Report

1.Model Analysis

Model Design

I used the bert-base-uncased pre-trained model, which consists of 12 Transformer layers, each containing 768 hidden units and 12 attention heads, amounting to a total of 110 million parameters.

Lora: In the BertSelfAttention component of the model, the computation of the query has been modified from using a standard Linear layer to a method based on Lora.

Bitfit: Since Bitfit only fine-tunes the biases in the original model without changing the architecture, it shares the same structure as the bert-base-uncased pre-trained model.

Loss Reduction Process:

Defining the loss function: With num_labels=2 set during model loading, the bert-base-uncased model uses the cross-entropy loss function.

Defining the optimizer: The default AdamW optimizer of the Trainer object is used.

Iterative training process: trainer.train()

Data is passed through the model for predictions. The loss between the predictions and the true labels is calculated.

The gradients are computed via backpropagation.

The optimizer updates the model parameters to minimize the loss.

The above steps are repeated in each training iteration.

Results of Validation and Testing:

Validation Results: In both MRPC and SST2 tasks, the validation results closely matched the training results, with the loss steadily decreasing as the number of training iterations increased.

Testing Results: For the MRPC task, the test accuracy of the model hovered around 80%, whether using Lora or Bitfit. Since the SST2 task had unlabelled test data, no testing was performed.

2. PEFT Discussion

Main Feature of Bitfit:

The primary characteristic of Bitfit is that only the bias parameters in the model are trained while freezing all other parameters. The advantage of this is high efficiency with low resource consumption. It is effective because the pre-trained model has already been trained on large amounts of data, and its parameters contain rich knowledge. Therefore, fine-tuning only the bias parameters allows the model's outputs to align more closely with specific downstream tasks, improving performance for those tasks. This approach ensures precision while saving the resources needed to retrain the entire model. The Bitfit paper also supports the assumption that fine-tuning primarily leverages the knowledge learned by the pre-trained model, rather than learning new language knowledge specific to a task.

Best Hyper-parameters:

The following hyperparameters were consistent across all model training runs:

learning_rate = 1e-3 (with the default linear learning rate scheduler),
per_device_train_batch_size = 16,

per_device_eval_batch_size = 16,

num_train_epochs = 5,

weight_decay = 0.01.

Most of the other TrainingArguments were set to default values.

Learning Rate:

Compared to full fine-tuning, Bitfit typically requires a lower learning rate because it only trains the bias parameters, freezing all other parameters. Since the number of parameters being trained is significantly fewer than in full fine-tuning, smaller learning rate steps are sufficient for the bias updates, avoiding over-adjustment or quick forgetting of previously learned knowledge, which could otherwise degrade performance.

3. PEFT Comparison:

Comparing the Performance of Bitfit with Lora on Two Datasets:

SST2: Both Bitfit and Lora achieved an accuracy around 0.9.

MRPC: Both Bitfit and Lora achieved an accuracy around 0.8.

Rank (r) Effect on LoRA's Parameter Count and Performance:

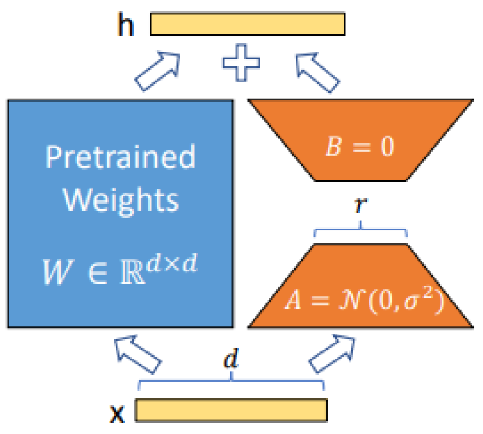

Figure 1: Our reparametrization. We only train A and B.

As seen from the above figure, the size of the matrices A and B to be trained is related to r. The larger the r setting, the more parameters need to be trained. For example, in the diagram:

A: d x r, B: r x d, so in total, 2 x r x d parameters need to be trained.

For the SST2 task:

r = 1, trainable params: 19970

r = 4, trainable params: 75266

r = 16, trainable params: 296450

Regarding the relationship between r and performance, the paper states: "We check the overlap of the subspaces learned by different choices of r and by different random seeds. We argue that increasing r does not cover a more meaningful subspace, which suggests that a low-rank adaptation matrix is sufficient."

My results on both tasks (r=1, 4, 16) indicate that changing r had little impact on performance, assuming other parameters remain constant.