Colab 連結

https://colab.research.google.com/drive/1fRFfINRcqqNsAdcSnqBEwUhpQeCyY_HX?usp=share_link

原始參數(linear 768 dropout 0.3)

```
100%|██████████| 250/250 [06:26<00:00,  1.54s/it]
[epoch 1] cost time: 422.0944 s
          loss     acc     f1      rec     prec
train | 0.4564, 0.7720, 0.7819, 0.8019, 0.7629
val   | 0.3614, 0.8383, 0.7956, 0.6992, 0.9546

100%|██████████| 250/250 [06:13<00:00,  1.49s/it]
[epoch 2] cost time: 409.1033 s
          loss     acc     f1      rec     prec
train | 0.2091, 0.9273, 0.9281, 0.9205, 0.9357
val   | 0.3059, 0.8790, 0.8619, 0.8294, 0.9134

100%|██████████| 250/250 [06:13<00:00,  1.49s/it]
[epoch 3] cost time: 409.1934 s
          loss     acc     f1      rec     prec
train | 0.1101, 0.9645, 0.9651, 0.9617, 0.9684
val   | 0.3345, 0.8909, 0.8841, 0.8749, 0.9068
```

```
correct = 0
for idx, pred in enumerate(res['pred']):
    if pred == res['label'][idx]:
        correct += 1
print('test accuracy = %.4f'%(correct/len(test_df)))
```

```
test accuracy = 0.8874
```

加入 scheduler (+0.026)

get_linear_schedule_with_warmup 為優化器提供一個線性的學習率調度器，同時還可以在訓練的前幾個 epoch 中預熱學習率。預熱學習率是指在訓練開始時，先使用一個較小的學習率，然後逐漸增加到預設的學習率。這樣做的目的是為了避免在訓練開始時，由於權重隨機初始化而導致的梯度爆炸或梯度消失問題，其中 num_warmup_steps 是預熱步數；num_training_steps 是總步數，結果有些過擬合，準確度還是明顯上升不少。

```
scheduler = get_linear_schedule_with_warmup(optimizer,
            num_warmup_steps=int(0.2*len(train_loader)),
            num_training_steps=len(train_loader)*parameters['epochs'])
```

```
scheduler.step()
```

```
100%|██████████| 250/250 [06:26<00:00,  1.55s/it]
[epoch 1] cost time: 422.4702 s
          loss     acc     f1      rec     prec
train | 0.4289, 0.7953, 0.7989, 0.7979, 0.7999
val   | 0.2520, 0.9067, 0.8909, 0.8703, 0.9312

100%|██████████| 250/250 [06:13<00:00,  1.49s/it]
[epoch 2] cost time: 409.4623 s
          loss     acc     f1      rec     prec
train | 0.1564, 0.9477, 0.9486, 0.9461, 0.9512
val   | 0.2766, 0.9127, 0.9017, 0.9030, 0.9168

100%|██████████| 250/250 [06:13<00:00,  1.49s/it]
[epoch 3] cost time: 409.4876 s
          loss     acc     f1      rec     prec
train | 0.0176, 0.9970, 0.9971, 0.9980, 0.9961
val   | 0.3464, 0.9137, 0.9000, 0.8816, 0.9322
```

```
correct = 0
for idx, pred in enumerate(res['pred']):
    if pred == res['label'][idx]:
        correct += 1
print('test accuracy = %.4f'%(correct/len(test_df)))
```

```
test accuracy = 0.9138
```

參考資料
https://stackoverflow.com/questions/60120043/optimizer-and-scheduler-for-bert-fine-tuning

縮小 learning rate

```
100%|██████████| 250/250 [06:21<00:00,  1.53s/it]
[epoch 1] cost time: 417.3081 s
          loss     acc      f1      rec     prec
train | 0.3937, 0.7985, 0.7965, 0.7925, 0.8005
val   | 0.1966, 0.9276, 0.9178, 0.9086, 0.9398

100%|██████████| 250/250 [06:07<00:00,  1.47s/it]
[epoch 2] cost time: 403.1317 s
          loss     acc      f1      rec     prec
train | 0.1478, 0.9473, 0.9468, 0.9442, 0.9495
val   | 0.1820, 0.9355, 0.9276, 0.9293, 0.9337

100%|██████████| 250/250 [06:08<00:00,  1.47s/it]
[epoch 3] cost time: 403.8954 s
          loss     acc      f1      rec     prec
train | 0.0348, 0.9915, 0.9914, 0.9894, 0.9934
val   | 0.2272, 0.9286, 0.9230, 0.9298, 0.9273
```

```
correct = 0
for idx, pred in enumerate(res['pred']):
    if pred == res['label'][idx]:
        correct += 1
print('test accuracy = %.4f'%(correct/len(test_df)))
```

```
test accuracy = 0.9188
```

以下結果因 Google Colab GPU 免費版達使用上限改用本機顯卡

增加 sample size 並改用 AdamW

增加 4000 單位訓練資料 1000 單位校驗資料，可以使模型更精準、抑制過擬合，Adam 和 AdamW 都是深度學習中常用的優化器。 Adam 是一種自適應學習率的優化算法，它可以自動調整每個參數的學習率，從而加快模型的收斂速度。而 AdamW 是在 Adam 的基礎上進行改進的算法，它引入了 L2 正則化項，可以有效地防止過擬合。在實際使用中，如果模型出現過擬合現象，可以嘗試使用 AdamW 來優化模型 123。

eps 是 Adam 和 AdamW 中的一個參數，它是一個很小的數值，用於防止除以 0 的情況發生。在計算梯度時，如果梯度值很小，那麼更新後的權重值也會很小，這樣就會導致學習率變得很大。為了避免這種情況發生，我們需要在計算

梯度時加上一個很小的數值 eps。

```
100%|██████████| 500/500 [12:43<00:00,  1.53s/it]
[epoch 1] cost time: 833.4126 s
         loss     acc      f1       rec     prec
train | 0.3307, 0.8441, 0.8417, 0.8384, 0.8451
val   | 0.1982, 0.9225, 0.9171, 0.9268, 0.9178

100%|██████████| 500/500 [12:18<00:00,  1.48s/it]
[epoch 2] cost time: 808.6009 s
         loss     acc      f1       rec     prec
train | 0.1115, 0.9623, 0.9620, 0.9664, 0.9577
val   | 0.1871, 0.9285, 0.9222, 0.9167, 0.9351

100%|██████████| 500/500 [12:18<00:00,  1.48s/it]
[epoch 3] cost time: 809.0458 s
         loss     acc      f1       rec     prec
train | 0.0273, 0.9940, 0.9939, 0.9947, 0.9932
val   | 0.2561, 0.9235, 0.9177, 0.9115, 0.9360
```

```
correct = 0
for idx, pred in enumerate(res['pred']):
    if pred == res['label'][idx]:
        correct += 1
print('test accuracy = %.4f'%(correct/len(test_df)))
```

```
test accuracy = 0.9316
```

參考資料

https://zhuanlan.zhihu.com/p/39543160

減少 epochs

從以上更動後的訓練過程可以發現，普遍會在第二回合後過擬合，於是我想將 epoch 停在第一回合結束，測試精準度是否更好，最後，雖只有一回但效果也很不錯

```
100%|██████████| 500/500 [12:46<00:00,  1.53s/it]
[epoch 1] cost time: 836.8678 s
         loss     acc      f1       rec     prec
train | 0.2621, 0.8952, 0.8953, 0.8875, 0.9032
val   | 0.1833, 0.9275, 0.9226, 0.9175, 0.9374
```

```
correct = 0
for idx, pred in enumerate(res['pred']):
    if pred == res['label'][idx]:
        correct += 1
print('test accuracy = %.4f'%(correct/len(test_df)))
```

```
test accuracy = 0.9228
```

總結，以上為【疊加】式更動模型參數，隨著訓練回合增加，validation 的 loss 也會跟著增加，所以這裡只訓練三回合，且文本消耗的資源好像比上次圖像辨識來得多，甚至需要切換 Google 帳號，為的就是免費的 GPU，另外也發現文本過擬合的情況比圖像嚴重許多，不過最後精準度卻意外地好。