Model2:

Data augmentation:

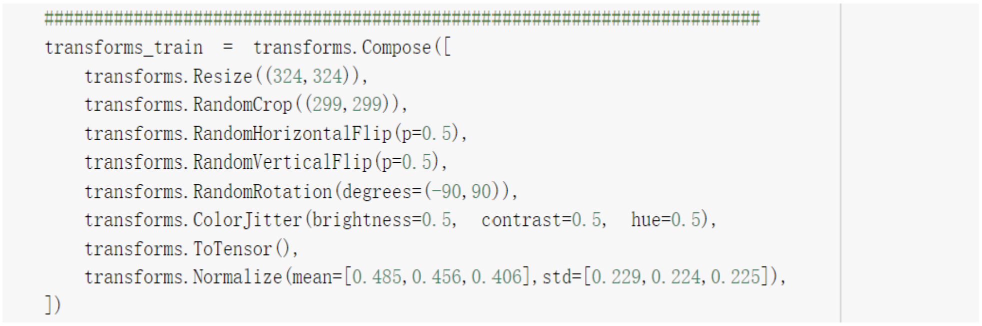
上一個 KAGGLE 分數為 0.86 的時候沒有使用 ColorJitter 來強化資料，加了之後再 train，將訓練 epoch 增加

Epoch= 80+80+20

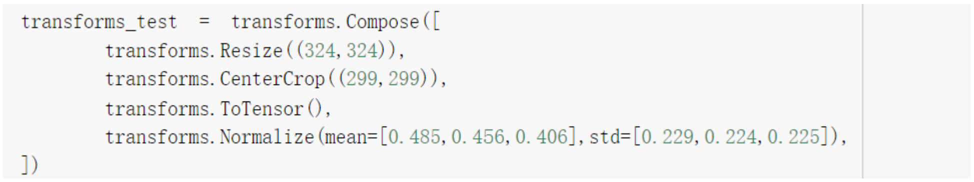
第一次 80epoch: learning rate 設置為 1e-3

第二個 80epoch: learning rate  設置為 1e-4

第三個 20epoch: learning rate  設置為 1e-5

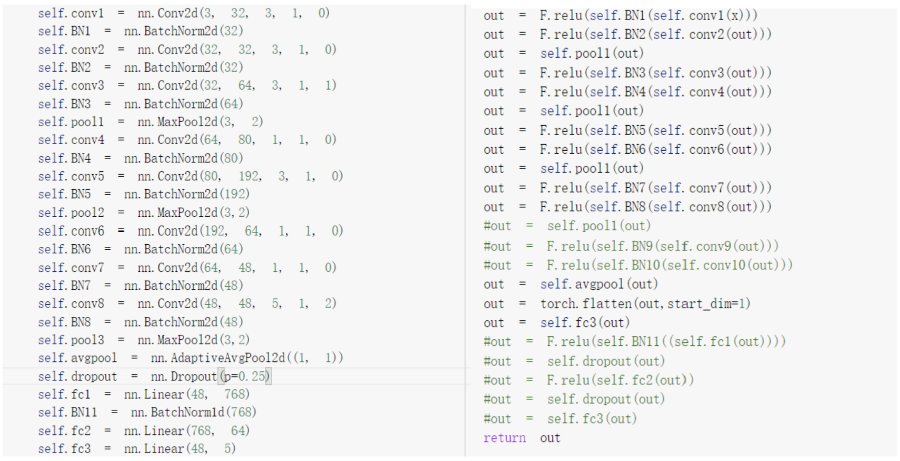


```
################################################################################
transforms_train  =  transforms.Compose([
    transforms.Resize((324,324)),
    transforms.RandomCrop((299,299)),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomVerticalFlip(p=0.5),
    transforms.RandomRotation(degrees=(-90,90)),
    transforms.ColorJitter(brightness=0.5,  contrast=0.5,  hue=0.5),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485,0.456,0.406],std=[0.229,0.224,0.225]),
])
```



```
transforms_test  =  transforms.Compose([
        transforms.Resize((324,324)),
        transforms.CenterCrop((299,299)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485,0.456,0.406],std=[0.229,0.224,0.225]),
])
```

其他的部分與上一個 pdf 檔一模一樣



```
self.conv1  =  nn.Conv2d(3,  32,  3,  1,  0)
self.BN1  =  nn.BatchNorm2d(32)
self.conv2  =  nn.Conv2d(32,  32,  3,  1,  0)
self.BN2  =  nn.BatchNorm2d(32)
self.conv3  =  nn.Conv2d(32,  64,  3,  1,  1)
self.BN3  =  nn.BatchNorm2d(64)
self.pool1  =  nn.MaxPool2d(3,  2)
self.conv4  =  nn.Conv2d(64,  80,  1,  1,  0)
self.BN4  =  nn.BatchNorm2d(80)
self.conv5  =  nn.Conv2d(80,  192,  3,  1,  0)
self.BN5  =  nn.BatchNorm2d(192)
self.pool2  =  nn.MaxPool2d(3,2)
self.conv6  =  nn.Conv2d(192,  64,  1,  1,  0)
self.BN6  =  nn.BatchNorm2d(64)
self.conv7  =  nn.Conv2d(64,  48,  1,  1,  0)
self.BN7  =  nn.BatchNorm2d(48)
self.conv8  =  nn.Conv2d(48,  48,  5,  1,  2)
self.BN8  =  nn.BatchNorm2d(48)
self.pool3  =  nn.MaxPool2d(3,2)
self.avgpool  =  nn.AdaptiveAvgPool2d((1,  1))
self.dropout  =  nn.Dropout(p=0.25)
self.fc1  =  nn.Linear(48,  768)
self.BN11  =  nn.BatchNorm1d(768)
self.fc2  =  nn.Linear(768,  64)
self.fc3  =  nn.Linear(48,  5)
```

```
out  =  F.relu(self.BN1(self.conv1(x)))
out  =  F.relu(self.BN2(self.conv2(out)))
out  =  self.pool1(out)
out  =  F.relu(self.BN3(self.conv3(out)))
out  =  F.relu(self.BN4(self.conv4(out)))
out  =  self.pool1(out)
out  =  F.relu(self.BN5(self.conv5(out)))
out  =  F.relu(self.BN6(self.conv6(out)))
out  =  self.pool1(out)
out  =  F.relu(self.BN7(self.conv7(out)))
out  =  F.relu(self.BN8(self.conv8(out)))
#out  =  self.pool1(out)
#out  =  F.relu(self.BN9(self.conv9(out)))
#out  =  F.relu(self.BN10(self.conv10(out)))
out  =  self.avgpool(out)
out  =  torch.flatten(out,start_dim=1)
out  =  self.fc3(out)
#out  =  F.relu(self.BN11((self.fc1(out))))
#out  =  self.dropout(out)
#out  =  F.relu(self.fc2(out))
#out  =  self.dropout(out)
#out  =  self.fc3(out)
return  out
```

最後 10 個 epoch:

```
==================== Epoch 10 ====================
Train Acc: 0.908670 Train Loss: 0.261129
  Val Acc: 0.845886   Val Loss: 0.509186
==================== Epoch 12 ====================
Train Acc: 0.904368 Train Loss: 0.270537
  Val Acc: 0.837775   Val Loss: 0.506230
==================== Epoch 14 ====================
Train Acc: 0.908008 Train Loss: 0.264551
  Val Acc: 0.841251   Val Loss: 0.501526
==================== Epoch 16 ====================
Train Acc: 0.914295 Train Loss: 0.255816
  Val Acc: 0.842410   Val Loss: 0.504503
==================== Epoch 18 ====================
Train Acc: 0.908339 Train Loss: 0.267466
  Val Acc: 0.843569   Val Loss: 0.501787
==================== Epoch 20 ====================
Train Acc: 0.904699 Train Loss: 0.260712
  Val Acc: 0.841251   Val Loss: 0.507455
```