Assignment 3

1. Dataset

For our dataset, we choose the TweetEval dataset to work on. For our assignment, we will only focus on sentiment classification of tweets. Each sample consists of a tweet and a corresponding sentiment label (0, 1 or 2).

- 0: Negative
- 1: Neutral
- 2: Positive

The dataset is also split into train, test and validation by default. We have 45615, 2000, 12284 samples for training, validation and testing respectively.

There are 3 main reasons for choosing this dataset. Firstly, it is a small dataset which makes it ideal for finetuning with limited computing power. Secondly, the nature of tweets is very diverse. For example, tweets can range from informal expressions to more serious discussion of topics. This wide range of language styles provides a realistic view of english sentences for sentiment analysis. Lastly, the dataset is well-labelled (i.e., the labels can be trusted) and has been used for other academic works.

2. Models

2.1 Base Model

For our baseline pretrained model, we choose DistilBERT *(distilbert-base-uncased)* as our base model. DistilBERT is a smaller model that was trained using the BERT base model. DistilBERT has 40% less parameters than BERT, while retaining 97% of its language understanding capabilities and being 60% faster (Sanh et al., 2019). With roughly 67 million parameters, DistilBERT was chosen for its lightweighted-ness. Moreover, DistilBERT is primarily aimed at being fine-tuned on tasks that uses the whole sentence to make decisions, such as sentiment classification.

2.2 LoRA (Low Rank Adaptation)

For our first finetuning approach, we choose LoRA. LoRA injects trainable low-rank matrices into query and value projection layers of the attention mechanism. When we are doing our finetuning, we only train these low-rank matrices. The parameters of DistilBERT are frozen.

The configuration for LoRA is as follows below:

- $r = 8$
- $lora\_alpha = 16$
- $lora\_dropout = 0.1$
- $bias = \text{'none'}$
- $task\_type = SEQ\_CLS\ (sequence\ classification)$

2.3 Adapters

For our second finetuning approach, we use adapters to improve the performance. Instead of tweaking the attention mechanism, adapters are small modules that are injected between transformer layers of a pre-trained model. During finetuning, only the parameters of adapters are changed while the parameters of the base model are frozen.

The configuration for our adapters is as follows below:

- $architecture = 'pfeiffer'$
- $reduction\_factor = 16$
- $non\_linearity = 'relu'$

3. Experimental Setup

3.1 Data Preprocessing

The dataset was tokenized using the DistilBERT tokenizer. Each sample was truncated and padded to a maximum length of 128 tokens to ensure consistent input dimensions for DistilBERT.

3.2 Training

For our training workflow, it is largely handled by the *Trainer* class in the transformers package. By providing the trainer with the model, training configuration, train and validation sets, *Trainer* handles the entire training loop, evaluation on validation set and logging of relevant metrics after each epoch.

Below are the most important configurations that were kept consistent across all experiments:

- $num\_train\_epochs = 3$
- $metric\_for\_best\_model = 'accuracy'$

After the entire training loop, we then evaluate the finetuned model on the test set, using both accuracy and F1 score as our metrics.

4. Results

Here are the results of the finetuning, correct to 2 decimal places.

	Baseline Model	LoRA	Adapters
Accuracy	0.34	0.68	0.67
F1	0.34	0.68	0.67

It is clear from the results that finetuning does play a significant role in improving the model's performance. Both LoRA and adapters doubled the baseline model's performance in sentiment classification. Interestingly, there were not much difference between LoRA and adapters. A possible reason for this is that despite the difference in architectures, the overall idea of LoRA and adapters are actually quite similar. Both methods involve inserting small trainable modules into the base model. As for training duration, there was also no significant difference. LoRA took ≈ 71 minutes while adapters took ≈ 66 minutes.

Here are the training loops for both LoRA and adapters.

LoRA

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.677200	0.681058	0.694000	0.693926
2	0.659500	0.667636	0.708000	0.707185
3	0.649800	0.659583	0.707500	0.708693

Adapter

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.668900	0.674761	0.702000	0.701090
2	0.653700	0.665356	0.712500	0.709456
3	0.643000	0.654384	0.712000	0.713033

5. Key Takeaways and Limitations

The biggest takeaway for me is the drastic increase in performance from finetuning. The performance of the model doubles even though full-finetuning was not conducted. With just a few injections of additional tunable parameters, the performance of the model on a specific task can be improved greatly. While this demonstrates the benefits of finetuning with parameter-efficient strategies, the resultant model is very niche and performs well at one and only one task only.

One possible limitation is that the number of epochs could be increased. We chose 3 epochs in light of limited compute. However, with more epochs, the finetuned model would have more opportunities to change the parameters of its modules. This could reveal clearer performance differences between LoRA and adapters.

References

Huggingface model: https://huggingface.co/distilbert/distilbert-base-uncased

Barbieri, F., Camacho-Collados, J., Anke, L. E., & Neves, L. (2020). TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. *Arxiv*. https://doi.org/10.18653/v1/2020.findings-emnlp.148

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1910.01108