**Computer Vision HW2 Report**

Student ID: R11921103

Name: 張銘軒

**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:** 在Bag of SIFT 準確率為61.0667%，Tiny Image準確率則為20.8%，從準確率跟confusion matrix可以看出Tiny Image容易讓分類器無法辨認出圖片的關鍵，進而無法建立出不同類別圖片的雛形；而Bag of SIFT 則有先建立vocabulary，透過萃取圖片的關鍵部分訓練分類器做更準確的判斷，再對test image做判斷得出結果，準確率因此好很多。KNN的部分在比較過後使用k=5，cdist metric=cityblock獲得最好的結果。build\_vocabulary中我選擇dsift的step=[2, 2]，kmeans共分600類，get\_bags\_of\_sifts中我選擇dsift的step=[1, 1]。

**Part 2. (25%)**

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

**Ans:**

|  |  |
| --- | --- |
| MyNet Validation Set Accuracy | ResNet Validation Set Accuracy |
| 0.8498 | 0.8308 |

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

**Ans:**

|  |
| --- |
| MyNet(  (nnet): Sequential(  (0): Conv2d(3, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (2): ReLU(inplace=True)  (3): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)  (4): Conv2d(64, 128, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (6): ReLU(inplace=True)  (7): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)  (8): Conv2d(128, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (10): ReLU(inplace=True)  (11): Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  (12): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (13): ReLU(inplace=True)  (14): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)  (15): Conv2d(256, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  (16): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (17): ReLU(inplace=True)  (18): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  (19): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (20): ReLU(inplace=True)  (21): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)  (22): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  (23): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (24): ReLU(inplace=True)  (25): Conv2d(512, 512, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  (26): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (27): ReLU(inplace=True)  (28): MaxPool2d(kernel\_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)  (29): AvgPool2d(kernel\_size=1, stride=1, padding=0)  (30): Flatten(start\_dim=1, end\_dim=-1)  (31): Linear(in\_features=512, out\_features=10, bias=True)  )  ) |
| ResNet18(  (resnet): ResNet(  (conv1): Conv2d(3, 64, kernel\_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)  (relu): ReLU(inplace=True)  (maxpool): MaxPool2d(kernel\_size=3, stride=2, padding=1, dilation=1, ceil\_mode=False)  (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)  )  ) |

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

**Ans:**

|  |  |
| --- | --- |
| MyNet | |
| Train Accuracy | Train Loss |
|  |  |
| Validation Accuracy | Validation Loss |
|  |  |

|  |  |
| --- | --- |
| ResNet | |
| Train Accuracy | Train Loss |
|  |  |
| Validation Accuracy | Validation Loss |
|  |  |

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

**Ans:**

Data Augmentation: 我先移除sample code中的Resize((32, 32))，並以RandomCrop((32, 32), padding = 4)代替。接下來再對圖片做RandomHorizontalFlip(0.5)

Model: 我的架構為一直做Conv2d=>BatchNorm2d=>ReLU，有時候會接著做MaxPool2d，反覆數次之後會對結果做AvgPool2d，最後再Flatten=>Linear產出分類

Loss Function: 仍然使用CrossEntropyLoss並無更動

Others: 我將batch加大為64會有較好的學習效果