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-encoding
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')
```

```
Using TensorFlow backend.
```

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")
#####train validation test(完全独立的，与训练过程无关的)
```

```
# >>>>>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_train 我们不需要训练集的第一列 #####
X_train = train.drop(labels = ["label"],axis = 1)
```

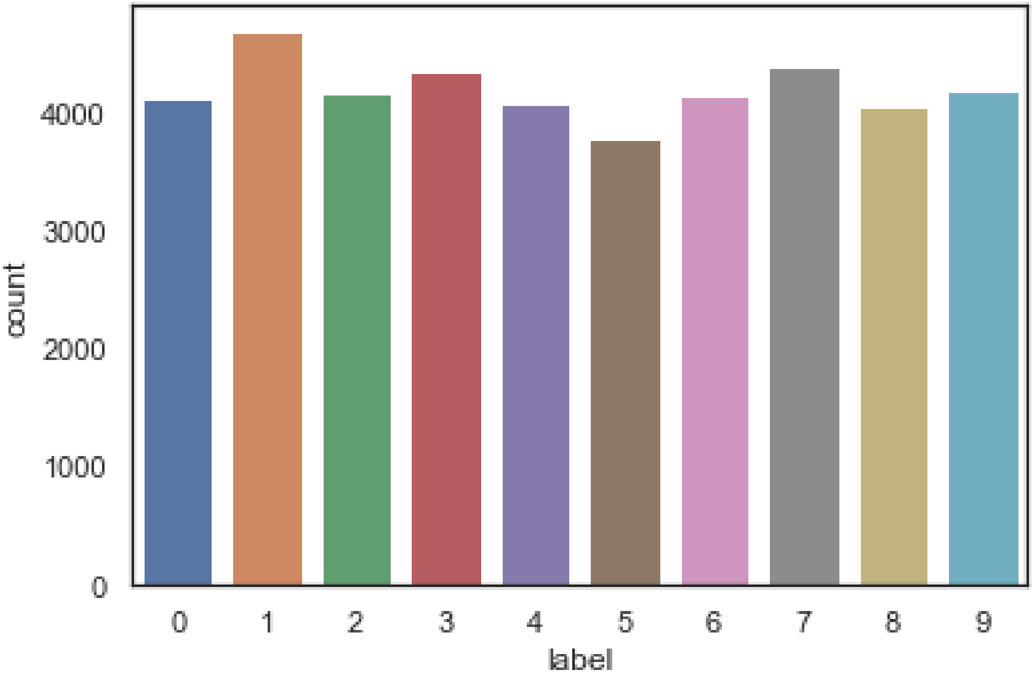
```
# 释放内存
del train

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)
0-255
```

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 : 9 -> [0,0,0,0,0,0,0,0,0,1])
```

```
##
Y_train = to_categorical(Y_train, num_classes = 10)
## one-hot encoding
```

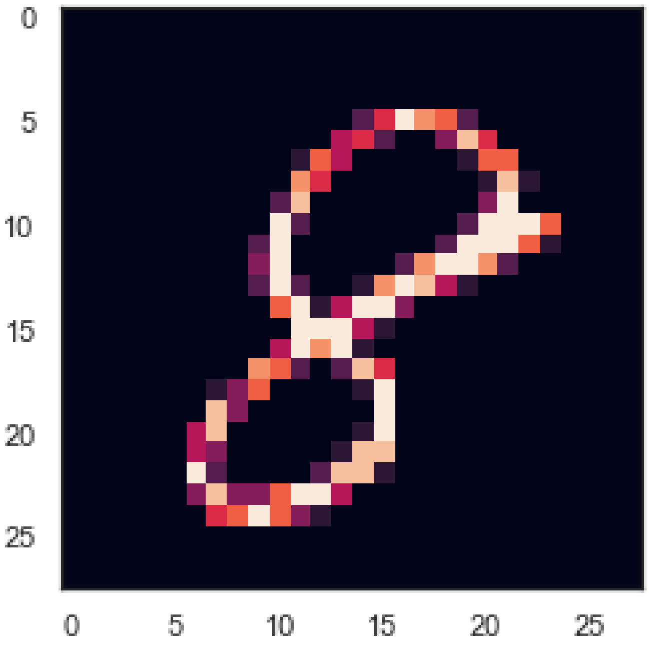
Labels are 10 digits numbers from 0 to 9. We need to encode these lables to one hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0]).

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 10 折交叉验证 10-fold 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])
```



3. CNN

3.1 Define the model

Type *Markdown* and LaTeX: *α*2α2

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from  keras.layers.normalization  import  BatchNormalization
from  keras.preprocessing.image  import  ImageDataGenerator
from  keras.callbacks  import  ReduceLROnPlateau
from  sklearn.model_selection  import  train_test_split
```

```
Using TensorFlow backend.
```

```
##### >>>>填写<<<< 我们的分配数量 num_classes,提示 我们的任务是手写体 0-9 的识别 <<<<<<<<<<#######
batch_size  =  64
num_classes  =  10
epochs  =  20
### >>> 填写<<<<   填写我们的输入 size  格式为(长，宽，通道数)   #######
input_shape  =  (28,28,1)

#构建 CNN  模型  这里我们利用 Sequential  序列累加  ######
model  =  Sequential()
##  第一个  卷积层  32 个 kernel  kernel 大小 3*3  输出的激活函数 relu  kernel 利用  He-正态分布  生成    ####
model.add(Conv2D(32,  kernel_size=(3,  3),activation='relu',kernel_initializer='he_normal',input_shape=input_shape))

### >>>  填写<<<<  请自行构建第二个卷积层，此时 kernel 的初始尝试用全零初始/全 1 初始/正态初始
model.add(Conv2D(32,  kernel_size=(3,  3),activation='relu',kernel_initializer='he_normal'))
### >>>  填写<<<<  构建一个最大池化层
model.add(MaxPool2D((2,  2),strides=2))

model.add(Dropout(0.20))
### >>>  填写<<<<  在下述卷积层内  构建一个 padding,  在之后构建一个 kernel  size  =  2  *2  的池化层
model.add(Conv2D(64,  (3,  3),  activation='relu',padding='same',kernel_initializer='he_normal'))
model.add(MaxPool2D(pool_size=(2,  2)))

#########################################################################################
model.add(Dropout(0.25))


model.add(Flatten())
```

```
###   >>> 填写<<<<  构建一个全联接  其中包含128 个神经元  并使用relu 激活函数
model.add(Dense(128, activation='relu'))
###############################################


model.add(BatchNormalization())
model.add(Dropout(0.25))

###   >>> 填写<<<<构建一个全联接，该全联接需要用特定的激活函数和适当的神经元个数  来实现我们的分类目标    提示：我们有多少个标签？什么激活适合最后的输出？
model.add(Dense(num_classes, activation='softmax'))


model.summary()
Model: "sequential_5"
_________________________________________________________
________
Layer (type)                 Output Shape              Param #

=========================================================
========
conv2d_16 (Conv2D)           (None, 26, 26, 32)        320


_________________________________________________________
________
conv2d_17 (Conv2D)           (None, 24, 24, 32)        9248


_________________________________________________________
________
max_pooling2d_7 (MaxPooling2 (None, 12, 12, 32)        0


_________________________________________________________
________
dropout_12 (Dropout)         (None, 12, 12, 32)        0


_________________________________________________________
________
conv2d_18 (Conv2D)           (None, 12, 12, 64)        18496


_________________________________________________________
________
```

```
max_pooling2d_8 (MaxPooling2 (None, 6, 6, 64)        0

_________________________________________________________
________
dropout_13 (Dropout)        (None, 6, 6, 64)        0

_________________________________________________________
________
flatten_4 (Flatten)         (None, 2304)           0

_________________________________________________________
________
dense_7 (Dense)            (None, 128)             295040

_________________________________________________________
________
batch_normalization_4 (Batch (None, 128)             512

_________________________________________________________
________
dropout_14 (Dropout)        (None, 128)            0

_________________________________________________________
________
dense_8 (Dense)            (None, 10)              1290

=========================================================
========
Total params: 324,906
Trainable params: 324,650
Non-trainable params: 256
_________________________________________________________
________
```