**Computer Vision HW2 Report**

Student ID: B09901075

Name: 陳駿瑋

**Part 1. (10%)**

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

**Ans:**

|  |  |
| --- | --- |
| **bag of sift** | **tiny image** |
|  |  |

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

|  |  |
| --- | --- |
| **Setting** | **Accuracy** |
| tiny image | 24.5% |
| bag of sift | 61.2% |

首先關於tiny image的部分，基本上就是把圖片resize - flatten - normalize，當成圖片feature。然後用cdist算test feature和train feature間的距離，最後用kNN決定結果。主要決定accuracy的因素是kNN中的k值和算距離使用的方法。cdist的部分，我試過hint中的minkowski distance (1-norm and 2-norm and 3-norm)，結果最佳的是minkowski distance (1-norm)， accuracy結果大概會落在0.21~0.25左右 (k = 7)。另外k值則是從1~11，從1開始accuracy會慢慢提升，但大概到8左右會慢慢開始下降，不過雖然有變化，也都大概是0.2 ~ 0.25左右。總結來說，在我測試結果中，此方法可以讓accuracy落在大概20%~25%，最後選擇的組合是minkowski distance (1-norm) 和 k = 7。

再來關於bag of sift的部分，首先使用dsift取得SIFT feature，然後用kmeans分群，得到vocab.pkl (像是圖片字典)。接下來取得預測結果的部分，一樣先取得圖片的SIFT feature，與vocab中的feature計算距離，統計出距離最近的feature，最後再使用kNN取得結果。在此方法中，我認為影響accuracy的因素主要有dsift中的step和與tiny image一樣，計算距離的方法和k值。(其實還有一個是在取SIFT feature時，可以不用全部使用，加快訓練速度，同時也不會使accuracy下降太多，但因為我選取的step花的時間還可以接受，所以我是直接使用全部feature) 而step我測試過2和3，step = [2,2]時的準確率較高，畢竟step越小，取feature越仔細，能提升accuracy還滿符合直覺，只是使用較小的step真的會大幅增加產出vocab.pkl的時間。另外k值的選取結果與tiny image類似，提高會讓accuracy上升，但太高的k可能會讓accuracy下降。(k = 1 和 k = 7時，accuracy大概差到3％) 最後是我認為最重要的distance計算方法的選擇，我測試了minkowski (1-norm and 2-norm)，其中在固定k = 7時，minkowski (1-norm)的accuracy達到0.612，是最佳結果，其他兩個會使accuracy降到0.51 (2-norm)和0.47 (3-norm)，差別非常大，而針對1-norm表現最好，我認為是因為SIFT feature是由histogram統計出來，所以計算距離時，也就使用1-norm不同dimension直接相減相加就好。

而針對tiny image和bag of sift的比較，畢竟tiny image只是單純resize圖片，沒有分析圖片，而bag of sift有分析圖片，取出sift feature，並且做分類，所以bag of sift比tiny image高不少算是可預期的結果。

**Part 2. (25%)**

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

**Ans:**

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Mynet (follow LeNet in lec06.pdf, p.98) | 63.82% |
| Resnet18 (with revised architecture) | 91.28% |

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

**Ans:**

相關的圖在文字下方。

Resnet跟一般的CNN主要差在使用residual/skip connections，從Resnet的source code可以發現，每個basic block的輸出都是output (input經過basic block中的Conv, BatchNorm, Relu後的結果) + identity (input 或是input經過downsample層) 後再經過Relu層。使用residual/skip connections是因為如果model太深，梯度往前傳時不斷減小，可能會導致梯度變零 (gradient vanish)，而透過residual/skip connections，可以解決此問題，也讓優化參數的過程更輕鬆。

|  |  |
| --- | --- |
| Resnet18 (with revised architecture) | Mynet (follow LeNet in lec06.pdf, p.98) |
|  |  |
|  |  |

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

**Ans:**

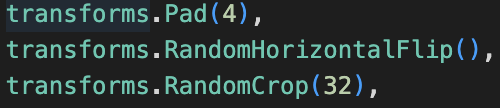
|  |  |  |
| --- | --- | --- |
|  | **Mynet** | **Resnet18** |
| **Train accuracy** |  |  |
| **Train loss** |  |  |
| **Valid accuracy** |  |  |
| **Valid loss** |  |  |

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

**Ans:**

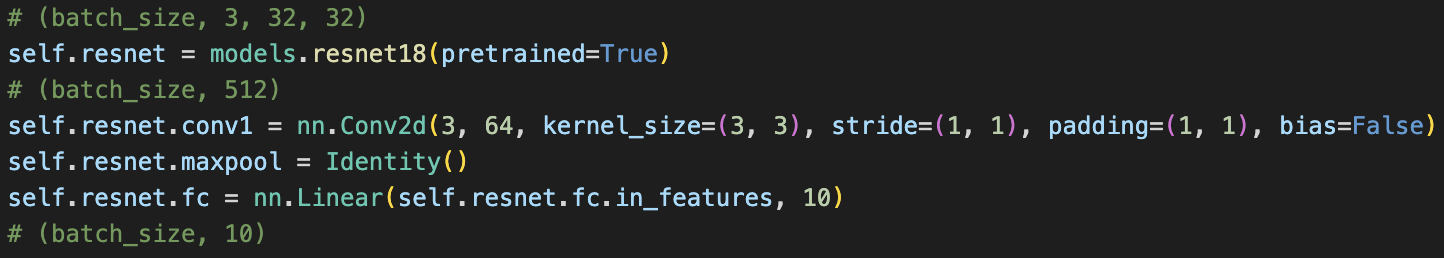
1. **Data augmentation**

原本有任意測試Rotation, ColorJitter, GrayScale等等，發現越多transform不一定較佳，因為transform種類很多，選取幾個也是問題，所以我查詢相關資料，最後選擇以下組合。

****

1. **Model architecture**

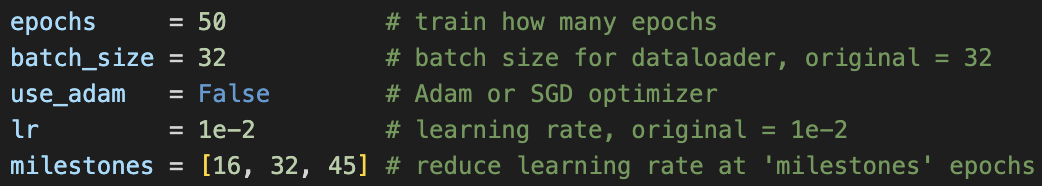
我最好的model達到0.9128的accuracy，是使用根據hint修改的resnet18，修改部分如下：



1. **Loss function**

optimizer = torch.optim.SGD(model.parameters(), lr=cfg.lr, momentum=0.9, weight\_decay=1e-6)

1. **Parameters**

****

雖然這裡epoch有到50，但實際train時，大概18就能達到9成的accuracy。