Advanced NLP

ASSIGNMENT 4

November 16, 2024

Ashna Dua
2021101072
ashna.dua@students.iiit.ac.in

Contents

1 Introduction

Quantization is a technique used to reduce the memory footprint and improve inference latency of machine learning models, particularly deep learning models. This report outlines the metrics measured before and after quantization, including memory footprint, inference latency, and perplexity. The analysis focuses on the impact of different quantization approaches on these metrics, providing both quantitative data and qualitative insights.

2 Metrics for Evaluation

2.1 Memory Footprint

Definition: The amount of memory used by the model before and after quantization.
Memory usage was calculated as follows for Custom Quantization:

```python
def measure_memory(self, model) -> Dict[str, float]:
        fp32_memory = 0
        int8_memory = 0

        for name, param in model.named_parameters():
            num_elements = param.nelement()

            if name in self.quantized_params:
                int8_memory += num_elements
            else:
                fp32_memory += num_elements * 4

        fp32_memory = fp32_memory / (1024 * 1024)
        int8_memory = int8_memory / (1024 * 1024)
        total_memory = fp32_memory + int8_memory

        return {'total_mb': total_memory}
```

Memory usage was calculated as follows for Quantization using bitsandbytes library:

```python
def measure_memory(self, model) -> float:
        memory_bytes = sum(p.nelement() * p.element_size() for p in model.parameters())
        return memory_bytes / (1024 * 1024)  # Convert to MB
```

2.2 Inference Latency

Definition: The time taken for the model to make predictions before and after quantization. Inference latency was measured in milliseconds (ms) for a standard input size as follows:

```
def measure_latency(self, model, input_texts: list, num_runs: int = 1) -> float:
        latencies = []
        with torch.no_grad():
            for input_text in input_texts[10:20]:
                inputs = self.tokenizer(input_text[:512], return_tensors="pt").to(self.device)
                for _ in range(num_runs):
                    start_time = time.time()
                    model(**inputs)
                    latencies.append(time.time() - start_time)

        return sum(latencies) / len(latencies)
```

2.3 Perplexity

Definition: A measurement of how well a probability distribution predicts a sample, commonly used in language models.
Measurement: Perplexity was computed using the Wikipedia dataset:

```
def calculate_perplexity(self, model, texts: List[str], max_length=512) -> float:
        total_loss = 0
        total_length = 0

        model.eval()
        with torch.no_grad():
            for text in texts:
                inputs = self.tokenizer(text[:max_length], return_tensors="pt").to(self.device)
                labels = inputs.input_ids

                outputs = model(**inputs)
                logits = outputs.logits

                shift_logits = logits[..., :-1, :].contiguous()
                shift_labels = labels[..., 1:].contiguous()

                loss_fct = CrossEntropyLoss(reduction='sum')
                loss = loss_fct(shift_logits.view(-1, shift_logits.size(-1)), shift_labels.view(-1))

                total_loss += loss.item()
                total_length += labels.size(1) - 1

        return torch.exp(torch.tensor(total_loss / total_length)).item()
```

3 Analysis

3.1 Results for Custom Quantization - Part 1

Model	Memory (MB)	Latency (sec)	Perplexity
Baseline	474.7	0.0862	31.62
Full-8bit	118.97	0.0549	inf
Selective-8bit	393.7	0.0673	69637184.00

Table 1: Custom Quantization

3.2 Results for Quantization using inbuilt libraries - Part 2

Model	Memory (MB)	Latency (sec)	Perplexity
Base	474.70	0.0951	31.62
8bit_linear	156.35	0.0566	31.83
4bit_linear	115.85	0.0302	35.62
4bit_nf4	115.85	0.0284	33.94

Table 2: Quantization using bitsandbytes library

4 Visualizations

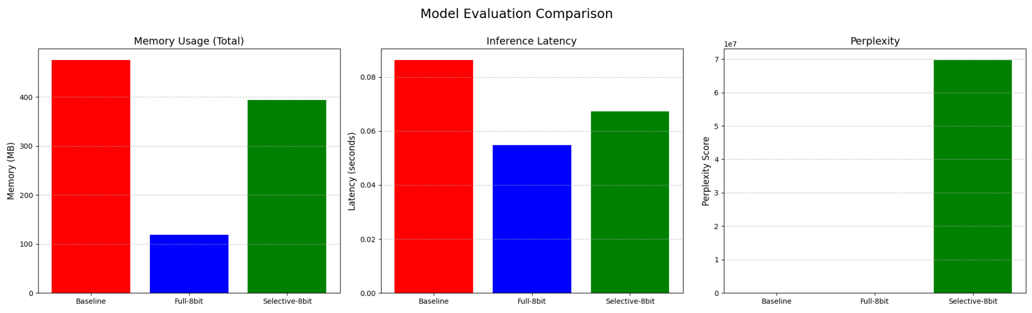

Figure 1: Custom Quantization Performance Comparison

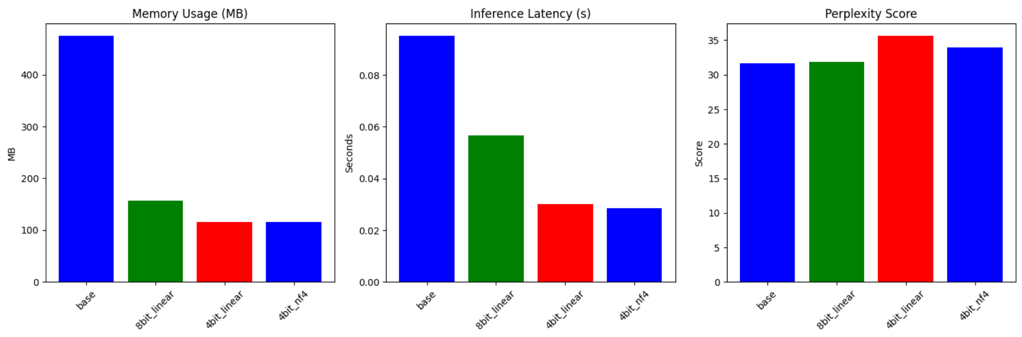

Figure 2: BitsAndBytes Quantization Performance Comparison

5 Observations and Questions

5.1 Analysis of Whole-Model and Selective Quantization

Quantization methods like whole-model quantization and selective quantization have distinct effects on model performance metrics, as shown in the results below:

- **Whole-Model Quantization:**
  - **Memory Usage:** There is a significant reduction in memory usage, dropping from 474.7 MB (baseline) to 118.97 MB for full 8-bit quantization.
  - **Latency:** Latency improves with whole-model quantization, decreasing from 0.0862 sec to 0.0549 sec, likely due to a reduction in model size and computational load.
  - **Perplexity:** However, perplexity becomes infinitely large (inf), indicating that full 8-bit quantization has caused a severe degradation in model accuracy, likely due to uniform quantization, especially of those layers and embedding matrix that don't need to be quantized.
- **Selective Quantization:**
  - **Memory Usage:** Memory usage is reduced to 393.7 MB, a more moderate reduction compared to the full 8-bit quantization. This is because only specific parts of the model are quantized, preserving some of the original precision.
  - **Latency:** Latency increases slightly, from 0.0549 sec (whole-model) to 0.0673 sec (selective), but this is still lower than the baseline latency of 0.0862 sec.
  - **Perplexity:** Perplexity increases drastically, reaching a value of 69,637,184. This extremely high perplexity suggests that selective quantization, while preserving some model precision, still leads to substantial performance degradation, particularly due to the quantization of critical parts of the model.

Whole-Model Quantization results in substantial memory savings and lower latency but causes severe accuracy loss, indicated by an infinite perplexity. **Selective Quantization** offers moderate memory savings with a slight increase in latency but also leads to a significant degradation in model performance, as seen in the extremely high perplexity score.

5.2 NF4 Quantization and Its Difference from Linear Quantization Scales

NF4 (Nonlinear Format 4-bit) quantization is a nonlinear quantization method designed to address the limitations of linear quantization by better handling weight distributions with a large dynamic range, often observed in transformer models.

- **Nonlinear Scale:**
  - In NF4 quantization, the quantization scale is adjusted dynamically and nonlinearly based on the input's distribution.
  - It uses a logarithmic-like mapping instead of a uniform mapping, which allows the representation of both small and large values with higher precision.

- The method leverages statistical properties of data, ensuring that more bits are allocated to commonly occurring values (near the center of the distribution), while fewer bits are allocated to outliers.

- **Adaptability:**
  - NF4 quantization is particularly suited for neural networks, where weights and activations often follow a non-uniform distribution, such as Gaussian or heavy-tailed distributions.
- **Enhanced Accuracy:**
  - This method minimizes the quantization error by preserving critical details in the weights that influence the model's predictions, thereby achieving better performance (e.g., lower perplexity in language models) compared to linear quantization.

Linear quantization, in contrast, uses a uniform quantization scale.

- **Uniform Scale:**
  - In this approach, the range of values (e.g., weights or activations) is divided into equally spaced intervals, and all values are mapped to the nearest interval.
  - The formula for mapping is:

$$q(x) = \text{round}\left(\frac{x - \min}{\text{step size}}\right),$$

    where the *step size* is calculated as:

$$\text{step size} = \frac{\max - \min}{2^b},$$

    and b is the number of bits used for representation.
- **Limitations:**
  - Uniform spacing results in poor precision for outliers, which can lead to significant quantization error.
  - It is less effective when the input data distribution is highly skewed or contains many outliers, common in large neural network models.

Transformers often have weight and activation distributions that are not uniformly distributed due to the nature of their multi-head attention and feed-forward mechanisms.

- NF4's ability to dynamically allocate precision ensures that **critical model parameters are better preserved**, resulting in:
  - **Improved perplexity** for language tasks.
  - **Minimal latency increase**, as the nonlinearity in NF4 is computationally efficient.
- By contrast, linear quantization is more prone to errors in these scenarios, leading to greater accuracy degradation, especially in tasks requiring high sensitivity to model parameters.

5.3 Impact of Linear vs. Nonlinear Quantization on Model Accuracy and Efficiency

- **Linear Quantization:**
  - **8-bit Linear Quantization:** Linear quantization at 8 bits reduces the model size significantly, from 474.7 MB (baseline) to 156.35 MB. This is a substantial memory savings. The latency improves to 0.0566 sec compared to the baseline latency of 0.0951 sec. However, there is a slight increase in perplexity (31.83) compared to the baseline perplexity (31.62), indicating a small drop in accuracy. The trade-off here is reduced memory usage and slightly faster inference at the cost of a minor increase in perplexity.
  - **4-bit Linear Quantization:** For 4-bit linear quantization, the model size further reduces to 115.85 MB, leading to even greater memory savings. The latency decreases to 0.0302 sec, a significant reduction compared to both the baseline and 8-bit linear models. However, the perplexity increases significantly to 35.62, indicating a noticeable loss in model accuracy. While the reduction in memory and latency is substantial, the drop in perplexity demonstrates that 4-bit linear quantization introduces a higher level of error, likely due to the greater compression of the model weights.
- **Nonlinear Quantization (NF4):**
  - **4-bit NF4 Quantization:** In contrast to linear quantization, nonlinear quantization (NF4) at 4 bits has a slightly lower memory usage compared to the 4-bit linear model (both are 115.85 MB). The latency improves even further to 0.0284 sec, which is the fastest among all the models tested. Despite this, the perplexity increases slightly to 33.94, which is lower than the 4-bit linear model's perplexity (35.62). This indicates that the NF4 quantization offers a more efficient trade-off in terms of both memory and accuracy compared to linear quantization at the same bit-depth. While the modelâs accuracy still suffers with a higher perplexity than the baseline, the NF4 quantization offers better performance efficiency.

5.4 Observations and Analysis of Quantization Impact on Model Performance

The three quantization approaches evaluated are:

- **Baseline Model**
- **Whole-Model Quantization**
- **Selective Quantization**
- **Linear and Nonlinear (NF4) Quantization**

5.5 Memory Usage

Memory usage refers to the amount of memory required to store the model's weights and activations during inference. This is a critical factor for deploying models on resource-constrained devices.

- **Baseline Model:** The baseline model consumes 474.7 MB of memory, which is the reference point for comparison.
- **Whole-Model Quantization:** Whole-model quantization, using 8-bit precision, leads to a significant reduction in memory usage. The memory usage is reduced to 118.97 MB, a reduction of about 75%.
- **Selective Quantization:** Selective quantization focuses on quantizing only certain layers of the model, leading to a more moderate memory reduction. The memory usage is 393.7 MB, a reduction of about 17%.
- **Linear and Nonlinear Quantization:** Linear 8-bit quantization reduces memory to 156.35 MB, while 4-bit linear quantization further reduces memory to 115.85 MB. Nonlinear (NF4) quantization at 4-bit yields a similar memory usage of 115.85 MB.

5.6 Latency (Speed)

Latency refers to the time taken by the model to make a prediction, which directly affects real-time applications. Lower latency is desired for faster response times.

- **Baseline Model:** The baseline model has a latency of 0.0951 seconds.
- **Whole-Model Quantization:** The full 8-bit quantization improves the latency, reducing it to 0.0549 seconds, a speedup of around 42%.
- **Selective Quantization:** Selective quantization results in a slight increase in latency (0.0673 seconds), though this is still faster than the baseline model. The increase in latency could be due to the overhead of quantizing only selected layers rather than the entire model.
- **Linear and Nonlinear Quantization:** Linear quantization at 8-bit reduces latency to 0.0566 seconds, while 4-bit linear quantization further reduces it to 0.0302 seconds. Nonlinear 4-bit NF4 quantization achieves the lowest latency at 0.0284 seconds, indicating that nonlinear quantization helps achieve faster inference speeds.

5.7 Perplexity (Accuracy)

Perplexity is a common metric for evaluating the performance of language models, where lower values indicate better accuracy. We discuss the change in perplexity after quantization to understand the trade-off between accuracy and efficiency.

- **Baseline Model:** The baseline model has a perplexity of 31.62.
- **Whole-Model Quantization:** The whole-model quantization results in a significant increase in perplexity, reaching "inf" (infinity) for the 8-bit quantization. This suggests a severe degradation in model accuracy, likely due to the quantization noise introduced in all layers.
- **Selective Quantization:** Selective quantization shows a large increase in perplexity, with a value of 69,637,184.0. This indicates that the model's performance has degraded substantially, likely due to the noise introduced by quantizing only specific layers.

- **Linear and Nonlinear Quantization:** Linear quantization at 8-bit results in a slight increase in perplexity (31.83), while 4-bit linear quantization leads to a larger increase in perplexity (35.62). Nonlinear (NF4) quantization at 4-bit gives a perplexity of 33.94, indicating that while accuracy decreases with lower bit-depths, NF4 quantization offers a better balance than 4-bit linear quantization.