student ID : 114527010

name : 黃子凌

1. Optimizer Selection

在指令分類作業中，我嘗試使用其他優化器，例如: AdamW、RMSprop、Adadelta。在評估模型指標 precision、recall、F1-score、support 後，SGD 的訓練結果是最正常的且穩定的，所以決定選擇 **SGD(隨機梯度下降)**當作優化器。

Working Principles of Common Optimizers

- **SGD**

  運作原理: 隨機抽樣 → 計算梯度 → 更新參數

  SGD 與一般梯度下降法(Batch Gradient Descent, BGD)不同的是，SGD 不需所有資料，僅隨機取一筆樣本或一個 mini-batch 來計算梯度，計算方式如下:

  $$\theta = \theta - \eta \cdot \nabla_\theta L_i(\theta)$$

  其中，θ是模型參數，η是學習率，$L_i(\theta)$是第i筆樣本的損失。

  雖然僅取一個 mini-batch 的更新較不精確，但其計算量小，速度快。隨機抽樣的好處是，每次更新使用不同的樣本，更新方向會有「抖動」，不像全梯度一樣穩定。而「抖動」能讓模型:

  1. 有機會跳出局部最小值 (local minima)，較有機會找到更好的解。
  2. 避免陷入 saddle point，在梯度幾乎為 0 時會繼續找梯度更低的點。
  3. 改善泛化能力，因為 SGD 更新有隨機性，不會完美對齊每個訓練樣本，對未知資料更穩健。

- **AdamW**

  AdamW 是 Adam 的改良版，主要針對 L2 正則化（weight decay）進行修正，使正則化效果更合理。以下是它的運作原理：

  原本 Adam 結合了 **Momentum**（一階矩估計）和 **RMSProp**（二階矩估計）

  $$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
  $$v_t = \beta_2 v_{t-1} + (1-\beta_2) {g_t}^2$$

$$\hat{m}_t = \frac{m_t}{1-{\beta_1}^t}$$

$$\hat{v}_t = \frac{v_t}{1-{\beta_2}^t}$$

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

其中 g_t 是梯度，β_1、β_2是動量衰減率，η是學習率。

Adam 對 L2 正則化（weight decay）其實把它當作梯度的一部分，與學習率自動縮放混在一起，這會造成正則化效果不如預期。將 weight decay 從梯度更新中分離出來，單獨做權重衰減：

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \eta\lambda\theta_t$$

其中，λ是 weight decay 係數。

此改進可以讓正則化不會被自適應學習率影響，提高模型泛化能力。

- **RMSprop（Root Mean Square Propagation）**

  運作原理: 計算梯度 → 計算梯度平方的指數移動平均（EMA） → 更新參數

  是一種自適應學習率的優化器，它主要用來解決梯度消失或梯度震盪的問題。RMSprop 希望每個參數使用自己的學習率，並且根據梯度的變化自動調整。若某個參數的梯度很大，學習率會自動縮小；若梯度很小，學習率會放大。避免在不同方向上更新不平衡的問題。其更新公式如下:

  先計算梯度:

  $$g_t = \nabla_\theta(\theta_t)$$

  計算梯度平方的指數移動平均（EMA）:

  $$v_t = \beta v_{t-1} + (1-\beta){g_t}^2$$

  其中，v_t是平均梯度平方，用來衡量梯度大小；β通常設 0.9。

  EMA 可以平滑梯度，減少震盪。

  更新參數：

  $$\theta_{t+1} = \theta_t - \cdot \frac{\eta}{\sqrt{v_t} + \epsilon} - g_t$$

  其中，η是全局學習率；ϵ是防止除零的小常數（如10^{-8}）。

Impact on Convergence Speed and Performance

- **SGD**

  雖然收斂速度不如其他優化器迅速，但其穩定性與泛化能力更具可信度。

  Convergence Speed:

  訓練與驗證曲線呈現穩定下降與上升，雖然收斂速度相對較慢，但這也代表模型在逐步學習，避免過快陷入過擬合。

  Loss 從約 1.1 降到 0.6，Accuracy 從約 60%提升到 90%，展現出穩健的學習過程。

  Performance:

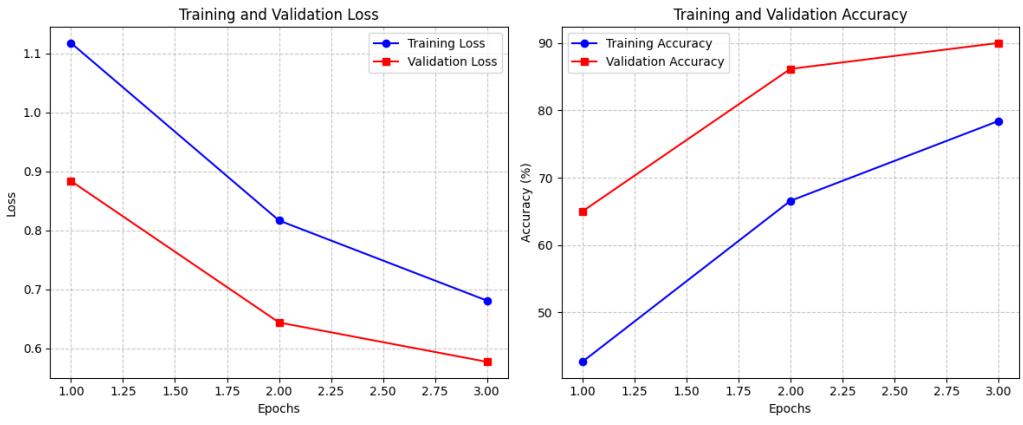
  Accuracy 為 90%。

  各類別的 Precision 與 Recall 表現均衡，尤其 Class 2 的 F1-score 為 0.9559。

  Confusion Matrix 中雖有些誤分類，但整體預測分布合理，反映出模型在真實情境下的泛化能力。

	precision	recall	f1-score	support
0	0.9080	0.8229	0.8634	96
1	0.8738	0.9184	0.8955	98
2	0.9286	0.9848	0.9559	66
accuracy			0.9000	260
macro avg	0.9035	0.9087	0.9049	260
weighted avg	0.9003	0.9000	0.8990	260

```
[[79 13  4]
 [ 7 90  1]
 [ 1  0 65]]
```



- **AdamW**

  雖然 AdamW 的結果看似理想，但過度完美的表現可能反映資料集特性不適合用來評估其泛化能力，需要驗證是否資料外洩或過擬合現象。

  Convergence Speed:

  在第 2 個 epoch 就達到 100% Accuracy 與近乎 0 的 Loss，收斂速度極快。Loss 從約 1.1 降到 0.6，Accuracy 從約 60%提升到 90%，但仍未達到飽和。

  Validation Accuracy 從一開始就維持在 100%，可能暗示資料集過於簡單或存在資料洩漏。

  Performance:

  Precision、Recall、F1-score 全為 1.0000，Classification Report 過於完美。

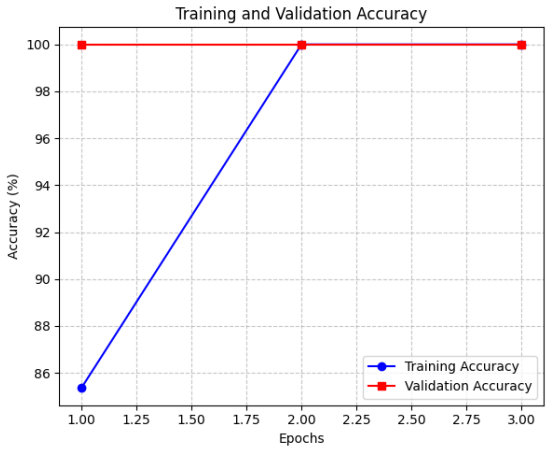
  Confusion Matrix 無誤分類，過於理想。

```
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000        96
           1     1.0000    1.0000    1.0000        98
           2     1.0000    1.0000    1.0000        66

    accuracy                         1.0000       260
   macro avg     1.0000    1.0000    1.0000       260
weighted avg     1.0000    1.0000    1.0000       260

[[96  0  0]
 [ 0 98  0]
 [ 0  0 66]]
```



- **RMSprop**

RMSprop 的結果雖然亮眼，但與 AdamW 一樣，過度完美的表現可能來自資料集特性或模型過度擬合。若要評估其真實效能，建議使用更具挑戰性的資料或進行 cross-validation。

Convergence Speed:

RMSprop 的訓練與驗證曲線與 AdamW 類似，快速達到 100% Accuracy 且 Loss 幾乎為 0。

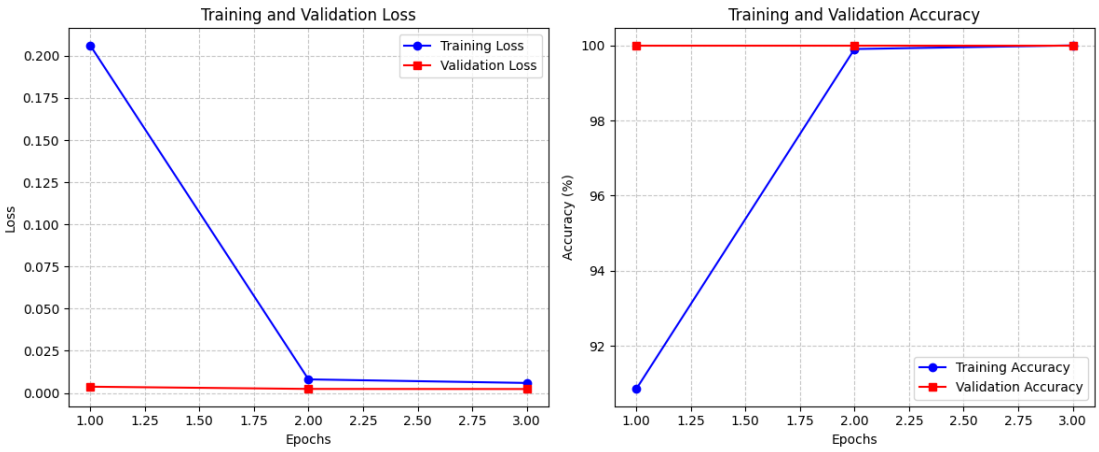
Validation Accuracy 從頭到尾都未變動，可能代表模型已記住資料而非真正學習。

Performance:

Precision、Recall、F1-score 全為 1.0000，Classification Report 過於完美。
Confusion Matrix 無誤分類，此結果在多數真實任務中不具代表性。

	precision	recall	f1-score	support
0	1.0000	1.0000	1.0000	96
1	1.0000	1.0000	1.0000	98
2	1.0000	1.0000	1.0000	66
accuracy			1.0000	260
macro avg	1.0000	1.0000	1.0000	260
weighted avg	1.0000	1.0000	1.0000	260

```
[[96  0  0]
 [ 0 98  0]
 [ 0  0 66]]
```



Justification for Choosing SGD

在本次分類任務中，AdamW 與 RMSprop 過度理想的表現反而引發對資料集適配性與模型泛化能力的疑慮。相較之下，SGD（Stochastic Gradient Descent）雖然收斂速度較慢，但其訓練過程更具穩定性，且模型在驗證集上的表現更貼近真實情境。

SGD 的訓練與驗證曲線呈現逐步改善，模型在不同類別上的 Precision 與 Recall 分布合理，並未出現過度擬合或資料洩漏的跡象。這使得 SGD 成為更可信的選擇，特別是在需要模型具備良好泛化能力與可解釋性的應用場景中。

2. Model Selection and Training Strategy

在這次指令分類，我用了 BERT 的中文預訓練模型（bert-base-chinese），並透過 HuggingFace 的 AutoTokenizer 與 AutoModel 進行初始化。Tokenizer 將原始文字轉換為 BERT 所需的輸入格式，包括 input_ids、attention_mask 和 token_type_ids。模型設計為三分類任務（替換 / 插入 / 刪除），最終輸出層定義為 Linear(256, 3)。

除了使用[CLS] token 作為句子表示外，我進一步強化模型架構，加入以下元件以提升效能與穩定性：

- 一層 Linear(768 → 256) 的預分類器（pre-classifier）
- ReLU 激活函數
- LayerNorm 正規化層
- Dropout(0.3) 以減少過擬合
- 最終分類層 Linear(256 → 3)

在資料前處理方面，我移除了重複句子，並採用 5-fold StratifiedKFold 交叉驗證，以確保各分類在訓練與驗證集中的分布均衡。測試集則以 30%的比例從原始資料中切分，並檢查並排除訓練、驗證、測試集之間的重疊。雖然本實驗設計採用 5-fold StratifiedKFold 交叉驗證以確保資料分布均衡，但考量訓練時間與硬體資源限制（如 CPU 過載），最終僅使用 fold 0 的訓練結果進行測試與分析。此策略雖可能略減模型在不同資料切分下的泛化評估完整性，但在本任務中已能提供穩定且具代表性的性能指標。

訓練策略如下：

- 優化器：我選擇使用 SGD 並設定 momentum=0.9。儘管 AdamW、RMSprop、Adadelta 等自適應型優化器在理論上具有較快的收斂速度，但在本任務中表現不穩定，常導致過擬合或驗證指標波動。相較之下，SGD 提供了最穩定的結果，訓練與驗證曲線規律，且在測試集上展現出平衡的 Precision、Recall 與 F1-score。
- 學習率：設定為 LEARNING_RATE = 2e-5，並根據實驗結果進行調整。
- 批次大小：訓練使用 TRAIN_BATCH_SIZE = 16，驗證與測試使用 VALID_BATCH_SIZE = 16。
- 訓練輪數：共訓練 EPOCHS 輪，依作業規定 EPOCHS = 3。
- 損失函數：使用 CrossEntropyLoss 並加入類別權重以處理標籤不平衡問題。
- 學習率排程器：採用 get_linear_schedule_with_warmup，前 10%的訓練步驟用於 warmup，以穩定初期訓練。

此訓練策略具備以下優勢：

- 類別加權(class_weights)與分層抽樣(Stratified Sampling)有助於減少模型對多數類別的偏誤。
- 本次僅使用 fold 0 進行訓練，SGD 在該折中表現穩定，驗證集指標均衡，適合作為本任務的優化器選擇。

3. **Ablation Study**

Tokenizer and Model Choice

我選擇了 bert-base-chinese 作為 tokenizer 與 pretrain model。雖然曾考慮使用其他中文模型，如 RoBERTa-wwm-ext，但初步實驗顯示 RoBERTa 在測試集上的 loss 曲線波動較大，F1-score 表現亦不穩定。相較之下，BERT 展現出更穩定的訓練，且與 HuggingFace 的 tokenizer 相容性更佳，所以最後就決定用 BERT。

Output Design and Architectural Enhancements

- **Baseline Design**: 直接使用 [CLS] token 接上 Linear(768 → 3) 層。此簡化架構容易造成過擬合，且驗證集表現不穩定。
- **Dropout Addition**: 在最終分類器前加入 Dropout(0.3) 可有效減少過擬合，並提升驗證集上的 F1-score。

- **ReLU Activation and LayerNorm**: 在中間層 Linear(768 → 256) 後加入 ReLU 激活函數與 LayerNorm 正規化，有助於穩定訓練過程並平滑 loss 曲線。
- **Classifier Design**: 採用兩層分類器 Linear(768 → 256 → 3) 取代直接的 Linear(768 → 3)，可提取更抽象的語意特徵，進而提升分類準確率。

Observations and Predicted Effects

整體來說，這些架構上的調整確實有幫助，讓模型在 Precision、Recall 和 F1-score 之間的表現更平衡。Dropout 和 LayerNorm 對模型的泛化能力和穩定性有明顯改善，而中間轉換層則讓模型更能抓住語意上的差異。我覺得這些設計如果用在更大的資料集或更複雜的分類任務上，應該也會有不錯的效果。