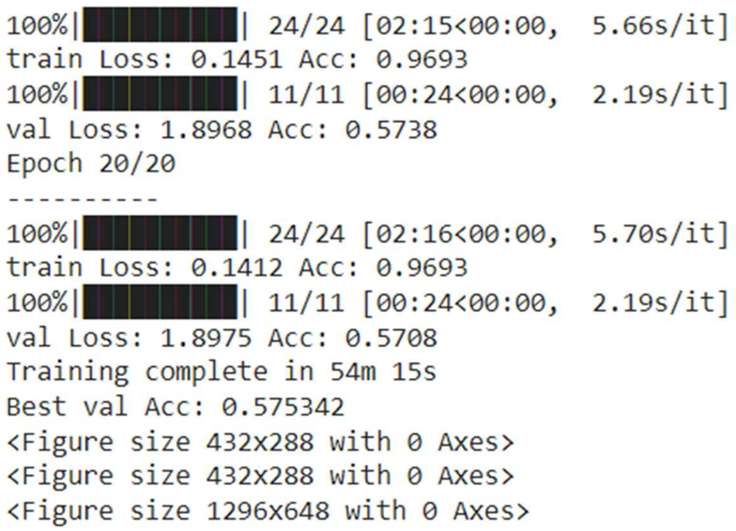
ML_Report

常博愛 資工三 408410086

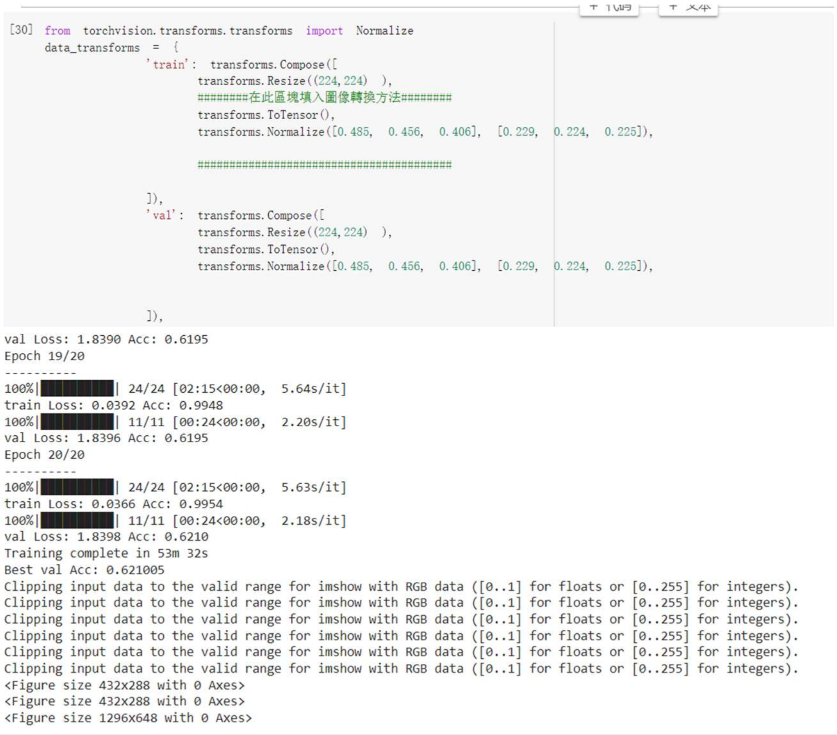
1.

以下所有方法未使用時的準確率統一為：**0.575342**



```
100%|██████████| 24/24 [02:15<00:00,  5.66s/it]
train Loss: 0.1451 Acc: 0.9693
100%|██████████| 11/11 [00:24<00:00,  2.19s/it]
val Loss: 1.8968 Acc: 0.5738
Epoch 20/20
----------
100%|██████████| 24/24 [02:16<00:00,  5.70s/it]
train Loss: 0.1412 Acc: 0.9693
100%|██████████| 11/11 [00:24<00:00,  2.19s/it]
val Loss: 1.8975 Acc: 0.5708
Training complete in 54m 15s
Best val Acc: 0.575342
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 1296x648 with 0 Axes>
```

Method 1：resize+normalize+totensor 準確率：0.6210



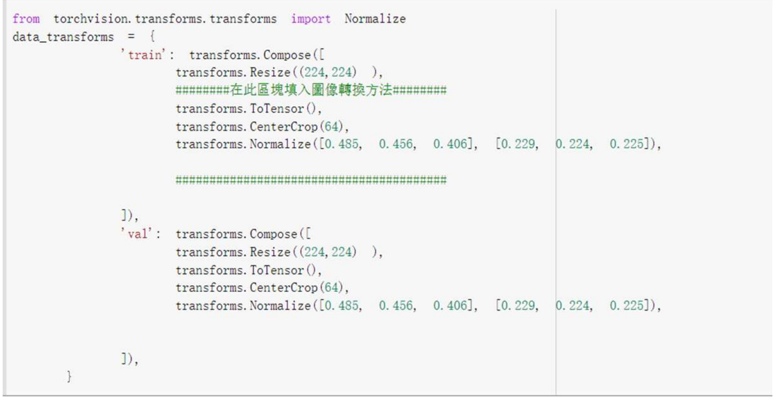
```
[30] from  torchvision.transforms.transforms  import  Normalize
     data_transforms  =  {
                    'train':  transforms.Compose([
                          transforms.Resize((224,224)  ),
                          ########在此區塊填入圖像轉換方法########
                          transforms.ToTensor(),
                          transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),

                          ########################################


                    ]),
                    'val':  transforms.Compose([
                          transforms.Resize((224,224)  ),
                          transforms.ToTensor(),
                          transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),


                    ]),
```

```
val Loss: 1.8390 Acc: 0.6195
Epoch 19/20
----------
100%|██████████| 24/24 [02:15<00:00,  5.64s/it]
train Loss: 0.0392 Acc: 0.9948
100%|██████████| 11/11 [00:24<00:00,  2.20s/it]
val Loss: 1.8396 Acc: 0.6195
Epoch 20/20
----------
100%|██████████| 24/24 [02:15<00:00,  5.63s/it]
train Loss: 0.0366 Acc: 0.9954
100%|██████████| 11/11 [00:24<00:00,  2.18s/it]
val Loss: 1.8398 Acc: 0.6210
Training complete in 53m 32s
Best val Acc: 0.621005
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 1296x648 with 0 Axes>
```

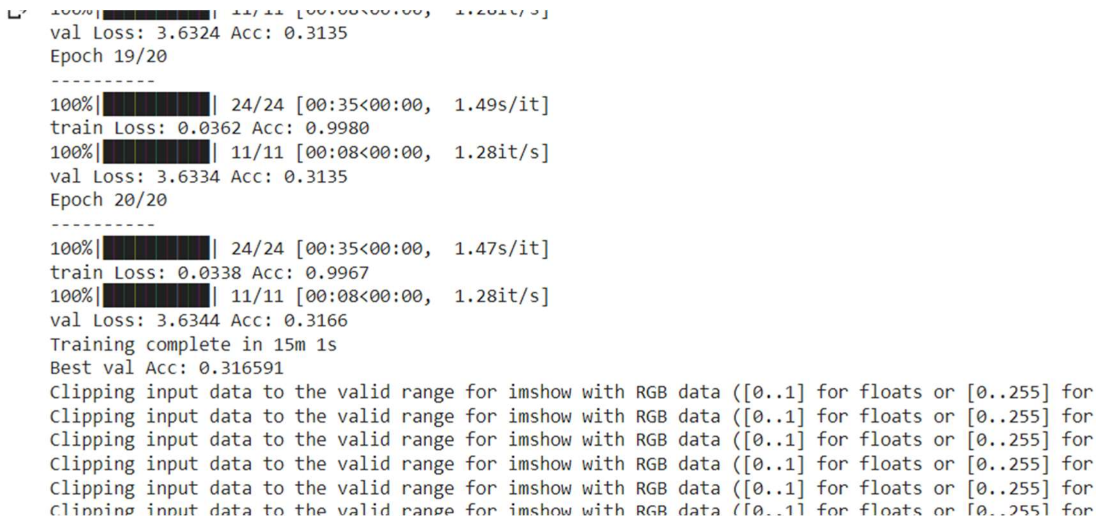
目的：用均值和標準差對 `Tensor` 進行歸一化處理。

結果：有提高訓練的準確度。
Note:由於 normalize 對實驗有正影響，所以接下來的方法皆有做 normalize 並觀察結果是否有提高精確度。

Method 2:centercrop 準確度：0.3166



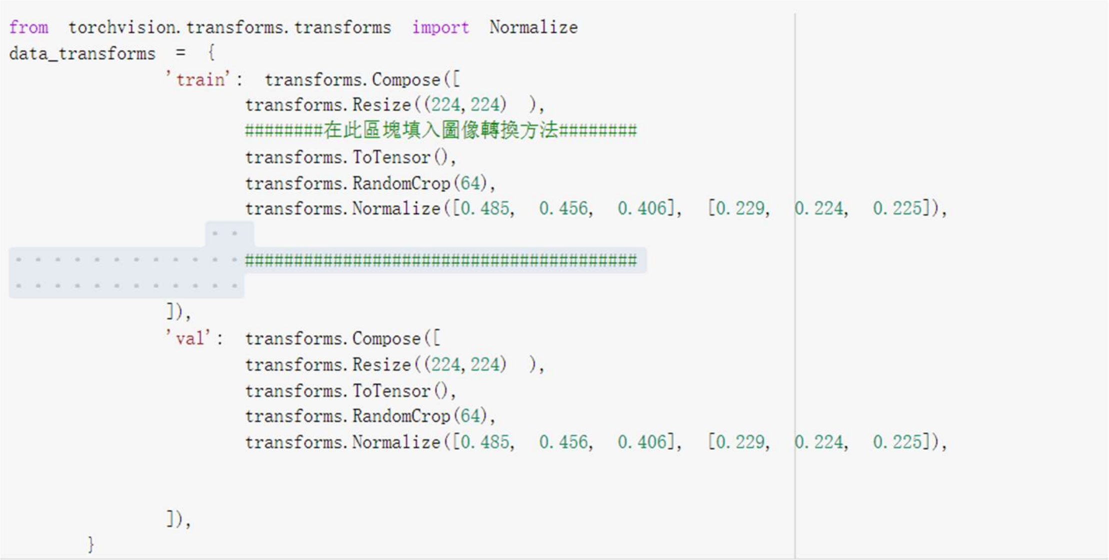
```
from  torchvision.transforms.transforms  import  Normalize
data_transforms  =  {
                'train':  transforms.Compose([
                        transforms.Resize((224,224)  ),
                        ########在此區塊填入圖像轉換方法########
                        transforms.ToTensor(),
                        transforms.CenterCrop(64),
                        transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),

                        ########################################


                ]),
                'val':  transforms.Compose([
                        transforms.Resize((224,224)  ),
                        transforms.ToTensor(),
                        transforms.CenterCrop(64),
                        transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),


                ]),
        }
```



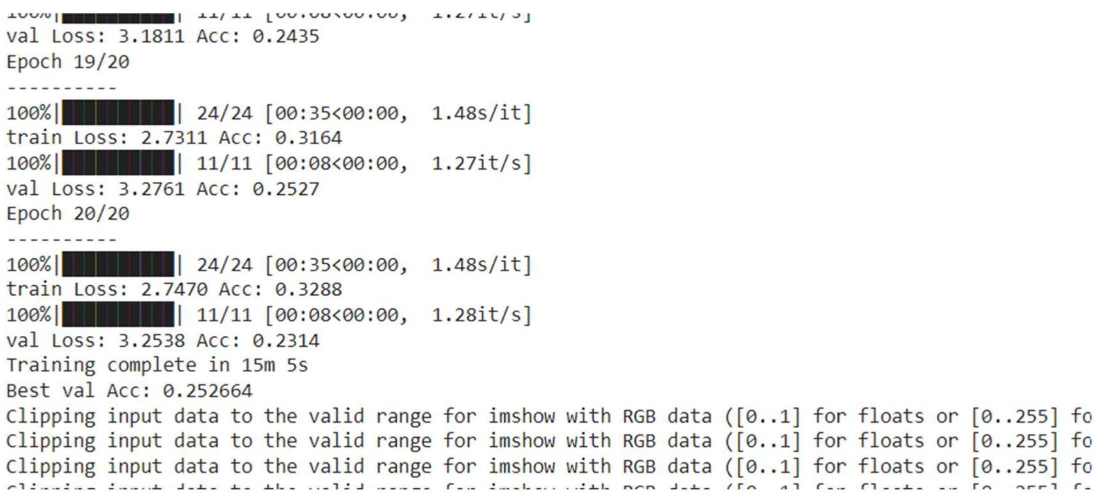
```
100%|██████████| 11/11 [00:08<00:00,  1.28it/s]
val Loss: 3.6324 Acc: 0.3135
Epoch 19/20
----------
100%|██████████| 24/24 [00:35<00:00,  1.49s/it]
train Loss: 0.0362 Acc: 0.9980
100%|██████████| 11/11 [00:08<00:00,  1.28it/s]
val Loss: 3.6334 Acc: 0.3135
Epoch 20/20
----------
100%|██████████| 24/24 [00:35<00:00,  1.47s/it]
train Loss: 0.0338 Acc: 0.9967
100%|██████████| 11/11 [00:08<00:00,  1.28it/s]
val Loss: 3.6344 Acc: 0.3166
Training complete in 15m 1s
Best val Acc: 0.316591
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for
```

目的：是從圖像的中心位置裁剪指定大小的圖像進行訓練
結果：準確率大幅下降，對比使用方法前產生了負影響。

Method 3: random crop 準確度：0.252664



```
from  torchvision.transforms.transforms  import  Normalize
data_transforms  =  {
                'train':  transforms.Compose([
                        transforms.Resize((224,224)  ),
                        ########在此區塊填入圖像轉換方法########
                        transforms.ToTensor(),
                        transforms.RandomCrop(64),
                        transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),

                        ########################################

                ]),
                'val':  transforms.Compose([
                        transforms.Resize((224,224)  ),
                        transforms.ToTensor(),
                        transforms.RandomCrop(64),
                        transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),


                ]),
        }
```



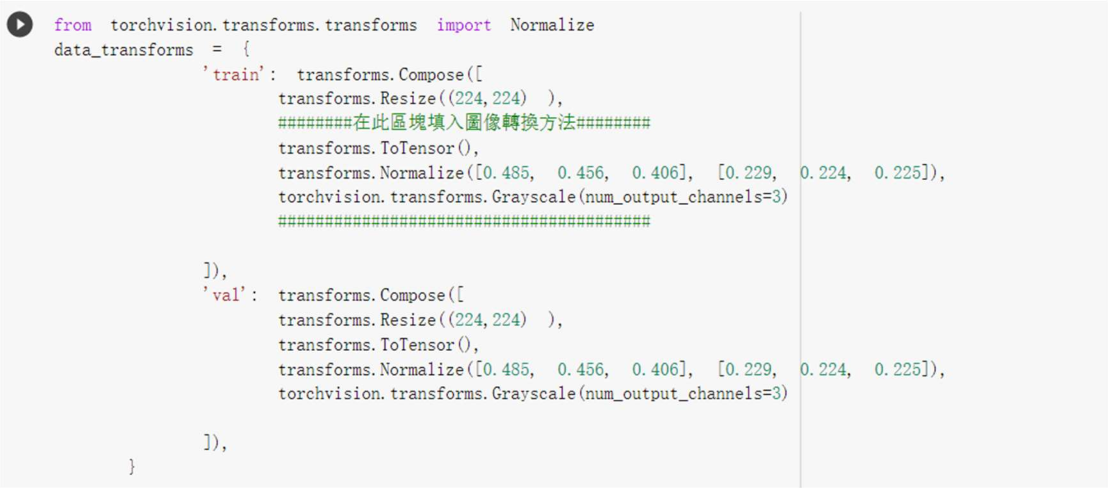
```
100%|██████████| 11/11 [00:08<00:00,  1.27it/s]
val Loss: 3.1811 Acc: 0.2435
Epoch 19/20
----------
100%|██████████| 24/24 [00:35<00:00,  1.48s/it]
train Loss: 2.7311 Acc: 0.3164
100%|██████████| 11/11 [00:08<00:00,  1.27it/s]
val Loss: 3.2761 Acc: 0.2527
Epoch 20/20
----------
100%|██████████| 24/24 [00:35<00:00,  1.48s/it]
train Loss: 2.7470 Acc: 0.3288
100%|██████████| 11/11 [00:08<00:00,  1.28it/s]
val Loss: 3.2538 Acc: 0.2314
Training complete in 15m 5s
Best val Acc: 0.252664
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] fo
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] fo
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] fo
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] fo
```

目的：是從圖像的隨機位置裁剪指定大小的圖像進行訓練

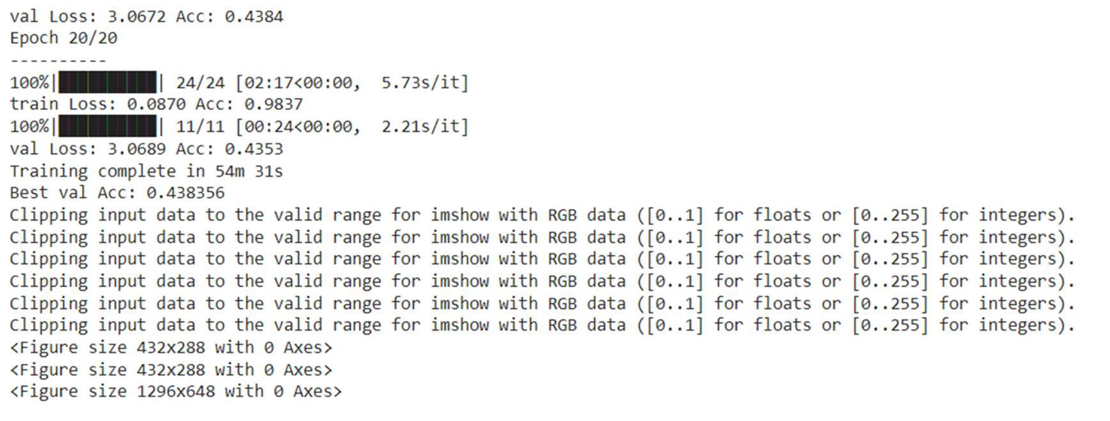
結果：準確率大幅下降，對比使用方法前產生了負影響。

Method 4: grayscale 準確度：0.438356

：



```
from  torchvision.transforms.transforms  import  Normalize
data_transforms  =  {
                'train':  transforms.Compose([
                        transforms.Resize((224,224)  ),
                        ########在此區塊填入圖像轉換方法########
                        transforms.ToTensor(),
                        transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),
                        torchvision.transforms.Grayscale(num_output_channels=3)
                        ########################################

                ]),
                'val':  transforms.Compose([
                        transforms.Resize((224,224)  ),
                        transforms.ToTensor(),
                        transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),
                        torchvision.transforms.Grayscale(num_output_channels=3)

                ]),
        }
```

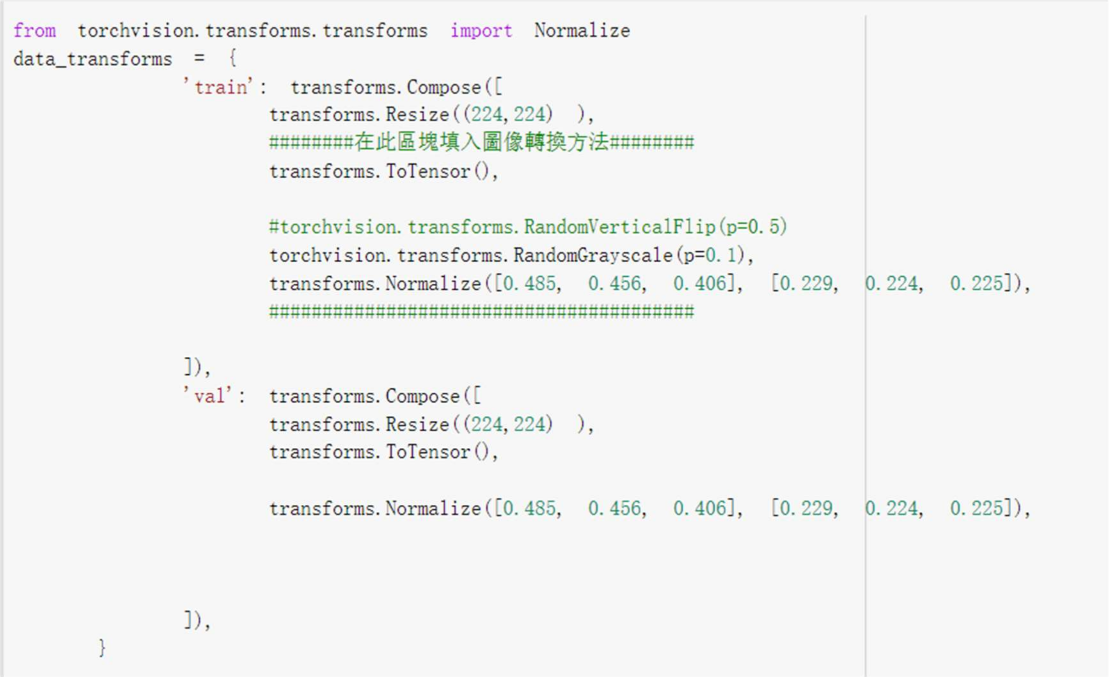


```
val Loss: 3.0672 Acc: 0.4384
Epoch 20/20
----------
100%|██████████| 24/24 [02:17<00:00,  5.73s/it]
train Loss: 0.0870 Acc: 0.9837
100%|██████████| 11/11 [00:24<00:00,  2.21s/it]
val Loss: 3.0689 Acc: 0.4353
Training complete in 54m 31s
Best val Acc: 0.438356
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 1296x648 with 0 Axes>
```

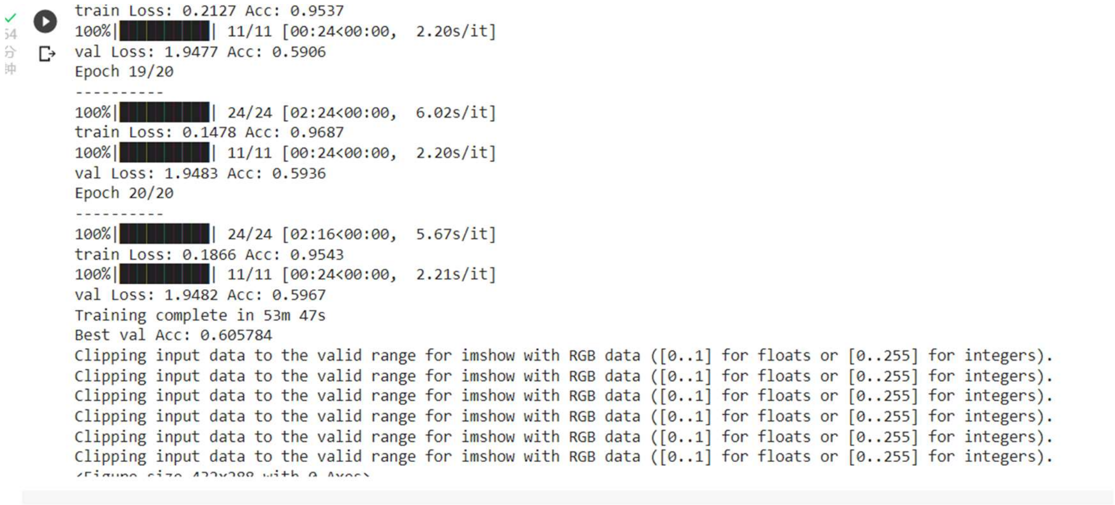
目的：將圖像轉換為灰度圖像進行訓練。

結果：準確率略有下降，對比使用方法前產生了負影響。

Method 5:random gray 準確度：0.605784



```
from torchvision.transforms.transforms import Normalize
data_transforms = {
            'train': transforms.Compose([
                  transforms.Resize((224,224) ),
                  ########在此區塊填入圖像轉換方法########
                  transforms.ToTensor(),

                  #torchvision.transforms.RandomVerticalFlip(p=0.5)
                  torchvision.transforms.RandomGrayscale(p=0.1),
                  transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
                  ########################################

            ]),
            'val': transforms.Compose([
                  transforms.Resize((224,224) ),
                  transforms.ToTensor(),

                  transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),



            ]),
      }
```

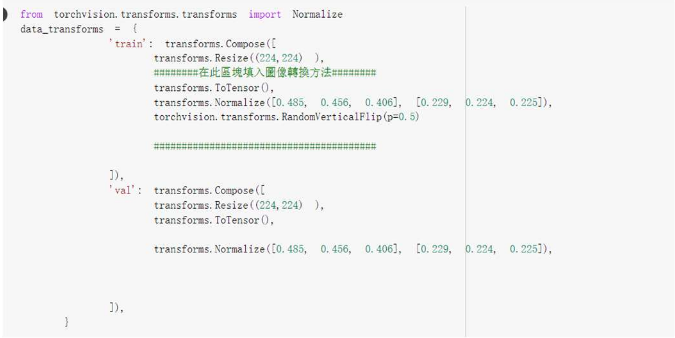


```
train Loss: 0.2127 Acc: 0.9537
100%|██████████| 11/11 [00:24<00:00,  2.20s/it]
val Loss: 1.9477 Acc: 0.5906
Epoch 19/20
----------
100%|██████████| 24/24 [02:24<00:00,  6.02s/it]
train Loss: 0.1478 Acc: 0.9687
100%|██████████| 11/11 [00:24<00:00,  2.20s/it]
val Loss: 1.9483 Acc: 0.5936
Epoch 20/20
----------
100%|██████████| 24/24 [02:16<00:00,  5.67s/it]
train Loss: 0.1866 Acc: 0.9543
100%|██████████| 11/11 [00:24<00:00,  2.21s/it]
val Loss: 1.9482 Acc: 0.5967
Training complete in 53m 47s
Best val Acc: 0.605784
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<Figure size 432x288 with 0 Axes>
```

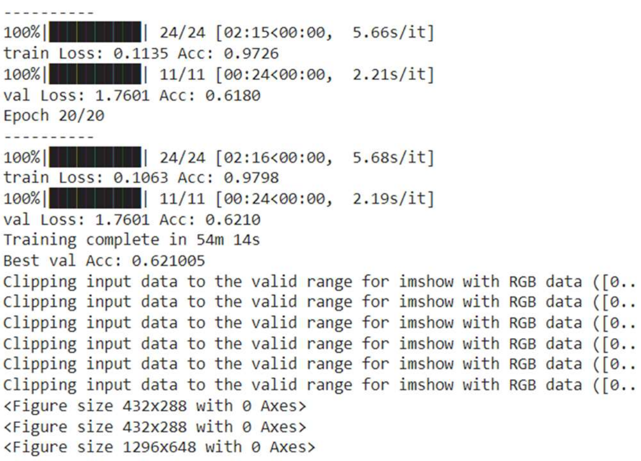
目的：以一定概率將圖像轉換為灰度圖像進行訓練。

結果：因共同使用了 normalize，且準確率對比單純使用 normalize 略有下降，對比使用方法前產生了輕微負影響。

Method 6:randomverticalflip 準確度：0.621005



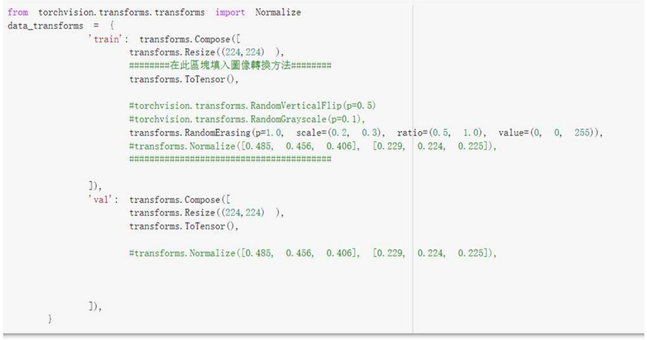
```
from torchvision.transforms.transforms import Normalize
data_transforms = {
            'train': transforms.Compose([
                  transforms.Resize((224,224) ),
                  ########在此區塊填入圖像轉換方法########
                  transforms.ToTensor(),
                  transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
                  torchvision.transforms.RandomVerticalFlip(p=0.5)

                  ########################################


            ]),
            'val': transforms.Compose([
                  transforms.Resize((224,224) ),
                  transforms.ToTensor(),

                  transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),



            ]),
      }
```



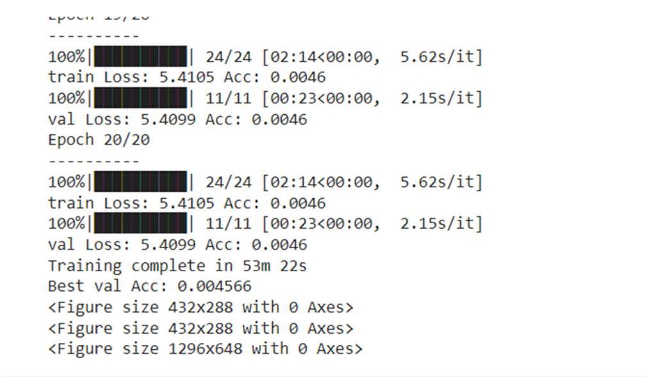
```
----------
100%|██████████| 24/24 [02:15<00:00,  5.66s/it]
train Loss: 0.1135 Acc: 0.9726
100%|██████████| 11/11 [00:24<00:00,  2.21s/it]
val Loss: 1.7601 Acc: 0.6180
Epoch 20/20
----------
100%|██████████| 24/24 [02:16<00:00,  5.68s/it]
train Loss: 0.1063 Acc: 0.9798
100%|██████████| 11/11 [00:24<00:00,  2.19s/it]
val Loss: 1.7601 Acc: 0.6210
Training complete in 54m 14s
Best val Acc: 0.621005
Clipping input data to the valid range for imshow with RGB data ([0..
Clipping input data to the valid range for imshow with RGB data ([0..
Clipping input data to the valid range for imshow with RGB data ([0..
Clipping input data to the valid range for imshow with RGB data ([0..
Clipping input data to the valid range for imshow with RGB data ([0..
Clipping input data to the valid range for imshow with RGB data ([0..
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 1296x648 with 0 Axes>
```

目的：以一定的概率對圖像進行垂直翻轉進行訓練。
結果：對比使用方法前無影響。

Method 6:Randomerasing 準確度：0.004566



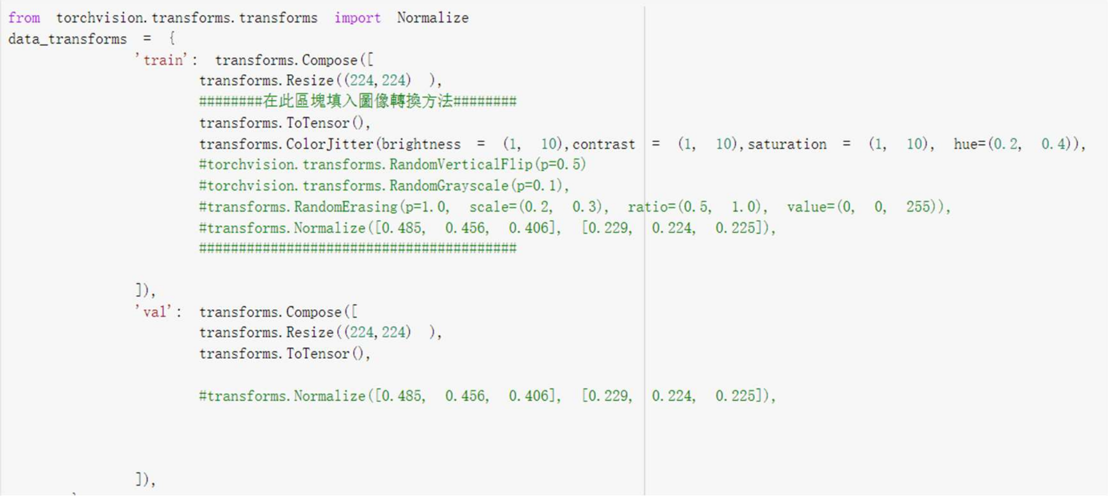
```
from  torchvision.transforms.transforms  import  Normalize
data_transforms  =  {
            'train':  transforms.Compose([
                  transforms.Resize((224,224)  ),
                  ########在此區塊填入圖像轉換方法########
                  transforms.ToTensor(),

                  #torchvision.transforms.RandomVerticalFlip(p=0.5)
                  #torchvision.transforms.RandomGrayscale(p=0.1),
                  transforms.RandomErasing(p=1.0,  scale=(0.2,  0.3),  ratio=(0.5,  1.0),  value=(0,  0,  255)),
                  #transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),
                  ########################################

            ]),
            'val':  transforms.Compose([
                  transforms.Resize((224,224)  ),
                  transforms.ToTensor(),

                  #transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),



            ]),
      }
```



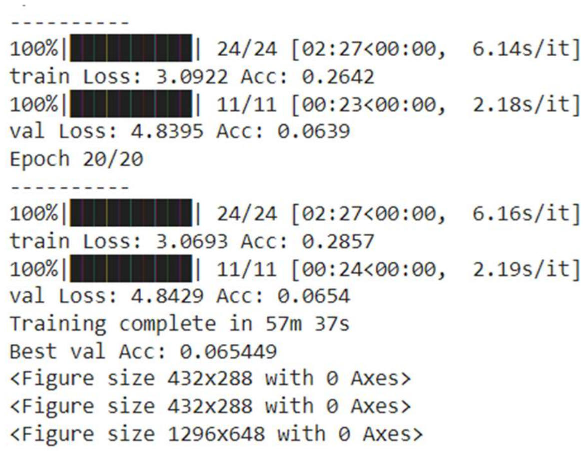
```
Epoch 19/20
----------
100%|██████████| 24/24 [02:14<00:00,  5.62s/it]
train Loss: 5.4105 Acc: 0.0046
100%|██████████| 11/11 [00:23<00:00,  2.15s/it]
val Loss: 5.4099 Acc: 0.0046
Epoch 20/20
----------
100%|██████████| 24/24 [02:14<00:00,  5.62s/it]
train Loss: 5.4105 Acc: 0.0046
100%|██████████| 11/11 [00:23<00:00,  2.15s/it]
val Loss: 5.4099 Acc: 0.0046
Training complete in 53m 22s
Best val Acc: 0.004566
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 1296x648 with 0 Axes>
```

目的: 隨機選擇圖像中的一塊區域，擦除其圖元，主要用來進行資料增強.
結果：對比使用方法產生**嚴重**負影響。

Method 7: colorjitter 準確度：



```
from  torchvision.transforms.transforms  import  Normalize
data_transforms  =  {
            'train':  transforms.Compose([
                  transforms.Resize((224,224)  ),
                  ########在此區塊填入圖像轉換方法########
                  transforms.ToTensor(),
                  transforms.ColorJitter(brightness  =  (1,  10),contrast  =  (1,  10),saturation  =  (1,  10),  hue=(0.2,  0.4)),
                  #torchvision.transforms.RandomVerticalFlip(p=0.5)
                  #torchvision.transforms.RandomGrayscale(p=0.1),
                  #transforms.RandomErasing(p=1.0,  scale=(0.2,  0.3),  ratio=(0.5,  1.0),  value=(0,  0,  255)),
                  #transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),
                  ########################################

            ]),
            'val':  transforms.Compose([
                  transforms.Resize((224,224)  ),
                  transforms.ToTensor(),

                  #transforms.Normalize([0.485,  0.456,  0.406],  [0.229,  0.224,  0.225]),



            ]),
```



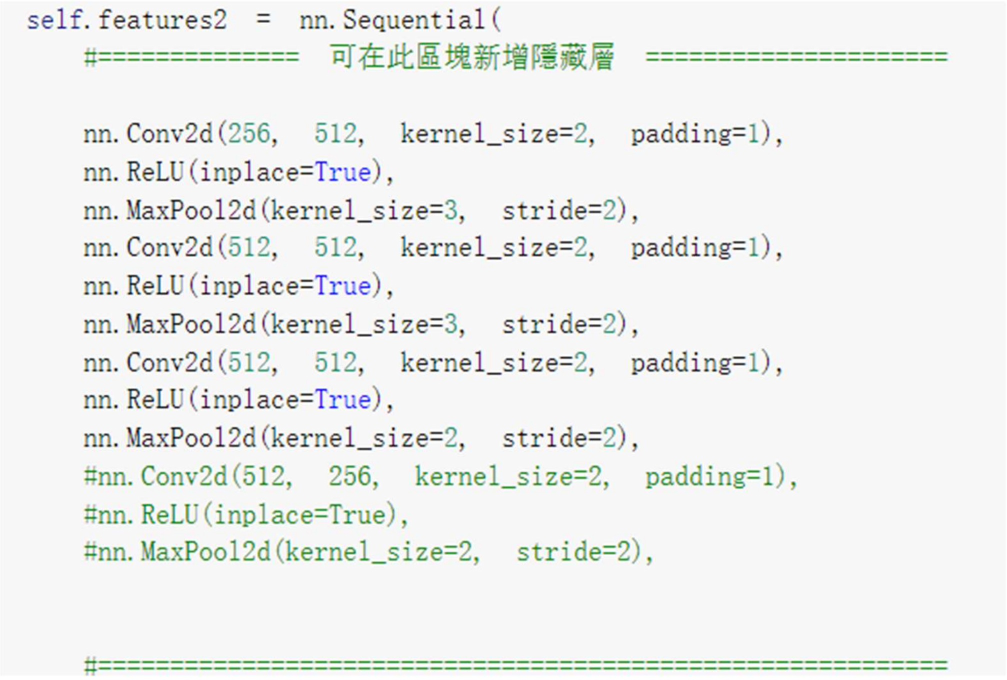
```
----------
100%|██████████| 24/24 [02:27<00:00,  6.14s/it]
train Loss: 3.0922 Acc: 0.2642
100%|██████████| 11/11 [00:23<00:00,  2.18s/it]
val Loss: 4.8395 Acc: 0.0639
Epoch 20/20
----------
100%|██████████| 24/24 [02:27<00:00,  6.16s/it]
train Loss: 3.0693 Acc: 0.2857
100%|██████████| 11/11 [00:24<00:00,  2.19s/it]
val Loss: 4.8429 Acc: 0.0654
Training complete in 57m 37s
Best val Acc: 0.065449
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 1296x648 with 0 Axes>
```

目的：隨機修改圖片的亮度、對比度和飽和度，常用來進行資料增強，尤其是訓練圖像類別不均衡或圖像數量較少時。
結果：對比使用方法有很大負影響。

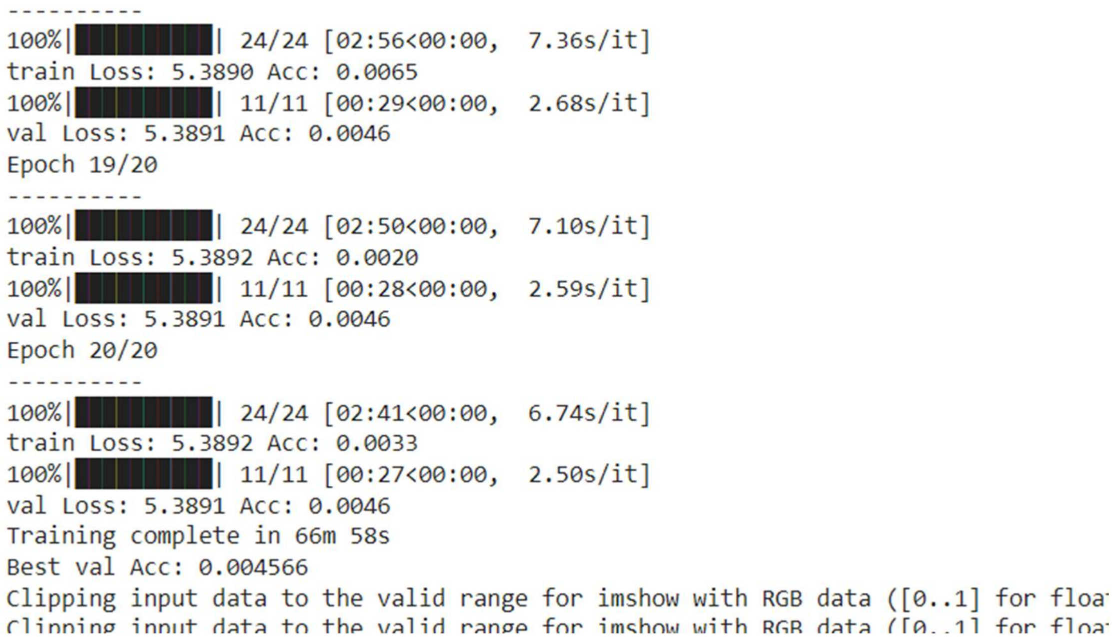
綜上對比，**Method1** 與 **Method6** 組合的準確率最高，為 **0.621005**，但為了簡化複雜步驟，選擇 **Method1**（**Resize+totensor+normalize**）為最佳組合。

2.

Method 1:升維至 **512** 維再進行卷積與最大池化



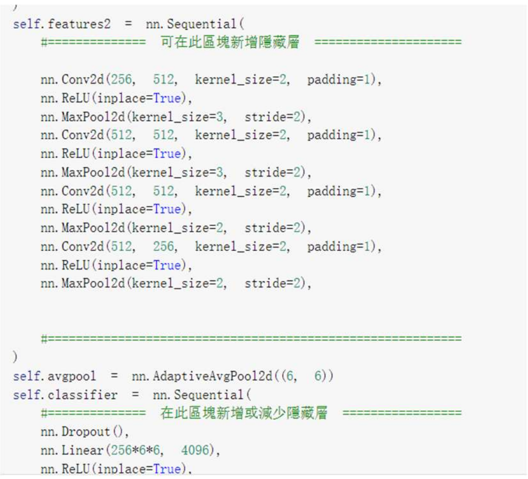
```
self.features2 = nn.Sequential(
    #==============  可在此區塊新增隱藏層  =====================

    nn.Conv2d(256, 512, kernel_size=2, padding=1),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2),
    nn.Conv2d(512, 512, kernel_size=2, padding=1),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2),
    nn.Conv2d(512, 512, kernel_size=2, padding=1),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=2, stride=2),
    #nn.Conv2d(512, 256, kernel_size=2, padding=1),
    #nn.ReLU(inplace=True),
    #nn.MaxPool2d(kernel_size=2, stride=2),


    #========================================================
```



```
----------
100%|██████████| 24/24 [02:56<00:00,  7.36s/it]
train Loss: 5.3890 Acc: 0.0065
100%|██████████| 11/11 [00:29<00:00,  2.68s/it]
val Loss: 5.3891 Acc: 0.0046
Epoch 19/20
----------
100%|██████████| 24/24 [02:50<00:00,  7.10s/it]
train Loss: 5.3892 Acc: 0.0020
100%|██████████| 11/11 [00:28<00:00,  2.59s/it]
val Loss: 5.3891 Acc: 0.0046
Epoch 20/20
----------
100%|██████████| 24/24 [02:41<00:00,  6.74s/it]
train Loss: 5.3892 Acc: 0.0033
100%|██████████| 11/11 [00:27<00:00,  2.50s/it]
val Loss: 5.3891 Acc: 0.0046
Training complete in 66m 58s
Best val Acc: 0.004566
Clipping input data to the valid range for imshow with RGB data ([0..1] for floa
Clipping input data to the valid range for imshow with RGB data ([0..1] for floa
```

由於升維過高，導致 noise 增加，同時因為 output channel 為 512 的關係，沒有使用預處理的 train 結果，因此準確率極低。

Method2：增加卷積層，先升維 512，再降維至 265，

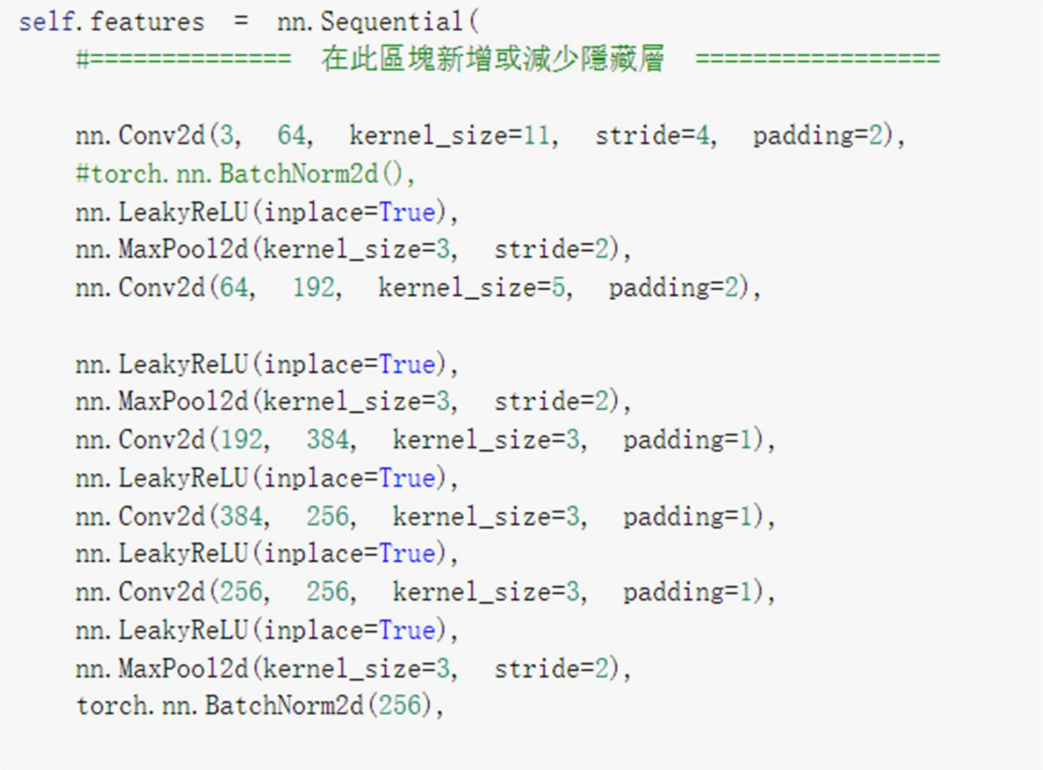


```
)
self.features2  =  nn.Sequential(
    #==============  可在此區塊新增隱藏層  ====================

    nn.Conv2d(256,  512,  kernel_size=2,  padding=1),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3,  stride=2),
    nn.Conv2d(512,  512,  kernel_size=2,  padding=1),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3,  stride=2),
    nn.Conv2d(512,  512,  kernel_size=2,  padding=1),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=2,  stride=2),
    nn.Conv2d(512,  256,  kernel_size=2,  padding=1),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=2,  stride=2),



    #=========================================================
)
self.avgpool  =  nn.AdaptiveAvgPool2d((6,  6))
self.classifier  =  nn.Sequential(
    #==============  在此區塊新增或減少隱藏層  ================
    nn.Dropout(),
    nn.Linear(256*6*6,  4096),
    nn.ReLU(inplace=True),
```

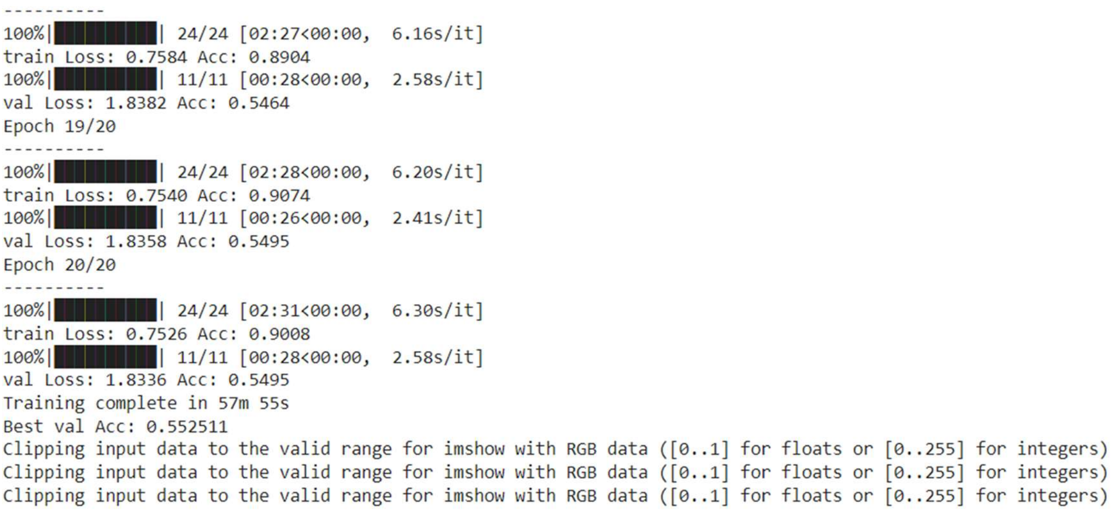
效果很差，val accuracy 在 0.0030 與 0.0046 間震盪

Method3 ：將 feature 中的 Relu 改為 LeakyRelu, 並 normalize 卷積層。

目的：當不啟動時，**LeakyReLU** 仍然會有非零輸出值，從而獲得一個小梯度，避免 **ReLU** 可能出現的神經元“死亡”現象。



```
self.features = nn.Sequential(
    #===============  在此區塊新增或減少隱藏層  =================

    nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
    #torch.nn.BatchNorm2d(),
    nn.LeakyReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2),
    nn.Conv2d(64, 192, kernel_size=5, padding=2),

    nn.LeakyReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2),
    nn.Conv2d(192, 384, kernel_size=3, padding=1),
    nn.LeakyReLU(inplace=True),
    nn.Conv2d(384, 256, kernel_size=3, padding=1),
    nn.LeakyReLU(inplace=True),
    nn.Conv2d(256, 256, kernel_size=3, padding=1),
    nn.LeakyReLU(inplace=True),
    nn.MaxPool2d(kernel_size=3, stride=2),
    torch.nn.BatchNorm2d(256),
```



```
----------
100%|██████████| 24/24 [02:27<00:00,  6.16s/it]
train Loss: 0.7584 Acc: 0.8904
100%|██████████| 11/11 [00:28<00:00,  2.58s/it]
val Loss: 1.8382 Acc: 0.5464
Epoch 19/20
----------
100%|██████████| 24/24 [02:28<00:00,  6.20s/it]
train Loss: 0.7540 Acc: 0.9074
100%|██████████| 11/11 [00:26<00:00,  2.41s/it]
val Loss: 1.8358 Acc: 0.5495
Epoch 20/20
----------
100%|██████████| 24/24 [02:31<00:00,  6.30s/it]
train Loss: 0.7526 Acc: 0.9008
100%|██████████| 11/11 [00:28<00:00,  2.58s/it]
val Loss: 1.8336 Acc: 0.5495
Training complete in 57m 55s
Best val Acc: 0.552511
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers)
```

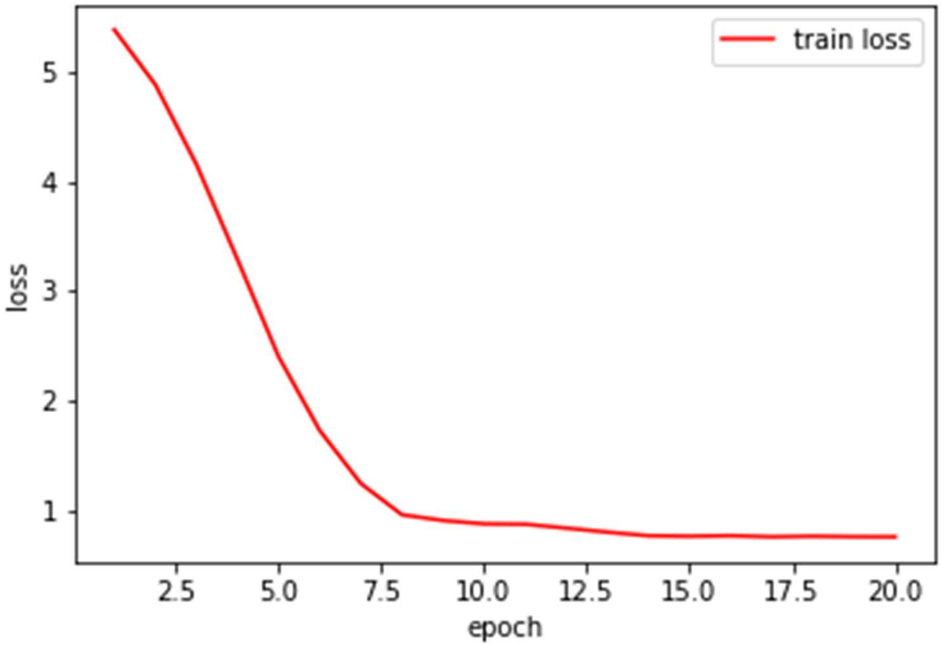
相比 samplecode accuracy 有所下降，但是三种方法最好的。

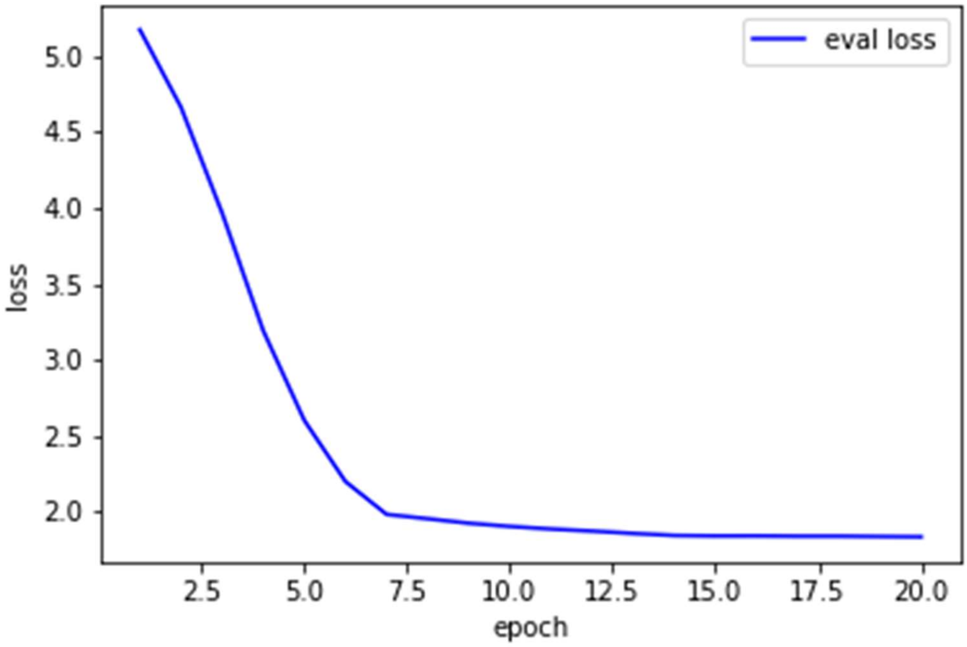
因此：最佳效果是 method3

Lr:0.001

Epoch:25

Batch_size:64


train loss
loss
5
4
3
2
1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
epoch


eval loss
loss
5.0
4.5
4.0
3.5
3.0
2.5
2.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
epoch