REDUCING LARGE LANGUAGE MODEL SIZE USING QUANTIZATION

Nouran El-Shora

1. Introduction

Modern Natural Language Processing (NLP) models such as **BERT** and **LLaMA** achieve state-of-the-art performance in tasks like text classification, translation, and question answering.

However, these models contain **hundreds of millions to billions of parameters**, making them:

- Memory-intensive
- Computationally expensive
- Difficult to deploy on edge devices (e.g., mobile phones, embedded systems)
- Power-hungry

For example:

- BERT-base ≈ 110 million parameters
- LLaMA-7B ≈ 7 billion parameters

Storing each parameter in **32-bit floating point (FP32)** format requires significant memory.

2. The Core Problem

If a model has:

$$N \text{ parameters}$$

Stored in 32-bit floating point format:

$$\text{Model Size} = N \times 32 \text{ bits}$$

For a 7B parameter model:

$$7 \times 10^9 \times 32 \text{ bits} = 224 \text{ billion bits}$$
$$= 28 \text{ GB approximately}$$

This makes deployment on embedded systems or small GPUs nearly impossible.

3. Solution: Quantization

3.1 What is Quantization?

Quantization reduces the number of bits used to represent model parameters.

Instead of:

FP32 (32-bit float)

We use:

- FP16 (16-bit float)
- INT8 (8-bit integer)
- INT4 (4-bit integer)

This reduces:

- Memory usage
- Bandwidth
- Inference latency

3.2 Mathematical Formulation

Given a floating-point weight w, quantization maps it to an integer value q:

$$q = \text{round}\left(\frac{w}{s}\right)$$

Where:

- s= scaling factor
- q= quantized integer value

To recover the approximate value:

$$\hat{w} = q \times s$$

This introduces small approximation error but greatly reduces storage cost.

4. Types of Quantization

4.1 Post-Training Quantization (PTQ)

- Applied after training
- No retraining required
- Fast and simple

4.2 Quantization-Aware Training (QAT)

- Simulates quantization during training
- More accurate
- Slightly more complex

5. Memory Comparison

Format	Bits per Parameter	Size Reduction
FP32	32 bits	1×
FP16	16 bits	2× smaller
INT8	8 bits	4× smaller
INT4	4 bits	8× smaller

Example for 7B model:

- FP32 → 28 GB
- INT8 → 7 GB
- INT4 → 3.5 GB

6. Coding Example – PyTorch Quantization

Example 1: Dynamic Quantization (Post-Training)

```
import torch
import torch.nn as nn
import torchvision.models as models

# Load pretrained model
model = models.resnet18(pretrained=True)

# Apply dynamic quantization
quantized_model = torch.quantization.quantize_dynamic(
    model,
    {nn.Linear},
    dtype=torch.qint8
)

# Compare model sizes
def model_size(model):
    torch.save(model.state_dict(), "temp.p")
    size = os.path.getsize("temp.p") / 1e6
    return size
```

```
import os
print("Original model size:", model_size(model), "MB")
print("Quantized model size:", model_size(quantized_model), "MB")
```

Example 2: Quantizing a Transformer Model (HuggingFace)

```
from transformers import AutoModelForSequenceClassification
import torch

model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")

quantized_model = torch.quantization.quantize_dynamic(
    model,
    {torch.nn.Linear},
    dtype=torch.qint8
)

print("Model successfully quantized.")
```

7. Advanced Quantization – 4-bit (LLMs)

Modern large models like LLaMA often use:

- 8-bit quantization
- 4-bit quantization (QLoRA)

Libraries used:

- bitsandbytes
- GPTQ

Example:

```
from transformers import AutoModelForCausalLM

model = AutoModelForCausalLM.from_pretrained(
    "meta-llama/Llama-2-7b-hf",
    load_in_4bit=True,
    device_map="auto"
)

print("4-bit quantized model loaded.")
```