

Computer Vision HW2 Report

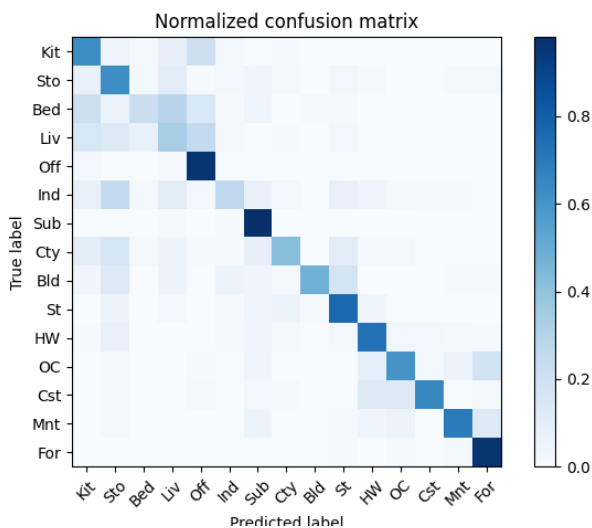
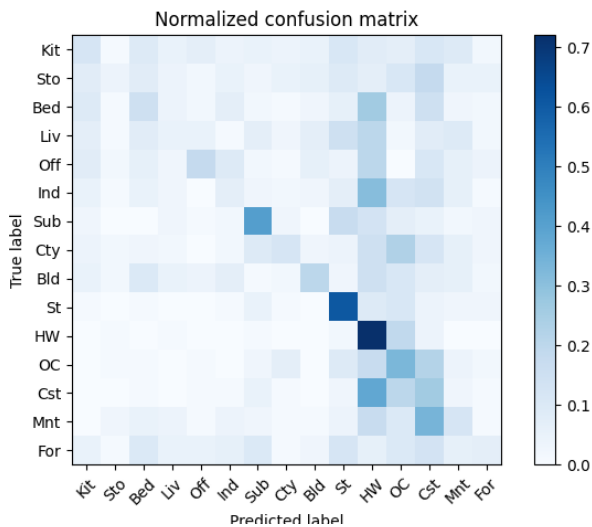
Student ID: R11921041

Name: 蔣沅均

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:

tiny image 的 accuracy 是 0.243，而 bag of sift 的是 0.622。兩個方法使用的 knn 分類器設定都一樣，因此可以比較出 bag of sift 相較於 tiny image 更能有效的抽取出利於場景分類的圖像特徵。

Part 2. (25%)

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

Ans:

	A - mynet	B - resnet18
accuracy	0.8182	0.8824

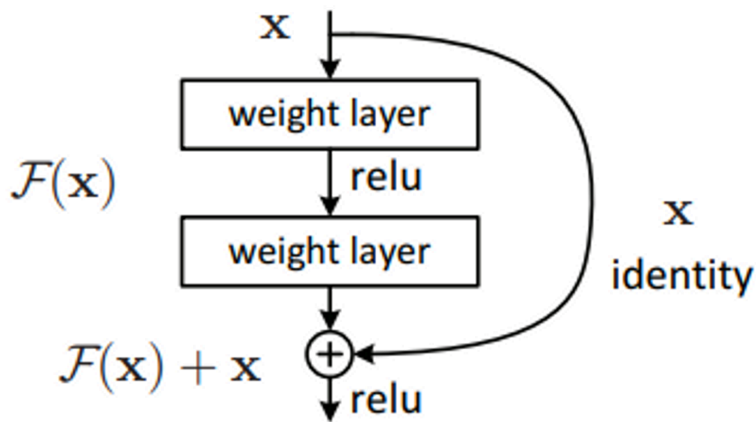
- Print the network architecture & number of parameters of both models. What is the main difference between ResNet and other CNN architectures? (5%)

Ans:

	A - mynet	B - resnet18
# of params	317,866	11,310,666
arch	<pre>MyNet((conv1): Conv2d(3, 32, kernel_size=(5, 5), stride=(2, 2), padding=(3, 3), bias=False) (bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (act1): GELU(approximate='none') (blk1): Sequential((0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): GELU(approximate='none') (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)) (blk2): Sequential((0): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False) (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): GELU(approximate='none') (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)) (down1): Sequential((0): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False) (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): GELU(approximate='none') (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)) (down2): Sequential((0): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False) (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (2): GELU(approximate='none') (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)) (avgpool): AdaptiveAvgPool2d(output_size=(1, 1)) (fc): Sequential((0): Linear(in_features=256, out_features=10, bias=True)))</pre>	<pre>1 ResNet18(2 (resnet): ResNet(3 (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False) 4 (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 5 (relu): ReLU(inplace=True) 6 (maxpool): Identity() 7 (layer1): Sequential(8 (0): BasicBlock(9 (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 10 (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 11 (relu): ReLU(inplace=True) 12 (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 13 (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 14) 15 (1): BasicBlock(16 (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 17 (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 18 (relu): ReLU(inplace=True) 19 (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 20 (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 21) 22) 23 (layer2): Sequential(24 (0): BasicBlock(25 (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False) 26 (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 27 (relu): ReLU(inplace=True) 28 (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 29 (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 30) 31 (1): BasicBlock(32 (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 33 (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 34 (relu): ReLU(inplace=True) 35 (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 36 (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 37) 38) 39 (layer3): Sequential(40 (0): BasicBlock(41 (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False) 42 (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 43 (relu): ReLU(inplace=True) 44 (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 45 (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 46) 47 (1): BasicBlock(48 (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 49 (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 50 (relu): ReLU(inplace=True) 51 (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 52 (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 53) 54) 55 (layer4): Sequential(56 (0): BasicBlock(57 (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False) 58 (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 59 (relu): ReLU(inplace=True) 60 (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 61 (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 62) 63 (1): BasicBlock(64 (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 65 (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 66 (relu): ReLU(inplace=True) 67 (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False) 68 (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 69) 70) 71 (avgpool): AdaptiveAvgPool2d(output_size=(1, 1)) 72 (fc): Sequential(73 (0): Linear(in_features=512, out_features=256, bias=False) 74 (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) 75 (2): GELU(approximate='none') 76 (3): Dropout(p=0.4, inplace=False) 77 (4): Linear(in_features=256, out_features=10, bias=True) 78) 79) 80)</pre>

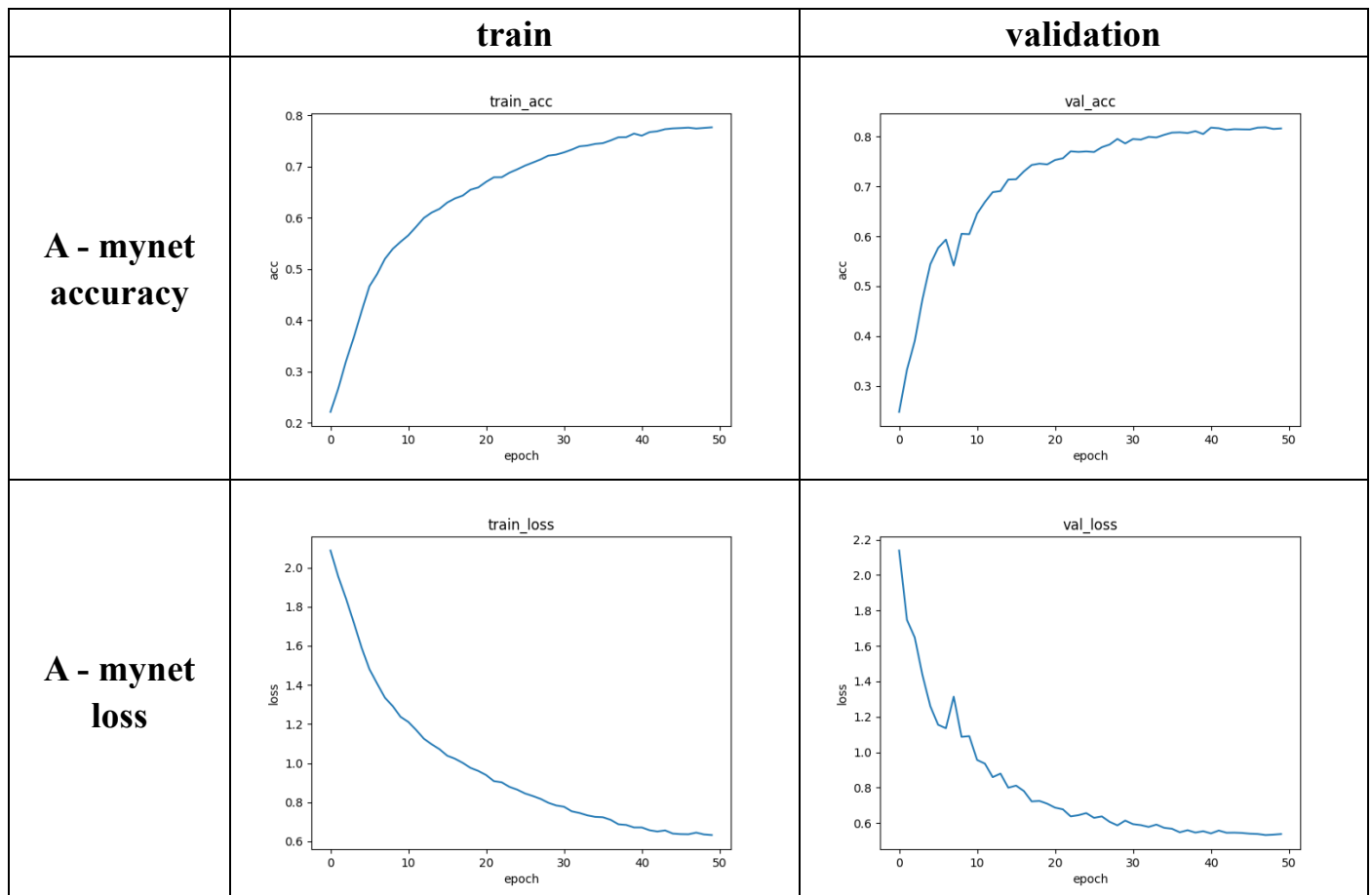
ResNet use residual connection.

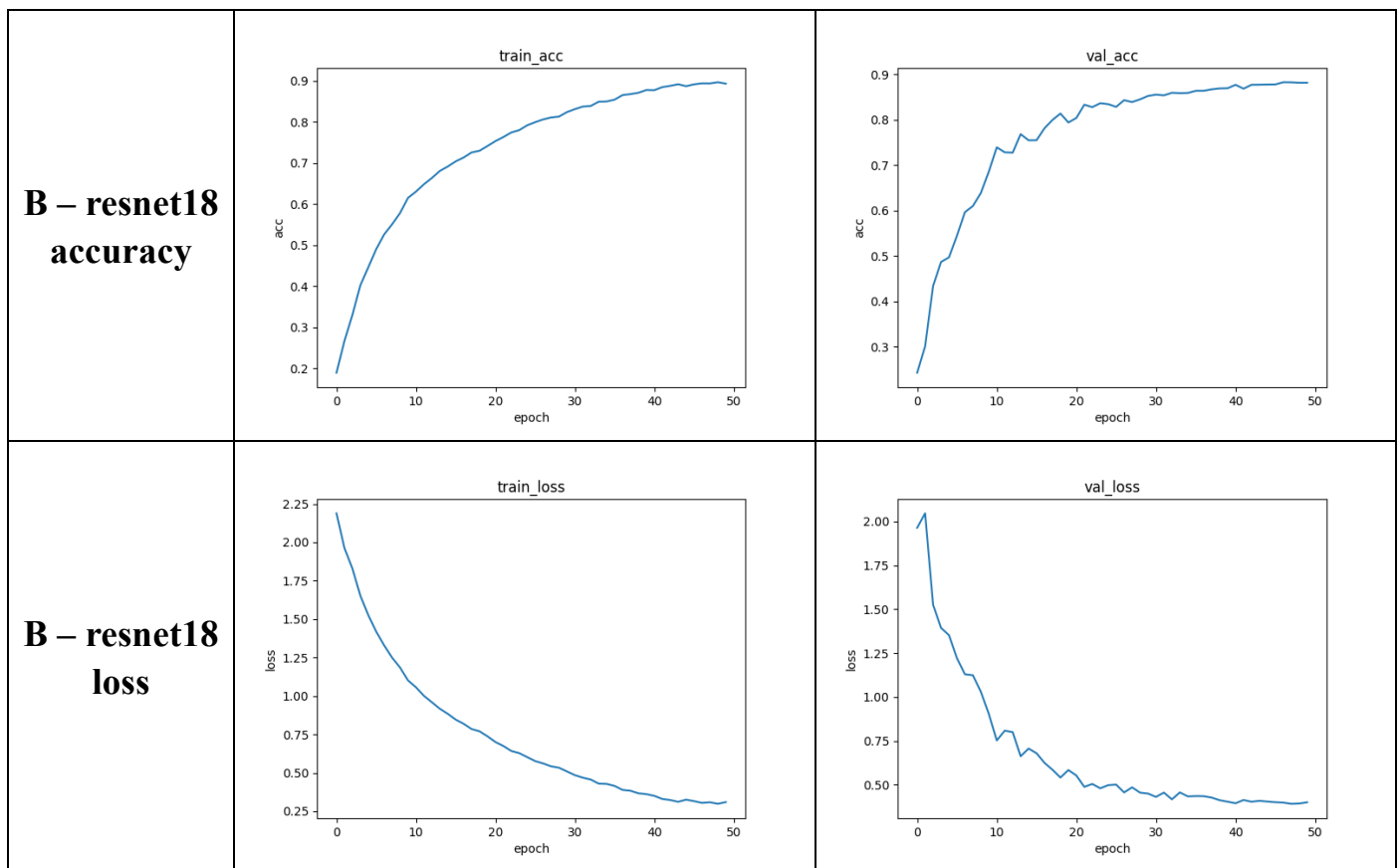
It should be easier for the solver to find the perturbations($F(x)$) with reference to an identity mapping(x) than to learn the function as a new one.



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

Ans:





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

Ans:

- data augmentation
 - Random Crop
 - TrivialAugment (<https://arxiv.org/abs/2103.10158>)
 - RandomErasing(<https://arxiv.org/abs/1708.04896>)
- model architecture
 - Based on Resnet18
 - Remove the first maxpool layer
 - Modify the fc module
- loss function
 - cross entropy