Machine learning HW3 b02901014 王崇勳

1. Supervised learning

a. 此部分使用 CNN 的方法做 training。Filter 的數量為 64-64-128-128-256-256，最後再通過一個 fully connected 的 layer，neuron 數為 512-10。Ativation 除了最後輸出時為 softmax，其他都是用 relu。Optimizer 使用 Adam，learning rate = 0.001。

b. 一開始只使用 5000 筆 labeled data，4500 筆作為 training data，500 筆作為 validation。在 train 40 epochs 可以再 kaggle 上面可以得到的分數為 0.47500。

c. 因為 labeled data 數量太少，因此使用 keras 裡面的 ImageDataGenerator 來擴增 training data。調整的參數有 width_shift_range=0.2，height_shift_range=0.2，horizontal_flip=Ture。接著利用 model.fit_generator()裡面的參數”samples_per_epoch”調整每個 epoch 需要產生出來的 training data。如此 40 epochs 在 kaggle 上得到的分數為 0.67040。

2. Semi-supervised learning (1) — self-training:

a. 此部分沿用 Supervised learning 的 CNN 架構。接著利用 label data train 出來的 model 去 label unlabeled data. 我使用 model.predict_proba()得到每一筆 unlabel data 屬於每一個 class 的機率，只有機率大於 0.99 的會被加入 training data，然後重新建造一個 model，再用新的 training data 去 train 這個 model。

b. Training 的過程中有設定 early stop，如果 val_acc 有 15 個 epoch 沒有變動，就不會再繼續 train 下去。另外還有設定 checkpoint，會記下 training 過程中 val_acc 最高的 model。此 model 用於 predict 最後的 testing data。

c. 每次用 labeled + unlabeled data train 完的 model 會單獨用 labeled data 在 train 一次，這樣的作法是因為 predict 出來的 unlabeled data 的 class 可能會不準，所以再用 labeled data 再 train 一次，這樣稱為 retrain 一次，如此希望提高正確率。實作中總共重複此步驟 5 次，在 kaggle 上得到的分數為 0.77940。

3. Semi-supervise learning (2) — autoencoder:

此部分用 CNN 做 autoencoder，其架構如 Figure.3。

```
x = Convolution2D(256,3,3,activation='relu',border_mode='same')(inputImg)
x = MaxPooling2D((2,2))(x)
x = Convolution2D(128,3,3,activation='relu',border_mode='same')(x)
x = MaxPooling2D((2,2))(x)
x = Convolution2D(64,3,3,activation='relu',border_mode='same')(x)
encoded = MaxPooling2D((2,2))(x)

x = Convolution2D(64,3,3,activation='relu',border_mode='same')(encoded)
x = UpSampling2D((2,2))(x)
x = Convolution2D(128,3,3,activation='relu',border_mode='same')(x)
x = UpSampling2D((2,2))(x)
x = Convolution2D(256,3,3,activation='relu',border_mode='same')(x)
x = UpSampling2D((2,2))(x)
decoded = Convolution2D(3,3,3,activation='sigmoid', border_mode='same')(x)
```

Figure.3

a.首先將 4500 筆 label data 加上 45000 筆 unlabel data 作為 autoencoder 的 training data。接著建立一個新的 model = Model(inputImg, encoded)，取這個 model 的 output 當作圖片的 feature。這些 feature 再做為 NN (架構如 Figure.4)的 input。最後再使用 NN 去預測 10000 筆 testing data。

```
model = Sequential()
model.add(Dense(64,input_shape=(128,)))
model.add(Activation('relu'))
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dense(classNum))
model.add(Activation('softmax'))

adam = Adam(lr=0.001)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
```

Figure.4

4. Compare and analyze the results

a. Self-learning

在沒有使用 unlabeled data 之前，正確率已可達到 0.67040，但是加入 unlabeled data 做 self-training 以後發現正確率變低了。有可能是因為 predict unlabeled data 時不夠準確，因此讓 model 變差。後來使用 retrain 五次的方式，正確率可以達到 0.70120。Table.1 裡面用來測試的 model 都使用擴增後的 training data，不論是只有 labeled data 或是 labeled + unlabled data 所使用的 Samples_per_epoch 皆為 45000。

	Kaggle public	Samples_per_epoch
Supervised training	0.67040	45000
No retrain	0.64360	45000/45000
With retrain (5)	0.70120	45000/45000

Table.1

接著對 samples_per_epoch 作調整，發現將此參數調大，在 kaggle 上面能得到最好的正確率也跟著變大。結果如 Talble.2。

Samples_per_epoch	Kaggle public
45000/45000	0.70120
75000/60000	0.75560

Table.2

前面的方法都是使用同一個 model retrain 五次，不過這樣比較容易 overfit，因此嘗試在 predict 完 unlabeled data 以後 rebuild model，發現有 rebulid model 能得到的最高正確率與有些許提高。結果如 Table.3。

	Kaggle public	Samples_per_epoch
Without rebuilding	0.75560	75000/60000
Rebuild	0.77940	75000/60000

Table.3

b. Autoencoder

此部分嘗試使用兩種方法做分類，分別為 SVM 與 NN，其結果如 Table.4。NN 相對於 SVM 來說較適合做多個 class 的分類，然後兩者正確率都不高。原因可能為從 autoencoder 裡面取出的 feautre 不夠好。在 train autoencoder 時，輸出的圖片應該和輸入圖片越像越好，但是將輸出圖片畫出來看發現十分模糊，很難看出圖片的內容。如此，取的的 feature 也就不夠好。

	Kaggle public
SVM	0.23560
NN	0.37700

Table.4