Reimagining LoRA: Parameter-Efficient Tuning for NLP Tasks

Deniz Bölöni-Turgut **Tony Chen** **Allison Avila** **Audrey Wang**

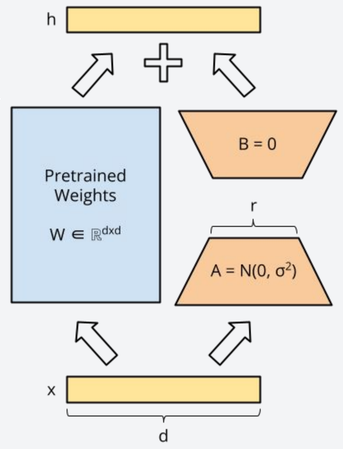
Background

Large language models such as GPT-2, RoBERTa, and GPT-3 have revolutionized natural language processing through pretraining on large corpora followed by task-specific fine-tuning. However, fine-tuning these models for each task requires updating all parameters, which becomes computationally expensive and impractical as model sizes scale into the billions of parameters.

The goal of this project is to replicate and evaluate LoRA (Low-Rank Adaptation), a parameter-efficient technique proposed to address the challenges of full fine-tuning. LoRA achieves task adaptation by freezing pretrained weights and injecting small, trainable low-rank matrices into existing layers - thereby significantly reducing the number of trainable parameters without sacrificing model quality.

Forward pass formula: $h = W_0x + \Delta Wx = W_0x + BAx$

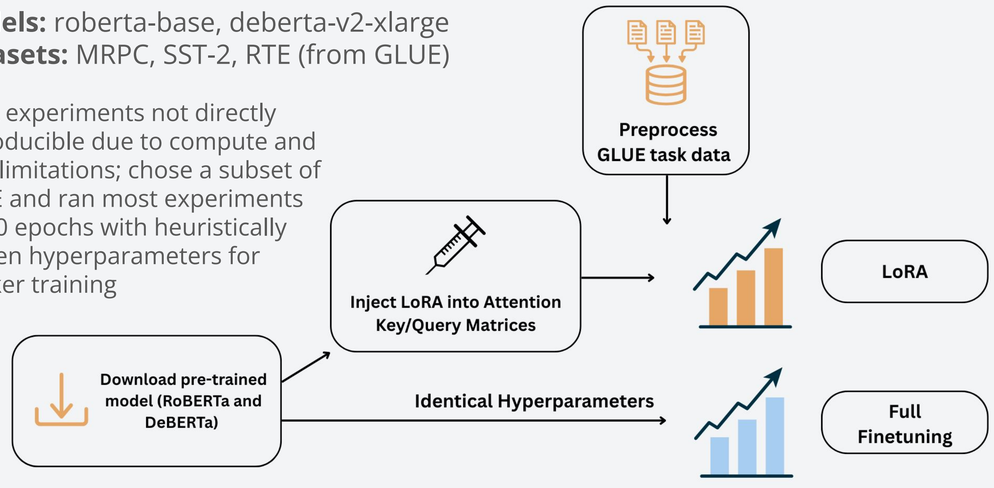
The LoRA paper introduces a novel and elegant method of injecting low-rank decompositions into attention layers of Transformer models. Its contributions include: (1) dramatically reduced trainable parameter count; (2) no increase in inference latency; (3) competitive or superior performance compared to other methods like adapters or prefix tuning; and (4) practical compatibility with a wide range of NLP architectures.



Methodology

Models: roberta-base, deberta-v2-xlarge
Datasets: MRPC, SST-2, RTE (from GLUE)

LoRA experiments not directly reproducible due to compute and time limitations; chose a subset of GLUE and ran most experiments for 10 epochs with heuristically chosen hyperparameters for quicker training



Results

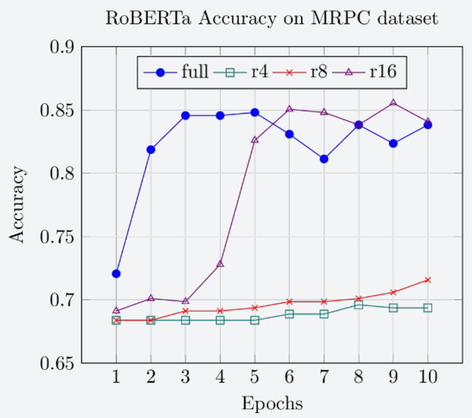


Figure 1: Comparison of RoBERTa model variants on the MRPC dataset across training epochs. full corresponds to fine-tuning all model weights. Aside from rank, identical hyperparameters are used across experiments.

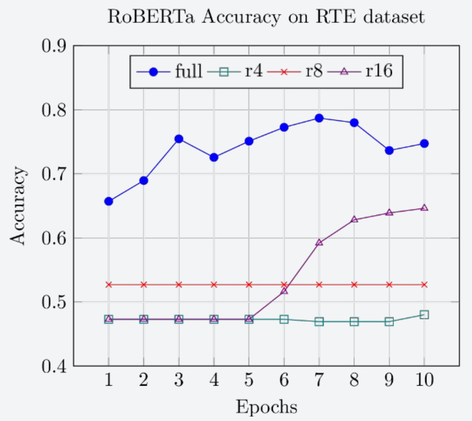


Figure 2: Comparison of RoBERTa model variants on the RTE dataset across training epochs.

r[x] represents training with injected LoRA layer with rank=alpha=x.

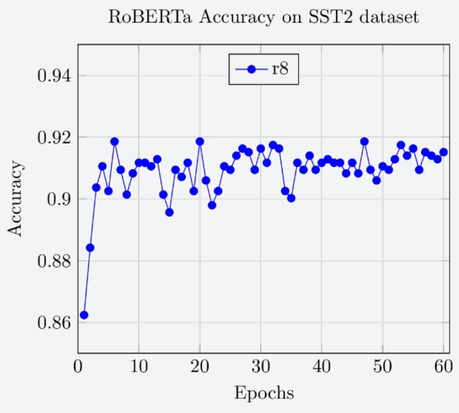


Figure 3: RoBERTa model on the SST2 dataset across 60 training epochs with injected LoRA layer of rank=alpha=8. Hyperparameters chosen according to Table 9 in [1]. Best accuracy 91.8% compared to 95.1% in Table 2 in [1].

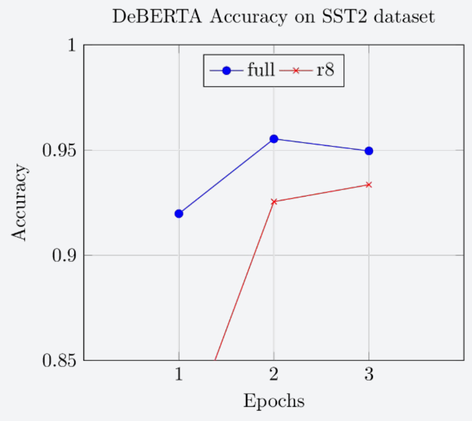


Figure 4: DeBERTa-v2-xlarge trained on the SST2 dataset for 3 training epochs with injected LoRA layer of rank=alpha=8 compared to the original model.

Conclusion

While we were not able to reproduce all of the experiments presented in [1], we made a couple key observations from our experiments.

LoRA greatly **reduces training time** across different models and datasets. For the experiment in Figure 4, training full DeBERTa model took 148 min compared with 101 min for LoRA model.

Optimizer states for LoRA models **require much less memory to store**. For RoBERTa model trained on MRPC (shown in Figure 1), optimizer state for full model was 951MB compared with 2.3MB for LoRA model with rank 8.

References

Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *ICLR* 1.2 (2022): 3.

Cornell University