AI CUP 2023 春季賽

真相只有一個: 事實文字檢索與查核競賽報告

隊伍：TEAM_3598

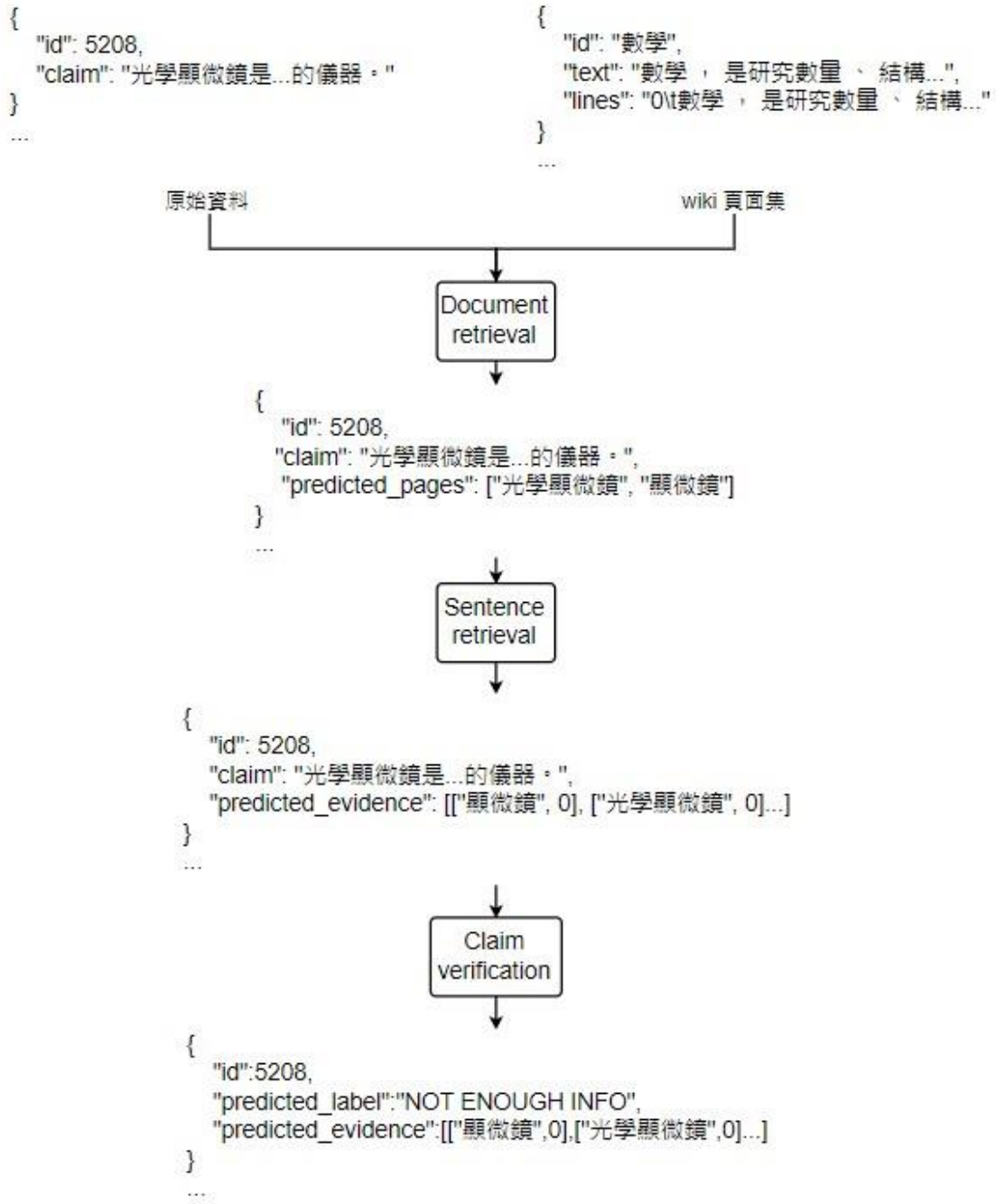
隊員：黃學智 （隊長）、李承哲、陳宥橋、朱祐麟

Private leaderboard：0.689 / Rank 5

壹、環境

- 作業系統：Windows 10
- 語言：Python
- 套件 （函式庫）：
  - 使用 Vscode 建立 Python 的開發環境
  - ipywidgets
  - pandarallel
  - pandas
  - scikit-learn
  - tensorboard
  - PyTorch
  - tqdm
  - transformers
  - numpy
  - jieba
  - TCSP
  - black
- CPU：I5-11400
- RAM：16 GB
- GPU：NVIDIA RTX 3060 Ti (8 GB)
- 預訓練模型：
  - Document retrieval：無
  - Sentence retrieval：hfl/chinese-lert-large
  - Claim verification：hfl/chinese-lert-large
- 由於使用環境繁多，僅列出一代表詳述之。

貳、演算方法與模型架構



1. Document retrieval

我們在這個階段使用了 tf-idf（Term Frequency-Inverse Document Frequency） 作為主要的文檔檢索演算法。tf-idf 是一種常用的特徵提取方法，用於衡量一個詞在文檔中的重要性。為了達到這個目標，我們調整了 TfidfVectorizer 的參數。這個向量化器是用於將文本轉換為 tf-idf 特徵向量的工具。以下是我們所做的調整：

- max_df = 0.8：這個參數指定了一個詞在超過多少文檔中出現時將被忽略。我們將其設置為 0.8，以避免過於常見的詞對文檔檢索的影響。

- min_df = 1：這個參數指定了一個詞在至少出現在一個文檔中的最小次數。我們將其設置為 1，以確保所有詞都被考慮到。
- use_idf = True：這個參數控制是否應用 idf 加權。我們將其設置為 True，以確保計算 tf-idf 特徵。
- sublinear_tf = True：這個參數控制是否應用子線性 tf 轉換。子線性 tf 轉換可以平滑高頻詞的影響，使得特徵更具鮮明性。我們將其設置為 True，以改善特徵的表現。
- ngram_range = (1,2)：這個參數指定了要提取的詞組的範圍。我們設置為 (1,2)，表示同時提取單個詞和雙詞詞組。這可以捕捉到更多的上下文信息。

通過這些參數的調整可以提高系統的效能和準確性，使我們能夠更快速地找到需要的信息。

2. Sentence retrieval

在第二階段，我們使用了以下參數進行 model 的訓練：

- Training Batch: 32
- Val Batch: 256
- Optimizer: AdamW
- Scheduler: ReduceLROnPlateau (factor=0.1, patience=2, mode='min') *step with val loss
- Loss function: BCEWithLogitsLoss (with class weight)

此外，bert model 輸出被改為 binary classification。

3. Claim verification

在第三階段，我們採用了 LERT 作為我們的訓練模型。在調整參數及方法測試都結束後，為了提高準確率，我們將原本使用的 hfl/chinese-lert-base 模型更換為 hfl/chinese-lert-large 模型。以下是我們在訓練模型時使用的參數：

- Batch Size （批量大小）：我們將批量大小設置為 32。這指定了在每次訓練迭代中一次處理的樣本數量。
- Seed （隨機種子）：我們設置了隨機種子為 42。這個種子用於生成隨機數，以確保訓練過程的可重現性。使用相同的種子可以確保每次執行時生成的隨機數序列相同。
- Learning Rate （學習率）：我們將學習率設置為 2e-5。學習率控制著模型參數在每次訓練迭代中的更新程度。適當的學習率可以促使模型更快地收斂到最佳解，但過高或過低的學習率可能導致訓練不穩定或收斂速度慢。
- Max Sequence Length（最大序列長度）：我們將最大序列長度設置為 256。這個參數限制了輸入文本的最大長度。長度超過這個值的文本將被截斷或進行其他處理。設置適當的最大序列長度可以平衡記憶體需求和模型性能。通過使用 hfl/chinese-lert-large 模型和適當的訓練參數，我

們期望能夠提高模型的準確率和表現能力。這將使我們的系統能夠更好地處理中文文本並提供更準確的結果。

參、創新性

在上述所描述的演算法和模型架構中，我們採用了幾種創新性的方法來提升系統的效能和準確性。

首先，在文件檢索階段，我們使用了 TF-IDF 作為主要的演算法。TF-IDF（Term Frequency-Inverse Document Frequency） 是一種常用的特徵表示方法，它可以量化文件中詞彙的重要性。透過調整 TF-IDF 向量化器（TfidfVectorizer） 的參數，如最小文件頻率 （min_df）、最大文件頻率（max_df）、使用 IDF 權重 （use_idf）和子線性 TF （sublinear_tf），我們可以優化文件檢索的準確性和效能。

其次，在第二、三階段的訓練中，我們採用了 LERT（Large-scale Evidence Retrieval Transformer） 模型作為我們的訓練模型，並將原本的 hfl/chinese-lert-base 模型升級為 hfl/chinese-lert-large 模型。LERT 模型是一種在大規模證據檢索任務上具有強大表現的語言模型，它能夠更好地理解和建模語義關係。透過使用更強大的模型，我們期望能提高系統的準確性和預測能力。

此外，在資料處理方面，我們採取了合併訓練資料集、調整訓練和驗證資料的比例等方法。合併訓練資料集可以增加訓練資料的量，有助於模型學習更多樣的資料，提高預測的準確性。而調整訓練和驗證資料的比例，我們發現將訓練資料比例設置為 10:1 時在預測測試資料時有更好的表現。這些方法都是為了最大限度地利用資料和提升模型的效能。

總之，我們在演算法和模型架構上採用了 TF-IDF 演算法、LERT 模型以及資料處理的創新方法，這些方法都有助於提高系統的效能和準確性。透過這些創新性的方法，我們能夠更好地處理文檔檢索和預測任務，從而提供更優秀的結果。

肆、資料處理

在模型訓練中，我們總共擁有兩份訓練資料集。為了增加訓練資料的量，我們將兩份資料集合併成一份訓練資料集，以擁有更多樣本以加強模型學習並提高預測準確性。

在劃分訓練資料和驗證資料時，我們測試了兩種不同的比例，8:2 和 10:1。這些比例代表將資料分成訓練集和驗證集的比例。通過這樣的設置，我們可以在訓練過程中使用更多的資料來訓練模型，同時保留一部分資料作為驗證集來評估模型的性能。在觀察實驗結果時，我們發現將訓練資料和驗證資料的比例設置為 10:1 時，模型在預測測試資料時表現更優秀。這意味著更大比例的訓練資料有助

於提升模型的泛化能力，使其能夠更好地應對未見過的測試資料。這樣的設置提供了更多的資料供模型學習，從而使模型能夠更準確地進行預測。

然後在第二階段，資料方面，將句子輸入改為：Claim + WikiPageName + 上一句 + 本句 + 下一句 （+ 中間會加 [SEP] token），原因是因為某一些句子只是一些數字，或者需要前/下一句才能知道它說什麼，所以就加了上一句及下一句的文字，讓 Model 比較好學習。

Training data：

原本 label 1 把所有句子合在一起，現在 label 1 會把每一句分開。

Origin:
Claim 1 Sent 1 + Sent 2 + Sent 3

Now:
Claim 1 Sent 1
Claim 1 Sent 2
Claim 1 Sent 3

然後進行 label 0 的標示，首先會數有多少 label 1，在 evidence 中 page 不存在的句子，就會標示為 label 0。

例如：一個 page 有 5 個句子，sent 1 和 sent 2 存在，那 sent 3 到 5 就是 label 0。

接著，我們再從 predicted_evidence 中取尚未標示為 label 0 的句子。為了避免 label 0 的資料過多，此時滿足條件的句子其中只有 10% 機率會被標示為 label 0 納入。

Val Data：同上，但不會針對不存在的句子做計數，所有存在的句子標為 1，不存在的標為 0，即不存在的不會因為機率或配合 1 的數量做刪減。

因此最後，訓練資料集大小為：

train_preprocessed length: 49374

0	37929
1	11445

伍、訓練方式

在第二階段 （sentence retrieval），我們以下方式對模型進行訓練：

1. 資料前處理：我們首先 Training Data 分成 Train 跟 Dev，然後把 Data 調整成我們的格式，轉成 pandas dataframe 然後包裝成我們的 Custom Dataset，在 Custom Dataset 中我們會把句子拼合增加 Special Token 然後輸入 Tokenizer 供給模型訓練。

2. 資料載入：然後我們會把 Dataset 包裝成 DataLoader，訓練用的 batch size 為 32，然後會隨機排次序，檢測用的 batch size 為 256。
3. 模型建立：首先載入 hfl/chineses-lert-large BertModel，然後在 pooler_output 增加 Dropout（機率為 0.3），最後把輸出改為 Binary Classification。
4. 優化器和學習率：使用 AdamW 優化模型參數，然後使用 ReduceLROnPlateau 配合 Validation loss 設定 learning rate（一開始為 2e-5）。
5. 訓練過程：Loss function 我們使用 BCEWithLogitsLoss 及計算 class weight，然後每跑一個 batch 就會更新參數一次。
6. 驗證和保存模型：我們使用了 classification report 及自行計算平均 `precision`、`recall`、`f1 score`，我們會把最高平均 f1 score 的模型儲存並後面使用。

而在第三階段 （claim verification） 中，我們使用了以下方式進行模型訓練:

1. 資料前處理：我們使用了 Hugging Face 庫中的 AutoTokenizer 從預訓練模型中載入分詞器 （tokenizer） 。並使用 AicupTopkEvidenceBERTDataset 類別來處理訓練資料和驗證資料，設定了最大序列長度（MAX_SEQ_LEN）。
2. 資料載入：我們使用 DataLoader 來將訓練資料和驗證資料進行批次（batch） 載入。設定了訓練資料的批次大小 （TRAIN_BATCH_SIZE） 和驗證資料的批次大小 （TEST_BATCH_SIZE）。
3. 設備配置：我們根據系統中可用的設備來選擇運算資源。如果系統中有可用的 GPU，我們使用 CUDA 來配置模型和資料處理的運算資源，以充分利用 GPU 的並行運算能力。如果系統中沒有可用的 GPU，我們會將模型和資料處理的運算資源放置在 CPU 上進行運算。儘管 CPU 的運算速度可能相對較慢，但仍然能夠完成模型的訓練以及推論任務。
4. 模型建立：我們使用 AutoModelForSequenceClassification 從預訓練模型中載入模型。
5. 優化器和學習率：我們使用 AdamW 優化器來優化模型參數，並根據訓練步驟的數量和學習率 （LR） 設定訓練的學習率。
6. 訓練過程：我們使用迴圈進行多個訓練迭代。在每個訓練迭代中，我們將批次資料送入模型進行預測，並計算損失 （loss）。然後，我們根據損失進行反向傳播 （backpropagation） 和參數更新。同時，我們記錄訓練損失和準確率到 TensorBoard 中。
7. 驗證和保存模型：在每個驗證步驟 （VALIDATION_STEP），我們使用驗證資料集 （eval_dataloader） 對模型進行評估，並記錄評估結果到

TensorBoard 中。同時，我們根據驗證準確率 （val_acc） 保存模型的 checkpoint。

以上是我們在第三階段訓練中的方式。透過這樣的訓練流程，我們能夠有效地利用資料進行模型的訓練，並持續地評估模型的性能。

陸、分析與結論

在第一階段中，我們使用 tf-idf 進行文本檢索，並取得了相當不錯的 precision 以及 recall，如下圖 ：

```
Precision: 0.6024289658247152
Recall: 0.9000830232465085
```

Figure 1

在第二階段中，我們把輸入句子增加了 Wiki page 的標題上一句及下一句嘗試增加模型辨識能力，測試中如果我們使用這個方法，public score 可以增加大約 0.03 ~ 0.05 的分數。在自己第二階段的測試資料集中，也能夠提升大約 0.03 f1 score。

而在第三部分，也就是論證驗證 （claim verification） 的過程中，我們嘗試了很多取得證據的方法。我們觀察了訓練數據，發現在標籤為 "Not enough info" 的情況下，後面的證據欄位會是空的。這對我們的模型訓練會產生不良的影響，使得驗證損失 （val_loss） 不斷上升，如下圖 ：

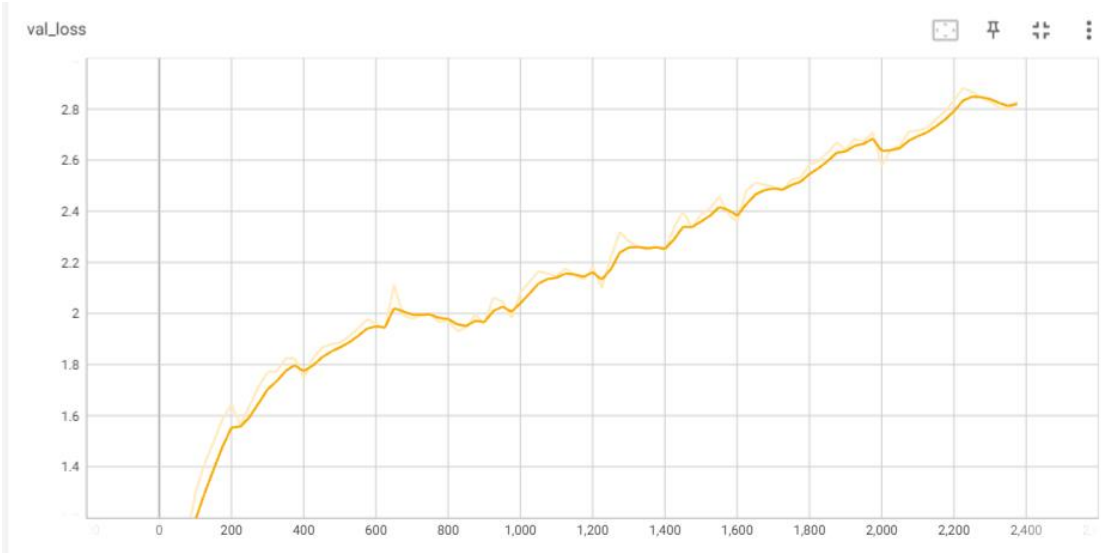


Figure 2

在發現了這個問題後，我們嘗試了兩種方法來解決這個問題：

方法一：

當我們的模型訓練遇到 "Not enough info" 標籤時，我們選擇使用 predicted_evidence 欄位中的第一個元素作為訓練時的證據列表（evidence_list）。為了避免 predicted_evidence 欄位也為空的情況，我們在第一階段設置了至少要選取一個 `predicted_page`。實作後，驗證損失（val_loss） 的變化如下:

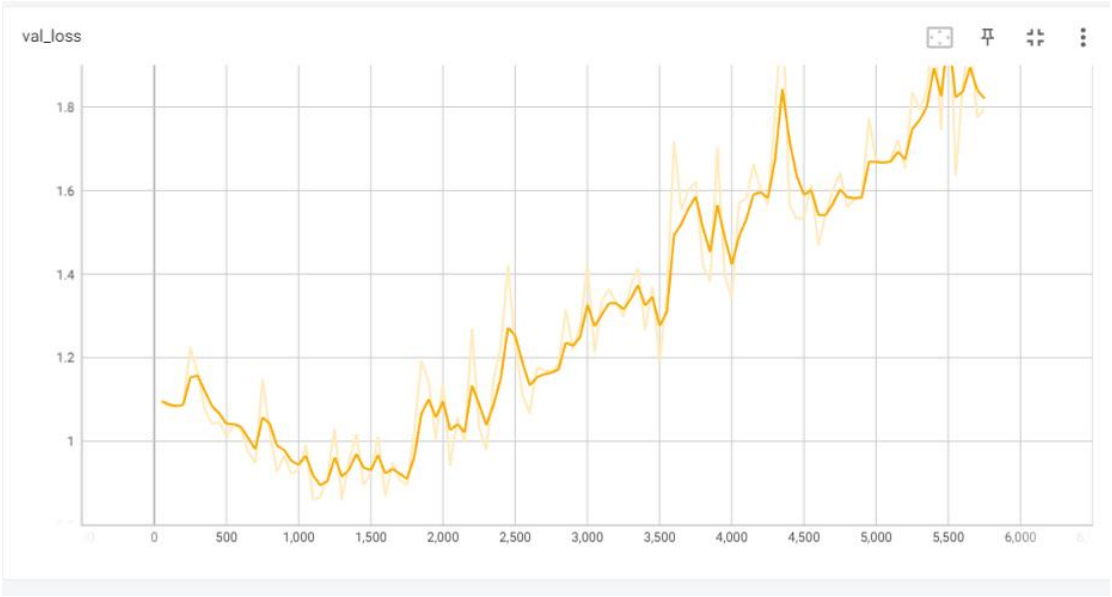


Figure 3（方法一 val_loss 之變化）

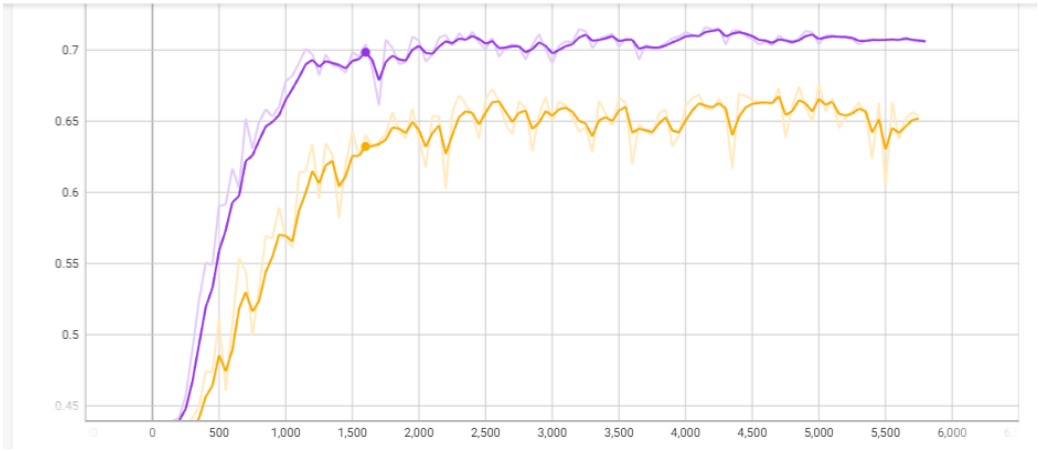
可以看到，驗證損失 （val_loss） 呈現了預期的先下降後上升的趨勢。而為了進一步提高驗證準確率 （val_acc），我們嘗試了以下另一種方法。

方法二:

我們直接取 predicted_evidence 作為訓練用的 evidence_list，儘管正確的 evidence 可能不在 predicted_evidence 中，但可以達到對三種結果 （`support`、`refute` 和 not enough info） 的公平性。測試結果如下圖:

Figure 4（方法二 val_loss 之變化）

兩者的 val_acc 和 val_loss 比較如下圖：



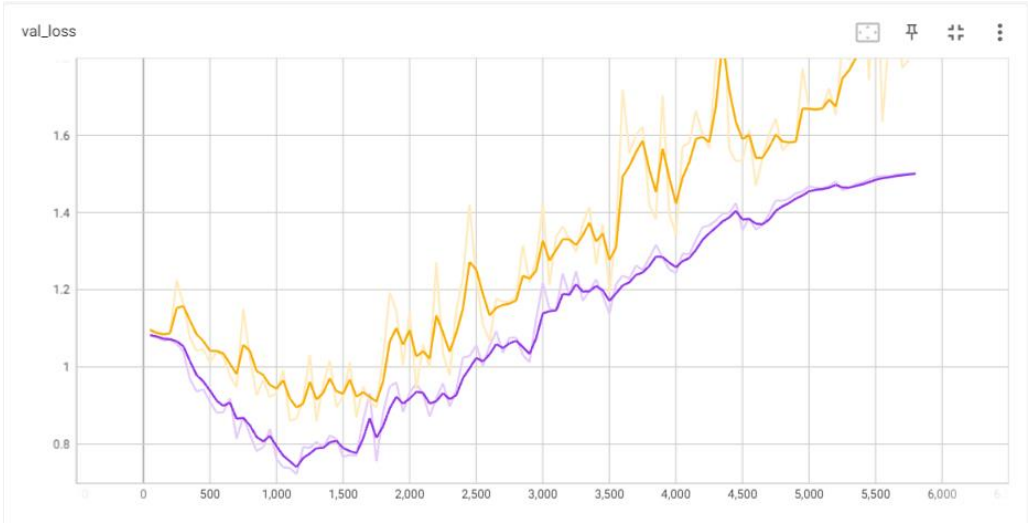


Figure 5（val_acc 之比較）

Figure 6（val_loss 之比較）

可以看到，方法二達到了更理想的 val_loss 和 val_acc 變化，因此我們最後實作也選了方法二作為我們第三部分的分析分法，在最後也取得了 0.598 的 public score 以及 0.689 的 private score。

柒、程式碼

GitHub 連結：https://github.com/shiroe345/AI_cup_2023_fifthplace.git

捌、使用的外部資源與參考文獻

無。

報告作者聯絡資料表

隊伍名稱	TEAM_3598	Private Leaderboard 成績	0.689	Private Leaderboard 名次	5
身分	姓名	學校＋系所中文全稱	學校＋系所英文中文全稱	電話	E-mail
隊長	黃學智, Wong Hok Chi Marco	國立成功大學資訊工程學系	National Central University Department of Computer Science & Information Engineering	0976-228-402	wonghokchi0402@gmail.com
隊員 1	李承哲 CHENG-JHE, LEE	國立成功大學資訊工程學系	National Cheng Kung University Department of Computer Science and Information Engineering	0966-291-402	gq4575@gmail.com
隊員 2	陳宥橋 YU-CHIAO, CHEN	國立成功大學資訊工程學系	National Cheng Kung University Department of Computer Science and Information Engineering	0983-083-605	chenforwork1368@gmail.com
隊員 3	朱祐麟 YU-LIN, CHU	國立成功大學資訊工程學系	National Cheng Kung University Department of Computer Science and Information Engineering	0968-936-832	dodo920306@gmail.com
指導教授資料					
	指導教授中文姓名	指導教授英文姓名	任職學校＋系所中文全稱	任職學校＋系所英文全稱	E-mail

教授 1	高宏宇	Hung-Yu, Kao	國立成功大學 資訊工程學系	National Cheng Kung University Department of Computer Science and Information Engineering	hykao@mail.ncku.edu.tw