Reducing the Size of Large Language Models Using Quantization

Sondos El-hariry

October 2025

1 Introduction

Modern transformer-based models like **BERT**, **GPT**, and **LLaMA** are powerful but extremely **resource-intensive**, often requiring **gigabytes of storage** and **significant memory** to load. This makes them **difficult to deploy** on edge devices, mobile phones, or even memory-limited servers.

Quantization is one of the most effective techniques to reduce the size and memory footprint of these models — with minimal performance loss.

2 Motivation

A typical BERT-base model uses **32-bit floating-point (FP32)** weights:

- **110 million parameters** $\times$ **4 bytes** = $\sim$**440 MB** just for weights.

By **quantizing to 8-bit integers (INT8)**, we reduce this to:

- **110 million** $\times$ **1 byte** = $\sim$**110 MB** $\rightarrow$ **4**$\times$ **smaller!**

3 Quantization: Core Idea

Quantization reduces the **precision** of the numbers used to represent the model's weights and activations.

Instead of using full **FP32**, we can convert weights to:

- **INT8** $\rightarrow$ 8-bit integer
- **INT4**, even **binary** in extreme cases

4 Mathematical Formulation: Quantization Formula

Quantization maps a float x to a lower-bit representation q using:

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

$$x \approx q \times \text{scale} + \min$$

Where:

- $\text{scale} = \frac{\max - \min}{2^k - 1}$
- k is the number of bits (e.g., 8 for INT8)

5 Code Example: Quantizing BERT using Hugging Face + bitsandbytes

Listing 1: Quantizing BERT to 8-bit

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

model_name =  bert-base-uncased

# Load model in 8-bit mode using bitsandbytes
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    load_in_8bit=True,  # <-- Quantization enabled
    device_map= auto
)

tokenizer = AutoTokenizer.from_pretrained(model_name)

# Example inference
inputs = tokenizer( hello world , return_tensors= pt ).to( cuda )
outputs = model(**inputs)

print( Logits: , outputs.logits)
```

Requirements: `pip install bitsandbytes accelerate`
Works best on models supported by **AutoGPTQ** or **bitsandbytes**

6 Performance Comparison

Model	Precision	Size (MB)	Speed
BERT (FP32)	32-bit	440 MB	Slow
BERT (INT8)	8-bit	110 MB	Fast
BERT (INT4)	4-bit	∼55 MB	Faster

Table 1: Quantization performance comparison

You can create this visualization with `matplotlib.pyplot.bar()` and display in your notebook.

7 Real-World Examples

- **Meta's LLaMA-2** is frequently quantized to **4-bit** for fast inference on laptops.
- **GGML / GPTQ** projects allow 13B+ models to run on **8GB RAM** machines using quantized formats.
- **DistilBERT** already reduces size via architectural changes — combine that with quantization for *super lightweight* deployments.

8 Conclusion

Quantization provides a powerful, deployment-friendly tradeoff:

- Up to **4–8× smaller model sizes**
- Lower memory & storage usage

- Faster inference
- Slight accuracy loss (often <1%)

$$\boxed{\text{Quantization} = \text{Compression without Compromise}}$$