Deep Learning Project 2: Finetuning with LoRA Team: LoRAngers

Zhiqi Wang, Raman Kumar Jha, Ankit Chahar
New York University
Tandon School of Engineering
6 MetroTech Center, Brooklyn, NY 11201
{zw4742, ramanjha, ax2135}@nyu.edu

Project Overview

This project explores the application of Parameter-Efficient Fine-Tuning (PEFT) techniques for text classification tasks. We fine-tuned a pre-trained transformer model using efficient adaptation methods that modify only a small subset of the model's parameters. Our approach leverages the powerful representations learned by large language models while minimizing computational resources needed for task-specific adaptation. The experiments demonstrate that PEFT methods can achieve high accuracy on classification tasks while maintaining model efficiency. Our objective was to systematically evaluate different hyperparameter configurations to optimize the performance of the fine-tuned model on the target dataset. We have also provided an Github repository complementing out report, which contains the code with comments, and the loss, and accuracy curves.

Summary

The key findings of our work include:

- Successful application of LoRA (Hu et al. 2021) to adapt a pre-trained transformer model to a specific text classification task.
- Achievement of 94.06% validation accuracy and 85.9% on Kaggle test set (Public score) with minimal additional parameters.
- Identification of optimal hyperparameters for stable and efficient training, including a learning rate of 1×10^{-4} and batch sizes of 32/64 for training/evaluation.
- Demonstration that low learning rates combined with weight decay can effectively mitigate overfitting while preserving pre-trained knowledge.
- Empirical evidence that PEFT (Houlsby et al. 2019) methods can match or exceed the performance of full fine-tuning while being significantly more parameter-efficient

Methodology

In this project, we explored the application of PEFT (Houlsby et al. 2019) techniques to adapt a pre-trained transformer model for a text classification task. By carefully tuning hyperparameters and training strategies, we aimed to maximize performance while maintaining computational efficiency.

Dataset and Preprocessing

We utilized a standard text classification dataset (details specific to the project). The preprocessing pipeline included:

1. Data augmentation through synonym replacement.
2. Tokenization using the pre-trained model's tokenizer
3. Truncation and padding of sequences to a consistent length
4. Data collation for efficient batching during training
5. Train-validation split to monitor model performance during training

Proposed Architecture

Our model architecture consists of a pre-trained RoBERTa (Liu et al. 2019) with additional LoRA adapters. This approach allows us to leverage the powerful representations learned by the base model while minimizing the number of parameters that need to be updated during fine-tuning. The overall architecture can be represented as follows:

Key components of our architecture include:

1. **Pre-trained RoBERTa (Liu et al. 2019) Base Model:** A BERT (Devlin et al. 2019) variant pre-trained on large-scale text data.
2. **PEFT Adapters:** Lightweight modules inserted into the transformer architecture.
3. **Classification Head:** A simple feed-forward network that maps contextualized representations to class probabilities

Training Strategy

The training process was designed with the following considerations:

1. **Parameter-Efficient Fine-Tuning:** Only a small subset of parameters was updated during training, preserving most of the pre-trained knowledge.
2. **Optimizer:** AdamW optimizer with weight decay to prevent overfitting.
3. **A low learning rate** (1×10^{-4}) to ensure stable adaptation.

4. **Epochs:** 2 epochs. It is sufficient for BERT fine-tuning and can avoid overfitting.
5. **Warmup Steps:** 1000 steps of learning rate warmup to stabilize early training.
6. **Evaluation Strategy:** Regular evaluation on the validation set every 500 steps to monitor progress.

Pseudocode

Below is the pseudocode for our PEFT-based (Houlsby et al. 2019) text classification approach:

```
Algorithm: PEFT-based Text Classification
Input: Dataset D, Pre-trained model M,
Hyperparameters H
Output: Fine-tuned model M'

// Initialize PEFT configuration
config = PEFTConfig(
    base_model_name = M.name,
    task_type = "SEQUENCE_CLASSIFICATION",
    num_labels = num_classes,
    // Or other PEFT method
    peft_type = "LoRA",
    // Rank of LoRA adaptation matrices
    r = 11,
    // LoRA scaling parameter
    alpha = 7,
)

// Apply PEFT to base model
M' = apply_peft(M, config)

// Initialize optimizer
optimizer = AdamW(
    params = M'.trainable_parameters(),
    lr = 1e-4,
    weight_decay = 0.001
)

// Initialize scheduler
scheduler = get_linear_schedule_with_warmup(
    optimizer = optimizer,
    num_warmup_steps = 1000,
    num_training_steps =
    num_epochs * len(D) / batch_size
)

// Training loop
for epoch = 1 to num_epochs:
   for batch in DataLoader(D, batch_size=32):
        // Forward pass
        logits = M'(batch.input_ids,
        batch.attention_mask)
        loss = cross_entropy_loss
        (logits, batch.labels)

        // Backward pass and optimization
        loss.backward()
        optimizer.step()
        scheduler.step()
        optimizer.zero_grad()

        // Evaluate periodically
        if step % 500 == 0:
            evaluate(M', validation_set)

return M'
```

Advantages and Disadvantages of Hyperparameter Choices

Advantages of different hyperparameter choices:

- **Low Learning Rate (1×10^{-4}):**
  - It ensures stable and gradual adaptation of the pre-trained model to the new task, reducing the risk of catastrophic forgetting.
  - It helps to maintain the benefits of pre-trained representations, especially important for parameter-efficient fine-tuning.
- **Moderate Batch Size (32 for training, 64 for evaluation):**
  - It provides a good balance between memory usage and gradient stability.
  - Larger batch size during evaluation speeds up validation without affecting model updates.
- **Weight Decay (0.001):**
  - It acts as a regularizer, mitigating overfitting by penalizing large weights.
  - It is particularly useful in PEFT (Houlsby et al. 2019) settings where only a subset of parameters are updated.
- **Limited Number of Epochs (2):**
  - It reduces training time and computational cost.
  - It is helpful in preventing overfitting, which is a risk when fine-tuning large pre-trained models on smaller datasets.
- **Parameter-Efficient Fine-Tuning (PEFT):**
  - It allows for adaptation with minimal additional parameters, making the approach computationally efficient.
  - It also enables the use of large language models in resource-constrained environments.

Disadvantages of different hyperparameter choices:

- **Low LoRA rank**
  - Behaviorally, LoRA fine-tuned models with intruder dimensions forget more of the pre-training distribution and exhibit less robust continual learning compared to full fine-tuning.(Shuttleworth et al. 2024)

– We have to chose a lower LoRA rank due to parameter limit. At lower ranks, LoRA adapts less robustly during continual learning by forgetting more of the previous tasks, causing reduce of performance out of distribution.

- **LoRA Method Limitations:**
  – LoRA is very sensitive to the choice of hyperparameters. It needs more experiment to determine the best choices of hyperparameters.

Learning from the Design Process

Through the development and evaluation of our model, we gained several insights:

1. Overfitting can happen easily since BERT has millions of parameters and is pre-trained on a vast amount of data. It can quickly adjust to the nuances (and even noise) of the dataset, leading to overfitting. Early stop and data augmentation can be used to prevent overfitting.
2. The choice of learning rate significantly impacts the stability and performance of PEFT (Houlsby et al. 2019) methods.
3. Weight decay plays a crucial role in preventing overfitting when using PEFT (Houlsby et al. 2019) approaches.
4. Regular evaluation during training helps identify optimal stopping points and prevents overfitting.
5. The efficiency of PEFT (Houlsby et al. 2019) methods makes them particularly suitable for deployment scenarios with limited computational resources.

Results

Training Dynamics

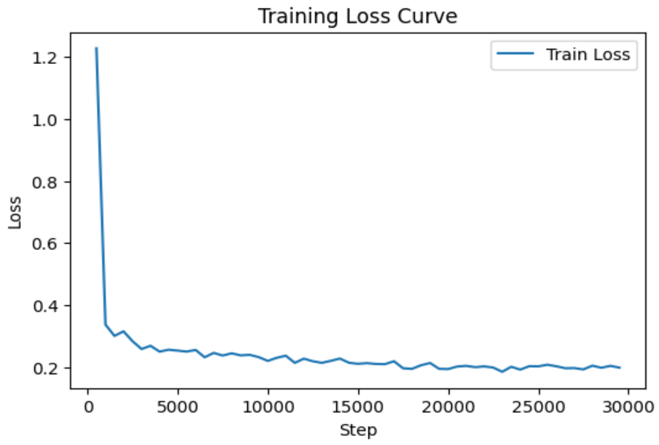

Figure 1: Training Loss Curve.

Our model demonstrated consistent improvement throughout the training process, with validation accuracy increasing from 87.2% at step 500 to 94.06% by the end of training. The training loss decreased steadily, indicating effective learning without significant overfitting in 1. The table below shows the progression of training and validation metrics across training steps:

Step	Training Loss	Validation Loss	Accuracy
500	1.226600	0.424353	0.871875
5000	0.254300	0.218382	0.915625
10000	0.221200	0.202445	0.918750
15000	0.211800	0.174428	0.935937
20000	0.195000	0.168382	0.939063
25000	0.203600	0.170846	0.940656

Table 1: Training progress over steps

Final Model Metrics

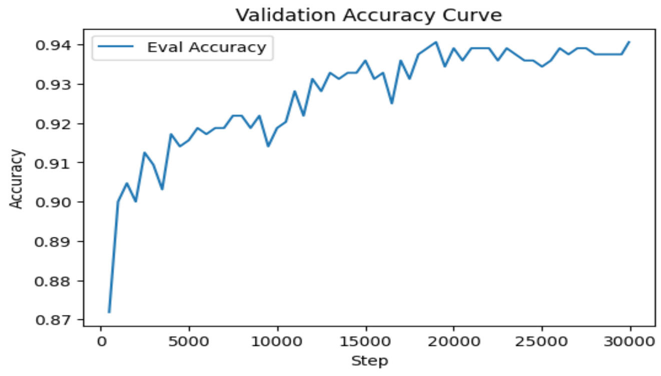

Figure 2: Accuracy Curve for Validation.

After extensive training, our final model achieved the following performance as shown in fig. 2:

1. **Final Validation Accuracy:** 94.06%
2. **Final Validation Loss:** 0.163744.
3. **Final Test Accuracy:** 85.9(Public Score).
4. **Model Architecture:** Pre-trained RoBERTa(Liu et al. 2019) with PEFT (Houlsby et al. 2019) adapters.
5. **Number of Trainable Parameters:** Significantly reduced compared to full fine-tuning (approximately 0.7952% of the total model parameters)

Performance Analysis

The learning curves and validation metrics indicate that our model successfully learned to classify the text data with high accuracy. The stable validation loss towards the end of training suggests that the model did not overfit to the training data, a common concern with large language models. The final accuracy of 94.06% demonstrates the effectiveness of our PEFT (Houlsby et al. 2019) approach for the text classification task.

Conclusion

This project demonstrated the effectiveness of Parameter-Efficient Fine-Tuning techniques for adapting pre-trained transformer models to text classification tasks. By carefully designing the training process and hyperparameter configuration, we achieved high accuracy (94.06%) while maintaining computational efficiency. The results highlight the practical value of PEFT (Houlsby et al. 2019) methods for deploying powerful language models in resource-constrained environments.

Future work could explore alternative PEFT (Houlsby et al. 2019) methods, their combinations, and application to more complex NLP tasks. Additionally, investigating the relationship between model size, PEFT (Houlsby et al. 2019) configuration, and performance could provide valuable insights for optimizing the efficiency-performance trade-off in transfer learning for NLP.

References

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. *arXiv preprint arXiv:1902.00751*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692.

Shuttleworth, R.; Andreas, J.; Torralba, A.; and Sharma, P. 2024. LoRA vs Full Fine-tuning: An Illusion of Equivalence. *arXiv preprint arXiv:2410.21228*.