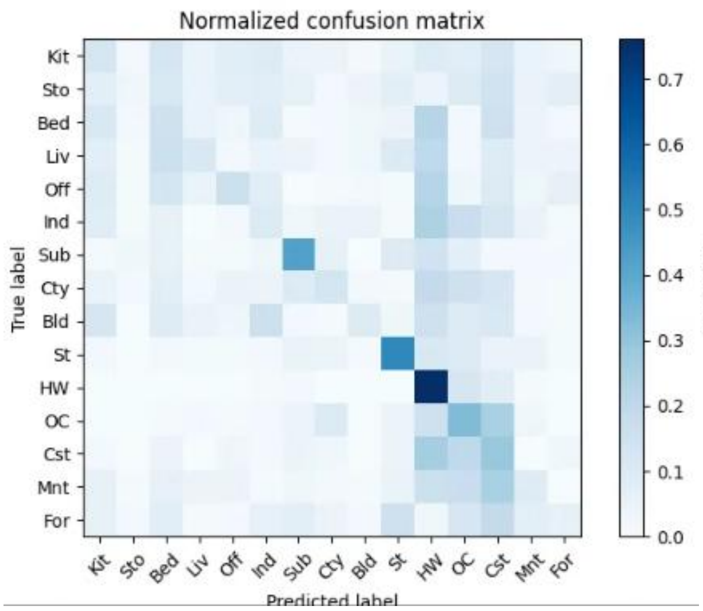
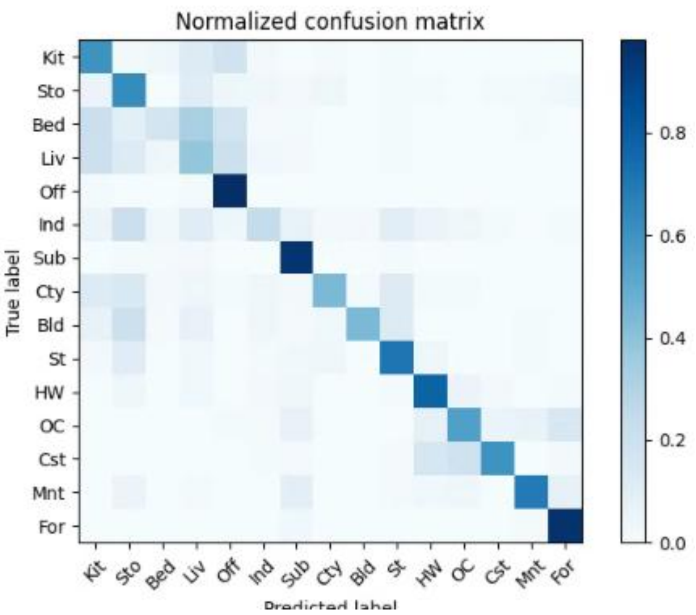
# Computer Vision HW2 Report

| |
|---|
| Student ID: r13945050 |
| Name: 張祐嘉 |

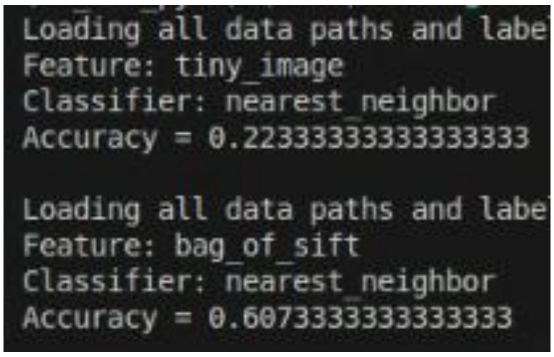
## Part 1. (10%)

**• Plot confusion matrix of two settings. (i.e. Bag of sift and tiny image) (5%)**
**Ans:**





**• Compare the results/accuracy of both settings and explain the result. (5%)**
**Ans:**



```
Loading all data paths and labe
Feature: tiny_image
Classifier: nearest_neighbor
Accuracy = 0.22333333333333333

Loading all data paths and labe
Feature: bag_of_sift
Classifier: nearest_neighbor
Accuracy = 0.6073333333333333
```

從 accuracy 可以明顯看出 Bag of sift 的準確度較高(0.607>0.223)，表示 sift 確實有抓出影像關鍵特徵，相比於 tiny image 只是將影像縮小並 flatten 成一維，某種程度已經損失了影像中二維空間的關聯性，而從 confusion matrix 中也可以明顯看出 Bag of sift 預測較準除了 bedroom, living room, indsidecity 分類較不好(推測可能為 indsidecity 很多照片僅包含建築物的一部分與 bedroom, living room 擷取的特徵可能很相近，導致不容易分類)，而 tiny image 只有在 street 的分類準確推測是街道線條型的特徵即便做 flatten 影響也不大

## Part 2. (25%)

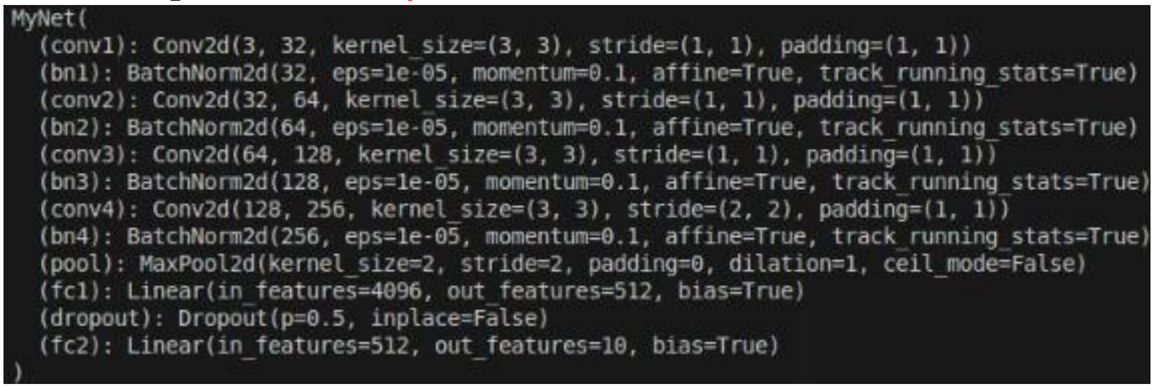
**• Report accuracy of both models on the validation set. (2%)**

**Ans:**

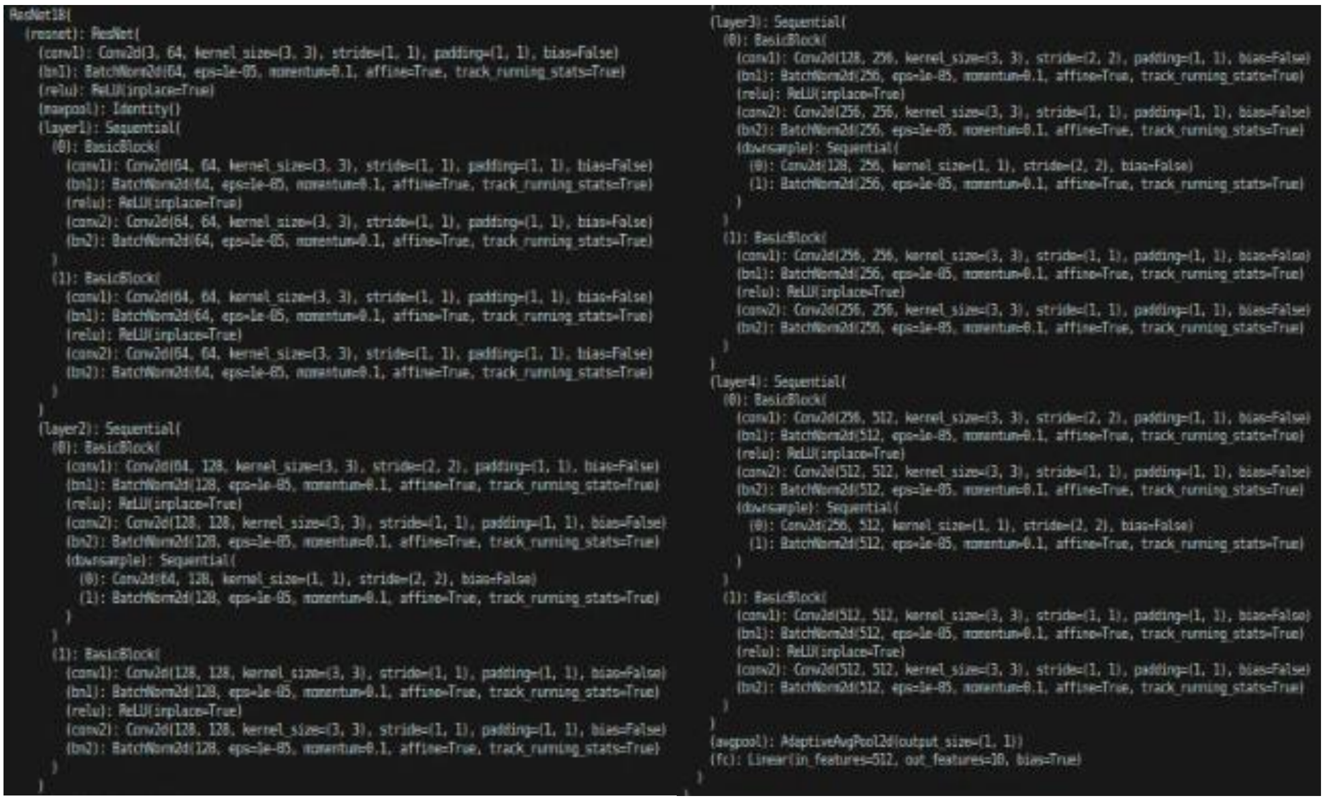
| | A | B |
|---|---|---|
| accuracy | 0.902 | 0.793 |

**• Print the network architecture & number of parameters of both models. What is the main difference between ResNet and other CNN architectures? (5%)**

**Ans: total parameters: mynet(2,492,170) ResNet18(11,173,962)**



```
MyNet(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv4): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (bn4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=4096, out_features=512, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=512, out_features=10, bias=True)
)
```



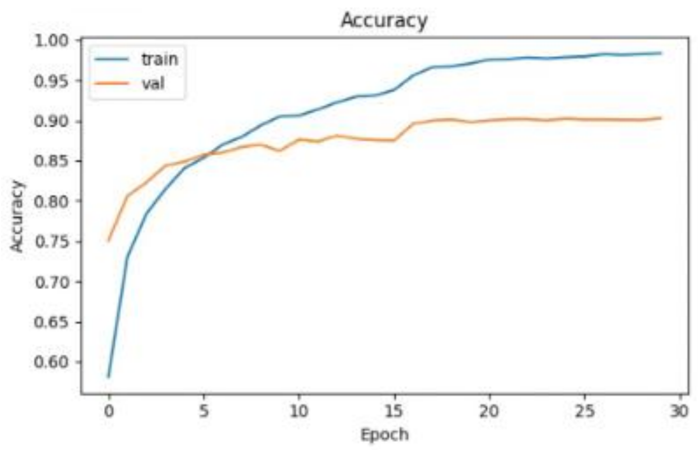
```
ResNet18(
  (resnet): ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): Identity()
    (layer1): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (layer2): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (layer3): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (layer4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

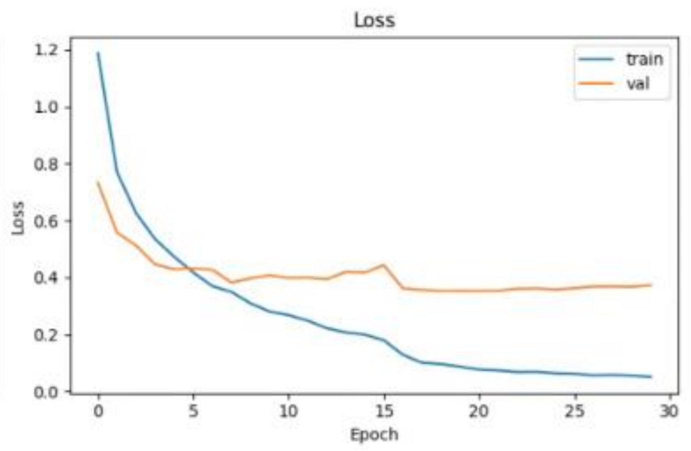
**以神經網路深度來說，Resnet18 的層數較多(18 層)，而 mynet 只用 4 層卷積與 2 層全連接層，而參數 Resnet18 比 mynet 多了近 5 倍，當然越多的參數理論上訓練效果越好，但也更會發生 overfitting 的狀況，也是我無法再往下增加層數的原因，而 Resnet18 有引入 Residual block(skip connection)的概念以緩解梯度消失的狀況，也可以有效限制深層模型 overfitting 的狀況，因此 Resnet18 performance 明顯較好**

**• Plot four learning curves (loss & accuracy) of the training process (train/validation) for both models. Total 8 plots. (8%)**

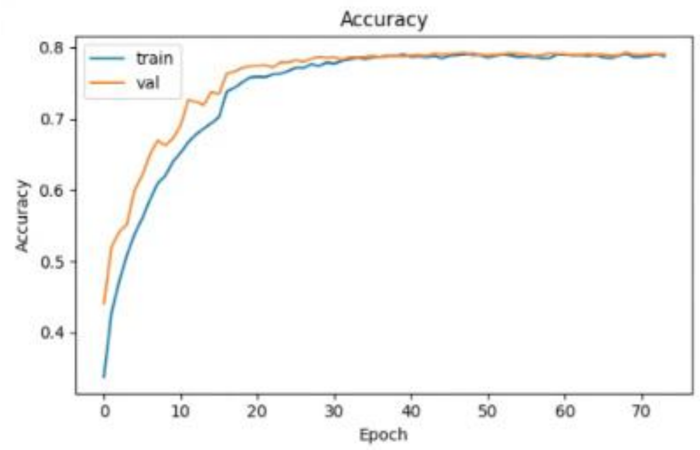
**Ans:**

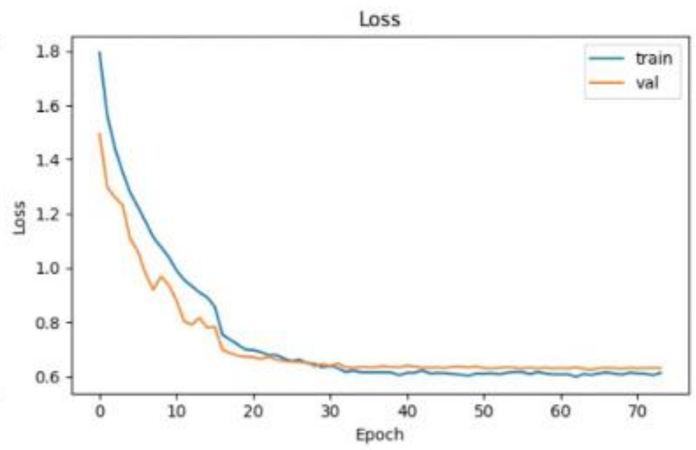
**Resnet18:**





**Mynet:**





因為我 early stop 的判斷是 10epoch 都沒 improve 才結束(容忍較大)，所以看起來是 resnet 的曲線看起來有一點 overfitting 的現象，但也可觀察到 pretrainweight 的作用(訓練初期準確度就很高)，因此也可能後面也多少會有 overfitting 但最可以後 reach 到約 90%的 val acc，(val 跟 train 都高)因此還算可以接受

**• Briefly describe what method do you apply on your best model? (e.g. data augmentation, model architecture, loss function, etc) (10%)**

**Ans:**

Data augmentation 主要使用水平翻轉, 隨機旋轉 15 度, 左右平移, 隨機放大區域，主要理由是觀察到驗證/測試圖像與訓練類似，不用做垂直翻轉或模糊化甚至顏色對比度的調整，反而甚至會使訓練準確度無法提升(卡在 89%左右)驗證集可想而知更低，至於訓練的過程有加入 unlabeled data 做半監督式學習，原本是每一個 epoch 都會抓一次 unlabeled data> threshold 的 data 出來丟入訓練但發現太耗時了，因此改為在 20-23 epoch 做挑選的動作，其餘都用一樣的 unlabeled data 做訓練，發現 validate acc 可能略低一些但影響並不大，而 threshold 的部分我則是根據 epoch 做動態調整(訓練初期 threshold 需比較高後期則可以逐漸將低)，以盡可能增加 training data，loss function 則沒有做太多調整一樣使用 cross entropy loss，也有加入 early stop 機制以偵測 overfitting 的狀況發生，至於超參數可以說幾乎是用 try and error 的方式 tune 出來