1. Introduction

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
%matplotlib inline


from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical # convert to one-hot-encodin
g
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau

sns.set(style='white', context='notebook', palette='deep')
```

2. Data preparation

2.1 Load data

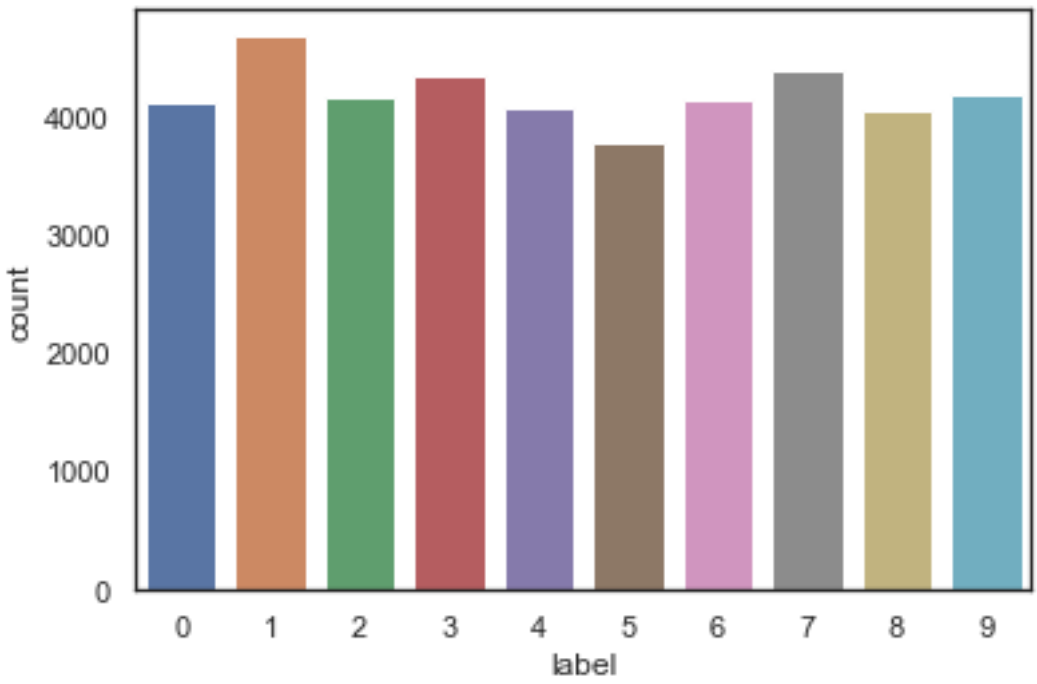
```
# >>>>>填写<<<< 利用pandas的load_csv函数，读取我们的train 和 test 数
据集合 变量已经给出 >>>>>填写<<<< ######
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")

# >>>>>填写<<<< 利用pandas的header选择，将label列传递给Y_train
 >>>>>填写<<<<
Y_train = train["label"]

# 因为train.csv中，第一列label在上述代码已经传递给Y_label，这里对于x_t
rain 我们不需要训练集的第一列 #####
X_train = train.drop(labels = ["label"],axis = 1)

# 释放内存
del train
```

```
g = sns.countplot(Y_train)
```

```
Y_train.value_counts()
```

Out[4]:

```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: label, dtype: int64
```



We have similar counts for the 10 digits.

2.2 Check for null and missing values

```
# 检查训练数据是否有空值
X_train.isnull().any().describe()
```

Out[24]:

```
count       784
unique        1
top       False
freq        784
```

```
dtype: object
```

```
# >>>>填写<<<< 检查训练数据是否有空值 >>>>填写<<<< ###
test.isnull().any().describe()
```

Out[25]:

```
count       784
unique        1
top       False
freq        784
dtype: object
```

I check for corrupted images (missing values inside).

There is no missing values in the train and test dataset. So we can safely go ahead.

2.3 Normalization

We perform a grayscale normalization to reduce the effect of illumination's differences.

Moreover the CNN converg faster on [0..1] data than on [0..255]. 标准化，将灰度值 0-255 映射到 0 - 1 区间

```
# Normalize the data
X_train = X_train / 255.0
###### >>>填写<<< 标准化测试集合 ########
test = test / 255.0
```

2.3 Reshape

```
# >>>>填写<<<<< 利用 reshape 函数， 将 X_train 变换成 (height = 28px, width = 28px , canal = 1)>>>>填写<<<<< ######
X_train = X_train.values.reshape(-1,28,28,1)
test = test.values.reshape(-1,28,28,1)
```

Train and test images (28px x 28px) has been stock into pandas.Dataframe as 1D vectors of 784 values. We reshape all data to 28x28x1 3D matrices.

Keras requires an extra dimension in the end which correspond to channels. MNIST images are gray scaled so it use only one channel. For RGB images, there is 3 channels, we would have reshaped 784px vectors to 28x28x3 3D matrices.

2.5 Label encoding

```
# 利用 0 1 编码 将 0-9 数字标签编码成 10 维向量 (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
Y_train = to_categorical(Y_train, num_classes = 10)
```

Labels are 10 digits numbers from 0 to 9. We need to encode these lables to one hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0]).

2.6 Split training and valdiation set

```
# Set the random seed
random_seed = 2
# 将训练集合按照9:1 分成训练集合 和验证集合 validation ####
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1, random_state=random_seed)
```

We can get a better sense for one of these examples by visualising the image and looking at the label.

```
# Some examples
g = plt.imshow(X_train[0][:,:,0])
```