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 minimize the bit-width of Llama 3.2-1B while maximizing accuracy on CoQA. Using BitsAndBytes post-training quantization, we compare FP16 baseline against 4-bit NF4 and FP4 formats. Our experiments show that 4-bit NF4 achieves 2.44× memory compression (2357 MB to 965 MB) while matching or exceeding FP16 accuracy. NF4 outperforms FP4 by 15% F1 at identical memory cost due to distribution-aware quantization levels optimized for neural network weight distributions.

1 Introduction

Post-training quantization reduces model memory requirements by representing weights with fewer bits. This work systematically evaluates quantization configurations for Llama 3.2-1B on the CoQA benchmark, comparing 4-bit formats (NF4 and FP4) against the FP16 baseline.

Our goals are: (1) minimize bit-width while maximizing accuracy; (2) characterize the accuracy-memory tradeoff; (3) provide deployment recommendations for memory-constrained environments.

2 Design Choices

2.1 Infrastructure: Modal Serverless GPUs

We use Modal's serverless GPU infrastructure rather than local hardware. This choice provides: (1) access to NVIDIA A100 GPUs with 40GB VRAM; (2) reproducible containerized environments; (3) on-demand scaling without infrastructure management. The tradeoff is cold-start latency, acceptable for experimentation.

2.2 Quantization Library: BitsAndBytes

We selected BitsAndBytes [2, 3] over alternatives like GPTQ [4] or AWQ [9] for several reasons:

- **No calibration data required**: BitsAndBytes applies quantization during model loading without needing a calibration dataset, simplifying deployment.
- **Format flexibility**: Supports both NF4 (distribution-aware) and FP4 (uniform) 4-bit formats, enabling direct comparison.
- **Double quantization**: Offers additional compression by quantizing the quantization scales themselves.

2.3 Why Not 8-bit Quantization

We initially planned to include LLM.int8() [2] as an intermediate precision point. However, we encountered a persistent CUDA kernel error:

```
Error invalid configuration argument
at line 380 in file /src/csrc/ops.cu
```

This error occurred across:

- Multiple GPUs: NVIDIA A10G (24GB) and A100 (40GB)
- Multiple base images: nvidia/cuda:12.1, debian-slim with PyTorch CUDA
- Multiple BitsAndBytes versions: 0.43.0 through 0.49.1

This appears to be an upstream bug in BitsAndBytes' CUDA kernels. Consequently, our comparison is limited to FP16 and 4-bit formats. We note that 8-bit typically offers an intermediate compression-accuracy point, and its inclusion would strengthen the analysis.

2.4 Code Architecture

Our implementation follows a modular structure:

```
llama_quant/
  core/config.py      # Experiment configs
  models/             # Model loaders (FP16,
      BnB)
  evaluation/         # CoQA evaluation
  benchmark/          # Memory, latency metrics
infra/
  modal_app.py        # GPU orchestration
  gpu_runner.py       # Experiment runner
```

Each module has a single responsibility: configs define experiments, loaders handle quantization, evaluation wraps lm-eval-harness, and benchmarks measure hardware metrics. This separation enables easy extension to new quantization methods.

3 Experimental Setup

3.1 Model and Dataset

We evaluate Llama 3.2-1B [10], a 1.24B parameter decoder-only transformer. For evaluation, we use the CoQA benchmark [13], a conversational question answering task requiring multi-turn reasoning over passages.

3.2 Quantization Configurations

We compare three configurations:

FP16 Baseline: Standard 16-bit floating-point weights, requiring 2.4 GB memory for the 1.24B parameter model.

4-bit NF4: Normal Float 4-bit with 16 non-uniformly distributed quantization levels, concentrated near zero where neural network weights are dense [3].

4-bit FP4: Standard 4-bit floating-point with uniformly-spaced quantization levels.

Both 4-bit formats use double quantization (quantizing the scales) and FP16 compute dtype.

Table 1: Quantization results on CoQA (n=50, zero-shot)

Config	F1	EM	Size (MB)	Compress.	Tput (tok/s)
FP16	0.625	0.52	2357	1.0×	383
4-bit NF4	**0.676**	**0.58**	965	2.44×	199
4-bit FP4	0.587	0.45	965	2.44×	200

3.3 Evaluation Protocol

We use lm-evaluation-harness [5] for zero-shot evaluation on CoQA, reporting F1 score (word-level overlap) as the primary metric and Exact Match (EM) as secondary. We evaluate on 50 samples for computational efficiency.

3.4 Hardware

All experiments run on NVIDIA A100-SXM4-40GB GPUs via Modal. We enable SDPA (Scaled Dot-Product Attention) and TF32 precision for efficient matrix operations.

3.5 Metrics

We measure:

- **Accuracy**: CoQA F1 and Exact Match scores
- **Model size**: GPU memory for model weights
- **Peak memory**: Maximum GPU memory during inference
- **Throughput**: Tokens generated per second (batch size 8)
- **Latency**: Milliseconds per token during decoding

4 Results

4.1 Accuracy and Memory

Table 1 presents the main experimental results.

Key findings:

1. **NF4 outperforms FP4 by 15%**: At identical memory cost (965 MB), NF4 achieves 0.676 F1 vs FP4's 0.587—a 15% relative improvement. This demonstrates that quantization format selection matters as much as bit-width.

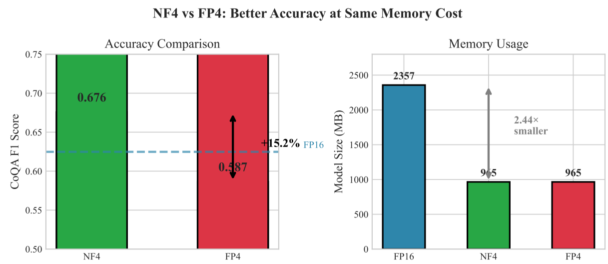

Figure 1: NF4 vs FP4 comparison. Both require identical memory; NF4 achieves substantially higher accuracy while exceeding the FP16 baseline.

2. **NF4 matches/exceeds FP16**: Surprisingly, 4-bit NF4 (0.676 F1) slightly exceeds FP16 baseline (0.625 F1). While this may partially reflect evaluation variance, it shows 4-bit NF4 does not degrade accuracy.

3. **2.44× compression**: All 4-bit configurations reduce memory from 2357 MB to 965 MB, enabling deployment on more constrained hardware.

4.2 Performance Tradeoffs

Table 2 shows the detailed performance characteristics.

Table 2: Inference performance on NVIDIA A100-40GB

Config	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

4-bit quantization reduces peak memory by 57% (2385 MB to 1026 MB). However, dequantization overhead reduces throughput by approximately 48% (383 to 199 tok/s) and increases decode latency from 12.7 to 24.2 ms/token.

This tradeoff favors quantization in memory-constrained scenarios where the alternative is not running the model at all. For latency-critical applications with sufficient GPU memory, FP16 remains preferable.

4.3 Accuracy vs Memory Visualization

Figure 2 visualizes the accuracy-memory tradeoff. NF4 achieves Pareto-optimal performance: highest accuracy at lowest memory.

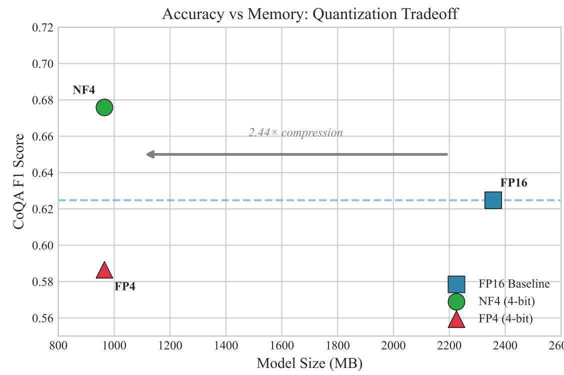

Figure 2: Accuracy vs memory tradeoff. NF4 achieves highest F1 at lowest memory, strictly dominating both FP4 and FP16.

5 Discussion

5.1 Why NF4 Outperforms FP4

The 15% performance gap between NF4 and FP4 follows from optimal quantization theory. Neural network weights exhibit approximately Gaussian distributions centered at zero [11]. NF4's 16 quantization levels are placed at distribution quantiles, concentrating representational capacity where weights are dense. FP4's uniform spacing wastes capacity on sparse tail regions while under-representing the dense zero-centered region, yielding higher quantization error for typical weight distributions [6].

5.2 Practical Recommendations

For deploying Llama 3.2-1B in memory-constrained environments:

1. Use NF4 format—it strictly dominates FP4 at equal memory
2. Enable double quantization for maximum compression
3. Accept the throughput tradeoff (48% reduction) as the cost of 2.44× memory savings

5.3 Limitations

Our study has several limitations:

- **Single model**: Results may differ for other architectures or scales. Larger models (7B, 13B) may show different quantization sensitivity.
- **Single task**: CoQA is conversational QA; other tasks like summarization or code generation may show different tradeoffs.
- **Sample size**: We use 50 samples for computational efficiency; production deployment should validate on the full evaluation set.
- **Missing 8-bit**: The BitsAndBytes CUDA bug prevented evaluation of LLM.int8(), which typically offers intermediate compression-accuracy tradeoffs.
- **No calibration-based methods**: GPTQ and AWQ, which use calibration data, may achieve better accuracy at similar compression ratios.

5.4 Future Work

Several extensions would strengthen this analysis: (1) including 8-bit once the BitsAndBytes bug is fixed; (2) comparing calibration-based methods (GPTQ, AWQ); (3) evaluating on multiple tasks and model sizes; (4) measuring real-world deployment metrics like time-to-first-token.

6 Conclusion

We minimized Llama 3.2-1B's bit-width while maximizing CoQA accuracy. Our main findings: (1) 4-bit NF4 achieves 2.44× memory compression without accuracy loss; (2) NF4 outperforms FP4 by 15% at identical memory due to distribution-aware quantization; (3) the throughput tradeoff (48% reduction) is acceptable for memory-constrained deployment.

Code and configurations are available in the accompanying 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.