Fine-Tuning GPT-2 on WikiText with Full Tuning and LoRA:

A Comparative Study

1. Introduction

The objective of this task is to fine-tune the "GPT-2" large language model (LLM) on the WikiText dataset using two approaches: full fine-tuning and Low-Rank Adapters (LoRA). LoRA is a Parameter-Efficient Fine-Tuning (PEFT) method that applies additional trainable parameters to a frozen pre-trained model, resulting in a more efficient fine-tuning process with fewer trainable parameters. This study compares the performance of both techniques in terms of quality, time, and resource utilization.

2. Dataset

The WikiText dataset is used for the fine-tuning process. It contains articles extracted from Wikipedia. A quick analysis of the WikiText dataset revealed that each word is typically represented by an average of 1.5 tokens, with most sequences not exceeding 512 tokens. These insights were essential for setting the max sequence length to 512 as well as in the process of producing the references and hypotheses, ensuring efficient memory usage and well-suited hyperparameter choices for fine-tuning and generation.

Dataset name	Train set	Validation set	Test set
wikitext-2-raw-v1	36718	3760	4358

3. Fine-Tuning Pipelines

Two pipelines were executed for comparison:

- **Full Fine-Tuning**: The entire model is trained on the WikiText dataset.
    - Command: `python main.py full --batch_size 4 --data_fraction 0.1 --epochs 2`
- **LoRA Fine-Tuning**: Only the LoRA parameters are trained, while the rest of the model remains frozen.
    - Command: `python main.py lora --batch_size 4 --data_fraction 0.1 --epochs 2 --lora_rank 4`

4. Comparison of Results

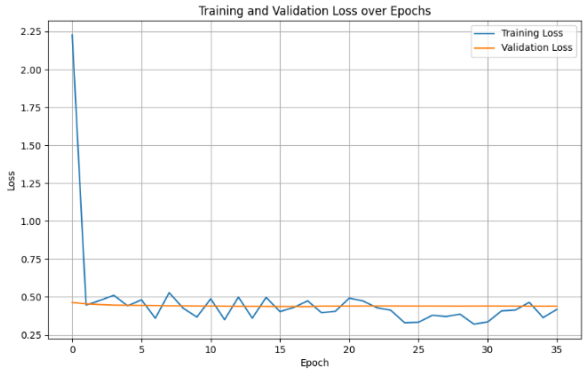
The table below compares the two fine-tuning approaches in terms of model size, trainable parameters, training time, memory usage, and performance metrics such as training loss, validation loss, and BLEU score.

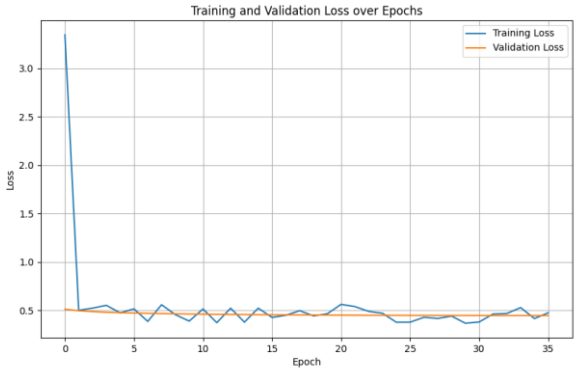
Metric	Full Fine-Tuning	LoRA Fine-Tuning
Model size (MB)	479	6.15
Trainable parameters (weights)	124,439,808	405,504
Training Time (s)	3694.07	2851.43

GPU memory usage (MB)	975.40 MB	19.34
CPU memory usage (MB)	257.81 MB	98.36
Best training loss	0.3004	0.3706
Best validation loss	0.4379	0.4443
bleu score	0.02	0.01
sacre-bleu score	1.36	1.33
Rouge1/rouge2/ rougeL/ rougeLsum	0.18/0.02/0.12/0.12	0.17/0.02/0.11/0.11
TER score	91.66	93.06
CHRF score	17.76	16.56
METEOR score	-	-

5. Losses during training

The following charts describe the loss dynamics over the training epochs for both full fine-tuning (on the left) and LoRA-based fine-tuning (on the right):





5. Discussion

From the comparison table and the training charts, several key points emerge:

- **Model Size**: As expected, LoRA fine-tuning significantly reduces the model size, resulting in a 6.15 MB model compared to 479 MB for full fine-tuning. This reduction is due to the fact that LoRA only trains a small set of additional parameters, while the rest of the pre-trained weights remain frozen.
- **Training Time and Memory Usage**: LoRA fine-tuning offers notable efficiency gains in both training time and memory usage. It completes the training in 2,851.43 seconds (compared to 3,694.07 seconds for full fine-tuning) and reduces GPU memory consumption by over 95%. This demonstrates the practicality of LoRA in resource-constrained environments, such as when training on consumer hardware with limited GPU memory.
- **Losses and Performance**:

- Both models exhibit convergence, but the full fine-tuning achieves slightly better training loss (0.3004 vs. 0.3706). However, the validation loss difference between the two approaches is minimal, indicating comparable generalization ability.
- The BLEU and SacreBLEU scores are very close between both approaches with slight upper hand to the full tuning approach, indicating comparable quality of text generation. However, both models score quite low on these metrics, suggesting potential more hyper parameterization and finetuning.
- ROUGE, TER, and CHRF scores also reveal minimal differences between the two methods, indicating that LoRA fine-tuning is competitive in terms of quality.
- Due to time constraints, the METEOR score was not included, but the evaluation code is ready to be executed.

- **Limitations**:
  - Due to hardware limitations (Intel i7-8750H and NVIDIA GTX 1060), training was performed on only 10% of the original dataset. This likely impacted the models' performance and scores.
  - Additional training epochs and hyperparameter optimization would likely improve both methods' performance.

7. Conclusion

In conclusion, LoRA fine-tuning proves to be a highly efficient alternative to full fine-tuning in terms of both memory and time, while achieving almost comparable performance in different metrics to the full tuning approach. This makes it a suitable option for environments where computational resources are limited. Further exploration with full dataset, hyper parameterization, and additional evaluation metrics (e.g., METEOR) would provide deeper insights into the comparative performance of both techniques.

8. Reproducibility

See Readme.md file in the project