學號： B06901180　 系級： 電機三　 姓名：鄭謹譯

1. 請描述你實作的模型架構、方法以及 accuracy 為何。其中你的方法必須為 domain adversarial training 系列 (就是你的方法必須要讓輸入 training data & testing data 後的某一層輸出 domain 要相近)。(2%)

先做Preprocessing，額外加上

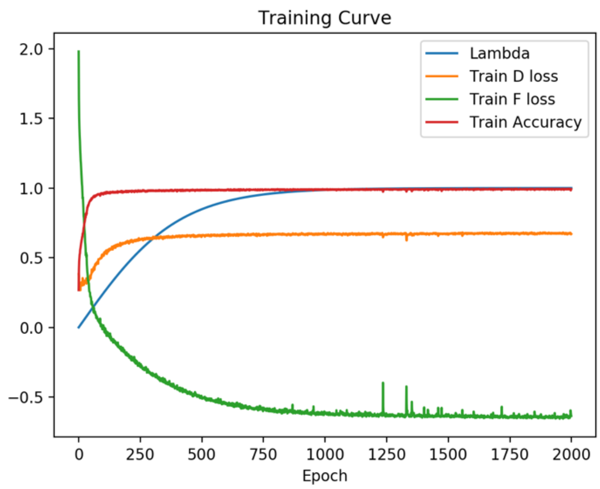
transforms.ColorJitter(brightness=(0.5),contrast=(0.5),saturation=(0.5))

模型架構分為三種，一是如下圖，二是FE為VGG16，三為Sample Code(即下圖去除最後四層Conv(512,512)

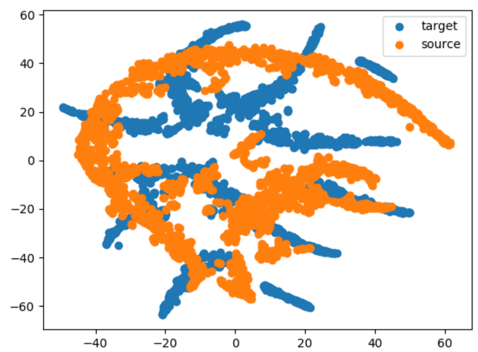
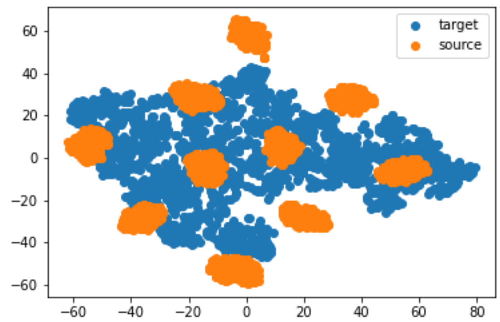
```
FeatureExtractor(
  (conv): Sequential(
    (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU()
    (15): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (16): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (18): ReLU()
    (19): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (20): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (21): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (22): ReLU()
    (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (24): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (25): ReLU()
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (28): ReLU()
    (29): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (30): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (31): ReLU()
  )
)
LabelPredictor(
  (layer): Sequential(
    (0): Linear(in_features=512, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): Linear(in_features=512, out_features=10, bias=True)
  )
)
DomainClassifier(
  (layer): Sequential(
    (0): Linear(in_features=512, out_features=512, bias=True)
    (1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=512, out_features=512, bias=True)
    (4): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Linear(in_features=512, out_features=512, bias=True)
    (7): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): Linear(in_features=512, out_features=512, bias=True)
    (10): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU()
    (12): Linear(in_features=512, out_features=1, bias=True)
  )
)
```

另外調整 $\lambda = \frac{2}{1+e^{-10\frac{current\ epoch}{total\ epoch}}} + 1$

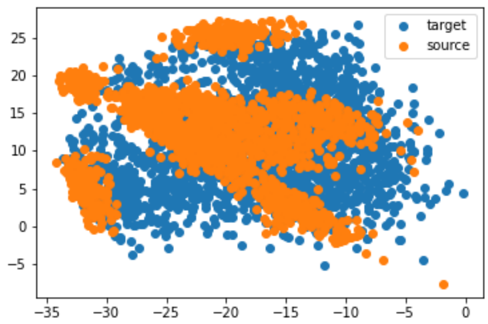
結果發現第一種模型和VGG16的模型在total epoch為200-300時Acc皆優於第三種模型，前二種在69-71間，而第三種僅僅67而已，但當把training epoch調高至2000後反而原始第三種模型可以達到76，而前兩種約在73-74附近，因此最後採用第三種模型，Training Curve如下圖。



2. **請視覺化真實圖片以及手繪圖片通過沒有使用 domain adversarial training 的 feature extractor 的 domain 分布圖。(2%)**

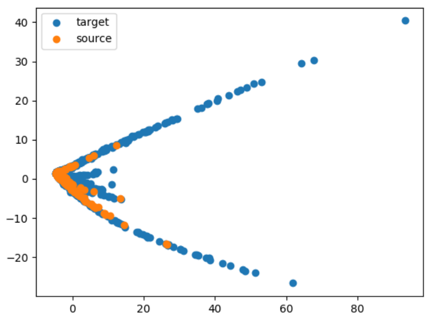
失敗組（疊太深，30層）	成功組 Extractor VGG16
	

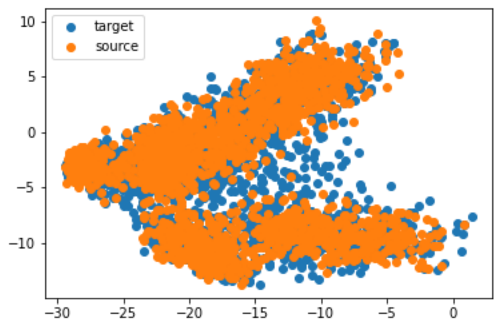
此為TSNE的結果 可發現左邊的source分群不明顯，右邊的結果清楚可見。



上圖為成功組的 PCA

3. **請視覺化真實圖片以及手繪圖片通過有使用 domain adversarial training 的 feature extractor 的 domain 分布圖。(2%)**





左圖為上圖失敗組的PCA，右圖為成功組的PCA，可以發現即使失敗組Source和Target分布的點也都蠻接近，成功組更幾乎完全重合，確實 domain adversarial 能讓兩個不同domain有效重合。