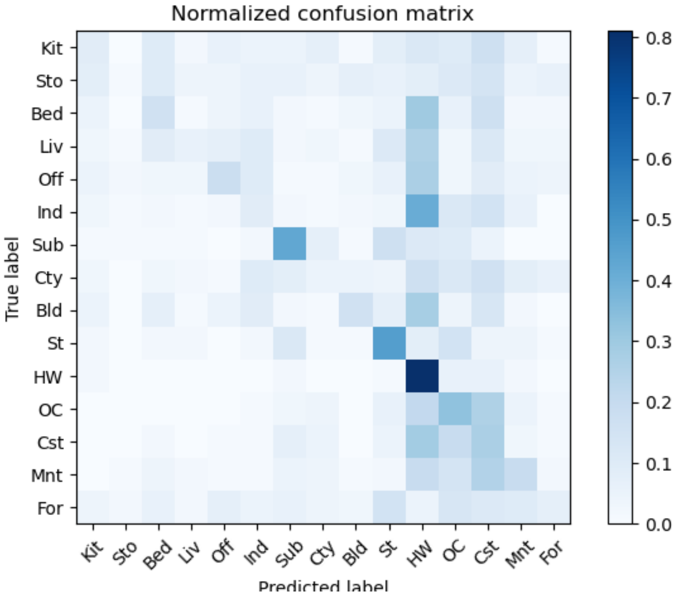
Computer Vision HW2 Report

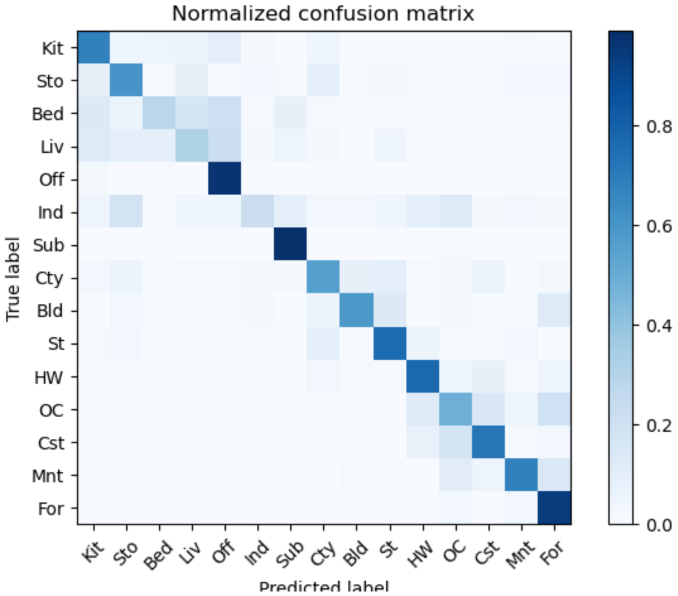
Student ID: R11921100
Name: 溫威領

Part 1. (10%)

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



-Tiny image-



-Bag of sift-

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

➢ Settings

Tiny image:

忽略圖片比例縮放至 16*16，然後 normalized 到 mean=0, std=1，再調整為 unit length。

Bag of sift:

dsift_step=[5, 5]，vocab_size=600，normalized histogram。

K-Nearest-Neighbor:

K=9，cdist_metric=”minkowski, p=0.5”

➢ Results

Predict Accuracy:

Bag of sift=0.224, Tiny image=0.6380

➢ Explain

Tiny image 是將圖片縮小作為特徵，而 Bag of sift 是將圖片取 sift 作為特徵，接著對要分類的圖片尋找相近的群集進行分類。Bag of sift 的 confusion matrix 對角線顏色較深代表分類正確率較高，而 Tiny image 的 confusion matrix 顏色分散代表分類正確率較低，由上述實驗結果可以觀察到 Bag of sift 的效果較好。

Part 2. (25%)

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

➢ **Mynet:** 0.5842

➢ **ResNet18:** 0.8908

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

➢ **Mynet:**

Network architecture:

```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
  (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): ReLU(inplace=True)
  (4): Dropout(p=0.5, inplace=False)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
  (6): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (8): ReLU(inplace=True)
  (9): Dropout(p=0.5, inplace=False)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
  (11): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (12): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (13): ReLU(inplace=True)
  (14): Dropout(p=0.5, inplace=False)
  (15): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
  (16): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (17): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (18): ReLU(inplace=True)
  (19): Dropout(p=0.5, inplace=False)
  (20): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=same, bias=False)
  (21): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (22): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (23): ReLU(inplace=True)
  (24): Dropout(p=0.5, inplace=False)
  (25): Flatten(start_dim=1, end_dim=-1)
  (26): Linear(in_features=1024, out_features=512, bias=True)
  (27): ReLU(inplace=True)
  (28): Dropout(p=0.5, inplace=False)
  (29): Linear(in_features=512, out_features=10, bias=True)
```

) **Number of parameters:** 1270218

➢ ResNet18:

Network architecture:

```
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)
```

) **Number of parameters:** 11181642

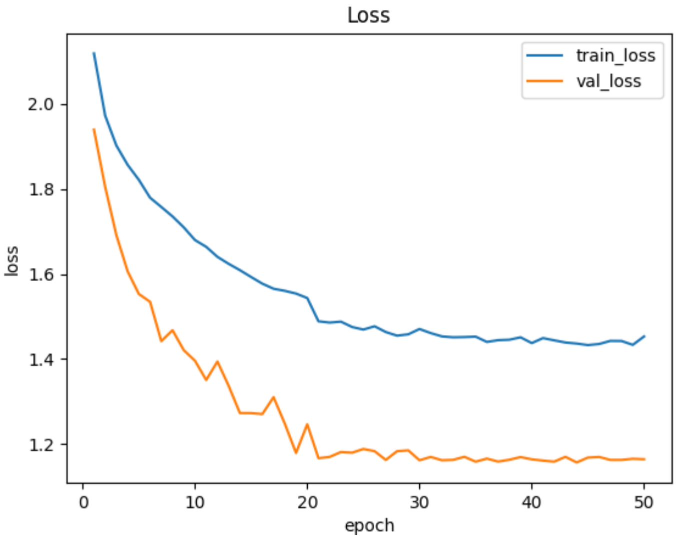
➢ Main difference:

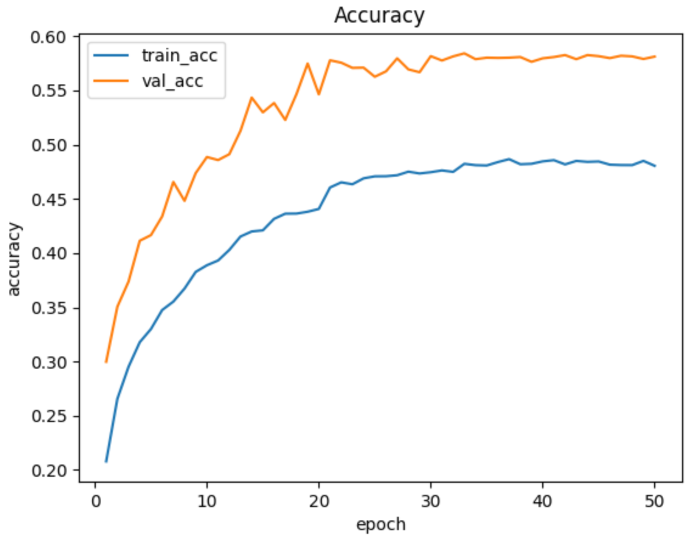
mynet 是自訂的 CNN model，使用 Conv2d、BatchNorm2d、MaxPool2d、ReLU、Dropout 為一個 Block 共重複五層 Block，最後 flatten 並使用二層 fully connect layer 進行分類。

ResNet18 則是使用定義好且預訓練的 model，但因為 cifar10 只有 10 類，所以在最後的 fully connect layer 調整為輸出 10 類。

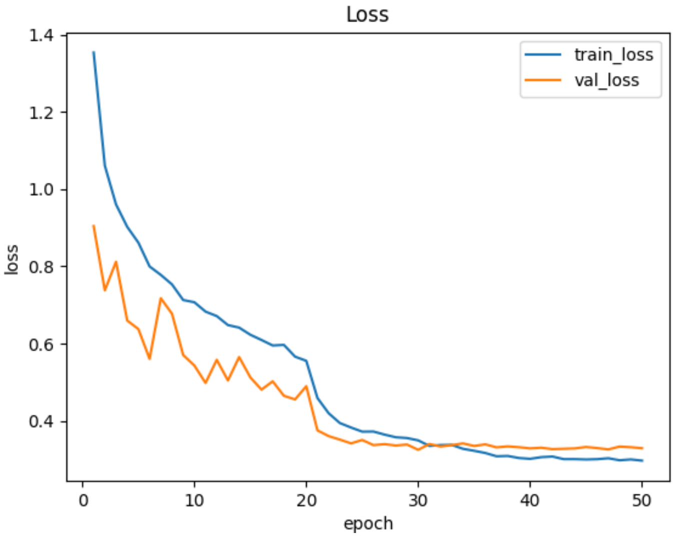
• Plot four learning curves (loss & accuracy) of the training process (train/validation) for both models. Total 8 plots. (8%)

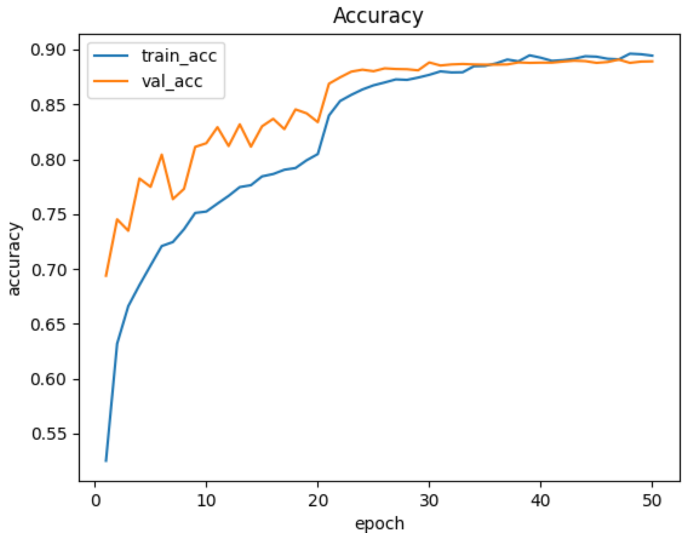
➢ Mynet:





➢ ResNet18:





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

- **Data augmentation:**

transforms.RandomGrayscale(0.1)

transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2)

transforms.RandomHorizontalFlip(p=0.5)

transforms.RandomRotation((-30, 30))

transforms.RandomResizedCrop((128, 128), scale=(0.3, 1))

- **Model architecture:** ResNet18 with pretrained weights
- **Loss function:** CrossEntropyLoss
- **Scheduler:** lr_scheduler.MultiStepLR, milestones=[20,35,45], gamma=0.1
- **Optimizer:** Adam with learning rate=1e-3
- **Batch size:** 128
- **Epochs:** 50