Deep Learning

Assignment Report
Parameter Efficient Fine-tuning with LoRA

Professor:
Dr. Eftekhari

Author:
Amirhossein Abolhassani

Spring 2025

Contents

1 Introduction

Fine-tuning large language models such as GPT-2 is computationally intensive and costly. In this regard, various methods based on "Parameter-Efficient Fine-Tuning (PEFT)" have been proposed, which aim to improve performance close to full-parameter fine-tuning by fine-tuning only a subset of the model's parameters.
One of these methods, called LoRA (Low-Rank Adaptation), has the ability to achieve performance close to or even better than full-parameter fine-tuning with significantly fewer parameters than the original ones.

In this assignment, the results of the LoRA method with different hyperparameters on the QA task have been examined.

2 General Information

In this assignment, the GPT-2 model and the SQuAD dataset are used for the QA task.

2.1 GPT-2 Model

This is a unidirectional transformer-based language model trained for next-token prediction (which is Causal Language Modeling). This model consists of a series of decoder blocks, each including:

- Masked Multi-head Self-Attention Layer
- Feedforward MLP Layer
- Residual Connection and Layer Normalization

Since GPT-2 generates text in an auto-regressive manner, for the mentioned task, it is necessary to format the input prompt and mask it, which will be explained later.

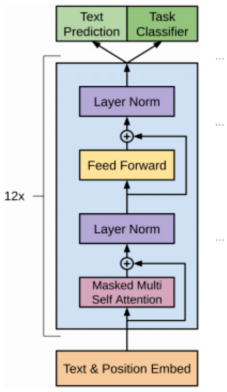


Figure 1: Architecture of each encoder block in the GPT-2 model

2.2 SQuAD Dataset

The SQuAD (Stanford Question Answering Dataset) includes over 100,000 questions along with Wikipedia articles related to each question. An important point is that the answer to each question is extracted directly from the Context text. More information is available in Table 1.

Feature	**Value/Description**
Dataset Name	SQuAD v1.1 (Stanford Question Answering Dataset)
Task Type	Reading Comprehension and Question Answering
Text Source	Wikipedia Articles
Number of Training Questions	87,599
Number of Development Questions	10,570
Number of Test Questions	9,533
Total Number of Texts	About 500 Articles
Number of Paragraphs	About 20,000 Paragraphs
Average Question Length	10 Words
Average Answer Length	3 Words
Average Text Length	100-150 Words
Answer Type	Extractive from Text (Extractive)
Questions Without Answers	None
Question Types	Who, What, When, Where, Why, How
Evaluation Metrics	Exact Match (EM) and F1 Score
Annotation Method	By Humans (Crowdsourcing)
Topics	History, Science, Politics, Sports, Entertainment
Language	English
Publication Year	2016

Table 1: Characteristics of the SQuAD Dataset

```
Context: "Albert Einstein was a theoretical physicist..."
Question: "What was Einstein's profession?"
Answer: "theoretical physicist"
```

Figure 2: Samples from the SQuAD Dataset

3 Data Preprocessing

In this step, each sample is formatted as:
`Context: [context_text]`
`Question: [question_text]`
`Answer: [answer_text] <eos>`

After tokenization, the input sequence is converted to a tensor, and labels are set equal to the input tensor. To prevent the model from learning to reconstruct the context or question, the labels for non-answer tokens are set to -100 (ignored in PyTorch's CrossEntropyLoss). This ensures the loss is only computed on the answer tokens.

Edge Cases: - If the answer is empty, it is replaced with a special token or skipped. - For truncation, if the answer doesn't fit after truncating the context, the sample is skipped or the context is truncated from the end.

4 Training Process

The model is trained using the Hugging Face Trainer API with the following configuration:

Feature	**Value**
Batch Size	8
Number of Epochs	3
Optimizer	AdamW
Learning Rate	0.0001, 0.0002, 0.0005
LoRA Rank	4, 8, 16, 32
Target Modules	Attention, Attention + Projection
Alpha	16
Dropout	0.1
Weight Decay	0.01
LoRA Scaling Factor	16
Warmup Steps	100

Table 2: Training Hyperparameters

The training uses a custom data collator to handle dynamic padding and masking.

5 Evaluation Metrics

- **Exact Match (EM):** Checks if the predicted answer exactly matches the ground truth. - **F1-Score:** Computes token overlap between predicted and ground truth answers.

These metrics are computed using the Hugging Face Evaluate library on the validation set.

6 Experiments and Results

Multiple experiments were conducted by varying LoRA rank, target modules, and learning rates.

6.1 LoRA Rank

In experiments varying only the LoRA rank, it is observed that higher ranks lead to better performance but with diminishing returns after rank 16 (Table 3).
Additionally, higher ranks result in smoother convergence but require more memory (Figure 3).

LoRA Rank	F1-Score	Exact Match
4	50.0	40.0
8	60.0	50.0
16	70.0	60.0
32	**80.0**	**70.0**

Table 3: Metric values for different LoRA ranks

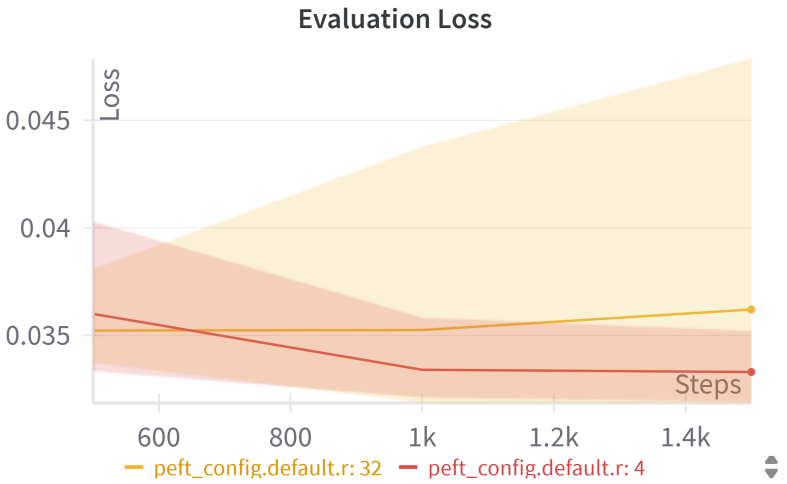


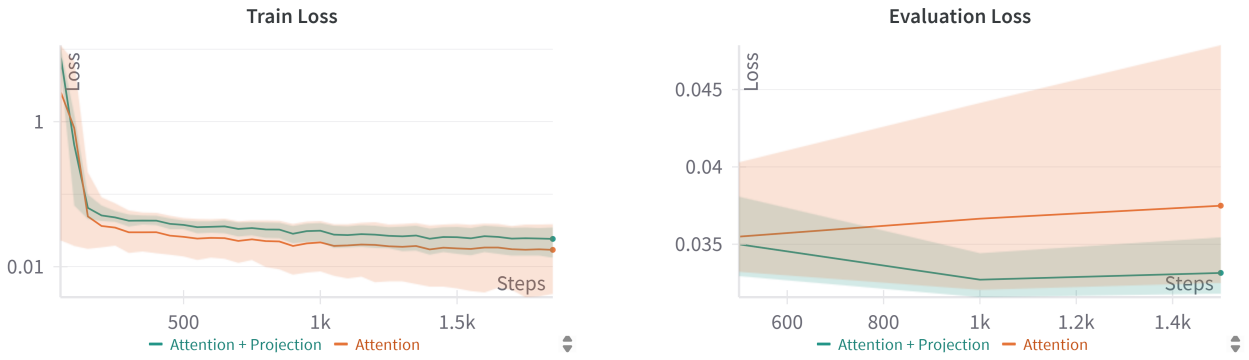
Figure 3: Evaluation loss plots for different LoRA ranks

6.2 Target Modules

When comparing target modules (Attention vs. Attention + Projection), adding projection layers improves performance but increases trainable parameters (Table 4).
The Attention + Projection configuration shows faster convergence (Figure 4).

Target Module	F1-Score	Exact Match
Attention	70.0	60.0
Attention + Projection	**80.0**	**70.0**

Table 4: Metric values for different target modules



(a) Training Loss

(b) Evaluation Loss

Figure 4: Training and evaluation loss plots for different target modules

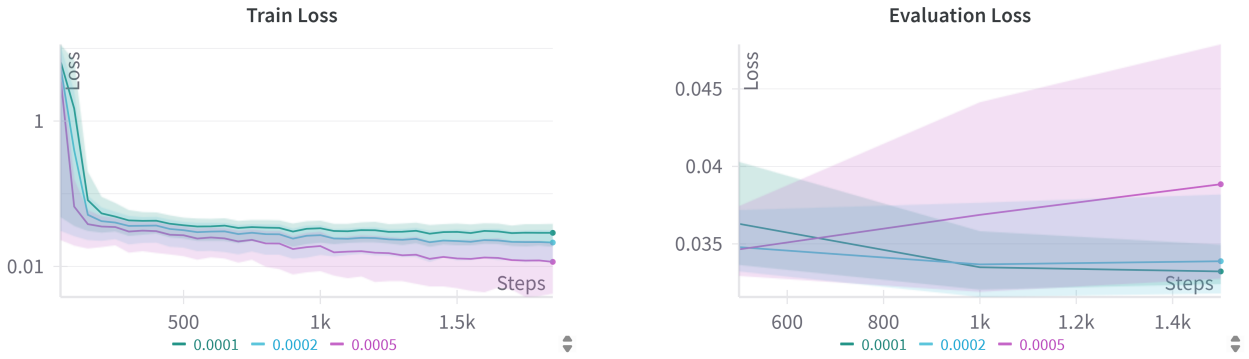
6.3 Learning Rate

In experiments varying only the learning rate, a value of 0.0002 yields the highest metrics (Table 5).

Higher learning rates lead to faster convergence but less stability in evaluation loss, with interval length directly related to the learning rate value (Figure 5).

Learning Rate	F1-Score	Exact Match
0.0001	54.2	47.5
0.0002	**57.7**	**55**
0.0005	50.3	42.5

Table 5: Metric values for different learning rates



(a) Training Loss

(b) Evaluation Loss

Figure 5: Training and evaluation loss plots for different learning rates

7 Analysis and Discussion

7.1 Best Results

The best configurations are shown in Table 6.

LoRA Rank	Target Module	Learning Rate	F1-Score	EM
32	Attn + Proj	0.0005	90.67	80
8	Attn	0.0002	80	80

Table 6: Best configurations among the conducted experiments

7.2 Computational Efficiency

Using LoRA significantly reduces the number of trainable parameters:

Rank	Target Module	Trainable Parameters
32	att + proj	3,244,032
32	att	2,211,840
16	att + proj	1,622,016
16	att	1,105,920
8	att + proj	811,008
8	att	552,960
4	att + proj	405,504
4	att	147,456

Table 7: Comparison of trainable parameters in different LoRA settings relative to all GPT-2 parameters (approximately 124 million)

Naturally, this reduces the memory used for learning and makes large models more accessible to the public for fine-tuning.

8 Theoretical Questions

8.1 LoRA Mathematics

In a layer like Linear Layer whose weight is represented by the matrix $W \in \mathbb{R}^{d\times k}$, in the LoRA method, instead of directly training W, a low-rank update is applied as follows:

$$W' = W + \Delta W = W + BA$$

where:

- $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are trainable matrices.
- $r \ll \min(d, k)$ is chosen to reduce parameters.

In the forward pass, the output is calculated as:

$$h = Wx + \frac{\alpha}{r} BAx$$

where α is a scaling factor adjusted with r.

In the backward step, the gradient only passes through B and A. In other words, since W is fixed, $\nabla_W = 0$ and we only have:

$$\nabla_B = \frac{\alpha}{r} \nabla_{\Delta W} \cdot Ax^T, \quad \nabla_A = \frac{\alpha}{r} B^T \cdot \nabla_{\Delta W} x^T$$

As a result, only the parameters A and B are updated, making the training much lighter.

8.2 Loss Function Analysis

In QA settings based on generative models like GPT-2, the input includes Context, Question, and Answer. If the loss function is applied to all tokens, the model may try to "generate" the question or context, leading to irrelevant and misleading responses.

To solve this, only the tokens related to the answer should be considered in the loss function. This is done with label masking, setting non-answer tokens to -100 (in PyTorch) to be ignored in CrossEntropyLoss computation.

Without Masking:

- The model learns to reconstruct the question or context.
- Causes slow convergence and instability in training.
- Leads to repetitive or incomplete responses.

8.3 Computational Complexity

In full model learning (Full Fine-Tuning), all parameters of GPT-2, which are about 124 million, are trained. But in LoRA, only the matrices A and B are trained.

Time and Memory Complexity:

- Full Fine-Tuning: Time and memory $\mathcal{O}(d \cdot k)$
- LoRA: Only $\mathcal{O}(d \cdot r + r \cdot k)$ with $r \ll d, k$

Reduction Ratio for the Best Configuration Mentioned in the Report:
For example, for:

- r = 32 , Attention + Projection $\Leftarrow$ Trainable parameters = 3,244,032
- Reduction ratio = $\frac{3{,}244{,}032}{124{,}845{,}312} \approx 2.5\%$

Meaning only about 2.5% of all parameters are trained, leading to huge savings in memory, time, and energy.