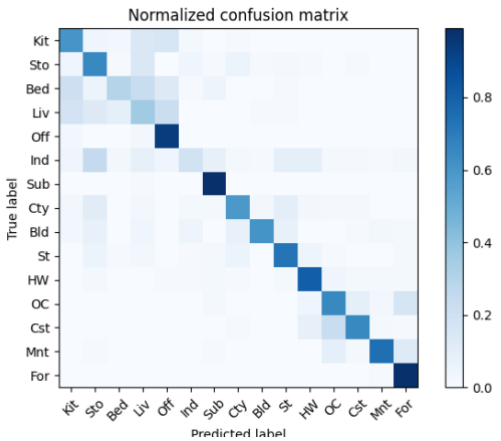
Computer Vision HW2 Report

Student ID: R12521601

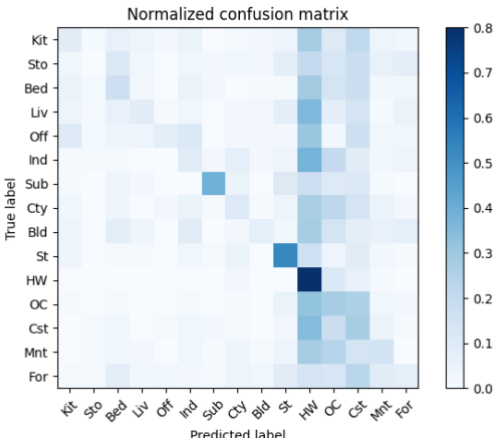
Name: 詹承諺

Part 1. (10%)

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



Bag of sift
(accuracy: 0.654)



Tiny image
(accuracy: 0.213)

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

Accuracy:
(1) Bag of sift: 0.654
(2) Tiny image: 0.213

Setting:
(1) Bag of sift:

計算距離的部分在 get_bags_of_sifts 的 function 中 cdist 是用 default 的 metric (‘euclidean’)，在每個 function 中都有先對 image 做 normalization。

(2) Tiny image:

先將 image resize 成 16*16，再將其 flatten 並 normalize。

(3) nearest_neighbor_classify

先將 image 做 normalize，在 nearest_neighbor_classify 的 function 中 cdist 的 metric 是用 'minkowski' 且 p 設為 0.5。KNN 的部分則是將 k 設置為 6。

Result:

Bag of sift 相較於 Tiny image 來說準確率高了許多，Bag of sift 的 confusion matrix 對角線顏色明顯較深，而 Tiny image 的 confusion matrix 中則較無明顯的對角線，許多圖片都會分類成 HW, OC, CST。

Part 2. (25%)

• Report accuracy of both models on the validation set. (2%)
Ans:
(1) MyNet: 0.8440
(2) ResNet18: 0.9126
• Print the network architecture & number of parameters of both models. What is the main difference between ResNet and other CNN architectures? (5%)
Ans:

(1) MyNet

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	1,792
BatchNorm2d-2	[-1, 64, 32, 32]	128
ReLU-3	[-1, 64, 32, 32]	0
MaxPool2d-4	[-1, 64, 16, 16]	0
Conv2d-5	[-1, 128, 16, 16]	73,856
BatchNorm2d-6	[-1, 128, 16, 16]	256
ReLU-7	[-1, 128, 16, 16]	0
MaxPool2d-8	[-1, 128, 8, 8]	0
Conv2d-9	[-1, 256, 8, 8]	295,168
BatchNorm2d-10	[-1, 256, 8, 8]	512
ReLU-11	[-1, 256, 8, 8]	0
MaxPool2d-12	[-1, 256, 4, 4]	0
Conv2d-13	[-1, 512, 4, 4]	1,180,160
BatchNorm2d-14	[-1, 512, 4, 4]	1,024
ReLU-15	[-1, 512, 4, 4]	0
MaxPool2d-16	[-1, 512, 2, 2]	0
Linear-17	[-1, 1024]	2,098,176
ReLU-18	[-1, 1024]	0
Dropout-19	[-1, 1024]	0
Linear-20	[-1, 512]	524,800
ReLU-21	[-1, 512]	0
Dropout-22	[-1, 512]	0
Linear-23	[-1, 10]	5,130

==========

Total params: 4,181,002
Trainable params: 4,181,002
Non-trainable params: 0

Input size (MB): 0.01
Forward/backward pass size (MB): 3.08
Params size (MB): 15.95
Estimated Total Size (MB): 19.04

```
MyNet(
  (conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (batchnorm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (batchnorm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (batchnorm3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv4): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (batchnorm4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=2048, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=10, bias=True)
  (relu): ReLU()
  (dropout): Dropout(p=0.5, inplace=False)
)
```

---

(2) ResNet18

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	1,792
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: 11,174,026
Trainable params: 11,174,026
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))
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU()
    (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)
  )
)
```

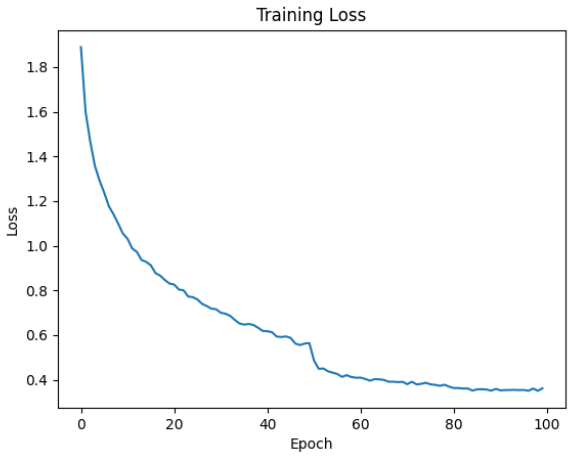
Main Difference:

ResNet 中 Convolution 的 stride 有(1,1)、(2,,2)兩種，但 MyNet 只有(1,1)一種。以 Convolution 層數來說，MyNet 只有 4 層，但 ResNet18 有 18 層。ResNet18 的 Trainable Parameter 數比 MyNet 多，整體 Total size 也較大。

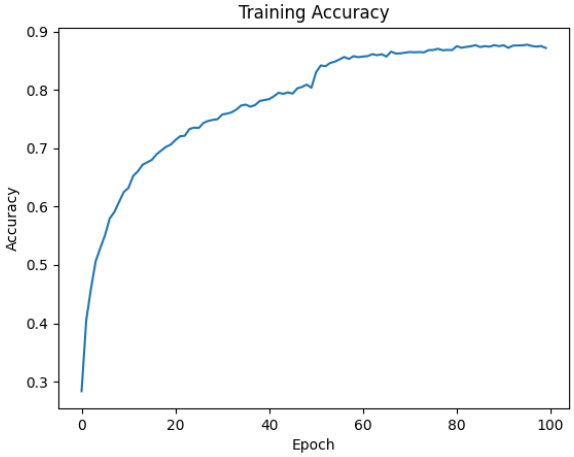
• Plot four learning curves (loss & accuracy) of the training process (train/validation) for both models. Total 8 plots. (8%)
Ans:

MyNet

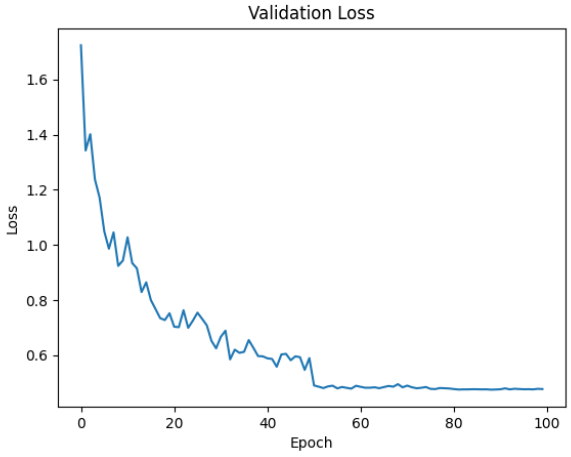
(1) Training Loss



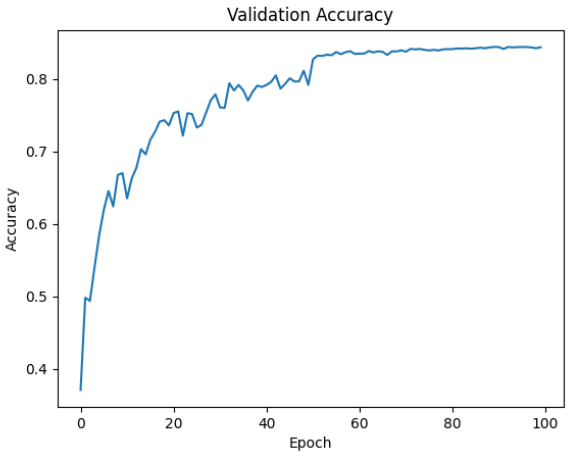
(2) Training Accuracy



(3) Validation Loss

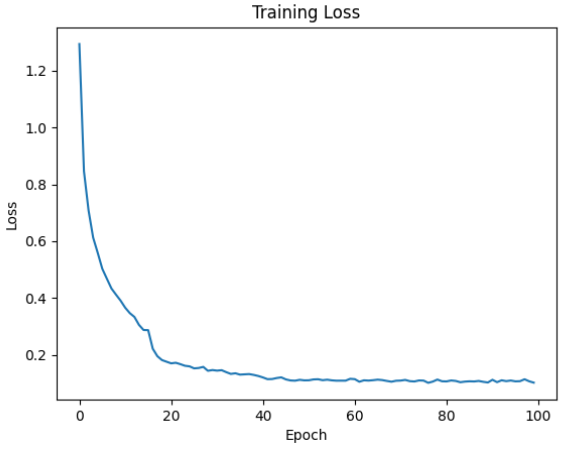


(4) Validation Accuracy

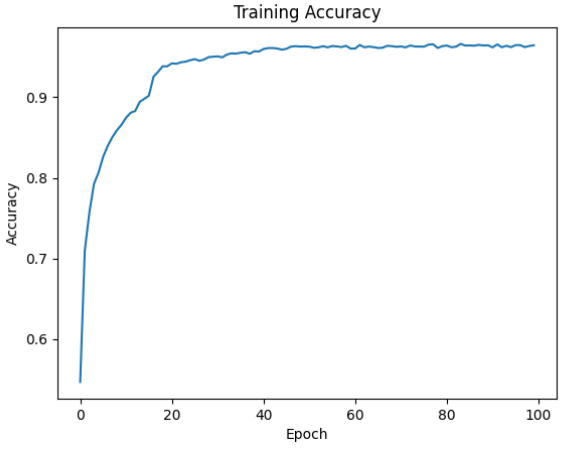


ResNet18

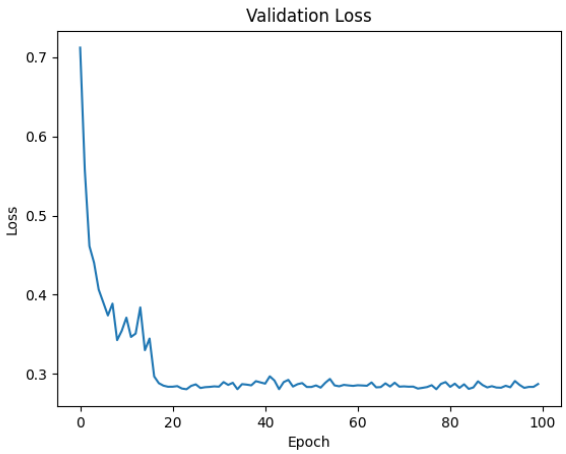
(1) Training Loss



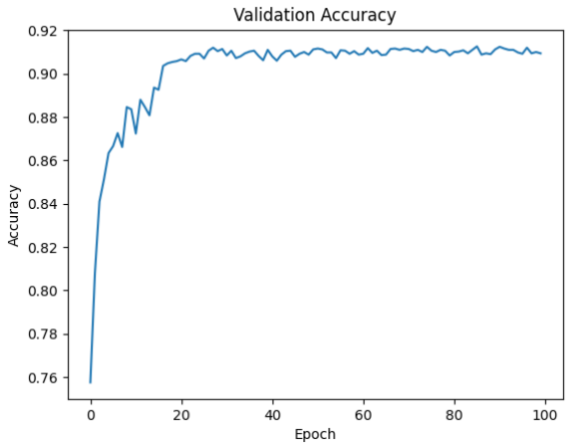
(2) Training Accuracy



(3) Validation Loss



(4) Validation Accuracy



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

Ans:

Data augmentation:

```
if split == 'train':
    transform = transforms.Compose([
        transforms.Resize((32,32)),
        ##### TODO: Data Augmentation Begin #####
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(15),
        transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
        transforms.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1),
        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])
    ])
```

除了原有的 Resize 跟 Normalize 外，還有做 RandonHorizontalFlip, RandomRotation, RandomAffine, ColorJitter, RandomCrop

Model Architecture:

MyNet 的部分，原來前面只有四個 Convolution 並做 Maxpooling，後來將 Convolution 後面做 batch normalization 及 ReLU 準確率就有顯著的進步。

```
x = self.pool(self.relu(self.batchnorm1(self.conv1(x))))
x = self.pool(self.relu(self.batchnorm2(self.conv2(x))))
x = self.pool(self.relu(self.batchnorm3(self.conv3(x))))
x = self.pool(self.relu(self.batchnorm4(self.conv4(x))))
x = x.view(-1, 512 * 2 * 2)
x = self.dropout(self.relu(self.fc1(x)))
x = self.dropout(self.relu(self.fc2(x)))
x = self.fc3(x)
```

ResNet18 的部分，則是在後面多加一層 Convolution Layer(縮小 kernel size 至 3)及做 ReLU，並將

Maxpool Layer 換成 Identity()，最後再加一層 Fully Connected Layer。

```
# (batch_size, 3, 32, 32)
self.resnet = models.resnet18(pretrained=True)
self.resnet.conv1 = nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1)
self.resnet.relu = nn.ReLU()
self.resnet.maxpool = Identity()

self.resnet.fc = nn.Linear(self.resnet.fc.in_features, 10)
```

Hyper Parameter (Config.py):

在 Config.py 中，Epoch 設為 100，batch size 調整成 128。Learning rate 則設為 0.01，並在 epoch 為

50, 80 的時候動態減少 learning rate。