ANLP-A4 Report

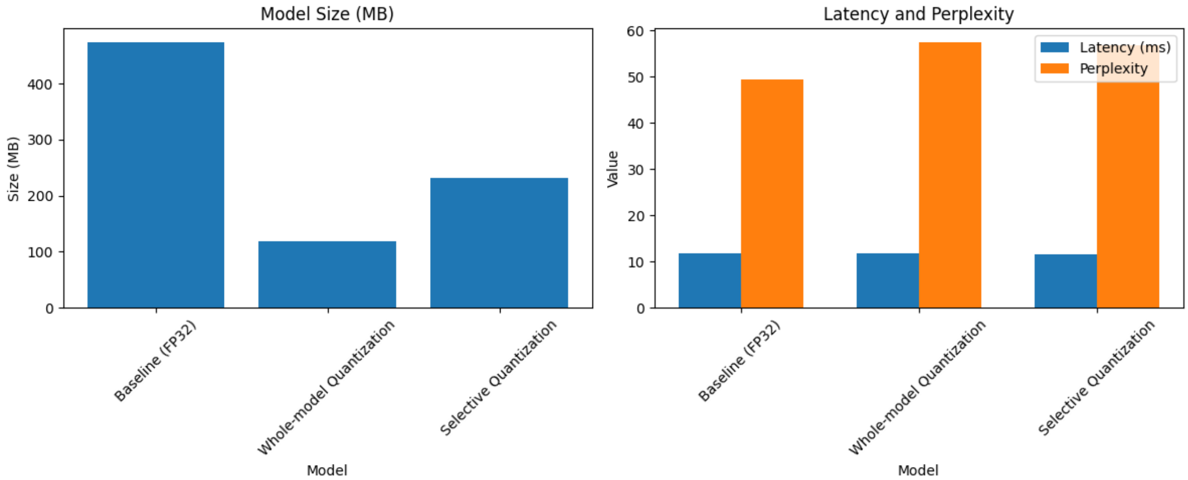
Saketh Reddy Vemula
saketh.vemula@research.iiit.ac.in

Instructor: Manish Shrivatsava
Course: Advanced NLP

2022114014

Model: `gpt2`

Part 1: Quantization from Scratch



Model Size (MB)

Model	Model Size	Latency (in ms)	Perplexity	Implications
Baseline (FP32)	474.40 M	11.74 ± 0.34 ms	49.27	The original model size is around 450 MB.
Whole-model Quantization	119.02 MB	11.72 ± 0.31 ms	57.51	**Whole-model Quantization** reduces the size to approximately 100 MB, a significant reduction indicating that the entire model has been compressed, resulting in notable memory savings.

Selective Quantization	231.70 MB	11.59 ± 0.36 ms	56.82	**Selective Quantization r**educes the size to around 200 MB, smaller than the baseline but larger than whole-model quantization. This is because this involves compressing only specific parts of the model, retaining precision in certain layers while compressing others.

Latency (ms) and Perplexity

- **Latency (ms)**:
    - All the three models have almost same Latency. With slight improvement for Selective Quantization.
    - Whole-model Quantization, although expected to have lowest latency, has approximately same latency because of hardware level optimizations present for FP32 models in Nvidia GPUs.
- **Perplexity**:
    - The baseline model has a perplexity around 50, while both quantized models exhibit slightly higher perplexities. This suggests that quantization introduces some degradation in predictive quality.

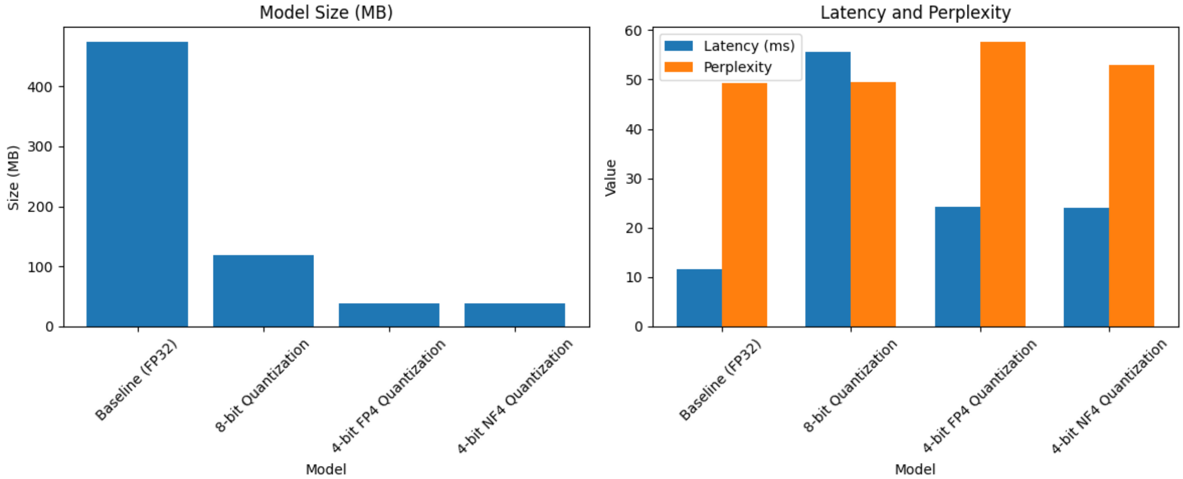
Summary of Findings

- **Whole-model Quantization**: Achieves the most significant memory and latency reductions, but at the potential cost of minor accuracy degradation.
- **Selective Quantization**: Strikes a balance between memory reduction and accuracy retention, with some reduction in model size and latency but potentially better retention of model accuracy compared to whole-model quantization.

Qualitative Insights

Quantization significantly improves memory usage and latency, which is beneficial for deployment on resource-constrained systems. Whole-model quantization may be ideal for applications where speed and memory efficiency are top priorities. Selective quantization, on the other hand, provides a balanced approach, potentially retaining more of the model's accuracy at a moderate memory and speed trade-off.

Part 2: Bitsandbytes Integration and NF4 Quantization



NF4 Quantization vs. Linear Quantization

Aspect	**Linear Quantization**	**NF4 (Nonlinear 4-bit) Quantization**
Definition	Maps model weights to a lower-precision format using a uniform, linear scale.	Maps weights using a non-linear, adaptive scale that preserves high-value precision.
Precision Levels	Typically uses 8-bit or 4-bit precision, with a uniform distribution across the range.	Specifically designed for 4-bit precision with a non-linear scale.
Scaling Method	Applies a fixed, linear scaling factor across all weight values.	Uses a non-linear scaling factor that adapts based on the distribution of values.
Distribution of Values	Each range segment has equal spacing, regardless of frequency.	More representation range is allocated to frequently occurring values, while less is given to rare extremes.
Representation of Outliers	Poorly represents outliers, as they are forced into a limited range.	Better captures outliers, since it uses a scaling that adapts to extreme values.
Memory Efficiency	Reduces memory usage by storing weights in lower-bit precision, but may lose significant precision at very low bits (e.g., 4-bit).	Highly memory-efficient, achieving lower memory usage while retaining more precision than linear 4-bit.
Impact on Model Accuracy	Potential accuracy loss, especially with 4-bit, due to lack of flexibility in representing diverse values.	Retains better accuracy by preserving precision where it's most needed, even with 4-bit quantization.

Latency and Speed	Faster inference compared to FP32, though may have higher latency compared to NF4 due to less optimized quantization.	Faster inference times due to better quantization efficiency and less need for frequent re-scaling.
Implementation Complexity	Simpler to implement, as it only requires a linear scaling factor.	More complex to implement because it requires a nonlinear scaling function.
Use Case Suitability	Suitable for models where some loss in precision is acceptable, or for higher bit depths (e.g., 8-bit).	Ideal for highly compressed models, especially when using 4-bit quantization, while aiming to maintain accuracy.

Overall:

- **Linear Quantization** is straightforward and effective for higher bit depths but can struggle with precision loss at very low bits. Can also be confirmed with the graph.
- **NF4 Quantization** uses a non-linear approach that maintains better model accuracy even at 4-bit precision, making it suitable for highly constrained environments. This method provides a better trade-off between memory efficiency and model accuracy compared to standard linear quantization at the same bit level.

Impact on Model Accuracy and Efficiency

From the plots we can see:

1. **Model Size**: All quantization methods (8-bit, 4-bit FP4, and 4-bit NF4) significantly reduce model size compared to the baseline FP32, with the 4-bit quantization methods (both FP4 and NF4) offering the smallest size.
2. **Latency**: As seen, 4-bit NF4 quantization provides reduced latency compared to the baseline and 8-bit quantization. This suggests that lower bit quantization reduces computational overhead, likely due to decreased memory usage. However, due to the issue with `bitsandbytes` library as mentioned in https://github.com/bitsandbytes-foundation/bitsandbytes/issues/6: Issue with Bitsandbytes, it takes longer inference time for quantized model.
3. **Perplexity (Accuracy)**: The 4-bit NF4 quantization maintains a similar perplexity to other quantization methods, indicating that its non-linear scaling helps preserve model accuracy despite the lower bit depth. The reduction in precision with linear 4-bit quantization might have resulted in a slight accuracy drop without NF4's adaptive scaling.

Summary

Using **Bitsandbytes**, we can apply various quantization methods to make models more memory-efficient:

- **8-bit quantization** generally offers a good trade-off between accuracy and model efficiency. However, increasing the model latency.

- **4-bit FP4** and **4-bit NF4 quantization** further reduce model size, with NF4 being a preferable choice as it preserves accuracy better due to its non-linear approach. But it increases the inference time.