

深度學習HW4

AI二 B1228005 胡樂麒

1. 模型與超參數設定

我使用的是MODEL_TYPE =HF, 分別使用MODEL_NAME = "google/vit-base-patch16-224" 和"google/vit-base-patch16-224-in21k"模型來訓練。使用colab環境。

參數設定
BATCH_SIZE = 32
EPOCHS = 3
LEARNING_RATE = 1e-4
IMAGE_SIZE = 128
PATCH_SIZE = 16
MODEL_NAME = "google/vit-base-patch16-224" 和 "google/vit-base-patch16-224-in21k"
MODEL_TYPE = "HF"
optimizer=Adam
loss_fn=CrossEntropyLoss()

2. 有無使用LoRA 的影響

從結果可以看出來這兩個模型的F1 score與Accuracy在沒有使用LoRA的情況下皆高於有使用LoRA。

在 vit-base-patch16-224 中：

不使用 LoRA: Accuracy = 90.87%, F1 = 89.81%

使用 LoRA: Accuracy = 87.5%, F1 = 85.76%

在vit-base-patch16-224-in21k中：

不使用 LoRA 時: Accuracy = 91.99%, F1 = 91.09%

使用 LoRA 時: Accuracy = 87.5%, F1 = 87.69%

出現此結果的原因可能是因為，在LoRA 設定中 r=8、lora_alpha=16, 參數較少，可能沒辦法有效地調整模型。所以在之後的修改會分別將參數改成r=24,lora_alpha=48來比較結果。

model_name	是否使用LoRA	Accuracy	F1 score
google/vit-base-patch16-224	有	0.875	0.8576
google/vit-base-patch16-224	無	0.9087	0.8981
"google/vit-base-patch16-224-in21k"	有	0.875	0.8769
"google/vit-base-patch16-224-in21k"	無	0.9199	0.9109

3.加入LoRA參數對模型的影響

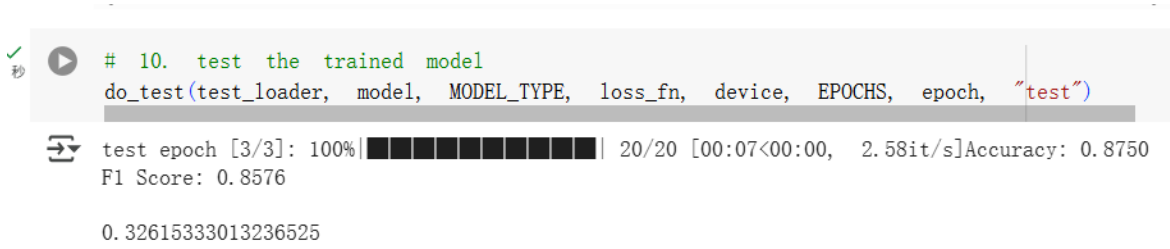
LoRA:不直接更新大型預訓練模型的權重,而是在某些權重矩陣旁邊插入一個「低秩的調整矩陣」來學習任務特定資訊。大幅減少需要訓練的參數量。

以下為參數調整前後的`可訓練參數量`

`可訓練參數數量: 443906 (0.51%)`

`可訓練參數數量: 1328642 (1.52%)`

由以下比較可知,在這次的作業中提高可訓練參數量,反而能提高模型的accuracy和F1 score。

MODEL_NAME = google/vit-base-patch16-224	
LoRA修改前	
 <pre> # 10. test the trained model do_test(test_loader, model, MODEL_TYPE, loss_fn, device, EPOCHS, epoch, "test") test epoch [3/3]: 100% ██████████ 20/20 [00:07<00:00, 2.58it/s]Accuracy: 0.8750 F1 Score: 0.8576 0.32615333013236525 </pre>	
LoRA修改後	

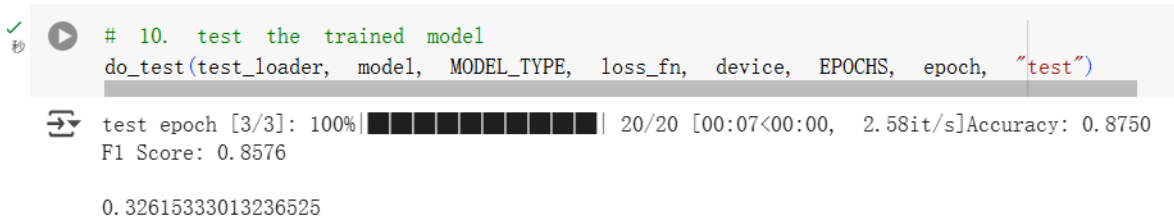

<div> <div> 7秒 </div> <div> <div># 10. test the trained model</div> <div>do_test(test_loader, model, MODEL_TYPE, loss_fn, device, EPOCHS, epoch, "test")</div> </div> <div> <div> test epoch [3/3]: 100% </div> <div> 20/20 [00:07<00:00, 2.60it/s]Accuracy: 0.9183 F1 Score: 0.9104 0.23416302390396596 </div> </div> </div>
<div>MODEL_NAME = "google/vit-base-patch16-224-in21k"</div>
<div>LoRA修改前</div>
<div> <div> 7秒 </div> <div> <div># 10. test the trained model</div> <div>do_test(test_loader, model, MODEL_TYPE, loss_fn, device, EPOCHS, epoch, "test")</div> </div> <div> <div> test epoch [3/3]: 100% </div> <div> 20/20 [00:07<00:00, 2.60it/s]Accuracy: 0.8750 F1 Score: 0.8569 0.3384483218193054 </div> </div> </div>
<div>LoRA修改後</div>
<div> <div> 7秒 </div> <div> <div>[103] # 10. test the trained model</div> <div>do_test(test_loader, model, MODEL_TYPE, loss_fn, device, EPOCHS, epoch, "test")</div> </div> <div> <div> test epoch [3/3]: 100% </div> <div> 20/20 [00:07<00:00, 2.60it/s]Accuracy: 0.8942 F1 Score: 0.8816 0.297581921890378 </div> </div> </div>

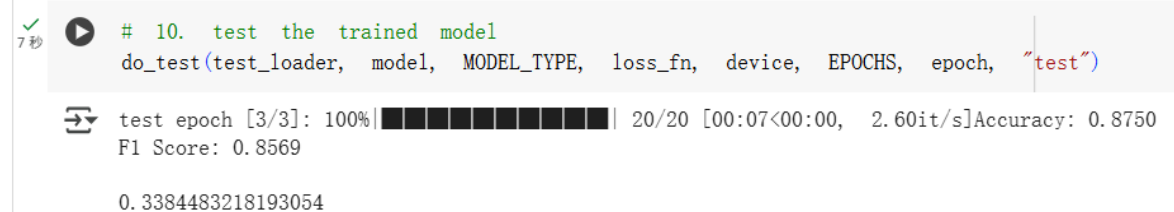
4.LoRA參數說明

LoRA參數說明	
r=8	矩陣的秩(rank)，控制 LoRA 矩陣的維度(越小訓練成本越低)
lora_alpha=16	Scaling factor，調整 LoRA 輸出的大小
lora_dropout=0.1	在 LoRA 上使用 dropout，防止過擬合
target_modules	指定哪些模組插入 LoRA，這裡是 "query", "key", "value" → Transformer
bias="none"	不調整原本模型的 bias(偏置項)
modules_to_save=["classifier"]	指定除了 LoRA 模組外，還要保留哪些模組參與訓練，這裡是保留 classifier 層

#調整r=24	擴大秩以增加可訓練參數量
#調整lora_alpha=48	提高縮放強度

5.Anything that can strengthen your report.

MODEL_NAME = google/vit-base-patch16-224 MODEL_TYPE=HF	
使用LoRA	
 <pre># 10. test the trained model do_test(test_loader, model, MODEL_TYPE, loss_fn, device, EPOCHS, epoch, "test") test epoch [3/3]: 100% ██████████ 20/20 [00:07<00:00, 2.58it/s]Accuracy: 0.8750 F1 Score: 0.8576 0.32615333013236525</pre>	
沒有使用LoRA	
 <pre># 10. test the trained model do_test(test_loader, model, MODEL_TYPE, loss_fn, device, EPOCHS, epoch, "test") test epoch [3/3]: 100% ██████████ 20/20 [00:07<00:00, 2.75it/s]Accuracy: 0.9087 F1 Score: 0.8981 0.3070465575903654</pre>	

MODEL_NAME = "google/vit-base-patch16-224-in21k" MODEL_TYPE=HF	
使用LoRA	
 <pre># 10. test the trained model do_test(test_loader, model, MODEL_TYPE, loss_fn, device, EPOCHS, epoch, "test") test epoch [3/3]: 100% ██████████ 20/20 [00:07<00:00, 2.60it/s]Accuracy: 0.8750 F1 Score: 0.8569 0.3384483218193054</pre>	
沒有使用LoRA	

✓ [30] # 10. test the trained model
7秒 do_test(test_loader, model, MODEL_TYPE, loss_fn, device, EPOCHS, epoch, "test")

↔ test epoch [3/3]: 100%|██████████████████| 20/20 [00:07<00:00, 2.77it/s]
Accuracy: 0.9199
F1 Score: 0.9109
0.2789418102242053