Lab 3-1 Report

Introduction:

這次 Lab 的目的在於讓我們熟悉各種加速深度學習訓練過程的方法，首先第一部分是學會如何實作 early stopping，接著了解使用 pretrained model 的好處以及效果。

Experiment setup:

我是在 win10 下使用 Anaconda 的 jupyter notebook 進行實驗，顯卡是 GTX1070。

Result:

(a)

我們預設 training 的 epoch 是 25，不算很多，因此我們將 patient 設成 5，另外設一個 flag 來判斷當超過 5 個 epoch 沒有得到更好的參數時即跳脫迴圈。

```
if phase == 'validation':
    loss_values.append(epoch_loss)
    acc_values.append(epoch_acc)
    # deep copy the model
    if epoch_acc > best_acc:
        best_acc = epoch_acc
        best_model_wts = copy.deepcopy(model.state_dict())
        j = 0
    else:
        j += 1
        if j == patient:
            end = 1
```

在 validation 的時候，只要得到更好的參數就將模型儲存起來，並將

j 設回 0，沒有就將 j 加 1，只要 5 次沒有更新參數就把 end 設成 1，

之後就會跳出迴圈。

原先的結果如下:

```
Epoch 23/24
----------
skewed_training Loss: 0.5539 Acc: 0.8317
validation Loss: 0.4843 Acc: 0.8583

Epoch 24/24
----------
skewed_training Loss: 0.5339 Acc: 0.8337
validation Loss: 0.5005 Acc: 0.8569

Training complete in 38m 20s
Best val Acc: 0.861224
Test set: Top1 Accuracy: 2934/3347 (87 %) , Top3 Accuracy: 3263/3347 (97 %)
class 0 : 276/368 75 %
class 1 : 114/148 77 %
class 10 : 206/231 89 %
class 2 : 440/500 88 %
class 3 : 285/335 85 %
class 4 : 258/287 89 %
class 5 : 379/432 87 %
class 6 : 143/147 97 %
class 7 : 93/96 96 %
class 8 : 267/303 88 %
class 9 : 473/500 94 %
```

使用 early stopping 之後:

```
Epoch 19/24
----------
skewed_training Loss: 0.5427 Acc: 0.8301
validation Loss: 0.5092 Acc: 0.8519

Epoch 20/24
----------
skewed_training Loss: 0.5580 Acc: 0.8297
validation Loss: 0.5187 Acc: 0.8507


Training complete in 28m 7s
Best val Acc: 0.852770
Test set: Top1 Accuracy: 2924/3347 (87 %) , Top3 Accuracy: 3256/3347 (97 %)
class 0 : 314/368 85 %
class 1 : 114/148 77 %
class 10 : 203/231 87 %
class 2 : 405/500 81 %
class 3 : 264/335 78 %
class 4 : 247/287 86 %
class 5 : 385/432 89 %
class 6 : 141/147 95 %
class 7 : 95/96 98 %
class 8 : 277/303 91 %
class 9 : 479/500 95 %
```

可以看到 epoch 減少到了 20 個，因為到這邊累計 5 次沒有更好的參數因此跳脫迴圈，而訓練時間減少了 10 分鐘左右。另外，準確率幾乎沒有任何影響，因為我們原本的 model 本來就會去紀錄最好的權重，並當作最後輸出的結果。

(b)

我們這部份使用的 CNN model 是 resnet18，使用資料是 skewed food11，首先這是尚未使用 pretrained model 的結果。

```
Test set: Top1 Accuracy: 1748/3347 (52 %) , Top3 Accuracy: 2683/3347 (80 %)
class 0 : 149/368 40 %
class 1 : 3/148  2 %
class 10 : 36/231 15 %
class 2 : 294/500 58 %
class 3 : 152/335 45 %
class 4 : 126/287 43 %
class 5 : 337/432 78 %
class 6 : 75/147 51 %
class 7 : 7/96  7 %
class 8 : 179/303 59 %
class 9 : 390/500 78 %
```

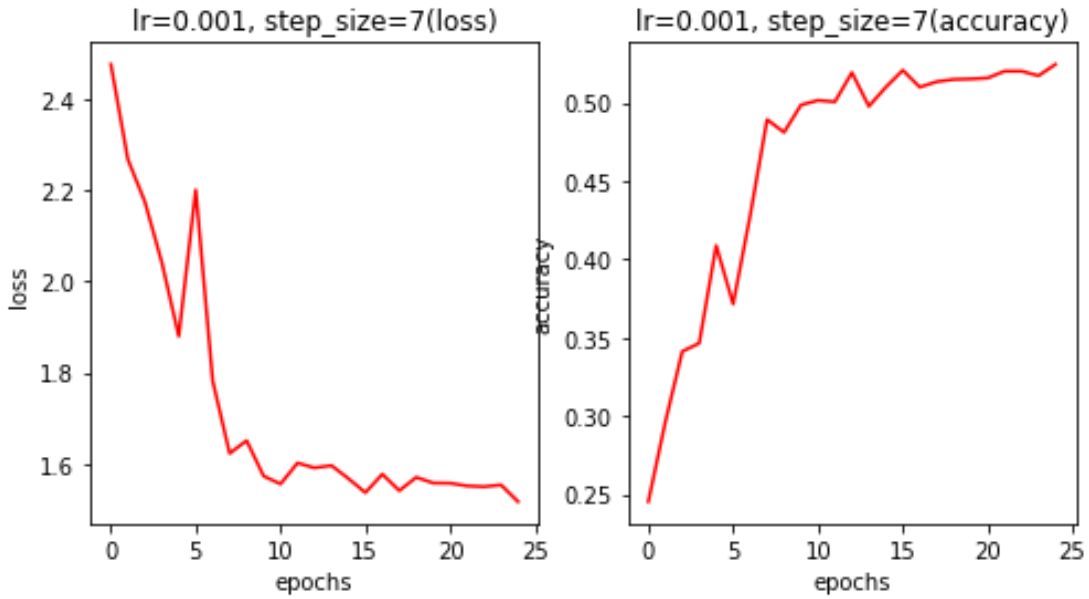
準確率很低，尤其是資料很少的類別，然後使用 pretrained model 的結果如下。

```
Test set: Top1 Accuracy: 2934/3347 (87 %) , Top3 Accuracy: 3263/3347 (97 %)
class 0 : 276/368 75 %
class 1 : 114/148 77 %
class 10 : 206/231 89 %
class 2 : 440/500 88 %
class 3 : 285/335 85 %
class 4 : 258/287 89 %
class 5 : 379/432 87 %
class 6 : 143/147 97 %
class 7 : 93/96 96 %
class 8 : 267/303 88 %
class 9 : 473/500 94 %
```

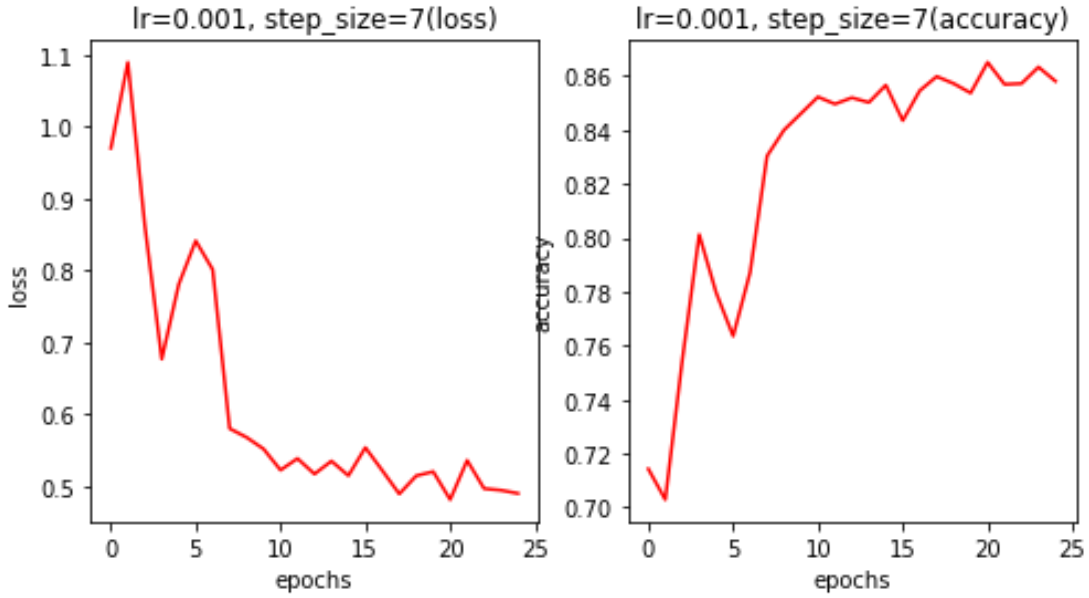
整體的結果有相當大的改善，平均準確率提升了 35%左右。

有關收斂速度的部份，下圖是 epoch 對 loss 以及 accuracy 的曲線

圖，尚未使用 pretrained model 的情況如下。



約莫在第 10 個 epochs 左右的時候收斂，而使用 pretrained model 的情況如下。



也是約莫在第 10 個 epochs 左右的時候開始收斂，因此我認為兩者收斂速度並無太大差異。

在調整 learning rate 的部份，通常在 transfer learning 當中，傾向於

使用較小的 learning rate，避免太快去破壞掉先前 model 已經 trained 好的權重，並防止失去某些資訊。在前面的實驗，我們使用的 learning rate 一開始是 0.01，每過 7 個 epoch 做一次下降的動作(乘以 0.1)。現在，我們降低一開始的 learning 到 0.005，得到的結果如下。

```
Test set: Top1 Accuracy: 2990/3347 (89 %) , Top3 Accuracy: 3280/3347 (97 %)
class 0 : 314/368 85 %
class 1 : 120/148 81 %
class 10 : 211/231 91 %
class 2 : 431/500 86 %
class 3 : 285/335 85 %
class 4 : 257/287 89 %
class 5 : 378/432 87 %
class 6 : 144/147 97 %
class 7 : 94/96 97 %
class 8 : 273/303 90 %
class 9 : 483/500 96 %
```

準確率有些微的提升，繼續下降到 0.0025，得到的結果如下。

```
Test set: Top1 Accuracy: 2978/3347 (88 %) , Top3 Accuracy: 3268/3347 (97 %)
class 0 : 298/368 80 %
class 1 : 118/148 79 %
class 10 : 206/231 89 %
class 2 : 438/500 87 %
class 3 : 276/335 82 %
class 4 : 254/287 88 %
class 5 : 396/432 91 %
class 6 : 140/147 95 %
class 7 : 94/96 97 %
class 8 : 281/303 92 %
class 9 : 477/500 95 %
```

並無明顯的提升情況。另外，我們測試改變 step_size(下降一次所經過 epochs 的數量)的結果，結果如下。

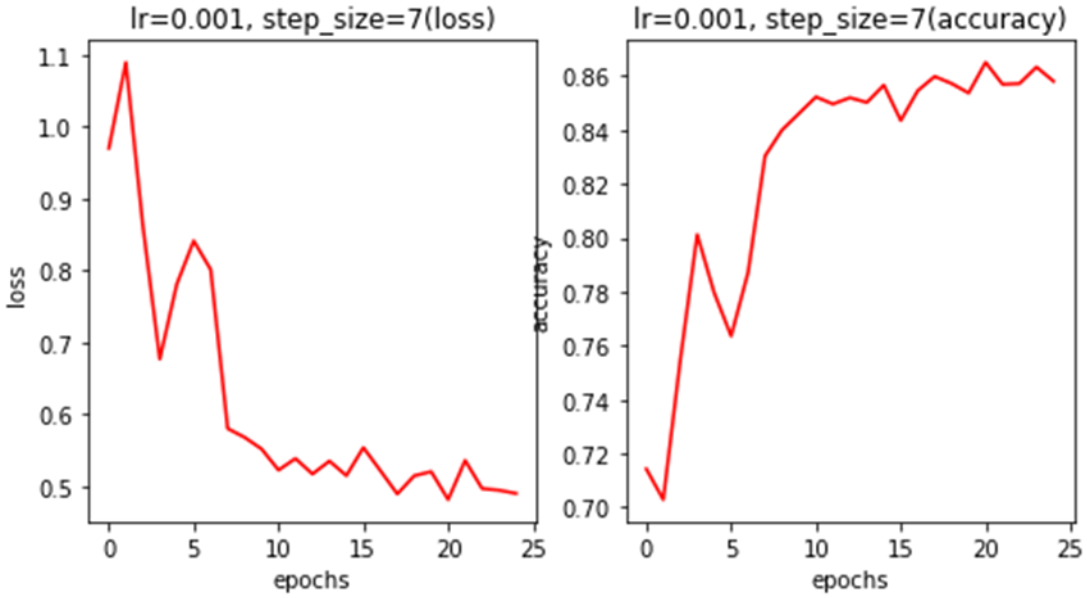
```
step_size = 1 :
Test set: Top1 Accuracy: 2295/3347 (68 %) , Top3 Accuracy: 2866/3347 (85 %)

step_size = 5 :
Test set: Top1 Accuracy: 2951/3347 (88 %) , Top3 Accuracy: 3256/3347 (97 %)

step_size = 14 :
Test set: Top1 Accuracy: 2948/3347 (88 %) , Top3 Accuracy: 3267/3347 (97 %)
```

當設成 1 的時候，準確率明顯的下降，另外以 7 為基準上升或下降都沒有明顯變化。總體來說，learning 一開始設成 0.005 左右，step_size 設成 7 左右的結果應該是最好的。

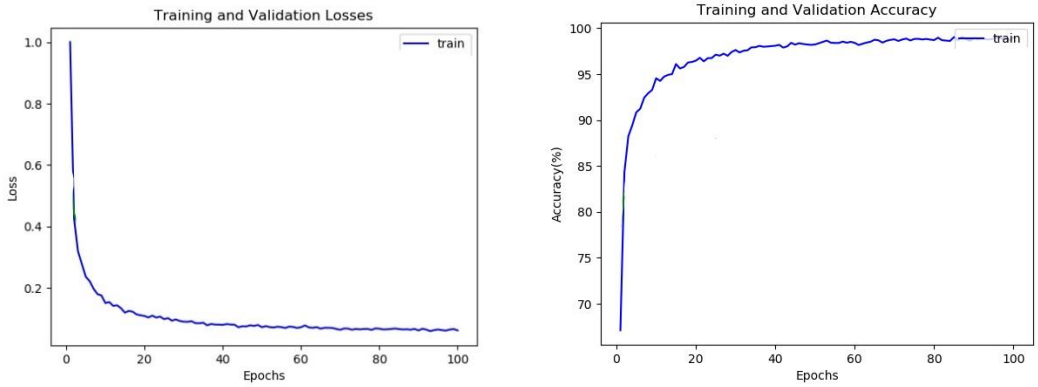
(C)

我選用 resnet50 作為 teacher model，並與 resnet18 的 student model 做比較。batch_size = 32, lr = 0.001

下圖為 resnet18 未經 Knowledge distillation 的訓練結果



下圖為經過 resnet50 Knowledge distillation 的訓練結果



可以發現在未經 Knowledge distillation 時，在 25 個 epochs 僅能收斂到 86%之 accuracy 及 0.5 的 loss，而經過 Knowledge distillation 之後，在 20 個 epochs 就達到了 95%以上的 accuracy 及 0.2 以下 loss 的優異表現

(d)

選擇的 tuner : TPE、Evolutionary


```
{
    "config":{
        "experimentName": "ResNet18(pre)_TPE",
        "maxExecDuration": "12h",
        "maxTrialNum": 15,
        "parentProject": "None",
        "model": "lab3_1d",
        "updatePeriod": 60,
        "tuner": {
            "builtinTunerName": "TPE",
            "classArgs": {"optimize_mode": "maximize"}
        }
    },

    "params": {
        "batch_size": 32,
        "lr": 0.001,
        "momentum": 0.9
    },

    "search_space": {
        "batch_size": {"_type": "choice", "_value": [8, 16, 32, 64, 128]},
        "lr": {"_type": "uniform", "_value": [0.1, 0.0001]},
        "momentum": {"_type": "choice", "_value": [0.1, 0.3, 0.5, 0.7, 0.9]}
    }
}
```

```
"config":{
    "experimentName": "ResNet18_Evolution_ES",
    "maxExecDuration": "12h",
    "maxTrialNum": 15,
    "parentProject": "None",
    "model": "lab3_1d",
    "updatePeriod": 60,
    "tuner": {
        "builtinTunerName": "Evolution",
        "classArgs": {"optimize_mode": "maximize", "population_size": 100}
    }
},

"params": {
    "batch_size": 32,
    "lr": 0.001,
    "momentum": 0.9
},

"search_space": {
    "batch_size": {"_type": "choice", "_value": [8, 16, 32, 64, 128]},
    "lr": {"_type": "uniform", "_value": [0.1, 0.0001]},
    "momentum": {"_type": "choice", "_value": [0.1, 0.3, 0.5, 0.7, 0.9]}
}
```