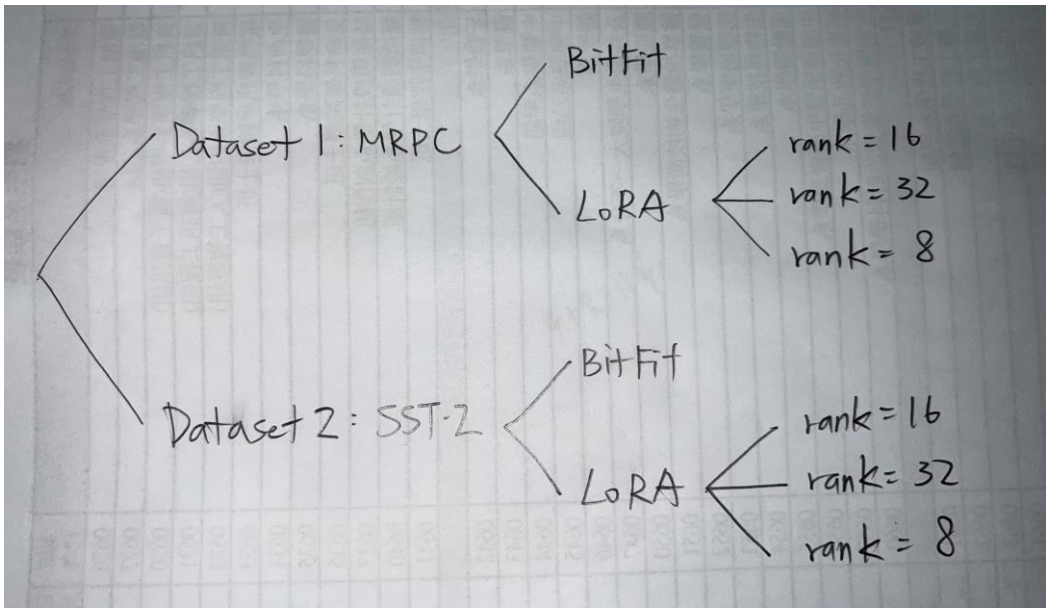
GAI Project 3 資訊 113 F74096027 吳瀚惟

1. Model Analysis

在本次 project，我使用了 **BERT 模型("bert-base-uncased")**作為預訓練模型，並且導入了 Glue benchmark 中的 **MRPC(Microsoft Research Paraphrase Corpus)**以及 **SST2(The Stanford Sentiment Treebank)**，分別在這兩組資料集上進行參數高效的微調(PEFT)。MRPC 是一個基於新聞來源的資料集，主要用於確定兩個句子是否具有相同的意思，它包含約 5800 對句子，這些句子備標註為"相同"或"不相同"；SST2 則是情感分析任務的資料集，它提供電影評論的句子，包含積極和消極評價的二分標籤，用於訓練模型以理解文本的情感傾向。我載入兩組資料集，首先對其進行前處理，使其變成適合輸入模型的形式，接著導入 BERT 預訓練模型，再分別撰寫用於評價模型指標的函數(MRPC: Accuracy/F1; SST2: Accuracy)，稍後套用於模型的驗證過程。我設計了使用 **BitFit** 和 **LoRA** 方法的模型，再分別調整超參數進行訓練，觀察輸出的結果，其中，更針對 LoRA 中的超參數"rank"做了分析，更改 r 值並觀察訓練結果有何不同。以下是基本程式架構以及訓練結果的截圖：

程式(訓練測試)架構：



●MRPC – BitFit:

```
{'eval_loss': 0.45934438705444336,
 'eval_accuracy': 0.8088235294117647,
 'eval_f1': 0.8682432432432433,
 'eval_runtime': 3.246,
 'eval_samples_per_second': 125.692,
 'eval_steps_per_second': 8.01,
 'epoch': 9.91}
```

eval_loss 從 0.6057 開始，經過 570 steps 的訓練後下降至 0.4593。

●MRPC – LoRA (rank = 16):

```
{'eval_loss': 0.7245796322822571,
 'eval_accuracy': 0.8553921568627451,
 'eval_f1': 0.8977469670710572,
 'eval_runtime': 3.8487,
 'eval_samples_per_second': 106.011,
 'eval_steps_per_second': 6.756,
 'epoch': 9.91}
```

eval_loss 從 0.5837 開始，經過 110 steps 的訓練後下降至 0.3270，隨後稍微回升。

●MRPC – LoRA (rank = 32):

```
{'eval_loss': 0.6160754561424255,
 'eval_accuracy': 0.8602941176470589,
 'eval_f1': 0.900523560209424,
 'eval_runtime': 3.708,
 'eval_samples_per_second': 110.033,
 'eval_steps_per_second': 7.012,
 'epoch': 9.91}
```

eval_loss 從 0.6050 開始，經過 110 steps 的訓練後下降至 0.2980，隨後稍微回升。

●MRPC – LoRA (rank = 8):

```
{'eval_loss': 0.8137052655220032,
 'eval_accuracy': 0.8578431372549019,
 'eval_f1': 0.9003436426116839,
 'eval_runtime': 3.6489,
 'eval_samples_per_second': 111.816,
 'eval_steps_per_second': 7.125,
 'epoch': 9.91}
```

eval_loss 從 0.6592 開始，經過 140 steps 的訓練後下降至 0.3446，隨後稍微回升。

●SST2 – BitFit:

```
{'eval_loss': 0.22979994118213654,
 'eval_accuracy': 0.9139908256880734,
 'eval_runtime': 6.9454,
 'eval_samples_per_second': 125.55,
 'eval_steps_per_second': 7.919,
 'epoch': 1.0}
```

eval_loss 從 0.3708 開始，經過 1050 steps 的訓練後下降至 0.2298。

●SST2 – LoRA (rank = 16):

```
{'eval_loss': 0.2146022468805313,
 'eval_accuracy': 0.9162844036697247,
 'eval_runtime': 8.2494,
 'eval_samples_per_second': 105.704,
 'eval_steps_per_second': 6.667,
 'epoch': 1.0}
```

eval_loss 從 0.2959 開始，經過 1050 steps 的訓練後下降至 0.2146。

●SST2 – LoRA (rank = 32):

```
{'eval_loss': 0.2056942582130432,
 'eval_accuracy': 0.9254587155963303,
 'eval_runtime': 8.3619,
 'eval_samples_per_second': 104.283,
 'eval_steps_per_second': 6.577,
 'epoch': 1.0}
```

eval_loss 從 0.2964 開始，經過 1050 steps 的訓練後下降至 0.2057。

●SST2 – LoRA (rank = 8):

```
{'eval_loss': 0.21130798757076263,
 'eval_accuracy': 0.9174311926605505,
 'eval_runtime': 8.215,
 'eval_samples_per_second': 106.147,
 'eval_steps_per_second': 6.695,
 'epoch': 1.0}
```

eval_loss 從 0.3312 開始，經過 1050 steps 的訓練後下降至 0.2113。

以上採用不同策略的八次訓練，**成果均超過助教設定的 baseline** (MRPC: accuracy = 0.8; SST2: accuracy = 0.88)。

2. PEFT Discussion

BitFit 是一種 PEFT 方法，它的全名是 Bias-term fine-tuning，其核心概念是**只對模型中的 bias 項進行微調，而將其他權重參數固定不變**；也就是把模型大部分的參數都凍結起來，只訓練一小部分的參數。這種方法可以大幅減少微調過程中的計算量和參數調整的複雜性，進而提高訓練的效率。同時，由於只對 bias 項進行微調，可以避免 overfitting 和泛化能力下降的問題。

我的實作程式碼如下圖所示。首先將模型中所有參數的 requires_grad 屬性設為 False，這代表這些參數在訓練過程中不會被更新。接著，去遍歷所有模型參數，只有當參數名稱中包含 bias 時，才會將它們的 requires_grad 屬性設為 True。這樣設定後，只有 bias 會在反向傳播過程中更新，實現了 BitFit 方法的核心概念：只調整 bias 項，保持其他權重不變，以減少訓練時所需要更新的參數數量，從而降低計算成本。

```
#  修改模型的訓練參數配置，僅對偏差進行訓練
for  param  in  model.parameters():
        param.requires_grad  =  False
for  name,  param  in  model.named_parameters():
        if  'bias'  in  name:
                param.requires_grad  =  True
```

以 MRPC 資料集，訓練使用 BitFit 方法的模型時，經過測試，我設定的最佳超參數配置如下：

```
#  設定訓練超參數
training_args  =  TrainingArguments(
        output_dir='./results',
        num_train_epochs=10,
        per_device_train_batch_size=16,
        per_device_eval_batch_size=16,
        gradient_accumulation_steps=4,
        learning_rate=5e-4,
        weight_decay=0.01,
        evaluation_strategy="steps",
        logging_dir='./logs',
        logging_steps=10
)
```

而以 SST2 資料集，訓練使用 BitFit 方法的模型時，經過測試，我設定的最佳超參數配置如下：

```
#  設定訓練超參數
training_args  =  TrainingArguments(
        output_dir='./results',
        num_train_epochs=1,
        per_device_train_batch_size=16,
        per_device_eval_batch_size=16,
        gradient_accumulation_steps=4,
        learning_rate=5e-3,
        weight_decay=0.01,
        evaluation_strategy="steps",
        logging_dir='./logs',
        logging_steps=50
)
```

在使用 BitFit 方法訓練時，我有特別關注並分析以下幾項超參數，嘗試合理的去調整它們，進行反復的測試，使訓練更穩定且高效：

(1) learning_rate: BitFit 因為只調整模型的 bias 參數，因此通常需要一個較大的學習率來快速達到收斂。如果學習率過低，模型可能需要更多的訓練時

間來適應資料或可能根本就無法收斂；學習率過高則可能導致訓練過程中的振盪過大，而使模型難以穩定。

(2) batch_size: 適當的 batch size 可以幫助模型有效的去學習，同時保持合適的記憶體使用率和訓練速度。過小的 batch size 可能會導致模型更新過於頻繁，影響訓練穩定性；過大則可能導致 out of memory 或學習效率降低。

(3) gradient_accumulation_steps: 聽完助教的講解，加上上網搜尋資料與實測，我了解到這是在記憶體受限但需要模擬更大 batch 效果時的一種技巧。適當增加累積步數可以在不增加記憶體負擔的情況下，實現更穩定的梯度下降。但如果累積的步數過高，則可能導致梯度更新延遲，影響模型的收斂速度。

(4) weight_decay: 這個參數有助於控制模型的 overfitting，對於簡單的微調任務來說，適當的權重衰減可以提高模型的泛化能力，然而，若設定的值過高可能會抑制模型學習，過低則可能無法有效防止 overfitting。

除了前文的討論內容之外，對比「全參數微調」(full fine-tuning)與「參數高效微調」(PEFT，如 BitFit 或 LoRA)時，學習率的設定通常會有顯著差異。**全參數微調**時，由於需要同時更新所有模型權重，通常會使用**較低的學習率**，以避免過度干擾已經預訓練好的模型權重，從而避免訓練不穩定。相對來說，**PEFT** 技術如 BitFit 只更新模型的一小部分參數(如 bias)，這允許使用**較高的**

學習率來加速特定參數的學習過程，而不會對整個模型的穩定性造成太大影響。

這種差異主要是因為 PEFT 的目標是在保持大部分預訓練參數固定的同時，只對少量參數進行調整。由於調整的參數少，需要較大的學習率來確保這些參數能夠在有限的訓練步驟中有效的更新和適應新任務。

3. PEFT Comparison

LoRA 的全名是 Low-Rank Adaptation，它是一種針對深度學習模型的 PEFT 方法，特別適合用於 transformer 架構。LoRA 的核心概念是**在模型的關鍵線性層之間插入低秩矩陣**。這些低秩矩陣允許模型在保持原始預訓練權重不變的情況下，透過訓練新增的參數來適應新任務。這種方法能夠有效的擴展模型的表達能力，同時最大限度的減少額外的參數和計算負擔。

我所設定的超參數如下：

```
peft_config = LoraConfig(
        r=16,  ###  or  8,  32
        lora_alpha=32,
        lora_dropout=0.1,
        bias="none",
        target_modules='all-linear',
        task_type=TaskType.SEQ_CLS
)
```

模型的訓練結果如第一點中所示。可以看到無論在 MRPC 還是 SST2 資料集上，BitFit 的訓練速度都相對較快。然而，在 MRPC 資料集上，採用 BitFit

方法訓練所得到的 accuracy/f1 為 0.8088/0.8682，效果並不如採用 LoRA 方法(以 rank=16 為例)所得到的 0.8554/0.8977。在 SST2 資料集上，採用 BitFit 方法訓練所得到的 accuracy 為 0.9140，效果亦不如採用 LoRA 方法(以 rank=16 為例)所得到的 0.9163。一般來說，BitFit 的性能表現會較 full fine-tuning 差，而 LoRA 通過在模型中加入低秩矩陣，能夠更靈活的調整參數而不破壞原有的模型結構，在某些情況下的表現甚至會優於 full fine-tuning。

LoRA 的 rank 參數(r)會直接影響模型可訓練的參數量。當 **rank 較高**時，**模型可訓練的參數(與比例)增加**，模型能學習到更多特徵，但計算量也較大；**rank 較低**時，**模型可訓練的參數(與比例)則較少**，計算雖更高效，但有可能捕捉不到所有有用的特徵。因此，在 MRPC 以及 SST2 資料集上，我分別將 rank 調整為 8, 16, 32，來找到最佳的平衡點，並分析了不同設定下的訓練成果。結果顯示，在 MRPC 與 SST2 資料集上，將 rank 設為 32 是三者之中的最佳選擇，其訓練成果分別為 0.8603/0.9005(MRPC, accuracy/f1)以及 0.9255(SST2, accuracy)。