Machine Learning Spring 2020 - HW12 Report

學號: b07902064 系級: 資工二 姓名: 蔡銘軒

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%)

   我所使用的 feature extractor 結構如下:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	640
PReLU-2	[-1, 64, 32, 32]	64
BatchNorm2d-3	[-1, 64, 32, 32]	128
Conv2d-4	[-1, 64, 16, 16]	36,928
Conv2d-5	[-1, 128, 16, 16]	73,856
PReLU-6	[-1, 128, 16, 16]	128
BatchNorm2d-7	[-1, 128, 16, 16]	256
Conv2d-8	[-1, 128, 8, 8]	147,584
Conv2d-9	[-1, 256, 8, 8]	295,168
PReLU-10	[-1, 256, 8, 8]	256
BatchNorm2d-11	[-1, 256, 8, 8]	512
Conv2d-12	[-1, 256, 4, 4]	590,080
Conv2d-13	[-1, 512, 4, 4]	1,180,160
PReLU-14	[-1, 512, 4, 4]	512
BatchNorm2d-15	[-1, 512, 4, 4]	1,024
Conv2d-16	[-1, 512, 2, 2]	2,359,808
Conv2d-17	[-1, 512, 2, 2]	2,359,808
PReLU-18	[-1, 512, 2, 2]	512
BatchNorm2d-19	[-1, 512, 2, 2]	1,024
Conv2d-20	[-1, 512, 1, 1]	2,359,808

   使用了類似 CNN 的架構，filter 的數量從64增加至512，並且將常用的 ReLU 用 PReLU 取代，讓 model 可以學習當 $x < 0$時要使用的斜率。在架構上，我偏好 Conv2d $\rightarrow$ ReLU $\rightarrow$ BatchNorm2d，而不是比較常見的 Conv2d $\rightarrow$ BatchNorm2d $\rightarrow$ ReLU。我認為將 ReLU 放在 BatchNorm 後面會破壞 BatchNorm 輸出的標準化的數字；而若先 ReLU 再 BatchNorm 可以讓下一層的 Conv 收到的輸入是經過標準化的。

   另外做的嘗試是將 MaxPool2d 改為使用 stride = 2 的 Conv2d 層取代。我認為比起 MaxPool2d 這樣固定的取樣，使用 stride = 2 的 Conv2d，可以讓 model 也學習在縮小圖片時，可以截取哪些特徵。不過因為 Conv2d 是有參數要訓練的，不像 MaxPool2d 只是做單純的運算，所以我在訓練時花的時間比使用 MaxPool2d 顯著增加了不少。

   我使用的 label predictor 架構如下:

Layer (type)	Output Shape	Param #
Linear-1	[-1, 512]	262,656
PReLU-2	[-1, 512]	1
Linear-3	[-1, 512]	262,656
PReLU-4	[-1, 512]	1
Linear-5	[-1, 10]	5,130

沿用了 Sample code 的架構，直接使用 Linear 搭配 PReLU 疊了三層來預測 label，跟一般 CNN model 裡的 FC 層架構類似。

我使用的 domain classifier 架構如下:

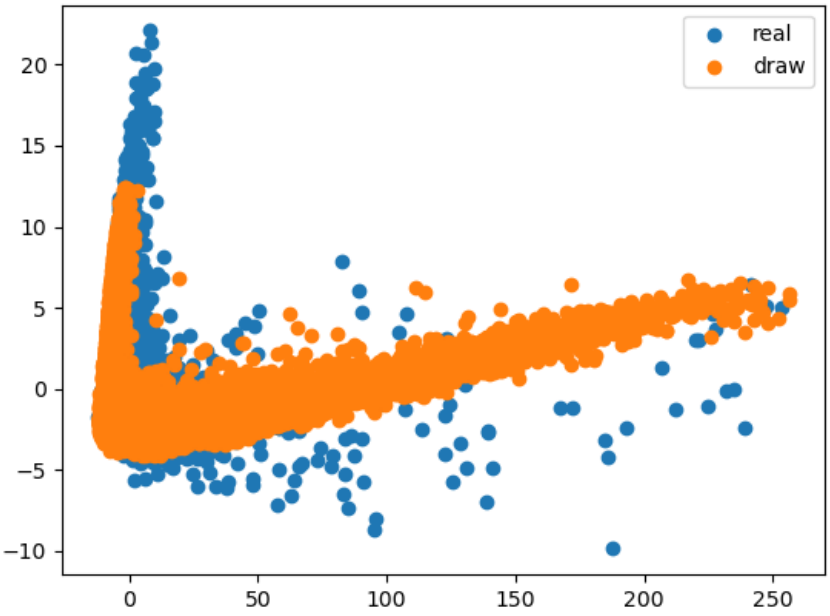
Layer (type)	Output Shape	Param #
Linear-1	[-1, 512]	262,656
BatchNorm1d-2	[-1, 512]	1,024
PReLU-3	[-1, 512]	1
Linear-4	[-1, 512]	262,656
BatchNorm1d-5	[-1, 512]	1,024
PReLU-6	[-1, 512]	1
Linear-7	[-1, 512]	262,656
BatchNorm1d-8	[-1, 512]	1,024
PReLU-9	[-1, 512]	1
Linear-10	[-1, 512]	262,656
BatchNorm1d-11	[-1, 512]	1,024
PReLU-12	[-1, 512]	1
Linear-13	[-1, 1]	513

這邊也直接沿用 Sample code 的架構，用 Linear $\rightarrow$ BatchNorm1d $\rightarrow$ PReLU 的方式疊了5層。

訓練時，使用與 Sample code 同樣的方式，是在一個 epoch 裡先訓練 domain classifier，接著訓練 domain classifier、label predictor 以及 feature extractor。所做的更改第一個是將 epoch 調高至2000個 epoch，並且使用了 adaptive lambda。比起使用一個固定的數字，我參考了 Sample code 裡提到的 paper，使用了 $\lambda = \frac{2}{1+e^{-10p}} - 1$，其中 p 是當前訓練進度，即當前 epoch 除以 epoch 總數。所以 λ 會從0慢慢趨近到1。

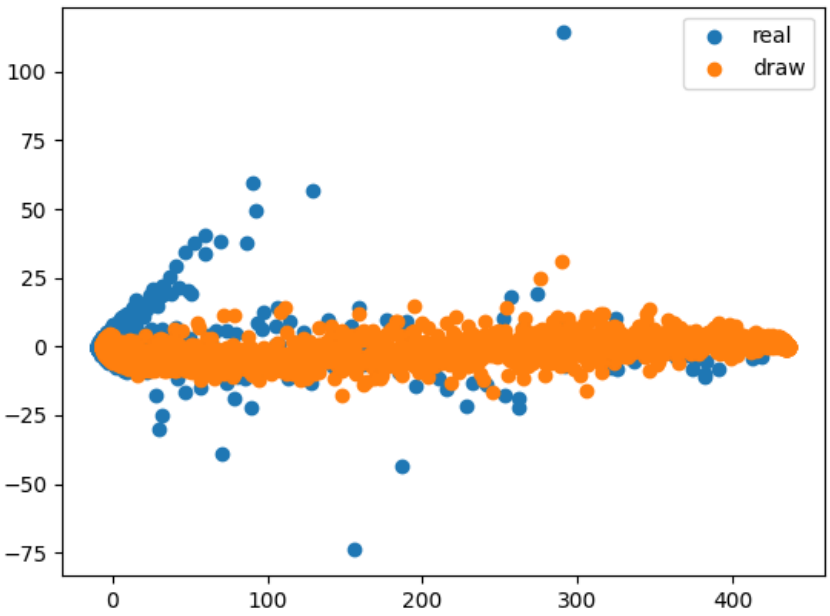
使用了以上的架構與訓練方法，在 training data 上取得0.9862的正確率，而在 kaggle publich score 為 0.77440。另外值得一提的是，在 feature extractor 裡使用 MaxPool2d時，在 training data 上得到的正確率約為 0.992，但在 kaggle 上的 public score 則略低，為0.76598。

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%)



這邊經過 feature extractor 後，直接使用 PCA 降至二維做圖。可以觀察到真實圖片跟手繪圖片的分佈有比較明顯的差異，尤其是圖片左半部。

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%)



同樣的將 feature extractor 的輸出直接用 PCA 降至二維。可以觀察到跟前一題的圖片比起來，兩者的分佈有更大的重疊，形狀也更為類似。說明了加入 domain adversarial training 後，feature extractor 確實能將兩類的圖片分類到更相似的 distribution上。