GAI 作業三報告

資訊系大四 F74092269 陳冠廷

1. Scoring Criteria

I. 挑選 GLUE 的兩個任務

選擇資料集的原因：資料量愈少愈好 XD。但是資料集(WNLI、RTE)資料數量過少且詭異，訓練起來表現不平穩而且不好訓練，所以最後選擇以下兩個資料集：

i. The Corpus of Linguistic Acceptability (CoLA)

訓練資料：8551 筆 / 驗證資料：1043 筆

此資料集是在分析句子是不是合乎語法的，並將相關性分為兩類（合文法與不合文法），可以視為一種分類問題，評價指標為 **Matthews correlation coefficient**。例如：Yes, she did. 標籤為 1 是依據合乎語法的句子；但是 Yes, she used.不合乎語法所以標籤為 0。

ii. Semantic Textual Similarity Benchmark (STS-B)

訓練資料：5706 筆 / 驗證資料：1465 筆

此資料集是在分析兩句子是不是相關的，並將相關性分為 0-5 分，可以視為一種迴歸問題，評價指標為 **Pearson and Spearman correlation coefficients**。例如：A plane is taking off. An air plane is taking off. 兩句話語意完全相同所以分數為 5.0；而 A man is playing a large flute. A man is playing a flute. 兩句話貌似相近但是少了 large 的語意，所以相似度為 3.8，依此類推。

- 使用 torch dataset 封裝

```
class CustomedDataset(Dataset):

    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = { key: torch.tensor(val[idx]) for key, val in self.encodings.items() }
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)
```

- 清除多餘空白，用 roberta 的特殊字元前處理句子格式

```
if DATA_NAME == "CoLA":
    # CoLA 會將句子放在 sentence 欄位、標籤放在 label
    for s in df["sentence"]:
        s = s.strip() # 除去不必要的空白
        # 包裝輸入成 tokenizer 預期的輸入
        ### <s>  : 句子開頭
        ### </s> : 句子結尾
        texts.append(f"<s> {s} </s>")

    labels = df["label"]

else:
    # STSB 會將句子放在 sentence1, sentence2 欄位、標籤放在 score
    for s1, s2 in zip( df["sentence1"], df["sentence2"] ):
        s1 = s1.strip() # 除去不必要的空白
        s2 = s2.strip() # 除去不必要的空白
        # 包裝輸入成 tokenizer 預期的輸入
        ### <s>  : 句子開頭
        ### </s> : 句子分割與結尾
        texts.append(f"<s> {s1} </s></s> {s2} </s>")

    labels = df["score"]
```

- Tokenize 資料(關閉添加特殊字元)

```
encodings = tokenizer(
    texts, truncation=True, padding=True,
    add_special_tokens=False                # 關閉 add_special_tokens -> 已經預處理時加入<s>, </s>
)

# 回傳的 label 分兩種格式
##### 分類 -> int
##### 回歸 -> float
return {
    "encodings": encodings,
    "labels": labels.astype(int) if DATA_NAME == "CoLA" else labels.astype(np.float32)
}
```

II. 訓練同時驗證(Model Snapshot)

訓練過程中同時驗證當前模型的分數，可以參考以下程式碼截圖：

```
from sklearn.metrics import matthews_corrcoef
from scipy.stats import spearmanr
```

```python
def compute_metrics(pred):

    # pred 可以透過 label_ids 和 predictions 分別取得類別與預測結果
    labels = pred.label_ids
    # preds 紀錄預測的結果 / score 紀錄真實標籤值
    preds = None
    score = None

    if DATA_NAME != "STS-B":
        # CoLA 資料集為分類任務需要找出預測分數最高的類別
        preds = pred.predictions.argmax(-1)
        score = matthews_corrcoef(labels, preds)
    else:
        # STS-B 是迴歸任務只需取出預測值即可
        preds = pred.predictions[:, 0]
        score = spearmanr(labels, preds).statistic

    if np.isnan(score): # 避免相關係數有 nan 出現
        score = 0

    # 根據資料集需求回傳對應格式 (分數名稱: 分數)
    return {"matthews-corr": score} if DATA_NAME != "STS-B" else {"pearson-corr": score}
```

III. PEFT v.s. Full-finetune

i. 模型選擇與基礎設置

實驗過程以 roberta-base 作為我的 base MLM model，對於 CoLA 分類任務設定 num_labels 為 2，而對於 STS-B 迴歸任務設定 num_labels 為 1，最長序列長度保留模型預設為 512。(num_labels 為 1 時，Huggingface 內部會自動將 loss function 設為 Min Square Error，不是 1 則設為 Cross Entropy Loss)

```python
# 定義不同資料集的label數量(1 -> regression -> MSELoss)
num_labels = 2 if DATA_NAME != "STS-B" else 1
# 讀入 huggingface 的 model 與 tokenizer
model = T.AutoModelForSequenceClassification.from_pretrained( MODEL_NAME, num_labels=num_labels )
tokenizer = T.AutoTokenizer.from_pretrained( MODEL_NAME )
```

ii. 模型可訓練參數統計

```python
def show_trainable_ratio(model):
    # 取得可訓練與不可訓練的參數量
    trainbale_params = model.num_parameters(only_trainable=True)
    all_params = model.num_parameters(only_trainable=False)

    # 設定與 peft 相同的輸出格式 ( {trainable} || {all} || {trainable%})
    ### :, -> 設定輸出格式千分位
    return f"trainable params: {trainbale_params:,} || all params: {all_params:,} || trainable%: {trainbale_params/all_params*1
```

```python
if PEFT_TYPE == "lora":
    # LoRA -> 採用 peft 套件的設定
    model = get_peft_model(model, LoraConfig(**lora_config))

elif PEFT_TYPE == "bitfit":
    # 凍結非 bias 的參數
    for name, param in model.named_parameters():
        if "bias" not in name:
            param.requires_grad = False
else:
    # full finetune 不用做任何改動
    pass

# 輸出可訓練的參數佔比
print(show_trainable_ratio(model))
```

可以發現 bitfit 可訓練的參數通常會比 LoRA 所需的參數量少（當 rank 不是設定的過低時），但相較於 full-finetune，bitfit 與 LoRA 幾乎都只有極少量的參數需要微調（未達 1%）。對於 Bitfit 需要手動關閉模型非 bias 的梯度計算，lora 可以直接呼叫 peft 設定

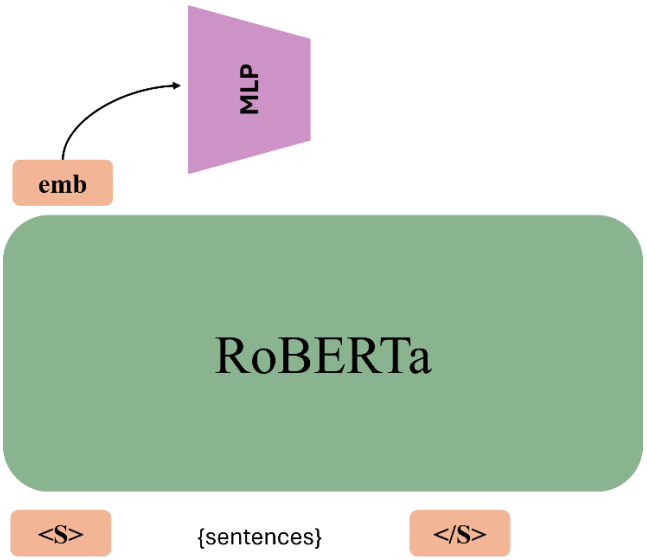
微調的方式	總參數量	可訓練參數量	可訓練參數比(%)
Full-finetune	124,647,170 (~125M)	124,647,170(~125M)	100.00
bitfit		102,914(~0.1M)	0.08
LoRA(r=2)	124,720,898 (~125M)	73,728	0.06
LoRA(r=8)	124,942,082 (~125M)	294,912(~0.3M)	0.24
LoRA(r=64)	127,006,466 (~127M)	2,359,296(~2M)	1.86
LoRA(r=256)	134,084,354 (~134M)	9,437,184(~9M)	7.04

2. Analysis

I. Model Analysis

由於測試資料沒有標籤，因此本次實驗只有驗證資料的分數。

i. Model Design



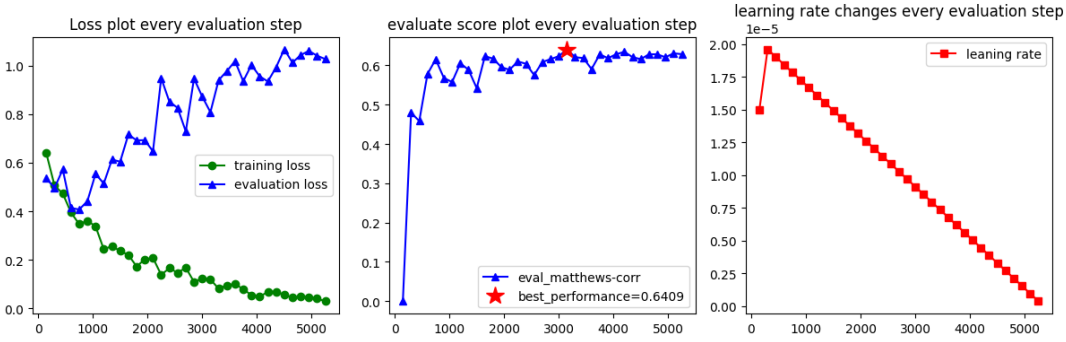
針對<S>的向量串接到 MLP 進行分類

ii. Loss Reduction

只呈現各種微調方式在兩個資料的最佳表現下的訓練過程（learning rate 的圖形可能較不準確，因為 warm-up step 與 snapshot step 不相同，但是都可以看出 linear schedular 的特性）

- **Full-finetune**
  - **CoLA**

    發現蠻詭異的現象 evaluate loss 上升但是評價分數卻也上升。同時，snapshot 記錄到了最佳分數為 0.6409

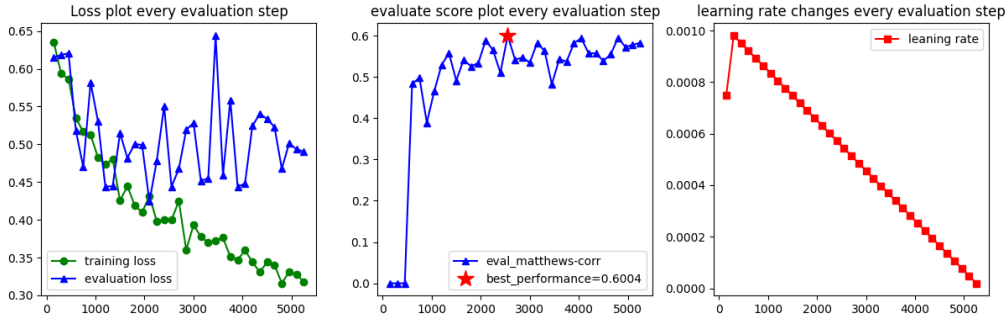


  - **STS-B**

    訓練過程與驗證過程都很完美，訓練與驗證的 loss 同步下降且收斂，驗證分數也收斂，並記錄到最佳表現為 0.9079

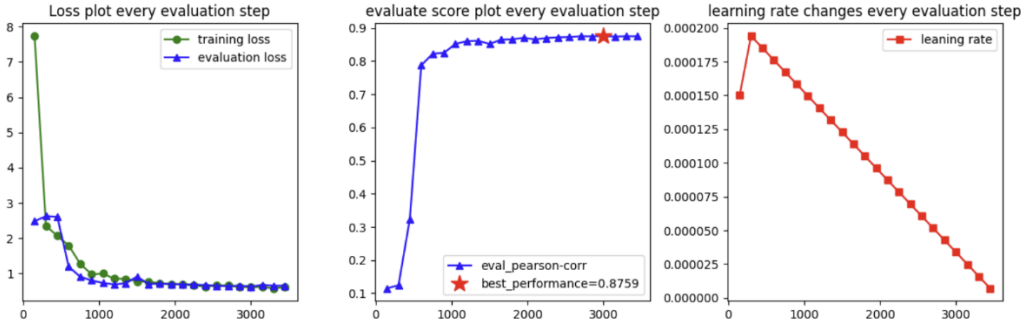
- **Bitfit**
  - **CoLA**

    一樣在驗證的 loss 上表現震盪且不穩定，分數大概在 0.5-0.6 間震盪，記錄到最佳表現為 0.6004 約略為 baseline



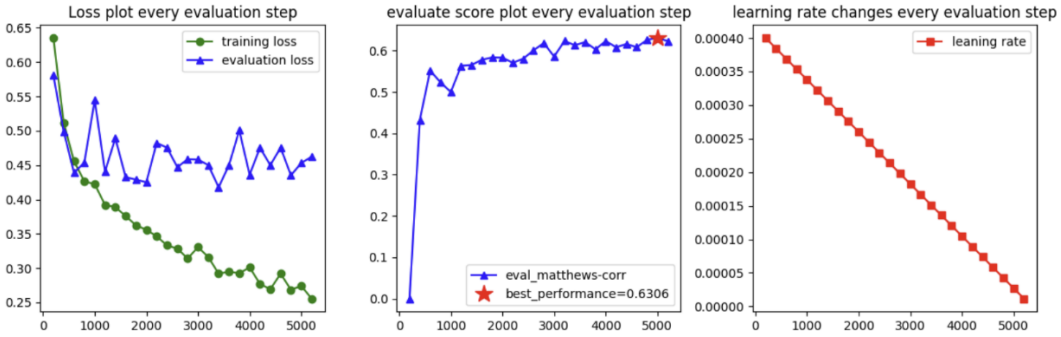
  - **STS-B**

    訓練過程與驗證過程都很完美，訓練與驗證的 loss 同步下降且收斂，驗證分數也收斂，並記錄到最佳表現為 0.8759



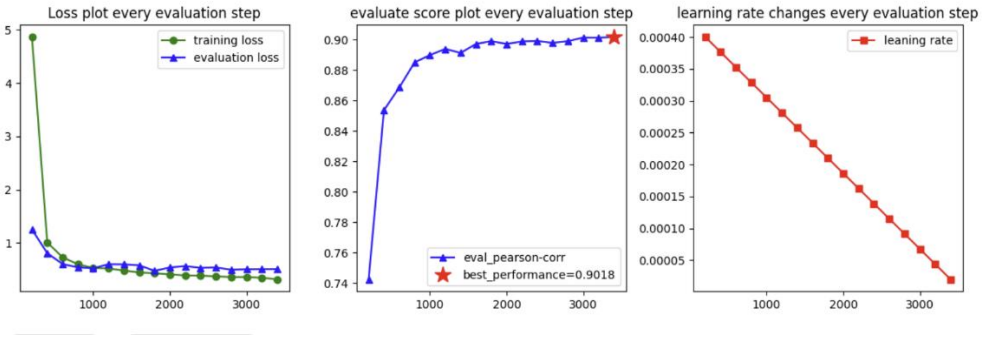
- **LoRA**
  - **CoLA**

    一樣在驗證的 loss 上表現震盪且不穩定，分數大概在 0.5-0.6 間震盪，記錄到最佳表現為 0.6004 約略為 baseline



- **STS-B**

  訓練過程與驗證過程都很完美，訓練與驗證的 loss 同步下降且收斂，驗證分數也收斂，並記錄到最佳表現為 0.9018



II. PEFT Discuss

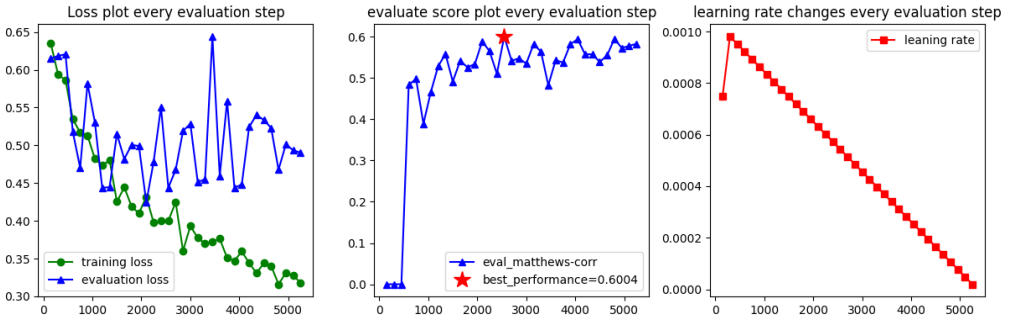
- **參數設定**

  部分參數參考至原始論文：roberta 建議 funetune 的學習率為 1e-5 或 2e-5，Bitfit 則是設定在 1e-4，但 CoLA 資料集我設置為 1e-3 才有比較好的表現，LoRA 的學習率亦是參考原論文設為 4e-4

	CoLA			STS-B		
	finetune	bitfit	lora	finetune	bitfit	lora
learning_rate	2e-5	1e-3	4e-4	2e-5	2e-4	4e-4
batchsize	16					
warmup_steps	200					
weight_decay	0.1					
adam_epsilon	1e-6					
epochs	10					

Bitfit 固定模型不是 bias 的部分，只學習 bias 參數，這樣的好處在於學習的參數少非常多，但是通常只調整 bias 很難對模型有很大的影響，即使模型中有 normalize 等方法有機會影響到其他參數的貢獻度，但仍比 finetune 的影響力差，因此調整參數上比較難，而且由於固定了絕大多數的參數，我們也會期待 bias 的改動大一點，因此學習率可以較大。反觀 finetune 因為整個模型都在更動，因此我們反而希望保留更多模型的預訓練知識，所以用小學習率微調，來避免破壞 pretrained model 的參數。

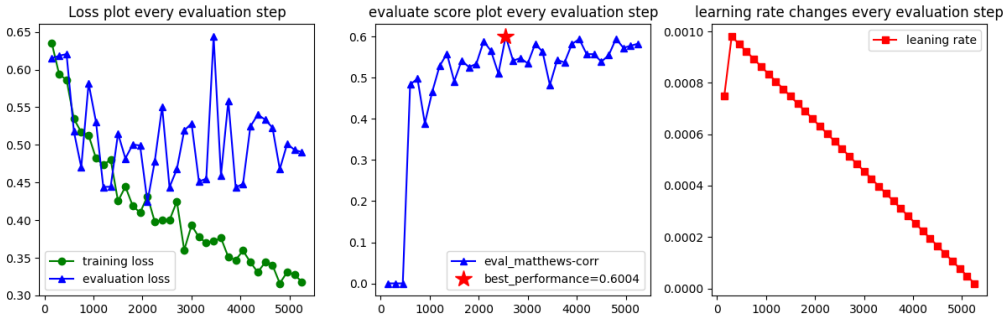
例如下圖分別為學習率是 2e-4 和 1e-3 的 bitfit 訓練在 CoLA 上，可以看出最好的表現有很大的差異，前者甚至低於 baseline，而微調的學習率為 2e-5 更是小的學習率，由此實驗即可看出 bitfit 在小學習率的表現較好。



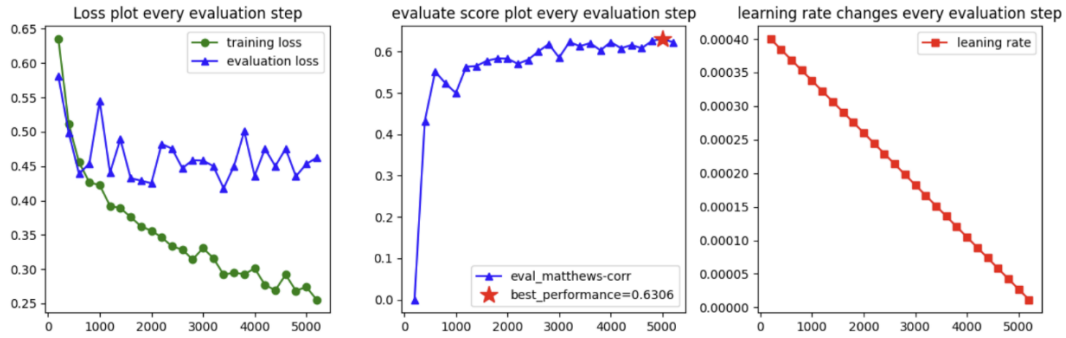
III. PEFT Comparison

- **Compare Bitfit and LoRA**
  - **CoLA**

以下是 bitfit 學習率為 0.001 的訓練過程：



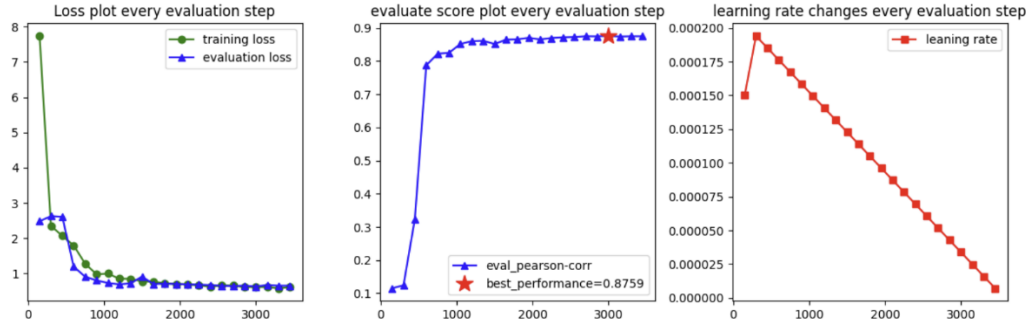
以下是 lora rank 為 8 的訓練過程：



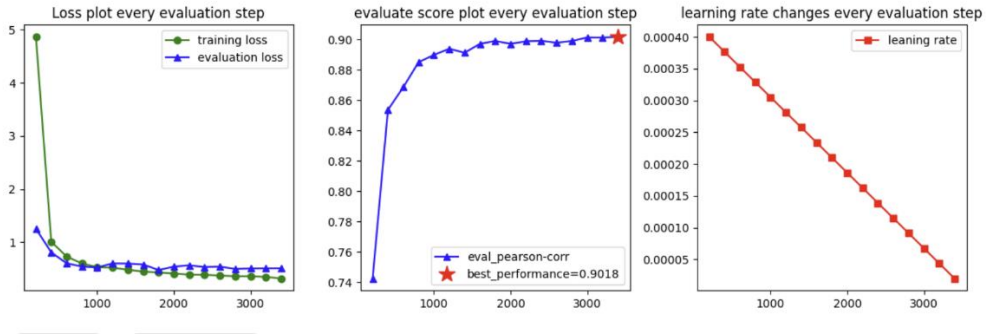
可以發現 bitfit 在 CoLA 上震盪較大而且不太收斂，表現也比 LoRA 差了 0.03。同時，在實作的時候發現 bitfit 在 CoLA 上面非常難調參數，幾乎都在 baseline(0.6)附近甚至比 baseline 差，但是 LoRA 幾乎可以隨便調都能穩定超過 0.6。

■ STS-B

以下是 bitfit 學習率為 0.0002 的訓練過程：



以下是 LoRA rank 為 2 的訓練過程：



可以發現 bitfit 在 STS-B 上一樣表現低於 LoRA，差了約 0.022。而且 LoRA 的最佳解似乎還有上升的機會，而且這只是 rank=2 的結果，意味著可訓練參數量還比 bitfit 少。

- **Compare different rank**

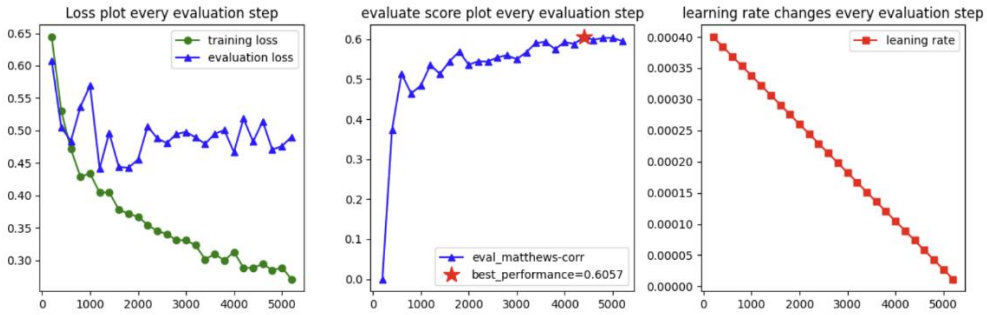
Lora 可以想做是近似 finetune 的過程，透過兩個低維度的梯形矩陣相乘來模擬 input 到 output 調整參數後的結果，當 rank 愈大時(input*rank)和(rank*output)兩個矩陣的大小都會變大，所需訓練的參數量也會變大，從上面的模型參數量表也可以看出來。

Lora 的 alpha 固定為 8 且 dropout 為 0.1 下做的比較：

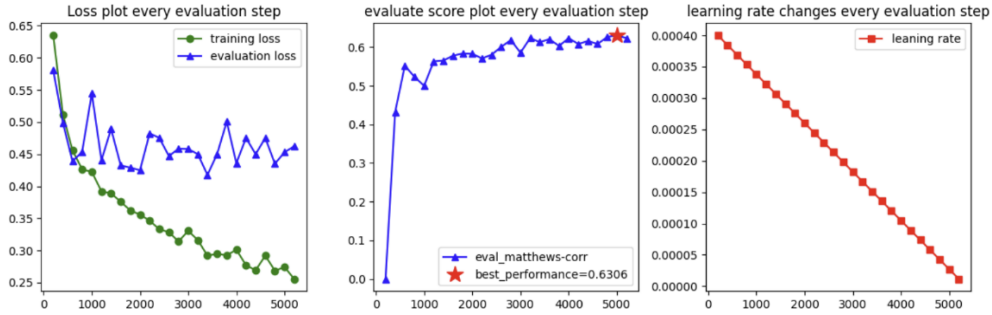
- CoLA

對 CoLA 而言 rank 還是有一些影響力的，特別是在 rank 為 8 時效果最好

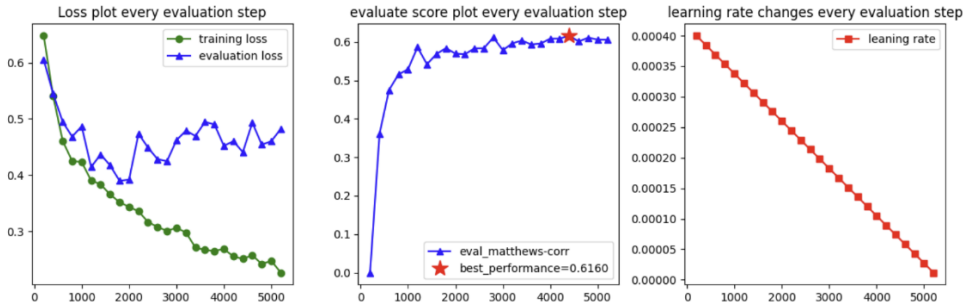
Rank=2



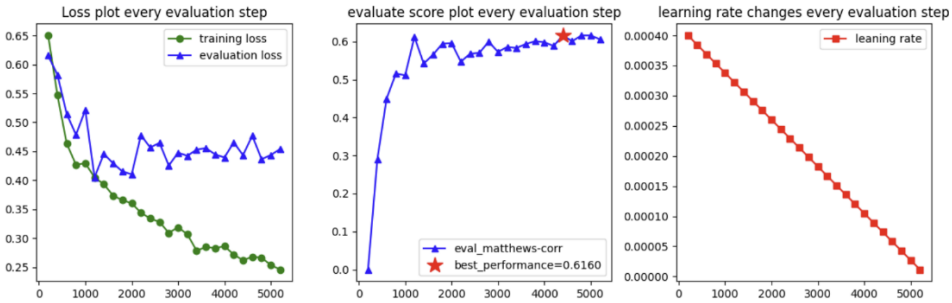
Rank=8



Rank=64



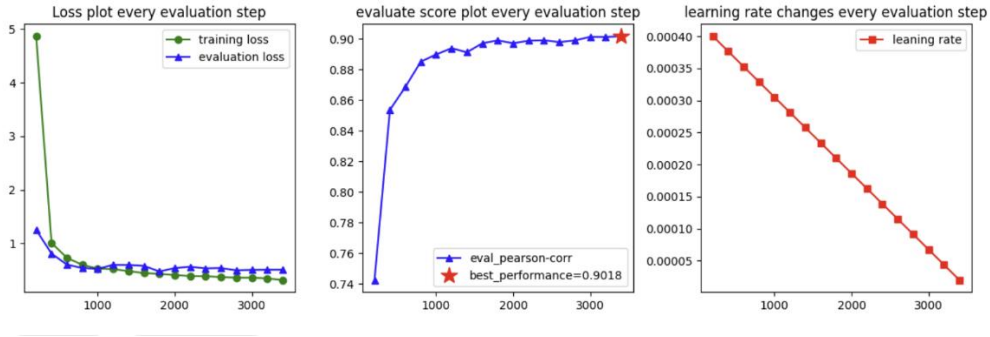
Rank=256



■ STS-B

對這個資料集而言，rank 的影響力不大。

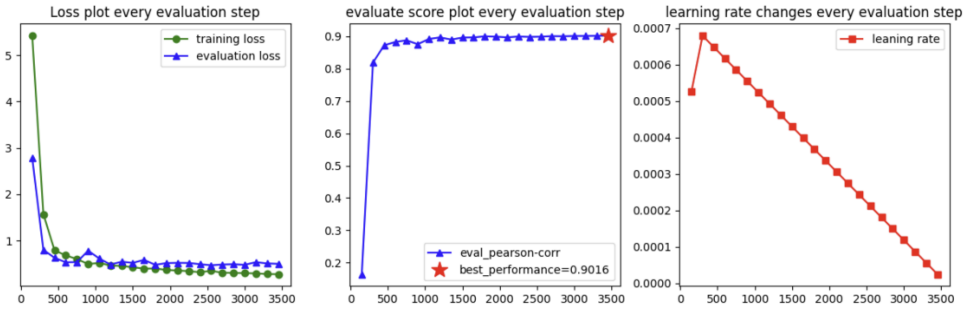
Rank=2



Rank=8

Rank=64

Rank=256



總而言之，rank 是一個可調整的參數，而且取決於資料會有不一樣的最佳解，但是由這兩個實驗都能發現，只要設小小的 rank 就能有不錯的小效果了，甚至超過 finetune

3. Reference

I. 於晨晨. (2021, October 31). GLUE 基準資料集介紹及下載. 知乎. https://zhuanlan.zhihu.com/p/135283598

II. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

III. Zaken, E. B., Ravfogel, S., & Goldberg, Y. (2021). Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. arXiv preprint arXiv:2106.10199.

IV. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.