學號：b06901153 系級：電機三 姓名：林瑩昇

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 distribution 要相近)。(2%)

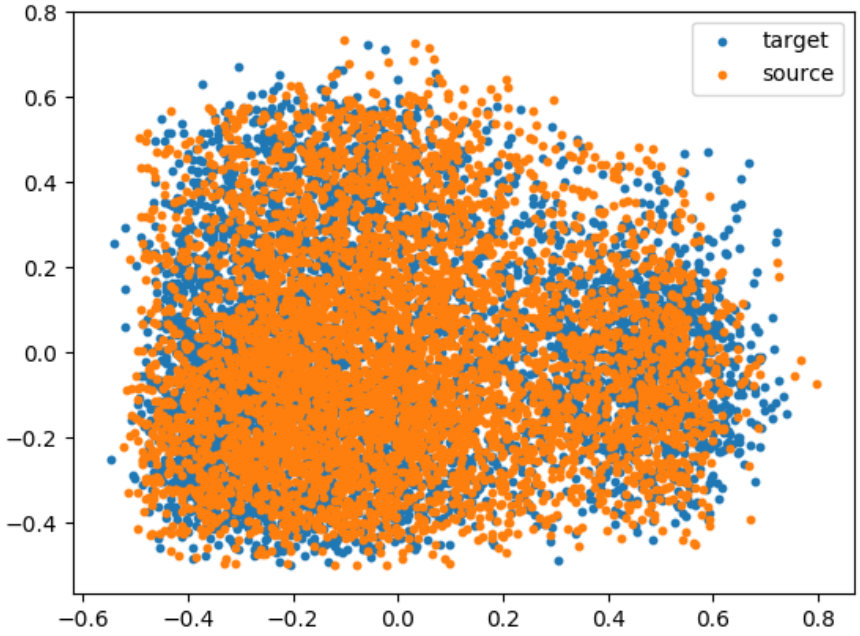
   (1) 我看 sample code 的時候就覺得 feature extractor的部分有點簡陋，跟在 hw3 時一樣，我記得那時候的model不足以把圖片中的資訊提出來，所以我就把feature extractor改成resnet式的model，相同channel數的會疊兩層再放maxpool，後來testing正確率從0.51上升至0.66

   (2) 後來又發現，discriminator與feature extractor的loss都很快就收斂了，就想到可以試試看用SGD來收斂至更好的解，所以我把optimizer都改成SGD，lr=0.01，epoch=400，testing正確率從0.66上升至0.73。

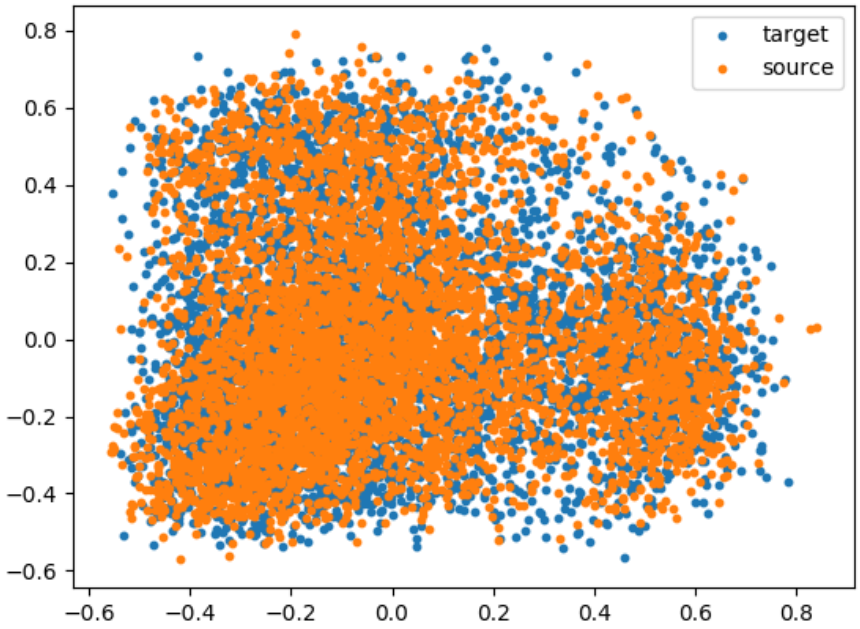
   (3) Feature extractor structure

```
self.conv = nn.Sequential(
    nn.Conv2d(1, 64, 3, 1, 1),
    nn.BatchNorm2d(64),
    nn.ReLU(),
    nn.Conv2d(64, 64, 3, 1, 1),
    nn.BatchNorm2d(64),
    nn.ReLU(),
    nn.MaxPool2d(2),

    nn.Conv2d(64, 128, 3, 1, 1),
    nn.BatchNorm2d(128),
    nn.ReLU(),
    nn.Conv2d(128, 128, 3, 1, 1),
    nn.BatchNorm2d(128),
    nn.ReLU(),
    nn.MaxPool2d(2),

    nn.Conv2d(128, 256, 3, 1, 1),
    nn.BatchNorm2d(256),
    nn.ReLU(),
    nn.Conv2d(256, 256, 3, 1, 1),
    nn.BatchNorm2d(256),
    nn.ReLU(),
    nn.MaxPool2d(2),

    nn.Conv2d(256, 256, 3, 1, 1),
    nn.BatchNorm2d(256),
    nn.ReLU(),
    nn.Conv2d(256, 256, 3, 1, 1),
    nn.BatchNorm2d(256),
    nn.ReLU(),
    nn.MaxPool2d(2),

    nn.Conv2d(256, 512, 3, 1, 1),
    nn.BatchNorm2d(512),
    nn.ReLU(),
    nn.Conv2d(512, 512, 3, 1, 1),
    nn.BatchNorm2d(512),
    nn.ReLU(),
    nn.MaxPool2d(2)
)
```

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%)



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%)



我原先預期兩個會有明顯的不同，但我用PCA將原先從feature extractor出來的512為向量降成2維後，畫出來卻沒有太大的不同。我就想說該不會我只train feature extractor出來的model與我用DaNN的train出來差不多吧，所以我在testing的時候用沒有用DaNN的feature extractor，出來的testing正確率掉到0.19，也就可以排除兩個model差不多的問題。所以我猜可能是因為兩個512維的vector雖然不太一樣，但經過降維之後(我總共用5000個data)剛好分布疊在一起，所以才會看起來差不多。

```python
def Into2dim(long_vec):
    # First Dimension Reduction
    transformer = KernelPCA(n_components=2, kernel='rbf', n_jobs=-1, random_state=0)
    kpca = transformer.fit_transform(long_vec)

    return kpca

# def plot_scatter(feat, label, savefig=None):
def plot_scatter(feat_source, feat_target, savefig=None):
    X_source = feat_source[:, 0]
    Y_source = feat_source[:, 1]
    X_target = feat_target[:, 0]
    Y_target = feat_target[:, 1]

    plt.scatter(X_target, Y_target, s=10)
    plt.scatter(X_source, Y_source, s=10)
    # plt.scatter(X, Y, c = label)
    plt.legend(['target', 'source'])
    if savefig is not None:
        plt.savefig(savefig)
    plt.show()
    return
```