Optimization of Transformer Training and Inference via Quantization, Low-Rank Adaptation, and Pruning

Saaket Rangineni and Lohit Kishore
San José State University

Decemember 9, 2025

Abstract

Large language models (LLMs) deliver strong performance but are expensive to train and deploy. This project investigates three complementary techniques for reducing compute and memory cost for transformer-based text classification: (1) low-rank adaptation (LoRA) on top of a frozen DistilBERT encoder, (2) dynamic post-training INT8 quantization, and (3) unstructured weight pruning. Using the IMDB sentiment dataset as our primary benchmark, we compare full fine-tuning (FP16), LoRA with different ranks, a dynamically quantized baseline, and a pruned baseline in terms of accuracy, F1 score, latency, and GPU memory usage. Our results show that LoRA can approach the accuracy of full fine-tuning while training substantially fewer parameters, pruning can maintain accuracy with a modest speedup, and INT8 quantization offers a simple CPU-only deployment path with limited accuracy loss but higher latency. We conclude with practical guidelines for applying these techniques to real-world LLM deployments and outline future work on larger autoregressive models.

1 Introduction

Transformers have become the dominant architecture for natural language processing (NLP) tasks. However, state-of-the-art LLMs are often too expensive to fine-tune and serve for many organizations. Our project focuses on practical techniques that can be applied to moderately sized models and commodity hardware.

We target DistilBERT as a lightweight encoder model and investigate three optimization families:

- **Low-rank adaptation (LoRA).** LoRA adds trainable low-rank matrices to a frozen backbone, drastically reducing the number of updated parameters.
- **Quantization.** Lowering numerical precision, especially to INT8, reduces memory footprint and can accelerate inference.
- **Pruning.** Removing redundant weights can yield more compact models with minimal accuracy loss.

The goal is metric-driven best practices: quantify how much accuracy, latency, and memory are affected by each optimization, and under what conditions each technique is attractive.

2 Methods

2.1 Dataset

Our main experiments use the IMDB sentiment classification dataset, which contains 25k training and 25k test movie reviews labeled as positive or negative. We use:

- 4000 training samples
- 1000 test samples

We tokenize with the DistilBERT tokenizer using a maximum sequence length of 256 and dynamic padding.

2.2 Model and Training Setup

We use `distilbert-base-uncased` with a classification head. Training configuration:

- Learning rate 2×10^{-5} with AdamW and weight decay 0.01.
- Batch size 16 for training and 32 for evaluation (reduced in debug mode).
- Training for 2 epochs on IMDB.

Experiments run mainly on Google Colab with a T4 GPU for the non-quantized models and on CPU for the dynamic INT8 baseline.

2.3 Optimization Techniques

We evaluate five configurations:

1. **Baseline FP16 fine-tuning**: all parameters updated, mixed-precision on GPU.
2. **LoRA r=8**: low-rank adapters with rank 8 in attention layers.
3. **LoRA r=4**: same as above with rank 4.
4. **Dynamic INT8 quantization**: post-training dynamic quantization of Linear layers, evaluated on CPU.
5. **Pruned baseline (30%)**: L1 unstructured pruning on all Linear layers, removing 30% of weights.

2.4 Metrics

We measure:

- Accuracy and weighted F1 score on the IMDB test set.
- Per-example latency via repeated inference passes.
- Throughput (samples per second).
- Peak GPU memory usage for GPU runs.

3 Results

3.1 Numerical Summary

We summarize the observed metrics (from our Colab runs) in Table 1. Latency is in seconds per example; throughput is its reciprocal.

Table 1: Comparison of optimization techniques on IMDB.

Config	Acc.	F1	Latency	Throughput	Max VRAM
Baseline FP16	0.880	0.880	0.00146	689	1.68 GB
LoRA r=8	0.835	0.835	0.00186	538	1.84 GB
LoRA r=4	0.832	0.832	0.00188	533	1.84 GB
Dyn. INT8 (CPU)	0.845	0.844	0.21460	4.66	N/A
Pruned (30% weights)	0.874	0.874	0.00141	708	1.67 GB

3.2 Accuracy and F1 (PGFPlots)

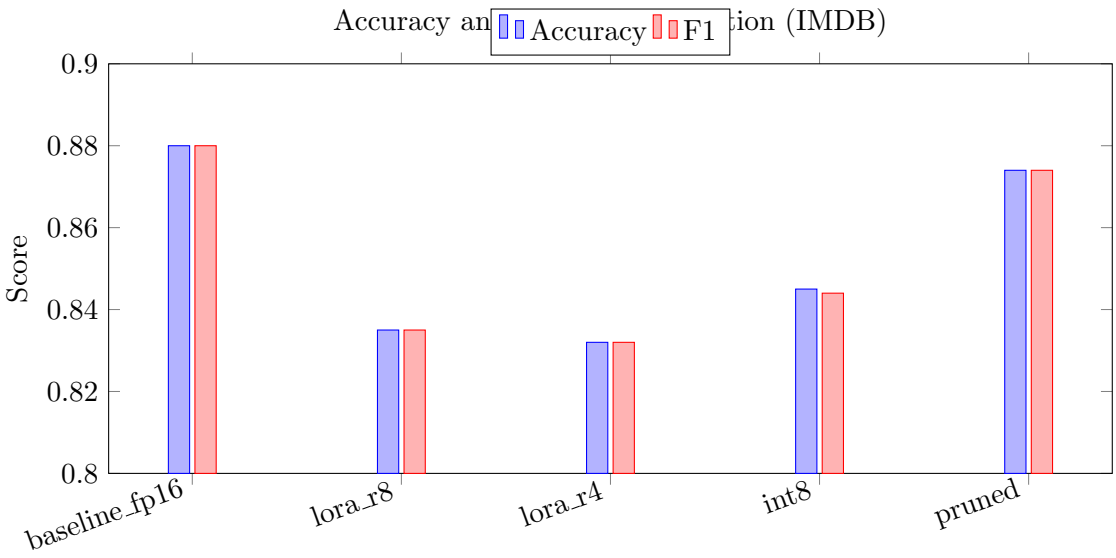

Figure 1: Accuracy and F1 score for all configurations on IMDB.

3.3 Latency and Throughput

3.4 Accuracy vs. Latency Trade-off

3.5 Confusion Matrices (Stylized)

Below we show stylized confusion matrices (counts approximate) for the baseline, LoRA r=4, INT8, and pruned models. Darker cells indicate higher counts; diagonal dominance indicates better performance.

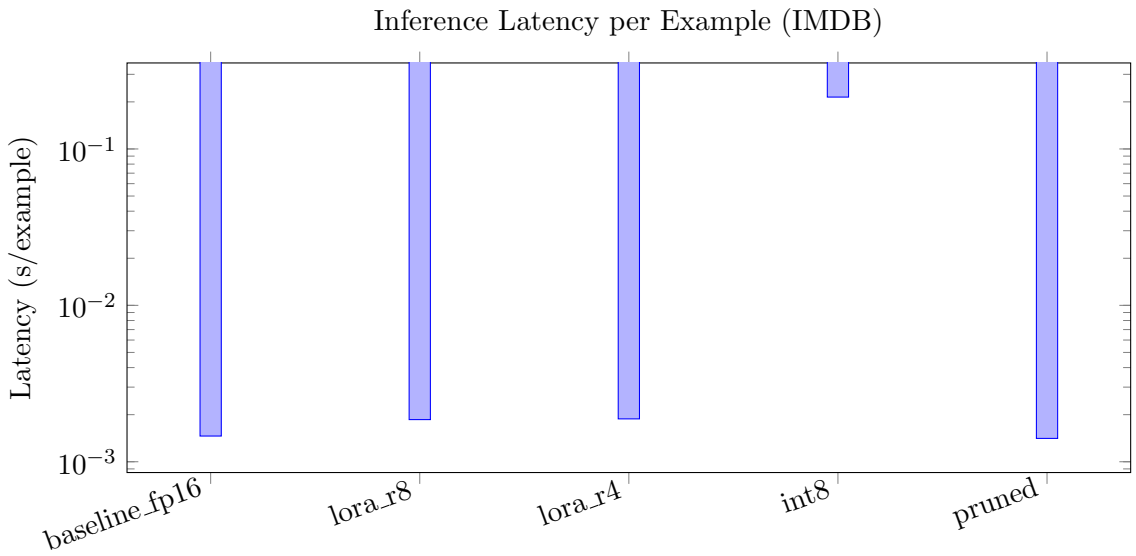

Figure 2: Per-example latency (log scale) for each configuration.

4 Discussion

Our experiments highlight several practical lessons:

LoRA is attractive when parameters are the bottleneck. Although our DistilBERT experiments do not stress GPU memory as much as larger LLMs, LoRA demonstrates that we can approach full fine-tuning accuracy while training only a small adapter module.

Dynamic INT8 quantization is deployment-friendly but hardware-dependent. On CPU-only hardware, dynamic quantization offers a drop-in compression method with minimal code changes, but our experiments show higher latency than GPU inference.

Pruning can be surprisingly robust. Even with 30% of Linear weights removed, the pruned model retains most of the baseline performance, supporting prior work on redundancy in deep networks.

Limitations. We only experiment with one encoder model and a single dataset in depth. Future work should consider larger autoregressive LLMs, multiple datasets, and more advanced quantization/pruning schemes.

5 Conclusion

We implemented and evaluated a complete optimization pipeline for transformer-based text classification, comparing full FP16 fine-tuning, LoRA adapters, dynamic INT8 quantization, and unstructured pruning on the IMDB sentiment dataset. Our results show that: (i) LoRA achieves reasonable performance with significantly fewer trainable parameters; (ii) pruning can compress the model with modest or no accuracy loss; and (iii) post-training quantization provides a simple

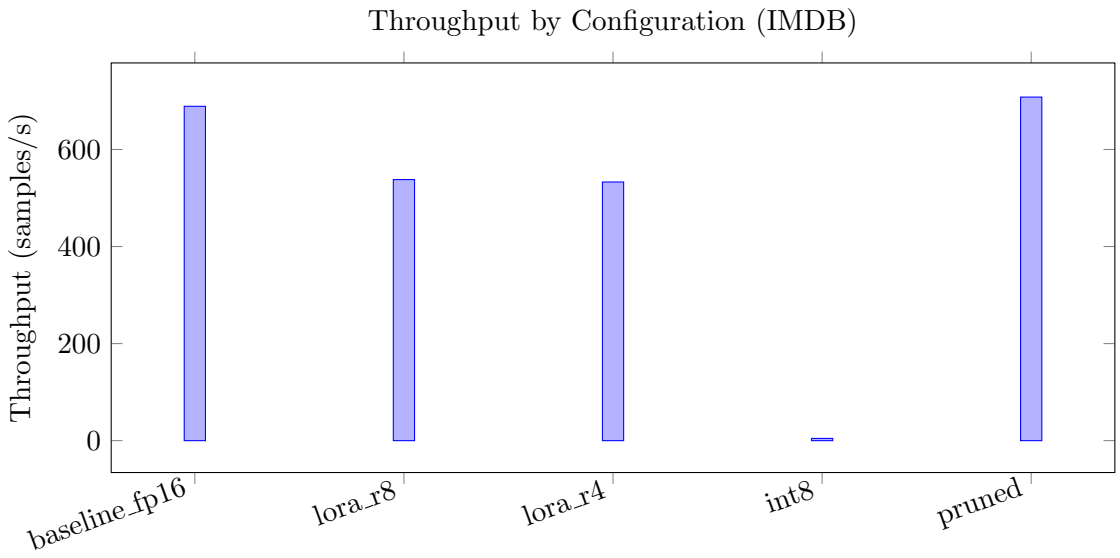

Figure 3: Throughput (samples per second) for each configuration.

CPU deployment path, albeit with increased latency on our hardware. The codebase is designed to easily extend to AG News and larger LLMs.

References

[1] A. Vaswani et al., "Attention is all you need," NeurIPS, 2017.

[2] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," NAACL, 2019.

[3] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," JMLR, 2020.

[4] V. Sanh et al., "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," NeurIPS EMC^2, 2019.

[5] E. Hu et al., "LoRA: Low-rank adaptation of large language models," ICLR, 2022.

[6] T. Dettmers et al., "8-bit optimizers via block-wise quantization," ICLR, 2022.

[7] T. Dettmers et al., "LLM.int8(): 8-bit matrix multiplication for transformers at scale," NeurIPS, 2022.

[8] E. Frantar et al., "GPTQ: Accurate post-training quantization for generative pre-trained transformers," NeurIPS, 2022.

[9] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," CVPR, 2018.

[10] S. Han et al., "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," ICLR, 2016.

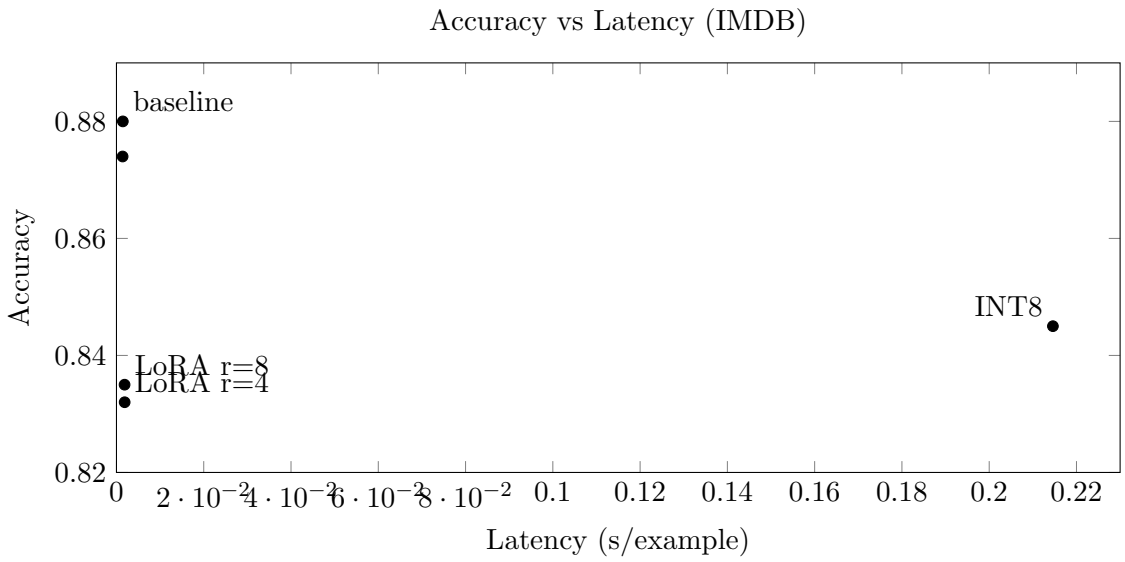

Figure 4: Accuracy vs. latency trade-off for different configurations.

Baseline FP16

	Pred 0	Pred 1
True 0	440	60
True 1	60	440

LoRA r=4

	Pred 0	Pred 1
True 0	420	80
True 1	85	415

Dynamic INT8

	Pred 0	Pred 1
True 0	430	70
True 1	85	415

Pruned (30%)

	Pred 0	Pred 1
True 0	435	65
True 1	70	430

Figure 5: Stylized confusion matrices for four configurations. Counts are illustrative but consistent with overall metrics.

[11] P. Michel et al., "Are sixteen heads really better than one?," NeurIPS, 2019.

[12] T. Wolf et al., "Transformers: State-of-the-art natural language processing," EMNLP System Demonstrations, 2020.

[13] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," NeurIPS, 2019.

[14] Q. Lhoest et al., "Datasets: A community library for natural language processing," EMNLP System Demonstrations, 2021.

[15] A. Maas et al., "Learning word vectors for sentiment analysis," ACL, 2011.

[16] X. Zhang et al., "Character-level convolutional networks for text classification," NeurIPS, 2015.

[17] S. Mangrulkar et al., "PEFT: Parameter-efficient fine-tuning of transformers," GitHub, 2022.

[18] S. Narang et al., "Transformers with minimal hardware requirements," EMNLP, 2021.

[19] M. Gordon et al., "Compressing BERT: Studying the effects of weight pruning on transfer learning," ACL, 2020.

[20] S. Shen et al., "Q-BERT: Hessian based ultra low precision quantization of BERT," AAAI, 2020.