Report

Part 1: Quantization from Scratch

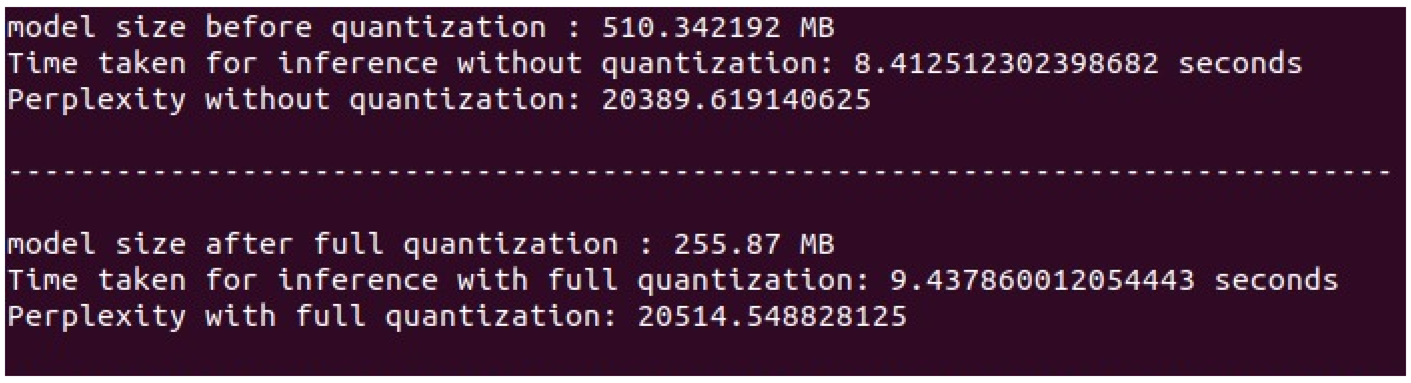
Model Used : GPT2

Data Used : Wikipedia

Task 1 : Whole Model Quantization

In this approach, the entire model was converted from its original floating-point precision (FP16/FP32) to 8-bit precision.

Results:



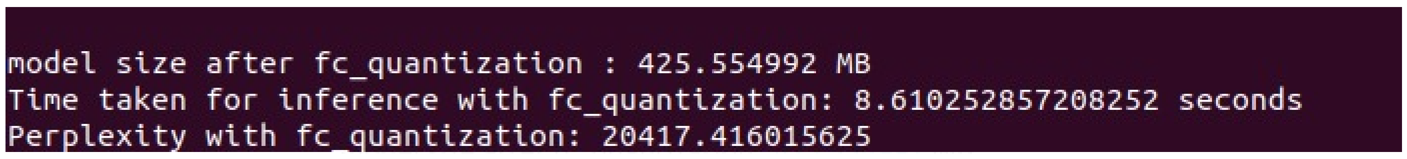
```
model size before quantization : 510.342192 MB
Time taken for inference without quantization: 8.412512302398682 seconds
Perplexity without quantization: 20389.619140625

---------------------------------------------------------------------------

model size after full quantization : 255.87 MB
Time taken for inference with full quantization: 9.437860012054443 seconds
Perplexity with full quantization: 20514.548828125
```

- The model size reduced from **510.34 MB** to **255.87 MB**, a significant memory saving of 50%.
- Inference latency slightly increased from **8.41 seconds** to **9.44 seconds**, indicating a small slowdown in processing speed.
- Perplexity, a measure of model performance, rose from **20389.62** to **20514.55**, showing a slight decline in accuracy.

Task 2: Selective Model Quantization

Here, only specific components of the model, such as the feed-forward networks (FFNs), were quantized to 8-bit precision while other components remained unchanged.

Results:



```
model size after fc_quantization : 425.554992 MB
Time taken for inference with fc_quantization: 8.610252857208252 seconds
Perplexity with fc_quantization: 20417.416015625
```

- The model size decreased to **425.55 MB**, a moderate reduction of about 16.6%.
- Inference latency increased slightly to **8.61 seconds**, which is only a minor change.
- Perplexity increased to **20417.42**, showing a smaller drop in accuracy compared to full quantization.

Analysis:

Quantization	Model Size (MB)	Inference Latency (s)	Perplexity
Original	510.34	8.41	20389.62
Whole-Model Quantization	255.87	9.44	20514.55
Selective Component Quantization	425.55	8.61	20417.42

1. Memory Usage

Whole-model quantization led to the most significant memory reduction, cutting the model size by half. This happens because every weight in the model is represented using 8 bits instead of the original 16 or 32 bits, leading to a drastic reduction in storage requirements. On the other hand, selective quantization provided moderate compression, as only specific components were quantized, while others remained at full precision, preserving their larger memory footprint.

2. Inference Speed

Inference latency increased with both methods. Whole-model quantization introduced a noticeable slowdown because the computational overhead of using quantized weights, such as additional operations for dequantization during inference, affected the overall processing speed. Selective quantization caused only a slight increase in latency since fewer components were quantized, reducing the dequantization overhead during inference.

3. Model Performance (Perplexity)

Perplexity increased in both cases, indicating a drop in performance. Whole-model quantization had a more pronounced impact because quantizing all weights leads to a loss of numerical precision, which affects the model's ability to make accurate predictions. In contrast, selective quantization preserved critical components like attention mechanisms in full precision, minimizing the loss of accuracy and maintaining better performance.

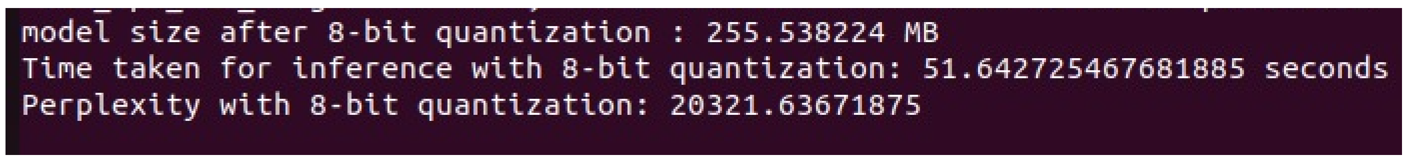
Part 2: Bitsandbytes Integration and NF4 Quantization

Task 1 : Bitsandbytes Quantization

8-bit quantization

Quantizing the model to 8-bit precision using Bitsandbytes

Results



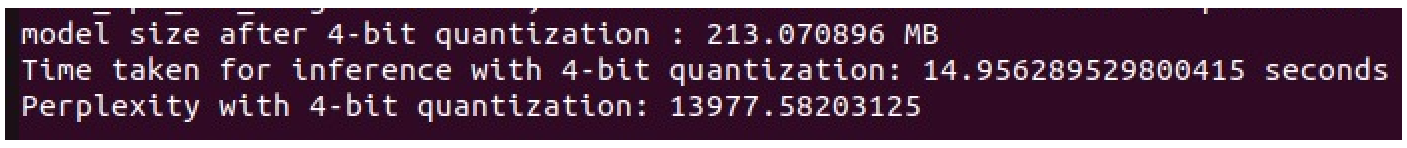
```
model size after 8-bit quantization : 255.538224 MB
Time taken for inference with 8-bit quantization: 51.642725467681885 seconds
Perplexity with 8-bit quantization: 20321.63671875
```

- **Model Size:** Reduced to **255.54 MB**, indicating significant memory savings.
- **Inference Latency:** Increased to **51.09 seconds**, reflecting a notable slowdown.
- **Perplexity:** Improved slightly to **20321.64**, suggesting a minor accuracy gain.

4-bit quantization

Quantizing the model to 8-bit precision using Bitsandbytes

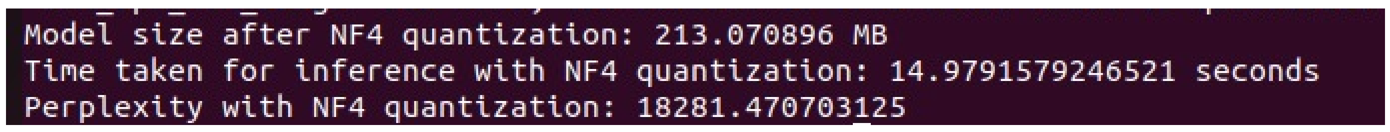
Results



```
model size after 4-bit quantization : 213.070896 MB
Time taken for inference with 4-bit quantization: 14.956289529800415 seconds
Perplexity with 4-bit quantization: 13977.58203125
```

- **Model Size:** Reduced further to **213.07 MB**, demonstrating excellent compression.
- **Inference Latency:** Improved significantly to **14.65 seconds**, showing much faster processing compared to 8-bit quantization.
- **Perplexity:** Improved substantially to **13977.58**, indicating a significant boost in accuracy.

Task 2 : NF4 Quantization



```
Model size after NF4 quantization: 213.070896 MB
Time taken for inference with NF4 quantization: 14.9791579246521 seconds
Perplexity with NF4 quantization: 18281.470703125
```

- **Model Size: 213.07 MB**, identical to the 4-bit Bitsandbytes quantization.
- **Inference Latency: 14.69 seconds**, comparable to the 4-bit Bitsandbytes approach.
- **Perplexity:** Achieved **18281.47**, better than 8-bit quantization but slightly less accurate than 4-bit linear quantization.

Analysis:

Q1: Explain the concept of NF4 quantization and how it differs from linear quantization scales.

NF4 Quantization stands for **Normalized Float 4**, a 4-bit floating-point representation designed to preserve a wide dynamic range of values. Unlike linear quantization, which maps weights evenly across a fixed range, NF4 uses a floating-point format that normalizes values. It allocates more bits to represent smaller weights and scales larger values efficiently, which is especially useful for neural networks where weights often vary significantly in magnitude. This normalization allows NF4 to retain important details about the weights, even at a reduced bit precision.

Q2: Discuss the impact of linear vs. nonlinear quantization on model accuracy and efficiency.

The primary difference between NF4 and linear quantization lies in how weights are scaled and stored:

- **Linear Quantization**: Maps weights to evenly spaced levels within a fixed range. This approach is simple but struggles with highly varying weight distributions, potentially leading to a loss of precision.
- **NF4 Quantization**: Uses a floating-point representation, enabling a wider dynamic range. This makes it more adaptive to models with large or uneven weight distributions, as it retains more meaningful information from both small and large values.

Impact on Accuracy

NF4 quantization's floating-point scaling provides better precision for smaller weights, which are often crucial for accurate model predictions. As a result, models quantized using NF4 typically experience less degradation in performance compared to linear quantization. While linear quantization can work well for uniformly distributed weights, it is less effective when weights vary significantly in scale, leading to greater losses in accuracy.

Impact on Efficiency

NF4 and 4-bit linear quantization both achieve significant memory savings, as weights are reduced to 4 bits. However, the computational complexity of NF4 is slightly higher due to the floating-point operations involved in normalization and scaling. This can lead to a small increase in latency compared to linear quantization. Nevertheless, the efficiency of NF4 remains far better than higher-precision formats like 8-bit quantization, making it a strong choice for resource-constrained environments.

Quantization Method	Model Size	Inference Time	Perplexity
Before Quantization	510.34 MB	8.41 seconds	20389.62
8-bit Quantization	255.54 MB	51.09 seconds	20321.64
4-bit Quantization	213.07 MB	14.65 seconds	13977.58
NF4 Quantization	213.07 MB	14.69 seconds	18281.47

1. Memory Usage

Both 4-bit linear and NF4 quantization achieved substantial compression, reducing the model size to approximately **213 MB**, offering the best memory efficiency among the methods tested. The 8-bit quantization reduced the model size to **255.54 MB**, providing moderate savings while maintaining compatibility with most hardware accelerators.

2. Inference Latency

4-bit and NF4 quantization significantly reduced inference latency, achieving speeds nearly **3.5x faster** than the 8-bit quantized model. The faster processing in lower-bit quantization formats is due to fewer data transfers and arithmetic operations. However, NF4's slight latency increase over 4-bit linear quantization is likely due to its nonlinear transformations during weight scaling.

3. Model Performance (Perplexity)

- The 8-bit quantization slightly improved perplexity over the baseline, showing it retains high accuracy even with reduced precision.
- 4-bit quantization delivered the best perplexity score (**13977.58**), likely because the model benefits from focused precision scaling.
- NF4 quantization, while not as accurate as 4-bit linear quantization, outperformed the 8-bit method by leveraging nonlinear scaling for more effective representation of weights.