National Tsing Hua University Fall 2023 11210IPT 553000 Deep Learning in Biomedical Optical Imaging Homework 4 Description

109081009 李承叡

● Task A: Model Selection

我選了下面兩張圖的 model 做測試

ResNet34_Weights.IMAGENET1K_V1:

These weights reproduce closely the results of the paper using a simple training recipe. Also available as `ResNet34_Weights.DEFAULT`.

acc@1 (on ImageNet-1K)	73.314
acc@5 (on ImageNet-1K)	91.42
min_size	height=1, width=1
categories	tench, goldfish, great white shark, ... (997 omitted)
num_params	21797672
recipe	link
GFLOPS	3.66
File size	83.3 MB

圖 1, ResNet34 的 model(和助教範例的類似)

MobileNet_V3_Large_Weights.IMAGENET1K_V1:

These weights were trained from scratch by using a simple training recipe.

acc@1 (on ImageNet-1K)	74.042
acc@5 (on ImageNet-1K)	91.34
min_size	height=1, width=1
categories	tench, goldfish, great white shark, ... (997 omitted)
num_params	5483032
recipe	link
GFLOPS	0.22
File size	21.1 MB

圖 2, MobileNet_V3_Large 的 model(和 34 的 pretrain 結果類似，且跑得比較快)

理由: 選這兩個 model 的原因是這兩個的 acc@1 和 acc@5 都比較相近，而在 file size 卻差了 4 倍左右。而兩者的複雜程度則是 ResNet34 比較複雜。

● Task B: Fine-tuning the ConvNet

1. 使用 ResNet34 的結果

```
import torch.nn as nn
import torch.nn.functional as F
from torchvision import models

model = models.resnet34(weights='IMAGENET1K_V1')

# # ConvNet as fixed feature extractor (freeze parameters)
# for param in model.parameters():
#         param.requires_grad = False

num_ftrs = model.fc.in_features
# Here the size of each output sample is set to 2.
# Alternatively, it can be generalized to ``nn.Linear(num_ftrs, len(class_names))``.
model.fc = nn.Linear(num_ftrs, 2)
model = model.cuda()
print(model)
```

圖 3, ResNet34 的 model

圖 4, ResNet34 跑完每次要花的時間(10.37)

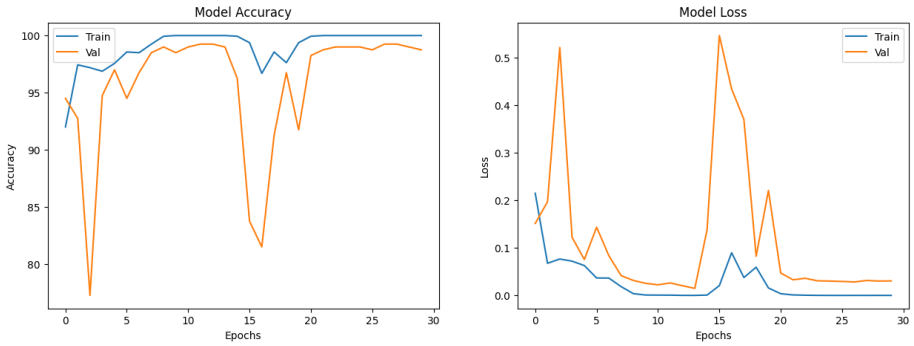

圖 5, ResNet34 的 accuracy 和 loss

圖 6, ResNet34 的最後 evaluation 的 accuracy

2. **使用 MobileNet_V3_Large 的結果**

```
import torch.nn as nn
import torch.nn.functional as F
from torchvision import models

model = models.mobilenet_v3_large(weights='IMAGENET1K_V1')

# # ConvNet as fixed feature extractor (freeze parameters)
# for param in model.parameters():
#         param.requires_grad = False

num_ftrs = model.classifier[3].in_features

# Here the size of each output sample is set to 2.
# Alternatively, it can be generalized to ``nn.Linear(num_ftrs, len(class_names))``.
model.classifier[3] = nn.Linear(num_ftrs, 2)
model = model.cuda()
print(model)
```

圖 7, MobileNet_V3_Large 的 model

圖 8, MobileNet_V3_Large 每次要花的時間(6.23)

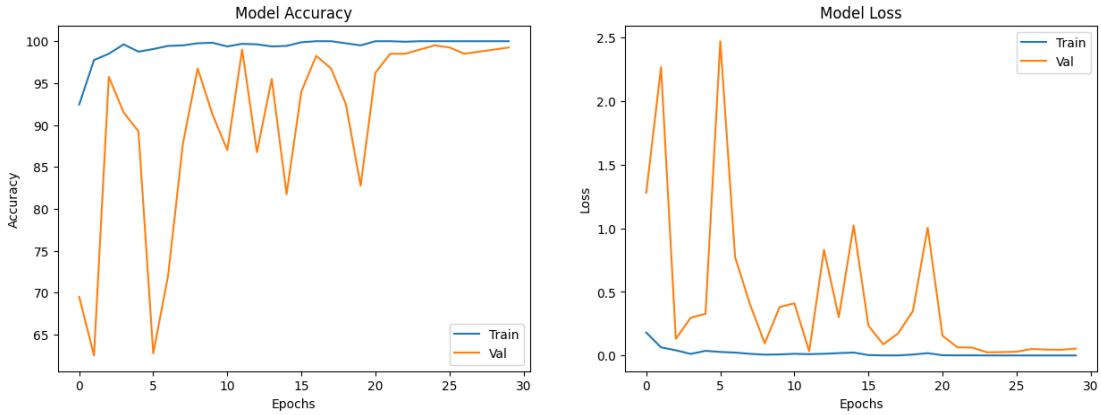

圖 9, MobileNet_V3_Large 的 accuracy 和 loss

圖 10, MobileNet_V3_Large 最後 evaluation 的 accuracy

Discussion: 從上述的比較來看 ResNet34 和 MobileNet_V3_Large 最終的 evaluation 結果其實是差不多，而時間上也和預期的一樣是 ResNet34 要花得比較久，不過很明顯可以看到 ResNet34 accuracy 的震盪頻率是比較低的，相反得 MobileNet_V3_Large 則震盪的非常多次，因為這兩個 model 在 pre train 的結果是差不多的，所以可能是因為 ResNet34 比較複雜，所以才比較不會一直震盪。

不過從 train 和 validation 的兩個表現結果來看，可能 ResNet34 有一點 overfitting 的狀況，MobileNet_V3_Large 則比較沒有。

● Task C: ConvNet as Fixed Feature Extractor

都 Fixed 後如下 ResNet34 和 MobileNet_V3_Large 的結果如下

1. ResNet34

圖 11, ResNet34 每次要花的時間(4.05)

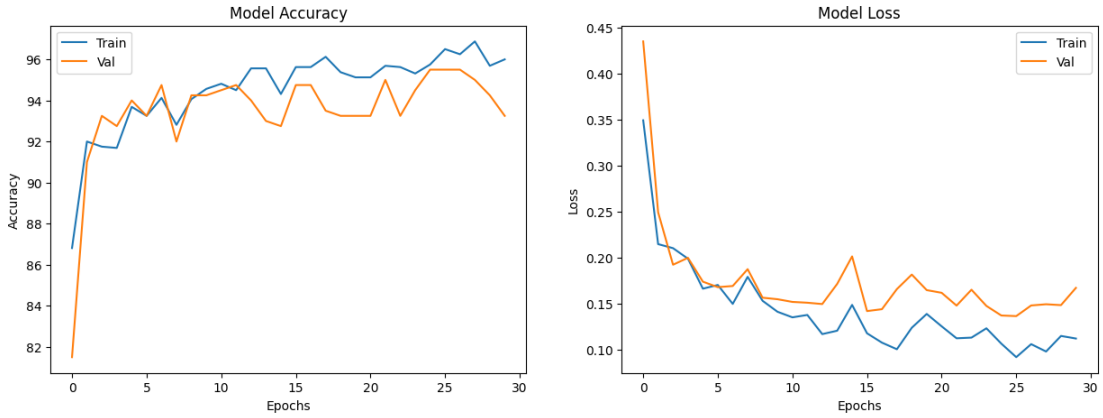

圖 12, ResNet34 accuracy 和 loss

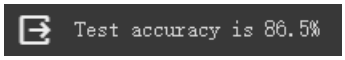

圖 13, test accuracy

2. MobileNet_V3_Large

圖 14, MobileNet_V3_Large 每次要花的時間(4.05)

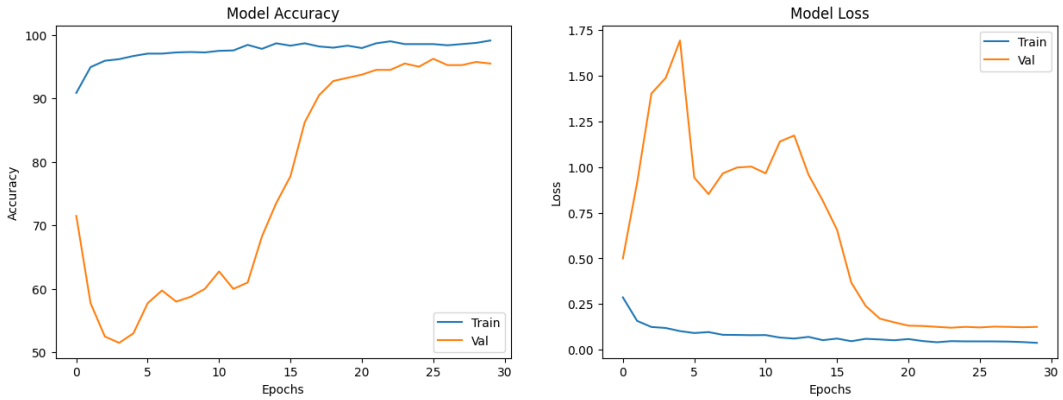

圖 15 MobileNet_V3_Large 的 accuracy 和 loss

```
Test accuracy is 82.0%
```

圖 1 所以 6 MobileNet_V3_Large test accuracy

Discussion: 在 fixed 後兩者在對 test 進行 evaluation 的結果是 ResNet34 比較好，然而在 training 和 validation 的表現上 ResNet34 卻比 MobileNet_V3_Large 還要差，並且 ResNet34 所花的時間也比較久。而震盪方面也是 ResNet34 比較震盪。

- **Task D: Comparison and Analysis**

因為我選的兩個 model 最主要的差別在於 model 的複雜程度，所以把 task B 和 C 比較後可以發現。

1. ResNet34(圖 5 和圖 12 比較)

ResNet34 是比較複雜的 model 在 train 和 validation 的 accuracy 震盪表現上，用 Task C 後震盪是比較多次的，並且 accuracy 也下降了，然而在進行 test 的表現上卻是使用 Task C 的預測表現比較好。

2. MobileNet_V3_Large(圖 9 和圖 15 比較)

MobileNet_V3_Large 是比較簡單的 model，train 和 validation 的 accuracy 震盪表現上用 Task B 後震盪是比較多次的，不過和 ResNet34 一樣在 accuracy 方面 Task B 是比較高的，但是在進行 test 的表現上卻是使用 Task C 的預測表現比較好。

所以最後我觀察到的是雖然在訓練的時候 task B 的表現都是比較好的，但是對於新的使用新的 test data 做測試，都可以發現使用 Task C 對於在陌生的 data 來說最終的 accuracy 是比較高的。

所以整體來看

Transfer model	Task B 震盪次數	Task C 震盪次數	(B)Train and validation performance	(C)Train and validation performance	Task B test Accuracy	Task C test Accuracy	執行速度
較複雜的 model	少	多	較好	較差	較低(80.5%)	較高(86.5%)	慢
較簡單的 model	多	少	較好	較差	較低 (79.75%)	較高(82.0%)	快

執行時間: Task B>Task C

Test Accuracy: Task B<Task C

Train and validation performance: Task B>Task C

● Task E: Test Dataset Analysis

In the original Lab 5's code, you may have encountered challenges in enhancing the performance on the test dataset.

Discussion: 這次最後的 test 的 accuracy 和在 train 做出來的 accuracy 差距會這麼大，我想有可能和 training data 和 test data 難度不同有關。

因為從 task D 的比較可以知道，實際上在 task B train model 的表現是比 task C 還要好的，從 train 和 validation 的都可以看出用 task B 的時候這兩者 accuracy 都是比較高的並且也是比較接近的，看起來比較沒有 overfitting。但是到了 evaluation 用 task B 對 test data 測試，跑出來的 accuracy 卻比 task C 還低了。

也就是說理論上，task B 應該要有更好的學習效果，因為畢竟 task B 是可以微調所有的 model 裡面的參數的，雖然有 overfitting 的風險(但此次從 train 和 validation 的 accuracy 來看 task B 反而比 Task C 還更沒有 overfitting)，但理論上微調更多的參數更能夠抓取 feature 提高 accuracy。然而這次卻是 task C 進行 test data 預測比較好，所以我認為這次的 test data 可能和用來 training 的 data 差異非常大，甚至有可能是因為 test data 根本就標錯了之類的原因。

而用比較複雜的 model 得出來的只有在使用 task C 的時候，進行 test 的 accuracy 才有明顯的比較高，然而在 training 時兩的 model 最後的 accuracy 其實是差不多的。

我認為複雜的 model 在多數參數被固定的情況下，因為比較複雜，所以處理的能力還是比同樣被固定的簡單 model 好的。所以這次因為 test data 的難度比較高，才會出現在 task C 的時候，使用複雜 model 的 accuracy 突然比簡單的還來的高。

也就是綜合來說，我猜測這次 performance on the test dataset 不好的原因可能在於 1.test data 本身的難度偏高(搞不好是照片糊糊的或是細節很不清楚)，並且再加上 2. test data 可能和用來 training 的 data 差異非常大，才造成 performance 不好。