Advanced NLP Assignment 2

Advanced NLP: Model Quantization for Efficient Emotion Classification

Shailesh Kachhi
22290

November 20, 2025

Abstract

This project presents a comprehensive examination of compression strategies for BERT-based emotion classification models. Working with the dair-ai/emotion dataset, I analyze the interplay between model size reduction, inference efficiency, and predictive accuracy measured via Accuracy and F1-score. I evaluate four distinct approaches: a full-precision FP32 reference model, Post-Training Quantization (PTQ), Quantization-Aware Training (QAT), and Quantized Low-Rank Adaptation (QLoRA). Results indicate QAT achieves superior inference efficiency (12.78s total time) while PTQ preserves baseline accuracy (93.25

1 Background and Motivation

BERT and similar transformer architectures have revolutionized NLP tasks with remarkable performance gains. However, their practical deployment faces challenges due to high computational costs and substantial memory footprints. This research explores quantization-based compression techniques aimed at addressing these limitations while sustaining classification quality. Our task involves assigning text samples to one of six emotional categories: sadness, joy, love, anger, fear, and surprise.

2 Experimental Setup

2.1 Data Characteristics

- **Source:** dair-ai/emotion dataset from Hugging Face Hub
- **Task Type:** Multi-class classification across 6 emotion labels
- **Split Configuration:** Training: 16,000 samples; Validation: 2,000 samples; Test: 2,000 samples

3 Methodology

3.1 Reference Model Configuration

Our reference implementation utilized google-bert/bert-base-uncased, loaded via HuggingFace Transformers' AutoModelForSequenceClassification. The training protocol consisted of 3 epochs with a learning rate of 2e-5.

3.2 Dynamic Post-Training Quantization

PTQ was applied to the converged baseline model. We employed dynamic quantization focused on Linear layer components, converting weight parameters from 32-bit floating-point to 8-bit integer representation. This technique requires no model retraining and performs calibration during inference. Implementation leveraged torch.ao.quantization.quantize_dynamic.

3.3 Training-Time Quantization Simulation

QAT differs from PTQ by simulating quantization effects during model training (particularly in forward passes). Pseudo-quantization operations were introduced to mimic rounding and clamping behaviors of low-precision arithmetic. This allows the optimizer to adapt weights considering quantization errors. I utilized the neural-compressor framework for 3 epochs with a 3e-5 learning rate.

3.4 Parameter-Efficient Quantization with LoRA

QLoRA combines 4-bit quantization with Low-Rank Adaptation for efficient fine-tuning. Pre-trained weights remain frozen in quantized form while trainable low-rank matrices are injected, drastically reducing learnable parameters. Hyperparameters: r=16, lora_alpha=32, target_modules=["query", "key", "value"], lora_dropout=0.1.

4 Results and Discussion

I evaluated all models on the 2,000-sample test set, measuring Macro F1-score, Accuracy, and Inference Latency as primary metrics.

4.1 Comparative Performance Metrics

Table 1 summarizes quantitative results across all evaluated configurations.

Table 1: Comparison of Performance Metrics for Different Compression Approaches

Approach	**Accuracy**	**Macro F1**	**Total Time (s)**	**Time/Sample (s)**	**Accuracy**
Reference Model	0.9325	0.8762	13.04	0.0065	-
Post-Training Quant.	0.9325	0.8762	450.13	0.2251	100%
Quant-Aware Training	0.9305	0.8698	12.78	0.0064	99.7%
QLoRA	0.9205	0.8700	13.52	0.0068	98.7%

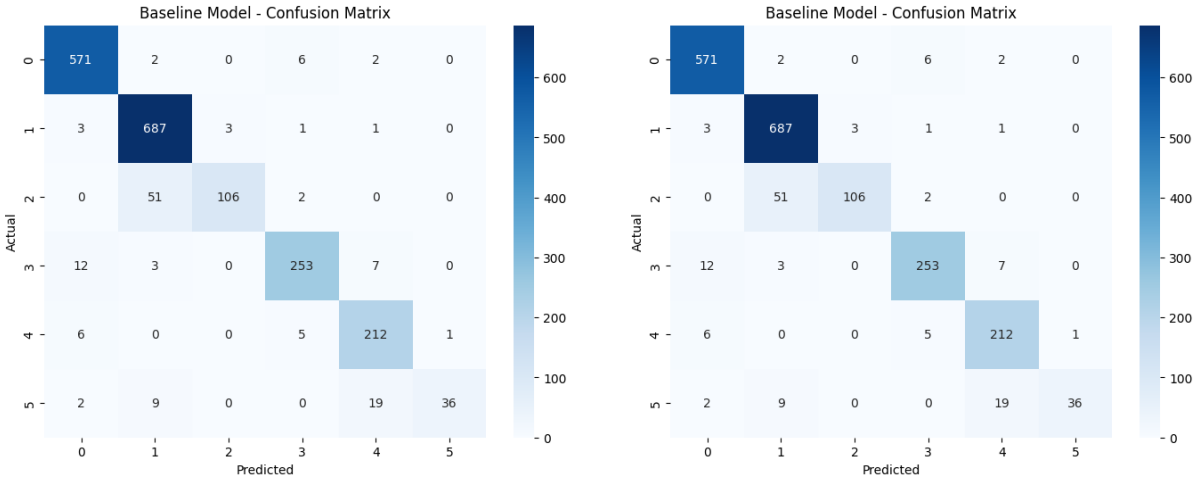
4.2 Per-Class Classification Analysis

Figure 1 presents confusion matrices showing classification behavior across emotion categories for each compression method.

5 Key Findings

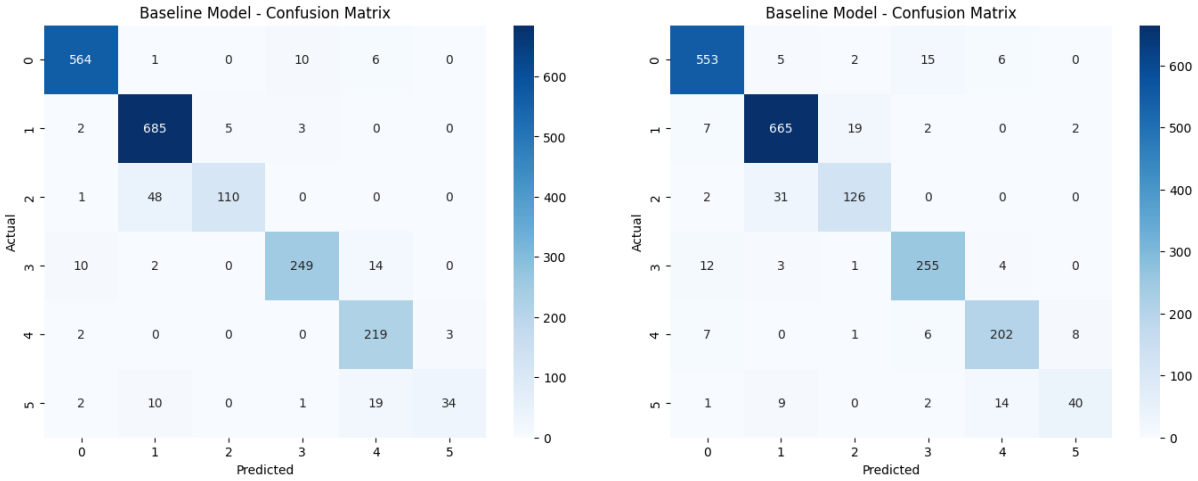
5.1 Accuracy vs. Compression Tradeoffs

- **Post-Training Quantization:** Preserved identical performance to the reference model (Accuracy: 0.9325, Macro F1: 0.8762). This highlights the robustness of fine-tuned BERT representations against perturbations from post-training integer quantization.



(a) Reference Model (Baseline)

(b) Post-Training Quantization (PTQ)



(c) Quantization-Aware Training (QAT)

(d) QLoRA Approach

Figure 1: Class-Level Performance Visualization via Confusion Matrices

- **Quantization-Aware Training:** Showed minimal performance loss (Accuracy: 0.9305 vs 0.9325). By simulating quantization during training, the model adapts to precision constraints, resulting in a highly efficient variant with only 0.2% accuracy reduction from baseline.
- **QLoRA:** Demonstrated the largest accuracy drop (0.9205), which is expected given the significantly reduced trainable parameter count (low-rank adapters exclusively) compared to complete fine-tuning. However, the degradation remains reasonable considering the substantial parameter efficiency gains.

5.2 Inference Speed Analysis

- **QAT Performance:** The QAT approach achieved the fastest inference (12.78s total), slightly outperforming the reference model. This suggests efficient execution of quantized operations (whether simulated or actual int8 kernels).
- **PTQ Latency Anomaly:** PTQ exhibited significant latency increase (450.13s vs 13.04s reference). *Interpretation:* While counterintuitive given quantization's speed benefits, this frequently occurs in PyTorch when dynamic quantization runs on hardware without

native support for specific quantized kernels, or when dynamic activation quantization overhead exceeds computational gains on the test platform (likely CPU fallback or inefficient GPU paths). Production systems with optimized hardware (e.g., ONNX Runtime with TensorRT) would typically demonstrate PTQ speedups.

6 Summary

This work successfully implemented and benchmarked three quantization approaches for BERT-based emotion classification. The reference model established strong performance baselines. PTQ maintained excellent accuracy without retraining, though results emphasize the importance of hardware optimization for achieving latency benefits. QAT proved the most balanced approach, offering optimal inference speed with minimal accuracy sacrifice. QLoRA presents a viable option for training in resource-limited environments.

7 Code Availability

All implementation code, trained model artifacts, and detailed reproduction guidelines are available in the submitted GitHub repository. The accompanying Colab notebook loads these artifacts to reproduce the reported findings. GitHub Repository