MACHINE LEARNING HW3 REPORT

B02901065 電機四 李洺曦

(1)Supervised learning：

我的model主要有以下結構：

```
80 model.add(Convolution2D(20,3,3,input_shape=(3,32,32)))
81 model.add(Activation("relu"))
82 model.add(MaxPooling2D((2,2)))
83 model.add(Convolution2D(50,3,3))
84 model.add(Activation("relu"))
85 model.add(MaxPooling2D((2,2)))
86 model.add(Convolution2D(100,3,3))
87 model.add(Activation("relu"))
88 model.add(MaxPooling2D((2,2)))
89 model.add(Flatten())
90 model.add(BatchNormalization(epsilon=1e-05, mode=0, axis=-1, momentum=0.99, weights=None, be
   ta_init="zero", gamma_init="one", gamma_regularizer=None, beta_regularizer=None))
91 model.add(Dense(output_dim=200))
92 model.add(Activation("sigmoid"))
93 model.add(Dropout(0.1))
94
95 ao = Adam(lr=0.0001)
96 model.add(Dense(output_dim=10))
97 model.add(Activation("softmax"))
98 model.compile(loss="categorical_crossentropy",
99               optimizer=ao,
100              metrics=["accuracy"])
```

在Dense層的部份，之所以選用sigmoid作為activative function，是因為在套用relu時發現效果沒有sigmoid好。

我將model的fit function實作於fit_model這個function中，並且有以下參數控制整個training的行為：max_time定義了最大epoch數，當training累積到此epoch數時即停止，每做一次epoch，當我發現validation的loss小於目前所紀錄最小loss-0.05時，即將此model使用save_weights method存於磁碟中，反之，則累積count_non_decay變數的值，表示在最好的model之下，model被over-train了幾次，而control在大於等於0時，代表了earlystopping的patience，當累積值大於此時，即停止train。在control等於-1時，則單純使用max_time去終止training，當control等於-2時，則使用max_per變數限制單一次呼叫fit_model時將會執行的epoch數。

在此項目中，我以control=-1之模式，並將max_time設為200個epoch，sw設為5之下，image generator會自動依照條件產生出剩下四份data，train過之後得到了理想的結果，在kaggle上的分數為0.622與0.626。

(2)Semi-supervised learning - Self-training：

在Self-training中，我先使用supervised learning去train過一次model（sw=5, control=10）之後便將unlabeled的data使用此model去做prediction，做出來的prediction其與理想情況（一個最大值為1，其餘為0）的root-maean-square須小於0.1，且根據其最大值有機率的不會被選中，將符合上述兩條件的data冠上label後加入training_data中，並從原本的unlabeled data集合中除去，使用control=-2, sw=1, max_per=20之模式去train model，直到unlabeled data剩餘不到1000筆或者總epoch數超越max_time為止。validation accuracy約為0.528。

(3)Semi-supervised learning - Autoencoder：

在此法中，我建立了一個autoencoder，結構如下：

```
encoding_dim = 256
input_img = Input(shape=(3*32*32,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(3*32*32, activation='sigmoid')(encoded)
autoencoder = Model(input=input_img, output=decoded)
encoder = Model(input=input_img, output=encoded)
x_train = np.array(data[0:len(data)-1000]).astype("float32") / 255.
x_vali = np.array(data[len(data)-1000:len(data)]).astype("float32") / 255.
```

將其train好之後，使用encoder的部份將image data作encode，並且使用NearestCentroid這個演算法，透過fit function之後對於encoded的testing data進行預測。

(4)Compare and analyze your results：

可以觀察到self training在validation的表現上並不如supervised好，推測由於第一次的fit function在val_acc約莫為0.5時即達到飽和，有些新label的data其label並不正確，導致整體表現沒有太大幅度的提升。