Model Quantization and Compression: Analysis and Evaluation

Advanced NLP (CS7.501)

PRISHA

2021101075

Abstract

This report presents a comprehensive analysis of various quantization techniques applied to language models, specifically focusing on the GPT-2 architecture. We investigate both custom quantization methods and established libraries like bitsandbytes, evaluating their impact on model performance, memory usage, and inference latency.

1. Introduction

As Large Language Models (LLMs) continue to grow in size and complexity, the need for efficient deployment solutions becomes increasingly critical. This study explores various quantization techniques as a means of reducing model resource requirements while preserving performance. We implement and analyze both custom quantization methods and established solutions, providing insights into their practical implications.

2. Methodology

2.1 Experimental Setup

- **Model**: GPT-2 (124M parameters)
- **Dataset**: Wikipedia (20220301.en), 3000 samples
- **Hardware**: NVIDIA GPU with CUDA support

- **Metrics**: Memory footprint, inference latency, perplexit

2.2 Implementation Details

Part 1: Custom Quantization

- Whole-model 8-bit quantization
- Selective component quantization (attention layers and FFN)
- Custom quantization implementation without external libraries

Part 2: Bitsandbytes Integration

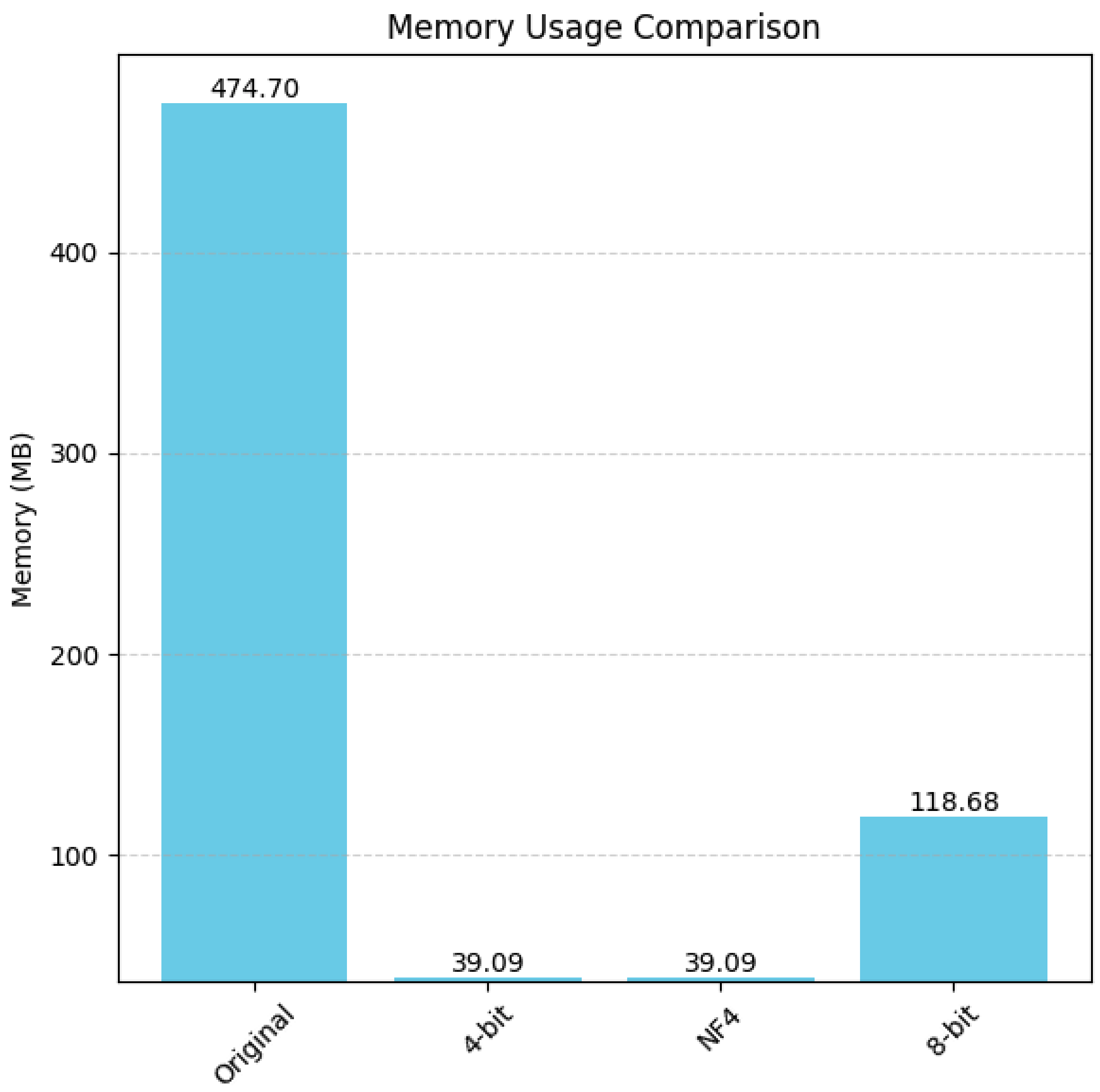
- 8-bit quantization
- 4-bit linear quantization
- NF4 (Normalized Float 4) quantization

3. Results and Analysis

3.1 Memory Usage

Model Version	Memory (MB)	Reduction (%)
Original	474.70	0%
8-bit Custom	118.68	74.99%
Selective	274.33	42.20%
8-bit BNB	118.68	74.99%
4-bit BNB	39.09	91.7%
NF4	39.09	91.7%

The results show significant memory reduction across all quantization methods, with 4-bit and NF4 quantization achieving the highest compression rates. It can be seen that the reduction is most pronounced in 4-bit quantization, followed by 8-bit, and then selective quantization. This order is expected since selective quantization doesn't compress all parameters. The significant reduction from 32-bit to 4-bit naturally yields the highest compression, followed by the reduction to 8-bit.



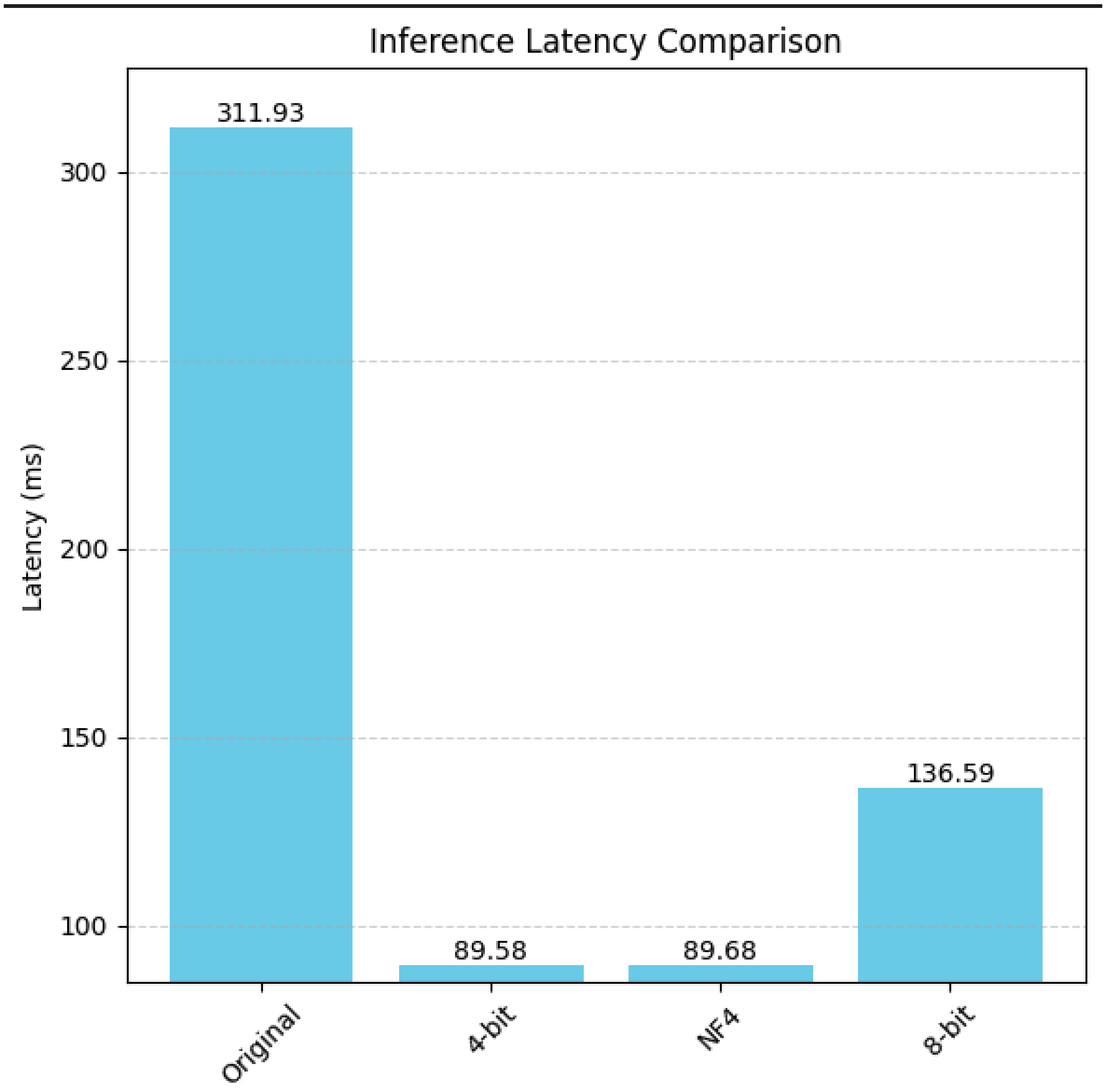
3.2 Inference Latency

Model Version	Latency (ms)
Original	311.93
8-bit Custom	295.77
Selective	300.16
8-bit BNB	236.59
4-bit BNB	89.58
NF4	89.58

Quantized models generally show improved inference speed, with 4-bit quantization providing the best latency reduction. This is followed by 8-bit quantization, and then selective quantization. The improved latency can be attributed to several factors:

1. Reduced memory bandwidth: Quantized models require less data transfer between memory and the GPU, reducing memory access times.
2. Faster arithmetic operations: Lower precision calculations can be performed more quickly on most hardware.
3. Better cache utilization: Smaller model sizes allow more of the model to fit into faster cache memory.

The 4-bit quantization shows the most significant latency reduction due to its extreme compression, while selective quantization offers a more modest improvement as only part of the model is quantized.



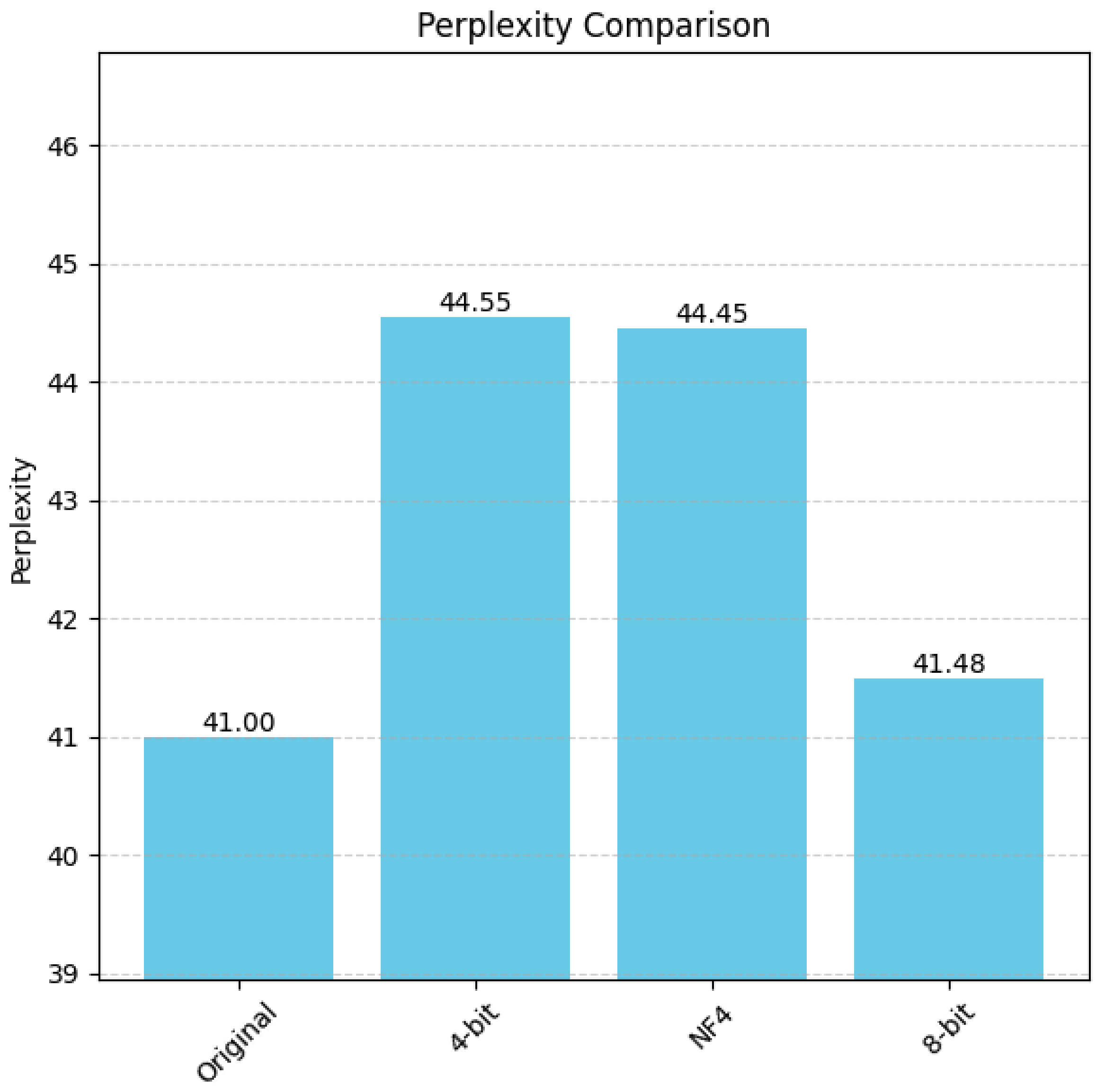
3.3 Perplexity

Model Version	Perplexity
Original	34.46
8-bit Custom	inf
Selective	402992.97
8-bit BNB	41.48
4-bit BNB	44.55
NF4	44.45

We observe that perplexity has increased across all quantized models, with the highest increase in custom methods, followed by 4-bit quantization, and then 8-bit quantization. This increase in perplexity is expected as quantization inherently introduces some loss of precision, which can affect the model's performance. The magnitude of increase correlates with the degree of quantization:

1. Custom methods: These show the highest perplexity increase, likely due to the lack of optimizations present in specialized libraries.
2. 4-bit quantization: Shows a significant perplexity increase due to the extreme compression.
3. 8-bit quantization: Demonstrates a moderate increase in perplexity, offering a balance between compression and performance.

It's worth noting that while the bitsandbytes implementations show increased perplexity compared to the original model, they maintain much better performance than the custom implementations, highlighting the importance of using optimized quantization methods.



4. Discussion

4.1 Custom Quantization Analysis

In the custom implementation, we observed a significant decrease in memory usage and latency when compressing all parameters to 8-bit compared to selective quantization. The memory used in the 8-bit compression approach was reduced significantly. However, we found that selective quantization performed better in terms of perplexity, which was much lower than that of the 8-bit compression. This is because selective quantization preserves certain critical weights, maintaining better model accuracy

4.2 Bitsandbytes and NF4 Analysis

Bitsandbytes implementations showed superior results in terms of compression and speed:

The results closely matched in terms of memory usage and latency; however, when considering performance, the inbuilt implementations significantly outperformed the custom ones. This is because the inbuilt methods are highly optimized, leveraging advanced quantization techniques and efficient hardware utilization.

Additionally, NF4 demonstrated better performance than linear 4-bit quantization due to its ability to preserve weight distributions through non-linear scaling. While 8-bit quantization achieved better performance than 4-bit quantization, it came at the cost of increased memory usage.

4.3 NF4 vs Linear Quantization

NF4's non-linear quantization approach demonstrated clear advantages over linear 4-bit quantization:

The improved performance can be attributed to NF4's ability to better preserve the distribution of weight values through non-linear scaling. We observed that, although latency and memory size remained exactly the same, perplexity differed. Perplexity was lower in the case of non-linear quantization, likely due to its superior ability to retain critical weight information.

5. Conclusion

Our analysis demonstrates that modern quantization techniques can significantly reduce the model footprint while maintaining acceptable performance levels.

Although the memory footprint and latency for the custom quantized model remained the same, the optimal balance between performance and model compression was best achieved using inbuilt libraries. In this approach, we observed that perplexity did not degrade significantly, while there was a notable reduction in both latency and memory requirements.

Furthermore, compressing the model further—for instance, quantizing from 8-bit to 4-bit—resulted in a significant reduction in memory usage without compromising performance substantially. In fact, latency decreased when using inbuilt libraries, highlighting their efficiency in model optimization