Computer Vision HW2 Report

Student ID: R11521608
Name: 李俊昇

Part 1. (10%)

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

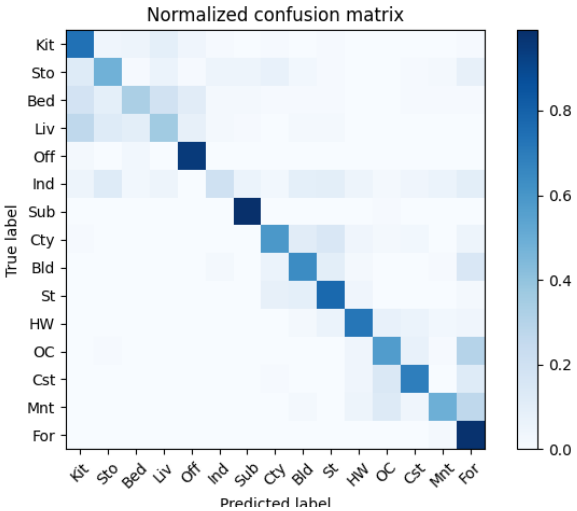


Fig1. Bag of SIFT (acc: 0.633)

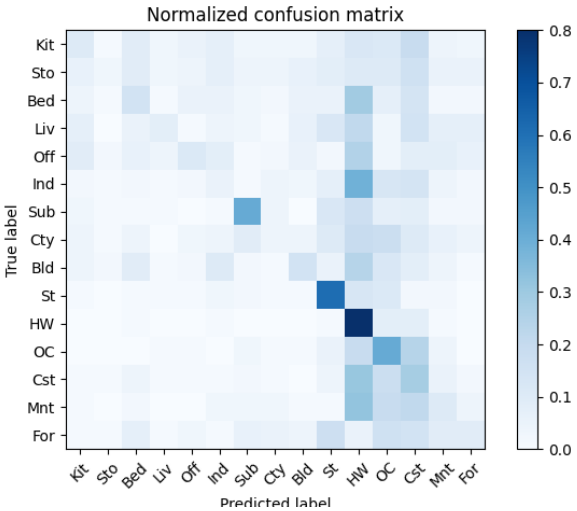


Fig2. Ting image (acc: 0.2273)

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

Accuracy:
(1) Bag of SIFT: **0.63** (2) Tiny of image: **0.2273**

Setting:
(1) Bag of SIFT:
Features 提取時；距離採用 cdist 進行計算，step 我設為 10，metric 我採用 euclidean (如同 build_vocabulary 函式的設定)；而繪製 histogram 時，我有進行正規化，扣除平均再除以總和。
(2) Tiny image:
我將每張照片 resize 成 16*16，接著再進行 flatten 和 normalize。
(3) nearest_neighbor_classify
計算距離時，我是採用 cdist，metric 選用 minkowski，而 p 設置為 0.3 發現可以過 strong baseline；而 KNN 部分我設置 k=6。

Result:
如 Fig1, 2 所示 Bag of SIFT accuracy 可達 0.63，confusion matrix 幾乎都在對角線；反之 Tiny image 只有 0.227，此方法大多會辨識成 Highway, OpenCountry, Coast 這三類。

Part 2. (25%)

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

Ans:

(1) MyNet: **0.8526** (2) ResNet18: **0.9034**

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

Ans:

Network architecture:

(1) MyNet

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 32, 32]	896
BatchNorm2d-2	[-1, 32, 32, 32]	64
Conv2d-3	[-1, 32, 32, 32]	9,248
BatchNorm2d-4	[-1, 32, 32, 32]	64
Conv2d-5	[-1, 64, 16, 16]	18,496
BatchNorm2d-6	[-1, 64, 16, 16]	128
Conv2d-7	[-1, 64, 16, 16]	36,928
BatchNorm2d-8	[-1, 64, 16, 16]	128
Conv2d-9	[-1, 128, 8, 8]	73,856
BatchNorm2d-10	[-1, 128, 8, 8]	256
Conv2d-11	[-1, 128, 8, 8]	147,584
BatchNorm2d-12	[-1, 128, 8, 8]	256
Conv2d-13	[-1, 256, 8, 8]	295,168
BatchNorm2d-14	[-1, 256, 8, 8]	512
Conv2d-15	[-1, 512, 8, 8]	1,180,160
BatchNorm2d-16	[-1, 512, 8, 8]	1,024
AdaptiveAvgPool2d-17	[-1, 512, 1, 1]	0
Linear-18	[-1, 10]	5,130

==========

Total params: 1,769,898

Trainable params: 1,769,898

Non-trainable params: 0

Input size (MB): 0.01

Forward/backward pass size (MB): 2.50

Params size (MB): 6.75

Estimated Total Size (MB): 9.27

```
----------------------------------------------------------------
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, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv6): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn6): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv7): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn7): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv8): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn8): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (avg_pool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=10, bias=True)
)
```

(1) ResNet18

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 32, 32]           1,728
       BatchNorm2d-2          [-1, 64, 32, 32]             128
              ReLU-3          [-1, 64, 32, 32]               0
          Identity-4          [-1, 64, 32, 32]               0
            Conv2d-5          [-1, 64, 32, 32]          36,864
       BatchNorm2d-6          [-1, 64, 32, 32]             128
              ReLU-7          [-1, 64, 32, 32]               0
            Conv2d-8          [-1, 64, 32, 32]          36,864
       BatchNorm2d-9          [-1, 64, 32, 32]             128
             ReLU-10          [-1, 64, 32, 32]               0
       BasicBlock-11          [-1, 64, 32, 32]               0
           Conv2d-12          [-1, 64, 32, 32]          36,864
      BatchNorm2d-13          [-1, 64, 32, 32]             128
             ReLU-14          [-1, 64, 32, 32]               0
           Conv2d-15          [-1, 64, 32, 32]          36,864
      BatchNorm2d-16          [-1, 64, 32, 32]             128
```

ReLU-17	[-1, 64, 32, 32]	0
BasicBlock-18	[-1, 64, 32, 32]	0
Conv2d-19	[-1, 128, 16, 16]	73,728
BatchNorm2d-20	[-1, 128, 16, 16]	256
ReLU-21	[-1, 128, 16, 16]	0
Conv2d-22	[-1, 128, 16, 16]	147,456
BatchNorm2d-23	[-1, 128, 16, 16]	256
Conv2d-24	[-1, 128, 16, 16]	8,192
BatchNorm2d-25	[-1, 128, 16, 16]	256
ReLU-26	[-1, 128, 16, 16]	0
BasicBlock-27	[-1, 128, 16, 16]	0
Conv2d-28	[-1, 128, 16, 16]	147,456
BatchNorm2d-29	[-1, 128, 16, 16]	256
ReLU-30	[-1, 128, 16, 16]	0
Conv2d-31	[-1, 128, 16, 16]	147,456
BatchNorm2d-32	[-1, 128, 16, 16]	256
ReLU-33	[-1, 128, 16, 16]	0
BasicBlock-34	[-1, 128, 16, 16]	0
Conv2d-35	[-1, 256, 8, 8]	294,912
BatchNorm2d-36	[-1, 256, 8, 8]	512
ReLU-37	[-1, 256, 8, 8]	0
Conv2d-38	[-1, 256, 8, 8]	589,824
BatchNorm2d-39	[-1, 256, 8, 8]	512
Conv2d-40	[-1, 256, 8, 8]	32,768
BatchNorm2d-41	[-1, 256, 8, 8]	512
ReLU-42	[-1, 256, 8, 8]	0
BasicBlock-43	[-1, 256, 8, 8]	0
Conv2d-44	[-1, 256, 8, 8]	589,824
BatchNorm2d-45	[-1, 256, 8, 8]	512
ReLU-46	[-1, 256, 8, 8]	0
Conv2d-47	[-1, 256, 8, 8]	589,824
BatchNorm2d-48	[-1, 256, 8, 8]	512
ReLU-49	[-1, 256, 8, 8]	0
BasicBlock-50	[-1, 256, 8, 8]	0
Conv2d-51	[-1, 512, 4, 4]	1,179,648
BatchNorm2d-52	[-1, 512, 4, 4]	1,024
ReLU-53	[-1, 512, 4, 4]	0
Conv2d-54	[-1, 512, 4, 4]	2,359,296
BatchNorm2d-55	[-1, 512, 4, 4]	1,024
Conv2d-56	[-1, 512, 4, 4]	131,072
BatchNorm2d-57	[-1, 512, 4, 4]	1,024
ReLU-58	[-1, 512, 4, 4]	0

```
          BasicBlock-59          [-1, 512, 4, 4]               0
              Conv2d-60          [-1, 512, 4, 4]       2,359,296
         BatchNorm2d-61          [-1, 512, 4, 4]           1,024
                ReLU-62          [-1, 512, 4, 4]               0
              Conv2d-63          [-1, 512, 4, 4]       2,359,296
         BatchNorm2d-64          [-1, 512, 4, 4]           1,024
                ReLU-65          [-1, 512, 4, 4]               0
          BasicBlock-66          [-1, 512, 4, 4]               0
AdaptiveAvgPool2d-67          [-1, 512, 1, 1]               0
              Linear-68                [-1, 10]           5,130
              ResNet-69                [-1, 10]               0
================================================================
```

Total params: 11,173,962

Trainable params: 11,173,962

Non-trainable params: 0

Input size (MB): 0.01

Forward/backward pass size (MB): 16.00

Params size (MB): 42.63

Estimated Total Size (MB): 58.64

```
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)
    )
)
```

Main different

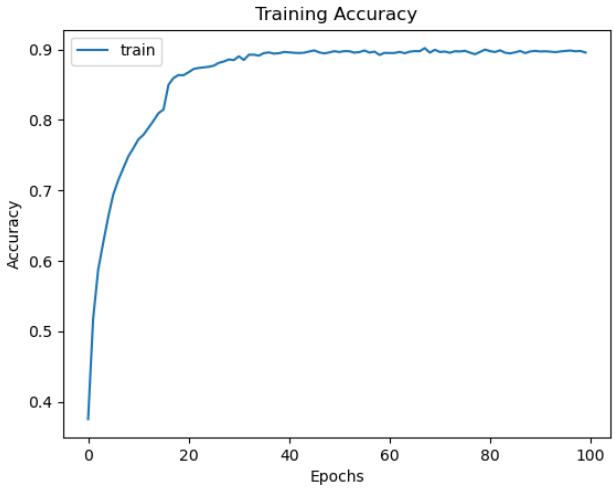
MyNet 只有 8 層卷積層，而 ResNet18 有 18 層，因此在可訓練的 parameters 上，ResNet18 也高出許多；在 ResNet18 中有進行 downsample，kernel size 為 1*1，不像 MyNet 的 kernel size 都是 3*3。

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

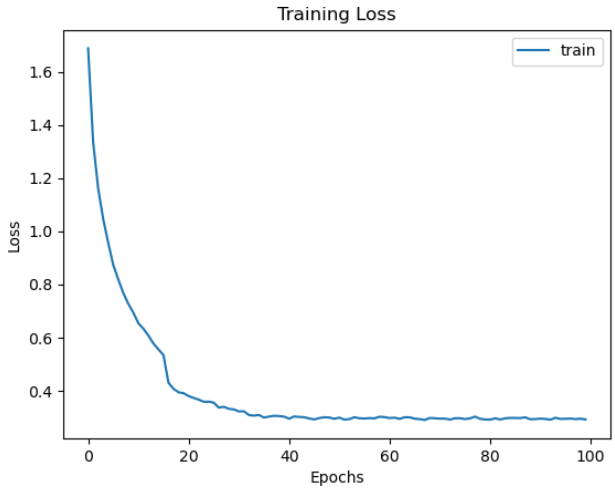
Ans:

MyNet

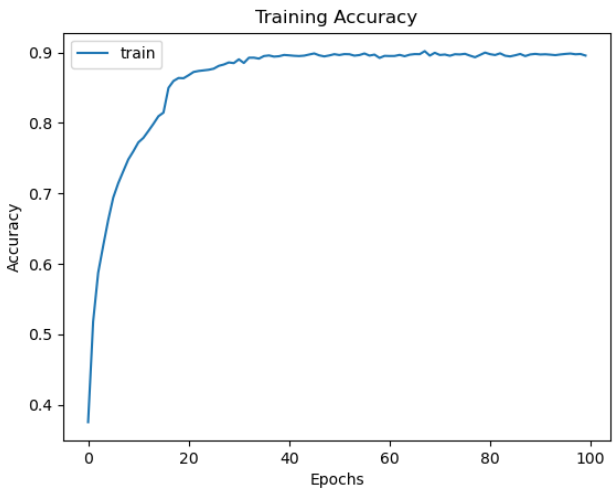
(1) training accuracy



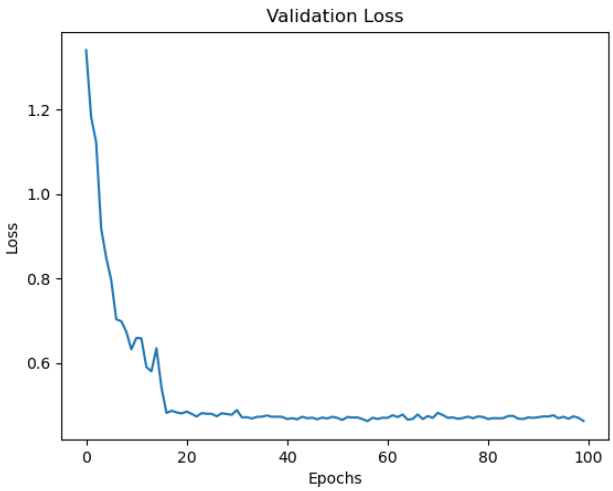
(2) training loss



(3) validation accuracy

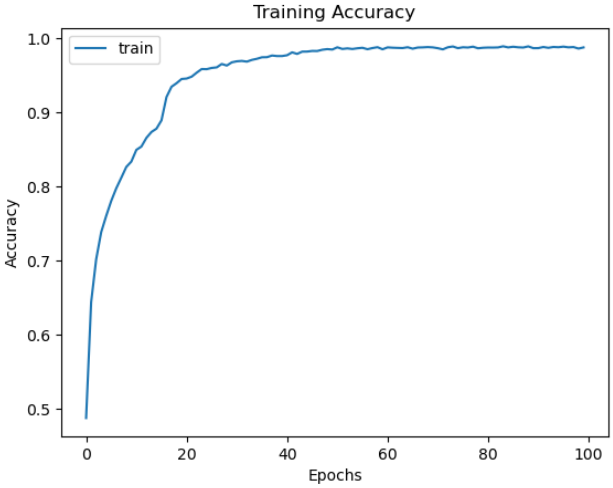


(4) validation loss

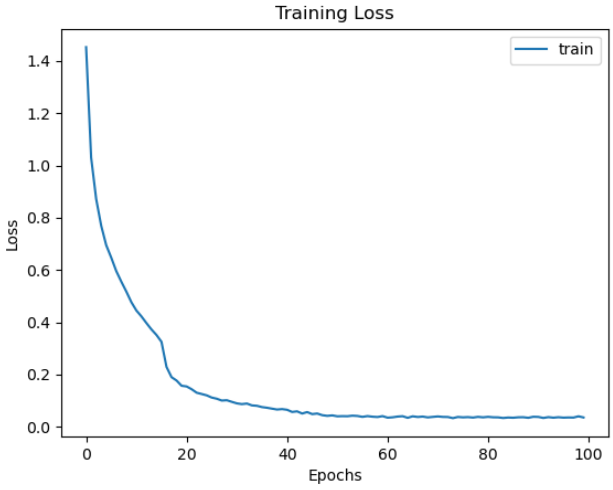


ResNet18

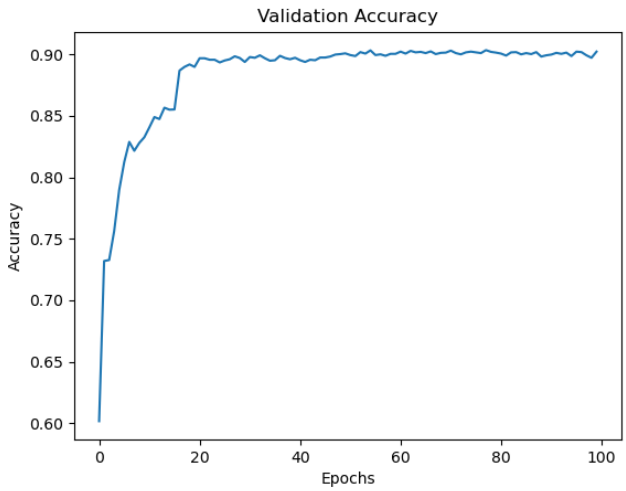
(1) training accuracy



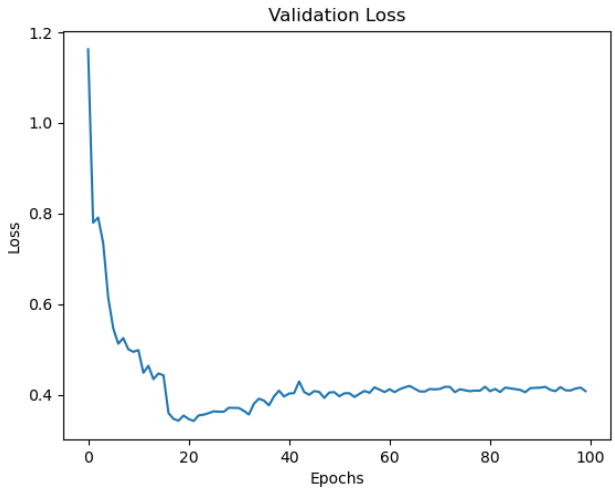
(2) training loss



(3) validation accuracy



(4) validation loss



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

Config:
如同 config.py 可見，我設置 trainging epoch 為 100，batch size 為 16，皆使用 Adam optimizer，learning rate 為 0.001，而在 epoch 分別為 16、45、64、93 時動態減少 learning rate。

Model architecture:
在 MyNet 裡，我有在其中加了 2 層 max pooling(2x2, stride=2)，效果會比較好，然後每層卷積層結束都進行 batch normalization。另外 ResNet18 部分，有參考註解裡的方式可以減少 kernel size 在第一層卷積層，因此我有設定 *self.resnet.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3, stride=1, padding=1, bias=False)* 使得第一層 kernel size 為 3，而不是正常的 7，然後也有嘗試把第一層的 max pooling 換成 Identity。

Data augmentation:
在訓練過程，有將圖片進行 Random HorizontalFlip、Random Rotation、Random Crop，如下圖。

```
if split == 'train':
    transform = transforms.Compose([
        transforms.Resize((32,32)),
        ##### TODO: Data Augmentation Begin #####
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(10),
        transforms.RandomCrop(32, padding=3),
        ##### TODO: Data Augmentation End #####
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ])
```