HW2 MNIST

404410030 資工三 鄭光宇

### **環境設置：**

#### 使用python，並使用ipython notebook環境。

#### 使用sklearn套件，裡面有的Cross-Validation工具做交叉驗證、使用t-SNE對資料可視化。

#### 使用matplotlib來繪製圖表。

#### 使用keras來快速搭建CNN模型。

### **需安裝套件：**

#### ipython, jupyter, theano, numpy, sklearn, matplotlib, seaborn, keras等。

### **資料集：**

#### 使用這次作業指定的MNIST。

#### MNIST：

##### 28x28大小的手寫數字，共有10種數字（0~9）。

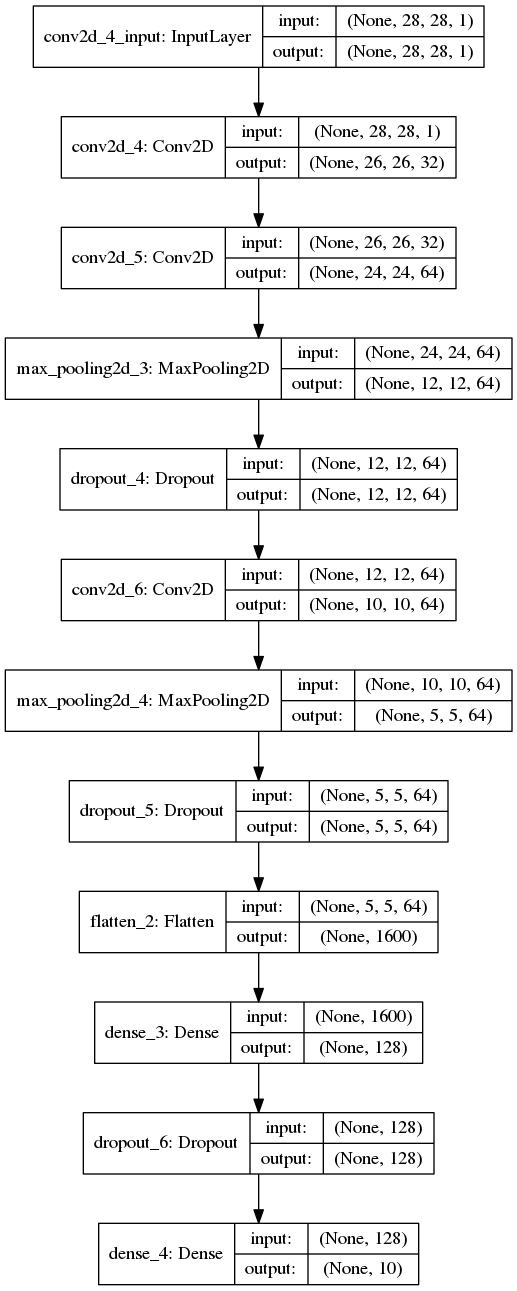
因為**keras**本身內建有**MNIST**資料集，所以沒有另外下載。

### **實驗結果：**

#### MNIST：

##### 我先將圖片數值scale到[0,1]的浮點數。

並使用如下CNN架構：



##### 與原始MNIST上CNN的LeNet-5較大不同之處，是我有使用**Dropout**，並多加了更多層Convolution->Pooling，在這樣的架構下，training set上的 5-fold Cross-Validation可以達到 **99.21±0.09%** 左右，**±0.09%** 是5-fold Cross-Validation的兩個標準差範圍。

輸出的activation function：softmax

選擇的Loss function：categorical\_crossentropy（多類別的對數損失）

Label Encoding：one-hot encoding

選擇的優化器：Adam (learning rate: 0.001)

##### **訓練集上測試得 10-fold Cross-Validation如下：**

|  |
| --- |
| 5-fold cross-validation |
| 99.17% |
| 99.10% |
| 99.24% |
| 99.37% |
| 99.16% |
| Accuracy: 99.21±0.09% |

##### 之後，在測試集上面驗證效果。

##### **測試集上的 Confusion Matrix如下：**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 978 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1130 | 0 | 1 | 0 | 2 | 0 | 2 | 0 | 0 |
| 2 | 1 | 0 | 1021 | 0 | 0 | 0 | 0 | 10 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1009 | 0 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 973 | 0 | 2 | 0 | 2 | 5 |
| 5 | 0 | 0 | 0 | 9 | 0 | 882 | 1 | 0 | 0 | 0 |
| 6 | 1 | 2 | 1 | 0 | 1 | 3 | 949 | 0 | 1 | 0 |
| 7 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 1024 | 0 | 1 |
| 8 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 969 | 1 |
| 9 | 1 | 0 | 0 | 2 | 3 | 1 | 0 | 1 | 1 | 1000 |

##### 對於每一個類別，效能評估基準如下表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| class | precision | recall | F1-score | support |
| 0 | 1.00 | 1.00 | 1.00 | 980 |
| 1 | 1.00 | 1.00 | 1.00 | 1135 |
| 2 | 1.00 | 0.99 | 0.99 | 1032 |
| 3 | 0.99 | 1.00 | 0.99 | 1010 |
| 4 | 1.00 | 0.99 | 0.99 | 982 |
| 5 | 0.99 | 0.99 | 0.99 | 892 |
| 6 | 1.00 | 0.99 | 0.99 | 958 |
| 7 | 0.99 | 1.00 | 0.99 | 1028 |
| 8 | 1.00 | 0.99 | 1.00 | 974 |
| 9 | 0.99 | 0.99 | 0.99 | 1009 |
| avg/total | 0.99 | 0.99 | 0.99 | 10000 |

##### 總結測試測試集的

##### Error rate：**0.65%**

##### Accuracy：**99.35%**

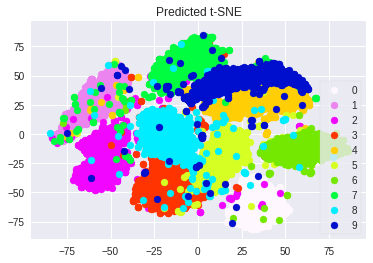
##### 可以看出效果非常好。

##### **資料可視化：**

###### 使用最近流行的t-SNE將資料降維、投影到2D平面上，使資料可視化。

###### 測試集上可視化結果如下：

###### 



第二張圖片是CNN預測的結果，可以看出預測結果與Ground Truth很接近，CNN很好地切開了各個類別。

**與其他方法的比較：**

與MNIST資料集網站上的「SVM, Gaussian Kernel」比較，  
我們使用CNN，可以得到0.65% 的error rate，而SVM是1.4%，可以說是有顯著的進步。

與MNIST資料及網站上的其他CNN相比，我們實作的方法好過大部分MNIST資料集網站上的CNN，得到很不錯的0.65% 錯誤率。效果較好的原因可能是因為使用比較晚期才出現的Dropout技巧，使得模型較不容易發生Overfitting。

### **結論：**

這次使用完CNN後，了解到它是個優秀的模型，在MNIST上能夠輕鬆得到99% 以上的正確率。

### **實作上遇過的問題：**

雖然調整資料輸入的batch size有助於幫助平行化，有更好的GPU使用率，  
不過過高的batch size有可能導致最後結果accuracy較低。

### **參考資料：**

#### 我的github：

#### <https://github.com/peter0749/Computer_Vision/tree/master/hw2_mnist_cnn>

#### PCA+SVM MNIST github:

#### <https://github.com/peter0749/Multimedia-Content-Analysis/tree/master/mnist>

#### Keras Documentation:

#### <https://keras.io/>