Deep Learning Assignment 4

Tags

https://prod-files-secure.s3.us-west-2.amazonaws.com/5c1bf980-0beb-4832-a47d-5ec27d860e62/7ae599d0-6a55-432c-8ae5-0eb1d064e9e8/A4.pdf

Self-supervised and transfer learning on CIFAR10

Use CIFAR10 to train a model on self-supervised task, fine-tune subset of the model's weights, and train a model in fully supervised setting in different weight initializations.

Use PyTorch ResNet18 for implementation.

在這個 Task 中，所有的圖片會被隨機的旋轉0, 90, 180, 270度。

神經網路會被訓練為辨認種不同的旋轉角度。這個訓練的結果(pre-trained model)可以接著被使用在後續的 fine-tune。

> 在我的 local 端遇到奇怪的錯誤，當要將圖片unnormalize的時候發生運行不出結果來的情況。所以後來又回到Colab運行。

💡 **Trouble Shooting**: trainloader 裡面的 num_workers 需要設置為 0 才能在 local 端運行。

官网教程dataiter = iter(trainloader)报错的解决办法_nnloveswc的博客-CSDN博客

Section 1 — Rotation Task on ResNet18

使用 CIFAR10 作為訓練資料，使用 Rotation 的方式對資料進行處理，並使用 ResNet18作為Learner 進行建模，使用經過 Rotation 的資料訓練圖片分類模型。

> *CIFAR10 的資料共有 10 classes 的圖片。*

首先定義對資料的處理方式，也就是**旋轉(rotation)**的規則。

```
def rotate_img(img, rot):
    if rot == 0: # 0 degrees rotation
        return img
    elif rot == 1:
        return transforms.functional.rotate(img, 90)
    elif rot == 2:
        return transforms.functional.rotate(img, 180)
    elif rot == 3:
        return transforms.functional.rotate(img, 270)
    else:
        raise ValueError('rotation should be 0, 90, 180, or 270 degrees')
```

定義一個 function `rotate_img()` ，當輸入 img (圖片)與 rot(角度) 時，對 img 進行相對應角度的旋轉。

並定義一個 CIFAR10Rotation class，繼承 CIFAR10 dataset 的父類別，引用 `rotate_img()` 對資料進行操作，最後實現 CIFAR10Rotation model 的 implementation。

```
[1,   100] loss: 1.274 acc: 43.03 time: 6.48
[1,   200] loss: 1.128 acc: 51.05 time: 7.10
[1,   300] loss: 1.107 acc: 51.54 time: 7.11
TESTING:
Accuracy of the network on the 10000 test images: 56.42 %
Average loss on the 10000 test images: 1.024
[2,   100] loss: 1.054 acc: 54.64 time: 6.24
[2,   200] loss: 1.025 acc: 56.62 time: 6.69
[2,   300] loss: 1.015 acc: 56.87 time: 7.19
TESTING:
Accuracy of the network on the 10000 test images: 58.61 %
Average loss on the 10000 test images: 0.986
[3,   100] loss: 0.984 acc: 58.45 time: 6.68
[3,   200] loss: 0.961 acc: 59.71 time: 6.98
[3,   300] loss: 0.957 acc: 59.40 time: 7.20
TESTING:
Accuracy of the network on the 10000 test images: 62.27 %
Average loss on the 10000 test images: 0.901
[4,   100] loss: 0.936 acc: 61.32 time: 6.68
[4,   200] loss: 0.932 acc: 61.03 time: 7.06
[4,   300] loss: 0.917 acc: 61.82 time: 7.03
TESTING:
Accuracy of the network on the 10000 test images: 63.25 %
Average loss on the 10000 test images: 0.880
[5,   100] loss: 0.915 acc: 61.82 time: 6.71
[5,   200] loss: 0.892 acc: 63.08 time: 6.83
```

```
[5,   300] loss: 0.889 acc: 63.19 time: 7.23
TESTING:
Accuracy of the network on the 10000 test images: 64.96 %
Average loss on the 10000 test images: 0.862
...
[44,   100] loss: 0.537 acc: 79.22 time: 6.58
[44,   200] loss: 0.536 acc: 79.27 time: 6.79
[44,   300] loss: 0.535 acc: 79.17 time: 7.10
TESTING:
Accuracy of the network on the 10000 test images: 78.06 %
Average loss on the 10000 test images: 0.561
[45,   100] loss: 0.537 acc: 79.30 time: 6.94
[45,   200] loss: 0.533 acc: 79.28 time: 6.95
[45,   300] loss: 0.541 acc: 78.96 time: 7.04
TESTING:
Accuracy of the network on the 10000 test images: 77.60 %
Average loss on the 10000 test images: 0.559
Finished Training
```

> Accuracy from 56% → 77.60%,
> Loss from 1 → 0.55

Section 2 — Fine-tuning on pre-trained model

第二個部分使用 fine-tune 的技巧，使用 pretrained model 上進行建模。將凍結(freeze)第四層與全聯階層之前的所有層，只更新最後的兩層中的參數。

> 發現到使用 .pt 檔案儲存整個 model 使用上更方便，不用特別確認模型的整個架構是否與預定義的相同，直接使用(fine-tune)即可。

Fine-tuning 時發現使用 pre-trained weight 相較於 randomized weight 可以更快的提升對模型的準確度。

但是在有 freezing 的階段只有一開始的 accuracy 速度攀升非常快，到了acc > 50% 以後每個 epoch 提升有限。

```
[1,   100] loss: 1.639 acc: 40.77 time: 6.17
[1,   200] loss: 1.379 acc: 49.68 time: 6.56
[1,   300] loss: 1.305 acc: 52.58 time: 6.66
TESTING:
Accuracy of the network on the 10000 test images: 55.96 %
Average loss on the 10000 test images: 1.231
[2,   100] loss: 1.227 acc: 55.88 time: 6.49
[2,   200] loss: 1.222 acc: 55.89 time: 6.55
[2,   300] loss: 1.209 acc: 56.29 time: 6.75
TESTING:
Accuracy of the network on the 10000 test images: 57.65 %
```

```
Average loss on the 10000 test images: 1.176
[3,   100] loss: 1.181 acc: 57.16 time: 6.44
[3,   200] loss: 1.160 acc: 58.10 time: 6.63
[3,   300] loss: 1.149 acc: 57.88 time: 6.93
TESTING:
Accuracy of the network on the 10000 test images: 59.25 %
Average loss on the 10000 test images: 1.134
[4,   100] loss: 1.117 acc: 59.50 time: 6.44
[4,   200] loss: 1.145 acc: 58.77 time: 6.63
[4,   300] loss: 1.130 acc: 58.95 time: 6.86
TESTING:
Accuracy of the network on the 10000 test images: 60.26 %
Average loss on the 10000 test images: 1.117
[5,   100] loss: 1.095 acc: 60.20 time: 6.62
[5,   200] loss: 1.097 acc: 60.33 time: 6.31
[5,   300] loss: 1.106 acc: 59.91 time: 6.45
TESTING:
Accuracy of the network on the 10000 test images: 61.19 %
Average loss on the 10000 test images: 1.096
...
[19,   100] loss: 0.913 acc: 67.24 time: 6.44
[19,   200] loss: 0.940 acc: 66.19 time: 6.35
[19,   300] loss: 0.939 acc: 66.67 time: 7.09
TESTING:
Accuracy of the network on the 10000 test images: 64.75 %
Average loss on the 10000 test images: 1.001
[20,   100] loss: 0.920 acc: 66.70 time: 6.47
[20,   200] loss: 0.928 acc: 66.96 time: 6.38
[20,   300] loss: 0.931 acc: 66.42 time: 6.99
TESTING:
Accuracy of the network on the 10000 test images: 64.07 %
Average loss on the 10000 test images: 1.023
Finished Training
```

```
[1,   100] loss: 2.037 acc: 26.94 time: 6.40
[1,   200] loss: 1.876 acc: 31.34 time: 6.30
[1,   300] loss: 1.840 acc: 33.59 time: 6.85
TESTING:
Accuracy of the network on the 10000 test images: 37.07 %
Average loss on the 10000 test images: 1.735
[2,   100] loss: 1.795 acc: 35.33 time: 6.57
[2,   200] loss: 1.776 acc: 35.18 time: 6.39
[2,   300] loss: 1.759 acc: 35.66 time: 7.01
TESTING:
Accuracy of the network on the 10000 test images: 40.29 %
Average loss on the 10000 test images: 1.665
[3,   100] loss: 1.733 acc: 37.01 time: 6.58
[3,   200] loss: 1.727 acc: 37.68 time: 6.26
[3,   300] loss: 1.738 acc: 36.58 time: 7.00
TESTING:
Accuracy of the network on the 10000 test images: 40.64 %
Average loss on the 10000 test images: 1.664
[4,   100] loss: 1.694 acc: 38.81 time: 6.65
```

```
[4,   200] loss: 1.725 acc: 37.46 time: 6.30
[4,   300] loss: 1.708 acc: 37.58 time: 6.95
TESTING:
Accuracy of the network on the 10000 test images: 40.54 %
Average loss on the 10000 test images: 1.643
[5,   100] loss: 1.689 acc: 38.89 time: 6.67
[5,   200] loss: 1.684 acc: 39.30 time: 6.26
[5,   300] loss: 1.685 acc: 39.07 time: 6.98
TESTING:
Accuracy of the network on the 10000 test images: 41.47 %
Average loss on the 10000 test images: 1.624
...
[18,   100] loss: 1.546 acc: 44.64 time: 6.56
[18,   200] loss: 1.540 acc: 45.39 time: 6.40
[18,   300] loss: 1.517 acc: 45.75 time: 7.15
TESTING:
Accuracy of the network on the 10000 test images: 45.72 %
Average loss on the 10000 test images: 1.532
[19,   100] loss: 1.539 acc: 44.33 time: 6.53
[19,   200] loss: 1.517 acc: 45.23 time: 6.46
[19,   300] loss: 1.532 acc: 45.12 time: 7.00
TESTING:
Accuracy of the network on the 10000 test images: 45.99 %
Average loss on the 10000 test images: 1.527
[20,   100] loss: 1.518 acc: 45.39 time: 6.62
[20,   200] loss: 1.524 acc: 44.91 time: 6.50
[20,   300] loss: 1.528 acc: 45.55 time: 7.03
TESTING:
Accuracy of the network on the 10000 test images: 45.89 %
Average loss on the 10000 test images: 1.525
Finished Training
```

> 訓練過程調整 learning rate 參數觀察 acc 的震盪幅度協助找到最佳解。
> lr = 0.01 → 0.001 → 0.0001

觀察結果

Using Pre-trained weight:

> Accuracy: 55.96% → 64.07%, (差距9%)
> Loss: 1.231 → 1.023

Using Randomized weight:

> Accuracy: 37.07% → 45.89%, (差距8%)
> Loss: 1.735 → 1.525

明顯可以發現到使用了 Pre-trained weight 的模型表現相較於使用 Random weight 的模型更好，光是初始的 Accuracy 就比 Random 高了將近 20%！

再者而目前僅僅讓 layer4 與 fc layer 權重能夠更新，對於是否經過pre-trained的模型表現的差異化更為明顯。

Section 3 — Re-Training model on Pre-Trained model

最後使用前面 rotation-task 訓練的 pre-trained model *重新訓練整個classification model*，並比較使用預訓練模型與隨機全種模型的表現差異。

```
[1,   100] loss: 1.576 acc: 41.07 time: 6.34
[1,   200] loss: 1.212 acc: 56.27 time: 6.72
[1,   300] loss: 1.101 acc: 60.99 time: 6.86
TESTING:
Accuracy of the network on the 10000 test images: 63.65 %
Average loss on the 10000 test images: 1.034
[2,   100] loss: 1.000 acc: 64.34 time: 6.43
[2,   200] loss: 0.931 acc: 67.43 time: 6.71
[2,   300] loss: 0.902 acc: 68.11 time: 6.65
TESTING:
Accuracy of the network on the 10000 test images: 67.53 %
Average loss on the 10000 test images: 0.931
[3,   100] loss: 0.855 acc: 70.34 time: 6.47
[3,   200] loss: 0.845 acc: 70.48 time: 6.51
[3,   300] loss: 0.826 acc: 71.20 time: 7.09
TESTING:
Accuracy of the network on the 10000 test images: 72.21 %
Average loss on the 10000 test images: 0.794
[4,   100] loss: 0.767 acc: 73.42 time: 6.73
[4,   200] loss: 0.757 acc: 73.59 time: 6.83
[4,   300] loss: 0.755 acc: 73.61 time: 6.93
TESTING:
Accuracy of the network on the 10000 test images: 73.58 %
Average loss on the 10000 test images: 0.769
[5,   100] loss: 0.715 acc: 75.17 time: 6.50
[5,   200] loss: 0.707 acc: 74.99 time: 6.56
[5,   300] loss: 0.706 acc: 75.39 time: 6.65
TESTING:
Accuracy of the network on the 10000 test images: 74.12 %
Average loss on the 10000 test images: 0.761
...
[19,   100] loss: 0.372 acc: 86.88 time: 6.55
[19,   200] loss: 0.362 acc: 87.55 time: 6.72
[19,   300] loss: 0.371 acc: 86.81 time: 7.03
TESTING:
Accuracy of the network on the 10000 test images: 81.28 %
Average loss on the 10000 test images: 0.553
[20,   100] loss: 0.359 acc: 87.73 time: 6.57
[20,   200] loss: 0.356 acc: 87.62 time: 6.72
[20,   300] loss: 0.365 acc: 87.61 time: 7.09
TESTING:
```

```
Accuracy of the network on the 10000 test images: 81.15 %
Average loss on the 10000 test images: 0.550
Finished Training
```

```
[1,   100] loss: 2.320 acc: 19.63 time: 6.68
[1,   200] loss: 1.896 acc: 31.17 time: 6.72
[1,   300] loss: 1.760 acc: 34.65 time: 6.95
TESTING:
Accuracy of the network on the 10000 test images: 43.24 %
Average loss on the 10000 test images: 1.541
[2,   100] loss: 1.584 acc: 41.49 time: 6.94
[2,   200] loss: 1.499 acc: 45.49 time: 6.79
[2,   300] loss: 1.452 acc: 46.80 time: 6.28
TESTING:
Accuracy of the network on the 10000 test images: 49.02 %
Average loss on the 10000 test images: 1.373
[3,   100] loss: 1.319 acc: 51.61 time: 6.57
[3,   200] loss: 1.263 acc: 54.35 time: 6.76
[3,   300] loss: 1.207 acc: 56.52 time: 6.47
TESTING:
Accuracy of the network on the 10000 test images: 61.76 %
Average loss on the 10000 test images: 1.088
[4,   100] loss: 1.122 acc: 59.72 time: 6.67
[4,   200] loss: 1.080 acc: 61.77 time: 6.76
[4,   300] loss: 1.052 acc: 62.30 time: 6.36
TESTING:
Accuracy of the network on the 10000 test images: 64.74 %
Average loss on the 10000 test images: 1.003
[5,   100] loss: 1.007 acc: 64.51 time: 6.53
[5,   200] loss: 0.968 acc: 65.34 time: 6.80
[5,   300] loss: 0.967 acc: 66.00 time: 6.15
TESTING:
Accuracy of the network on the 10000 test images: 68.85 %
Average loss on the 10000 test images: 0.912
...
[19,   100] loss: 0.444 acc: 84.50 time: 6.66
[19,   200] loss: 0.451 acc: 84.32 time: 6.89
[19,   300] loss: 0.449 acc: 84.30 time: 7.16
TESTING:
Accuracy of the network on the 10000 test images: 79.99 %
Average loss on the 10000 test images: 0.600
[20,   100] loss: 0.439 acc: 84.63 time: 6.52
[20,   200] loss: 0.443 acc: 84.51 time: 6.77
[20,   300] loss: 0.445 acc: 84.52 time: 7.02
TESTING:
Accuracy of the network on the 10000 test images: 80.51 %
Average loss on the 10000 test images: 0.592
Finished Training
```

觀察結果

Using Pre-trained weight:

> Accuracy from 63% → 81%,
> Loss from 1.0 → 0.5

Using Random weight:

> Accuracy from 43% → 80%,
> Loss from 1.5 → 0.59

根據觀察，使用pre-trained model 再繼續做 re-training 的動作能夠大幅提高**初始的 Accuracy** 數值(63% vs 43%)，Randomly weighted 的 model 約在第四個 epoch 之後才慢慢接近 Pre-train weighted model。

結論

兩個訓練方式**最後結果差距並不大(81% vs 80%)**，反倒是繼續訓練下去說不定 random 的 accuracy 會更高。

不過使用Pre-trained model 進行訓練的優勢在於可以**省下更多的運算資源**，並可以在相似的工作重複利用曾經訓練好的模型提取訓練過的特徵(rotation task)，提供新訓練模型基準。