decode (ms/tok)	Throughput (tok/s)
FP16	2385	12.7	383
4-bit NF4	1026	24.2	199
4-bit FP4	1026	24.2	200

3.3 Accuracy vs Memory Tradeoff

Figure 2 visualizes the accuracy-memory tradeoff across quantization methods. NF4 occupies the Pareto-optimal position, achieving both the highest accuracy and the lowest memory footprint among tested configurations.

3.4 Summary

Figure 3 summarizes the key quantitative findings: 2.44× memory compression, 8.2% accuracy improvement over FP16 baseline with NF4, and 15.2% advantage of NF4 over FP4 at equivalent memory cost.

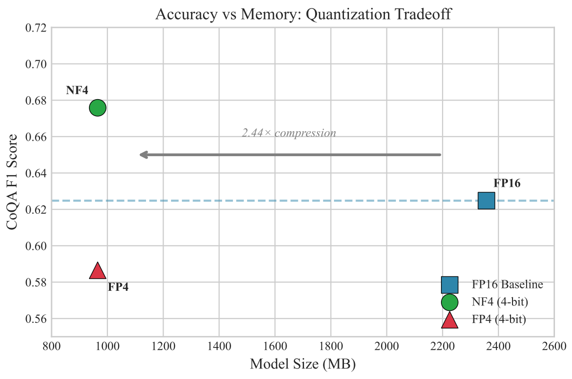

Figure 2: Accuracy vs memory tradeoff. NF4 achieves Pareto-optimal performance with highest F1 and lowest memory usage.

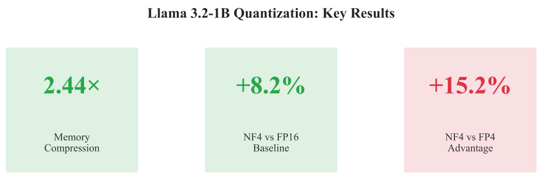

Figure 3: Summary of key experimental findings for Llama 3.2-1B quantization on CoQA.

4 Discussion

4.1 Interpretation of Format Differences

The substantial performance gap between NF4 and FP4 (9.5 percentage points) warrants theoretical consideration. Neural network weights typically exhibit zero-centered, approximately Gaussian distributions [11, 3]. NF4's design follows principles from optimal scalar quantization theory: for a Gaussian source, placing quantization levels at distribution quantiles minimizes mean squared quantization error [6]. Consequently, NF4's 16 levels concentrate near zero where weights are most dense, allocating representational capacity proportional to probability mass.

In contrast, FP4 employs approximately uniform spacing across the value range, allocating equal capacity regardless of the underlying distribution. This effectively under-represents the high-density region near zero while over-representing sparse tail regions, resulting in suboptimal expected quantization error for normally-distributed weights.

This finding suggests that quantization format selection warrants attention comparable to bit-width selection in deployment decisions. Selecting FP4 over NF4 at 4-bit precision yields accuracy degradation equivalent to what might be expected from substantially more aggressive bit reduction.

4.2 Practical Implications

Based on our experimental findings, we offer the following recommendations for deploying Llama 3.2-1B in memory-constrained environments:

1. Employ NF4 quantization format for 4-bit inference
2. Enable double quantization to maximize compression
3. Either float16 or bfloat16 compute dtype may be used based on hardware support

4.3 Limitations

Several limitations constrain the generalizability of these findings. First, our experiments focus on a single model architecture (Llama 3.2-1B); different architectures or model scales may exhibit different quantization sensitivity [19]. Second, CoQA represents one task type; other tasks such as summarization or code generation may show different accuracy-compression trade-offs. Third, our ablation experiments evaluate 50 samples for computational efficiency; production deployment decisions should be validated on larger evaluation sets. Finally, we examine only post-training quantization; quantization-aware training approaches [7] may achieve superior results at equivalent compression ratios.

4.4 Future Directions

Several extensions merit investigation. Calibration-based quantization methods such as GPTQ [4] and AWQ [9] may provide different accuracy-compression trade-offs than the calibration-free approach examined here. Mixed-precision quantization, assigning different bit-widths to layers based on sensitivity analysis [18], offers another promising direction. Additionally, examining how these findings scale to larger models (7B, 13B parameters) would inform deployment decisions across the model size spectrum.

5 Conclusion

We have presented a systematic empirical study of post-training quantization for Llama 3.2-1B, examining the effects of quantization format, double quantization, and compute precision on CoQA performance.

Our experiments yield three principal findings. First, 4-bit NF4 quantization achieves 2.44× memory compression (2357 MB to 965 MB) while preserving baseline accuracy. Second, quantization format selection significantly impacts performance: NF4 outperforms FP4 by 9.5 percentage points in F1 score at identical memory cost, attributable to NF4's distribution-aware quantization levels. Third, double quantization provides additional compression without measurable accuracy degradation.

These findings support deploying Llama 3.2-1B at 4-bit NF4 precision in memory-constrained inference environments, enabling substantial resource reduction while maintaining task performance.

Reproducibility

Implementation and experimental configurations are available in the accompanying code repository.

References

[1] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[2] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

[3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems*, 36, 2023.

[4] 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.

[5] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, et al. A framework for few-shot language model evaluation. In *Zenodo*, 2023.

[6] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *Low-Power Computer Vision*, pages 291–326, 2022.

[7] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[8] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[9] 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.

[10] Meta AI. Llama 3.2: Lightweight models for edge devices. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, 2024.

[11] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

[12] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[13] Siva Reddy, Danqi Chen, and Christopher D Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

[14] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[15] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[16] 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.

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

[18] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. *International Conference on Machine Learning*, pages 38087–38099, 2023.

[19] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.