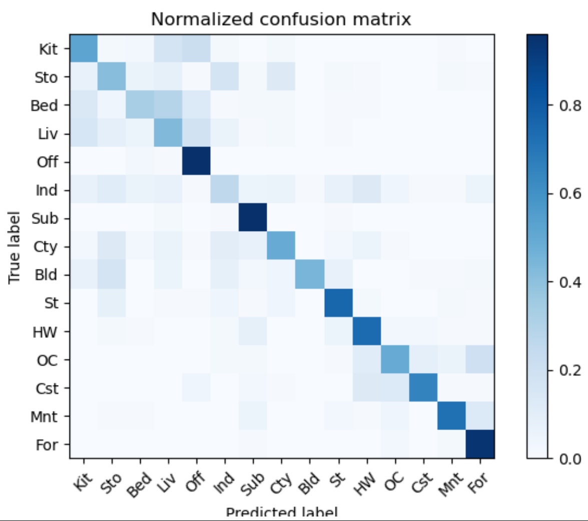
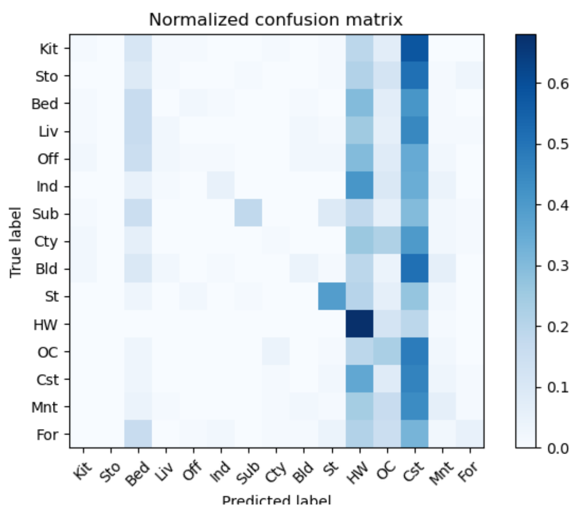
Computer Vision HW2 Report

Student ID: R10921a36
Name: 石子仙

Part 1. (10%)

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

Ans:

Bag of sift	Tiny image
	

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

Ans:

- Setting of bag of sift and accuracy

	euclidean	cosine	jensenshannon	cityblock
k=5	0.523	0.555	0.5933	0.6073
k=7	0.533	0.56	0.6087	0.6073

- Setting of Tiny image and accuracy

	euclidean	cosine	jensenshannon	cityblock
k=5	0.157	0.157	0.142	0.188
k=7	0.162	0.158	0.145	0.189

上表標記紅色的為 best accuracy，綜合兩種表，我的程式碼最終設定為螢光筆底色的，k=7，cdist metric='cityblock'。

Confusion matrix 對角線上的點越深代表預測準確率越高，預測能力越好。可以從上表以及第一題的 confusion matrix 來看，Tiny image representation 的預測準確率比較

差，多半集中在 label HW 和 Cst。個人認為，有可能是僅從 tiny image 沒辦法提供足夠的資訊，而且也許只看一小張圖片的話，可能每張圖的 tinmy image 都會有很相似的地方，一張圖片需要靠整張圖片的資訊合併起來才有意義，所以改把照片用 bag of sift 表示就能保留較多有用的資訊。

Part 2. (35%)

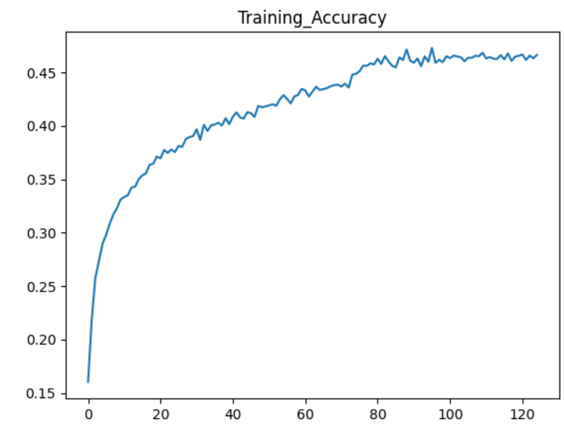
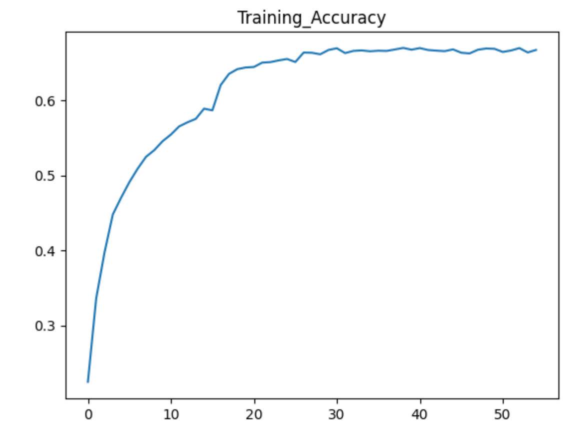
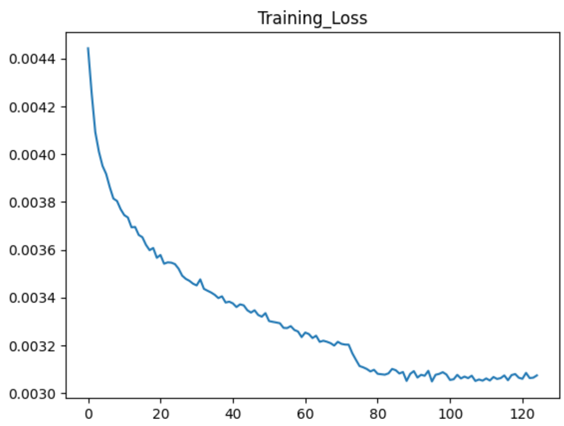
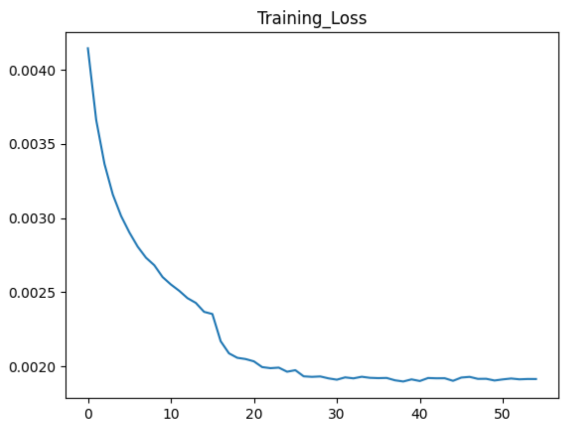
• Compare the performance on residual networks and LeNet. Plot the learning curve (loss and accuracy) on both training and validation sets for both 2 schemes. 8 plots in total. (20%)

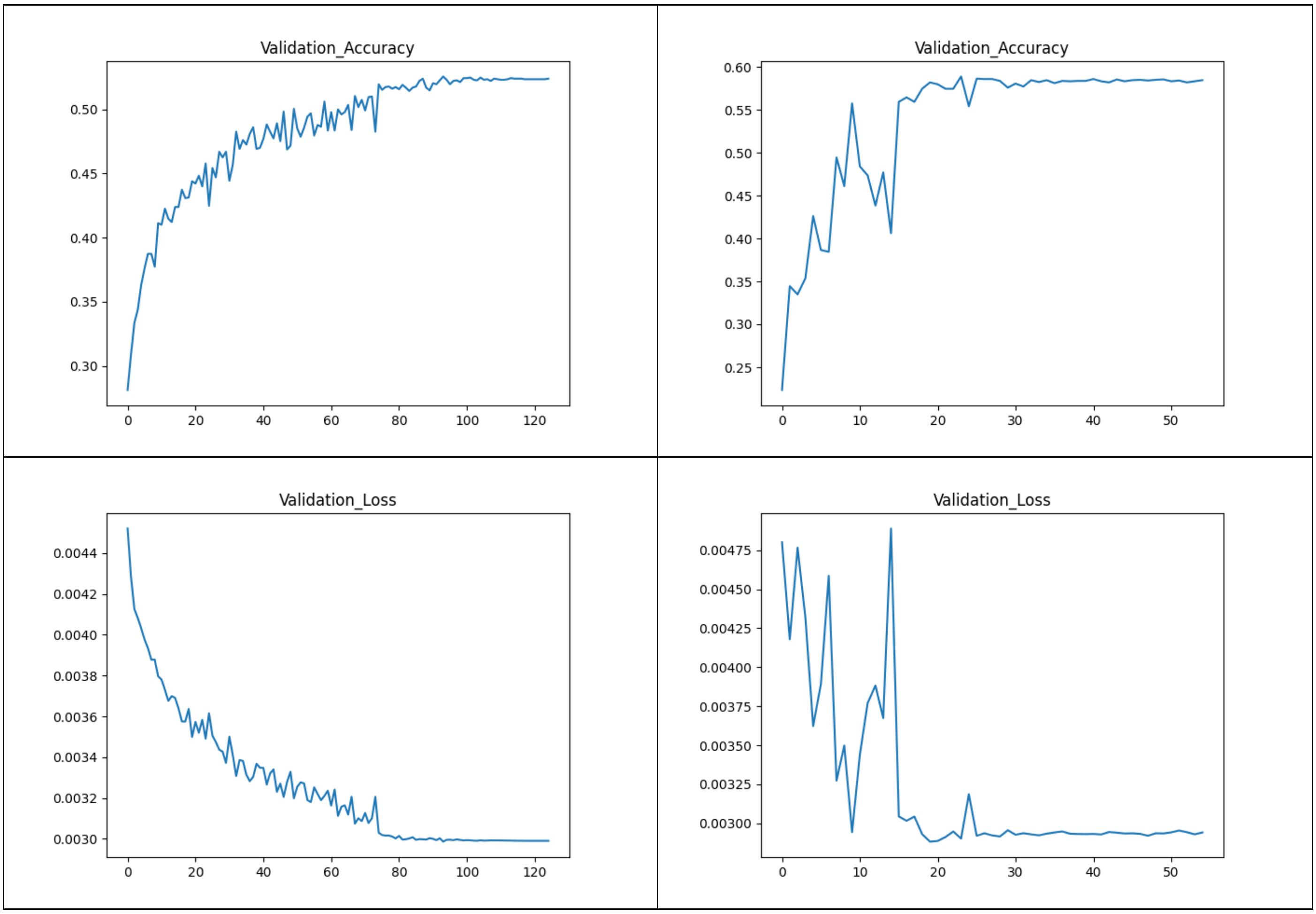
Ans:

- Accuracy

	LeNet	myResNet
Accuracy on public test	0.5992	0.6672

- Learning curve

LeNet	myResNet
	
	



在本題我有將原先助教提供的 optimizer 從 SGD 改成 Adam，scheduler 從 MultiStepLR 改成 ReduceLROnPlateau，若有連續 5 個 epoch model 沒有進步的話就調小 learning rate 為原本的 0.1 倍，並設定 early stop 為 30 個 epoch。

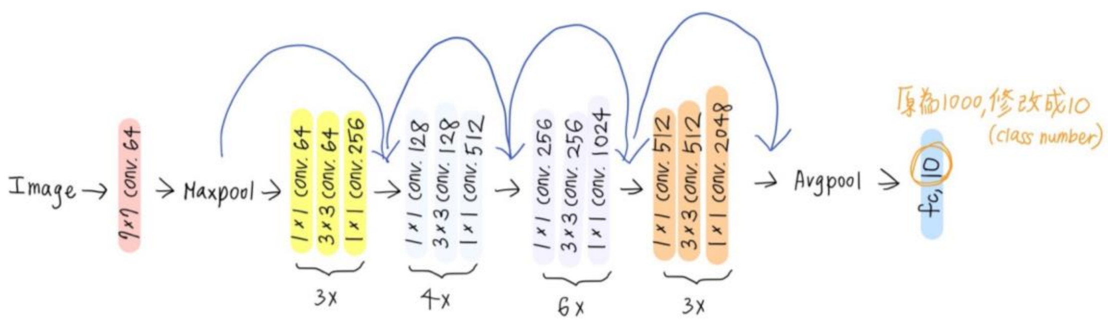
由上表 Accuracy 表格可得出 myResNet 的 performance 比 myLeNet 好很多(0.6672 v.s. 0.5992)，myResNet 的收斂速度比 myLenet 快。

• Attach basic information of the model you use including model architecture and number of the parameters. (5%)

Ans:

我使用 Pretrained resnet50，因為 model architecture 要佔好幾頁，這邊先用畫的，完整 print 出來的架構附在 report 最後。

- **Model Architecture:**



- **Number of the parameters:**

  Total Trainable Params: 23,528,522

  由於要把每一層都列出來會非常長，詳細附 report 的最後面。第一層 conv1 就是 3*64*7*7 = 9408，layer1.0.conv1.weight 是 64*64 = 4096，以此類推，最後再加總。

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

Ans:

- 用來通過 baseline 的 model，我是使用 pretrained resnet50，並加入一些修改，最終 public test accuracy 為 0.8468。

- 我有做了以下改變：
  1. 修改參數
  2. 修改 optimizer 與 learning rate scheduler
  3. 修改 data augmentation
  4. 加入 semi-supervised

- 以下以各點詳細說明：
  1. 修改參數

     以下列的不是全部的 cfg，是比較重要的，另外我有加入 early_stop 來控制 training process。

```
 'split_ratio': 0.9,
 'batch_size': 512,
 'num_epoch': 300,
 'early_stop': 30,
 'optimizer': 'Adam',
 'optim_hparas': {'lr': 0.001,
                  'weight_decay': 0.001,
                  'betas': (0.4, 0.999)}
}
```

  2. 修改 optimizer 與 learning rate scheduler
     - 將 Optimizer 從 SGD 改成 Adam，

```
'optimizer': 'Adam',
 'optim_hparas': {'lr': 0.001,
             'weight_decay': 0.001,
             'betas': (0.4, 0.999)}
}
```

- Learning rate scheduler 改成

```
optim.lr_scheduler.ReduceLROnPlateau(optimizer,
                                     mode='min', patience=5, cooldown=3)
```

後面使用

```
scheduler.step(val_loss)
```

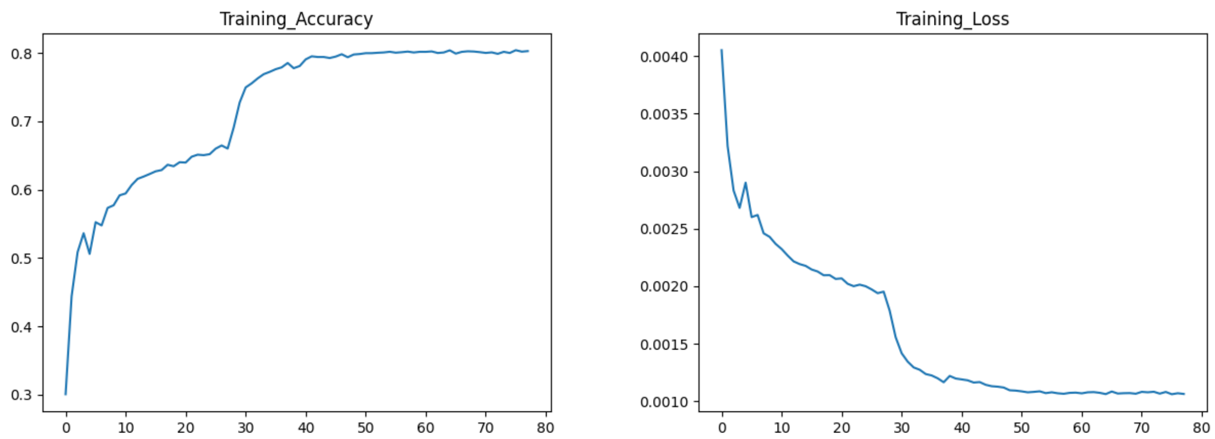
意思是如果 validation loss 在 5 個 epoch 下沒下降的話，就將 learning rate 改成原本的 0.1 倍（此為 default 的值我沒修改），改成這個後 accuracy 上升了約 0.01。

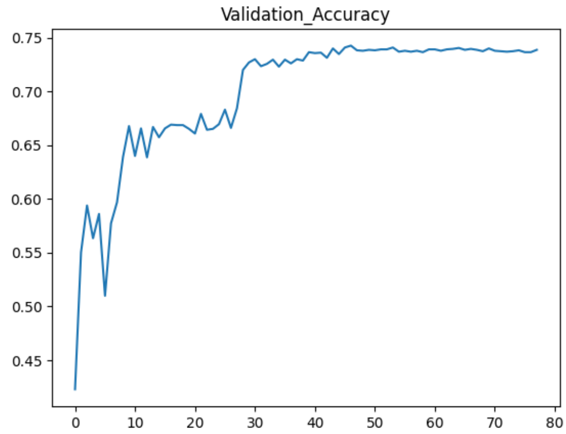
3. 修改 data augmentation

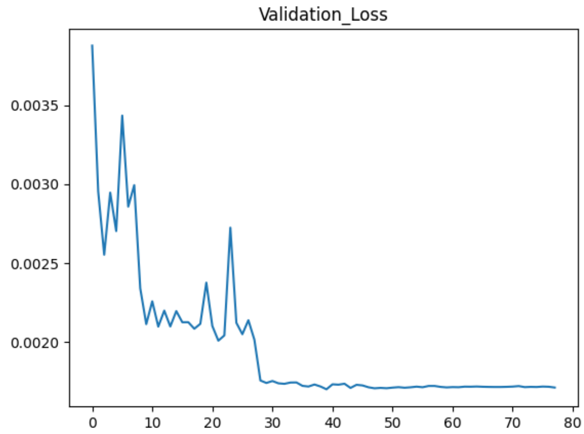
   加入 torchvision.transforms 下的 AutoAugment、RandomRotation、RandomHorizontalFlip、RandomAffine

4. 使用 semi-supervised

   在 validation accuracy 大於 0.7 且 validation accuracy 有進步時，就會開始使用 semi-supervised，令 confidence threshold 為 0.9，將 unlabeled data（這邊有將 public test 也拿來用）加入到 training set 中。由下圖的 learning curve 可以看到，在開始做 semi-supervised 後（validation accuracy 大於 0.7），training 與 validation 的 loss 與 accuracy 都有明顯的下降與提升。







Model Architecture 與 Number of parameters 的詳細

- **Model Architecture:**

詳細如下：

```
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): Bottleneck(
    (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
    (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace=True)
    (downsample): Sequential(
      (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer2): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

```
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=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): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer3): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (4): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
```

```
    )
    (5): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer4): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=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)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Sequential(
    (0): Linear(in_features=2048, out_features=10, bias=True)
  )
)
```

- 詳細 number of model parameter：

```
+-------------------------------+------------+
|            Modules            | Parameters |
+-------------------------------+------------+
|          conv1.weight         |    9408    |
|           bn1.weight          |     64     |
|            bn1.bias           |     64     |
|      layer1.0.conv1.weight    |    4096    |
|       layer1.0.bn1.weight     |     64     |
|        layer1.0.bn1.bias      |     64     |
```

```
|      layer1.0.conv2.weight      |   36864    |
|       layer1.0.bn2.weight       |     64     |
|        layer1.0.bn2.bias        |     64     |
|      layer1.0.conv3.weight      |   16384    |
|       layer1.0.bn3.weight       |    256     |
|        layer1.0.bn3.bias        |    256     |
|  layer1.0.downsample.0.weight   |   16384    |
|  layer1.0.downsample.1.weight   |    256     |
|   layer1.0.downsample.1.bias    |    256     |
|      layer1.1.conv1.weight      |   16384    |
|       layer1.1.bn1.weight       |     64     |
|        layer1.1.bn1.bias        |     64     |
|      layer1.1.conv2.weight      |   36864    |
|       layer1.1.bn2.weight       |     64     |
|        layer1.1.bn2.bias        |     64     |
|      layer1.1.conv3.weight      |   16384    |
|       layer1.1.bn3.weight       |    256     |
|        layer1.1.bn3.bias        |    256     |
|      layer1.2.conv1.weight      |   16384    |
|       layer1.2.bn1.weight       |     64     |
|        layer1.2.bn1.bias        |     64     |
|      layer1.2.conv2.weight      |   36864    |
|       layer1.2.bn2.weight       |     64     |
|        layer1.2.bn2.bias        |     64     |
|      layer1.2.conv3.weight      |   16384    |
|       layer1.2.bn3.weight       |    256     |
|        layer1.2.bn3.bias        |    256     |
|      layer2.0.conv1.weight      |   32768    |
|       layer2.0.bn1.weight       |    128     |
|        layer2.0.bn1.bias        |    128     |
|      layer2.0.conv2.weight      |   147456   |
|       layer2.0.bn2.weight       |    128     |
|        layer2.0.bn2.bias        |    128     |
|      layer2.0.conv3.weight      |   65536    |
|       layer2.0.bn3.weight       |    512     |
|        layer2.0.bn3.bias        |    512     |
|  layer2.0.downsample.0.weight   |   131072   |
|  layer2.0.downsample.1.weight   |    512     |
|   layer2.0.downsample.1.bias    |    512     |
|      layer2.1.conv1.weight      |   65536    |
```

```
|       layer2.1.bn1.weight        |   128   |
|        layer2.1.bn1.bias         |   128   |
|      layer2.1.conv2.weight       |  147456 |
|       layer2.1.bn2.weight        |   128   |
|        layer2.1.bn2.bias         |   128   |
|      layer2.1.conv3.weight       |  65536  |
|       layer2.1.bn3.weight        |   512   |
|        layer2.1.bn3.bias         |   512   |
|      layer2.2.conv1.weight       |  65536  |
|       layer2.2.bn1.weight        |   128   |
|        layer2.2.bn1.bias         |   128   |
|      layer2.2.conv2.weight       |  147456 |
|       layer2.2.bn2.weight        |   128   |
|        layer2.2.bn2.bias         |   128   |
|      layer2.2.conv3.weight       |  65536  |
|       layer2.2.bn3.weight        |   512   |
|        layer2.2.bn3.bias         |   512   |
|      layer2.3.conv1.weight       |  65536  |
|       layer2.3.bn1.weight        |   128   |
|        layer2.3.bn1.bias         |   128   |
|      layer2.3.conv2.weight       |  147456 |
|       layer2.3.bn2.weight        |   128   |
|        layer2.3.bn2.bias         |   128   |
|      layer2.3.conv3.weight       |  65536  |
|       layer2.3.bn3.weight        |   512   |
|        layer2.3.bn3.bias         |   512   |
|      layer3.0.conv1.weight       |  131072 |
|       layer3.0.bn1.weight        |   256   |
|        layer3.0.bn1.bias         |   256   |
|      layer3.0.conv2.weight       |  589824 |
|       layer3.0.bn2.weight        |   256   |
|        layer3.0.bn2.bias         |   256   |
|      layer3.0.conv3.weight       |  262144 |
|       layer3.0.bn3.weight        |   1024  |
|        layer3.0.bn3.bias         |   1024  |
| layer3.0.downsample.0.weight |  524288 |
| layer3.0.downsample.1.weight |   1024  |
|  layer3.0.downsample.1.bias  |   1024  |
|      layer3.1.conv1.weight       |  262144 |
|       layer3.1.bn1.weight        |   256   |
```

```
|      layer3.1.bn1.bias       |   256    |
|     layer3.1.conv2.weight    |  589824  |
|      layer3.1.bn2.weight     |   256    |
|       layer3.1.bn2.bias      |   256    |
|     layer3.1.conv3.weight    |  262144  |
|      layer3.1.bn3.weight     |   1024   |
|       layer3.1.bn3.bias      |   1024   |
|     layer3.2.conv1.weight    |  262144  |
|      layer3.2.bn1.weight     |   256    |
|       layer3.2.bn1.bias      |   256    |
|     layer3.2.conv2.weight    |  589824  |
|      layer3.2.bn2.weight     |   256    |
|       layer3.2.bn2.bias      |   256    |
|     layer3.2.conv3.weight    |  262144  |
|      layer3.2.bn3.weight     |   1024   |
|       layer3.2.bn3.bias      |   1024   |
|     layer3.3.conv1.weight    |  262144  |
|      layer3.3.bn1.weight     |   256    |
|       layer3.3.bn1.bias      |   256    |
|     layer3.3.conv2.weight    |  589824  |
|      layer3.3.bn2.weight     |   256    |
|       layer3.3.bn2.bias      |   256    |
|     layer3.3.conv3.weight    |  262144  |
|      layer3.3.bn3.weight     |   1024   |
|       layer3.3.bn3.bias      |   1024   |
|     layer3.4.conv1.weight    |  262144  |
|      layer3.4.bn1.weight     |   256    |
|       layer3.4.bn1.bias      |   256    |
|     layer3.4.conv2.weight    |  589824  |
|      layer3.4.bn2.weight     |   256    |
|       layer3.4.bn2.bias      |   256    |
|     layer3.4.conv3.weight    |  262144  |
|      layer3.4.bn3.weight     |   1024   |
|       layer3.4.bn3.bias      |   1024   |
|     layer3.5.conv1.weight    |  262144  |
|      layer3.5.bn1.weight     |   256    |
|       layer3.5.bn1.bias      |   256    |
|     layer3.5.conv2.weight    |  589824  |
|      layer3.5.bn2.weight     |   256    |
|       layer3.5.bn2.bias      |   256    |
```

```
|     layer3.5.conv3.weight      |   262144   |
|      layer3.5.bn3.weight       |    1024    |
|       layer3.5.bn3.bias        |    1024    |
|     layer4.0.conv1.weight      |   524288   |
|      layer4.0.bn1.weight       |    512     |
|       layer4.0.bn1.bias        |    512     |
|     layer4.0.conv2.weight      |  2359296   |
|      layer4.0.bn2.weight       |    512     |
|       layer4.0.bn2.bias        |    512     |
|     layer4.0.conv3.weight      |  1048576   |
|      layer4.0.bn3.weight       |    2048    |
|       layer4.0.bn3.bias        |    2048    |
| layer4.0.downsample.0.weight |  2097152   |
| layer4.0.downsample.1.weight |    2048    |
|  layer4.0.downsample.1.bias  |    2048    |
|     layer4.1.conv1.weight      |  1048576   |
|      layer4.1.bn1.weight       |    512     |
|       layer4.1.bn1.bias        |    512     |
|     layer4.1.conv2.weight      |  2359296   |
|      layer4.1.bn2.weight       |    512     |
|       layer4.1.bn2.bias        |    512     |
|     layer4.1.conv3.weight      |  1048576   |
|      layer4.1.bn3.weight       |    2048    |
|       layer4.1.bn3.bias        |    2048    |
|     layer4.2.conv1.weight      |  1048576   |
|      layer4.2.bn1.weight       |    512     |
|       layer4.2.bn1.bias        |    512     |
|     layer4.2.conv2.weight      |  2359296   |
|      layer4.2.bn2.weight       |    512     |
|       layer4.2.bn2.bias        |    512     |
|     layer4.2.conv3.weight      |  1048576   |
|      layer4.2.bn3.weight       |    2048    |
|       layer4.2.bn3.bias        |    2048    |
|          fc.0.weight           |   20480    |
|           fc.0.bias            |     10     |
+------------------------------+------------+
Total Trainable Params: 23528522
```