Machine Learning HW3　　電機四 黃馨平 b02901039

1.Supervised learning:

method:

用 5000 筆 label data 以及 2D（3 層）的 CNN 來 train，分別使用 16、32、64、64、256 的 layer，其中有使用 activation 將結果控制在數值範圍內，以及使用 dropout 來避免 overfitting。我將 train 到 acc>0.9 的 model 當做方法 2.的判斷依據。

performance:

kaggle 的正確率為 0.59380。

發現只用 5000 筆 data 可以達到不會太差的 performance，非常令我意外。

code：

```
model.add(Convolution2D(16, 3, 3, input_shape=(3,32, 32)))
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.3))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(output_dim=256))
model.add(Activation('relu'))
model.add(Dropout(0.3))
model.add(Dense(output_dim=10))
model.add(Activation('softmax'))
```

2.Semi-supervised learning (1):

method：

將方法 1.中做出來的 acc>Y 的 model 當作一個初始判斷的依據。一直重複以下步驟來判斷 unlabel 是屬於哪一類（將他們變 label），直到 label 的數量超過 40000：

1.用上一個 train 好的 model 來判斷 unlabel 的種類和 confidence

2.如果一筆 unlabel 的 confidence>X，就將他加入 label 中

3.用相同 CNN 結構 train 一個新的 model，直到 acc>Y

如果已經有超過 40000 筆的 label，就不再加入新的 unlabel，繼續將他 train 下去。

performance：

當 Dropout=0.4，X=0.7，Y=0.9 時，kaggle 的正確率為 0.61140

當 Dropout=0.3，X=0.7，Y=0.9 時，kaggle 的正確率為 0.58820

當 Dropout=0.3，X=0.9，Y=0.9 時，kaggle 的正確率為 0.58280

當 Dropout=0.4，X=0.6，Y=0.9 時，kaggle 的正確率為 0.58680

可以發現同一種 method 調整不同參數出來的正確率還是都差不多。

然後當 Dropout 比較高，confidence>0.7 左右時，做出來的正確率最好。

code：

```
while unlabel 數目>10000 :
```

```
pre_temp=model.predict(data_unlabel)
for i in range(data_unlabel.shape[0]): //如果 confidence>0.7，就納入 label
    if np.max(pre_temp[i])>0.7:
        count_unlabel[i]=np.argmax(pre_temp[i]
更新新的 label 和 unlabel
建立新的 CNN 如同方法 1.
while score[1]<0.9: //train 到 acc>0.9
    model.fit(data_label,class_label,batch_size=100,nb_epoch=5)
keep training…
```

3.Semi-supervised learning (2):

method:

將 45000 筆 unlabel 直接根據 pixel intensity 分類。

看一筆 unlabel 和 5000 筆 label 中的哪一筆 mse 最小，就把他分為那一筆的種類。接著用和方法 1.相同結構的 CNN 來 train。

performance:

在測試的時候分別用了 autoencoder 和 pixel intensity 作為分類的依據，也分別用了兩種方法 a.看跟 label 中的哪一筆最近，就是那筆的種類 b.看跟每一類共 500 筆的 mse 和，哪一個最小，就是那一類。總結如下:

a. with pixel intensity: kaggle 的正確率為 0.31120
a. with autoencoder: kaggle 的正確率為 0.27800(可能 autoencoder train 的不夠久)
b. with pixel intensity: kaggle 的正確率為 0.18800
b. with autoencoder: kaggle 的正確率為 0.23660

結果發現方法 b.根本就不行，而方法 a.的準確率也很低。可見得可能要用 deep autoencoder 或是其他方法會比較好。

code：

```
for i in range(45000):
    for j in range(5000): //看跟誰的 pixel intesity 的 MSE 最小
        if rmse(x_train[j],x_test[i]) < temp:
            temp=rmse(x_train[j],x_test[i])
            c=j
    classes[i][int(c/500)]=1 //把 45000 筆 data 的每一筆直接分類
```

autoencoder:

```
encoding_dim = 256
input_img = Input(shape=(3072,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(3072, activation='sigmoid')(encoded)
autoencoder = Model(input=input_img, output=decoded)
encoder = Model(input=input_img, output=encoded)
encoded_input = Input(shape=(encoding_dim,))
decoder_layer = autoencoder.layers[-1]
decoder = Model(input=encoded_input, output=decoder_layer(encoded_input))
```

4.Compare and analyze your results

分析分別在 1.2.3.中有描述，總之，原本根據 5000 筆 label data 的結果還行。cluster 的方法不管是根據 pixel 還是 autoencoder，結果都不太好。比較稍微好的是重覆 predict 和加入可信賴的 unlabel data 這招，即方法 2.。可能還可以試試看用其他的 cluster 方法來做！或是可以試試看加入 unlabel data 時 weight by confidence。