In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries
 installed
# It is defined by the kaggle/python docker image: https://github.com/k
aggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) w
ill list the files in the input directory

import os
#print(os.listdir("../input"))

# Any results you write to the current directory are saved as output.



# Import modules

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
import cv2

#keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import np_utils
import sklearn.metrics as metrics
```

```
Using TensorFlow backend.
```

In [2]:
```
train = pd.read_csv("../input/emnist-balanced-train.csv",delimiter =
',')
test = pd.read_csv("../input/emnist-balanced-test.csv", delimiter = ','
)
mapp = pd.read_csv("../input/emnist-balanced-mapping.txt", delimiter =
' ', \
                   index_col=0, header=None, squeeze=True)
```

In [3]:
```
#! cat ../input/emnist-balanced-mapping.txt
```

In [4]:
```
print("Train: %s, Test: %s, Map: %s" %(train.shape, test.shape, mapp.sh
ape))
```

```
Train: (112799, 785), Test: (18799, 785), Map: (47,)
```

In [5]:
```
# Split x and y
train_x = train.iloc[:,1:]
train_y = train.iloc[:,0]
del train

test_x = test.iloc[:,1:]
test_y = test.iloc[:,0]
del test
```

In [6]:
```
print(train_x.shape,train_y.shape,test_x.shape,test_y.shape)
```

```
(112799, 784) (112799,) (18799, 784) (18799,)
```

In [7]:
```
# Constants
HEIGHT = 28
WIDTH = 28

def rotate(image):
    image = image.reshape([HEIGHT, WIDTH])
    image = np.fliplr(image)
```

```
    image = np.rot90(image)
    return image
```

In [8]:
```
# Flip and rotate image
train_x = np.asarray(train_x)
train_x = np.apply_along_axis(rotate, 1, train_x)
print ("train_x:",train_x.shape)
```

```
train_x: (112799, 28, 28)
```

In [9]:
```
test_x = np.asarray(test_x)
test_x = np.apply_along_axis(rotate, 1, test_x)
print ("test_x:",test_x.shape)
```

```
test_x: (18799, 28, 28)
```

In [10]:
```
# Normalise
train_x = train_x.astype('float32')
train_x /= 255
test_x = test_x.astype('float32')
test_x /= 255
```

In [11]:
```
print(train_x[1])
plt.imshow(train_x[1], cmap=plt.get_cmap('gray'))
```

```
[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.01568628
  0.01568628 0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
```

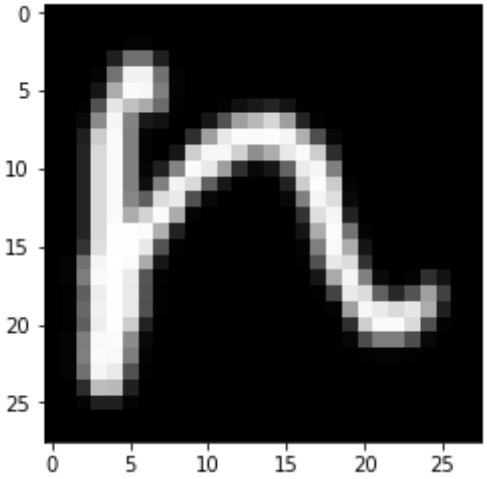
```
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.1254902  0.44313726
 0.44313726 0.1254902  0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.01568628 0.4509804  0.94509804
 0.94509804 0.44313726 0.01568628 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.08627451 0.6745098  0.98039216
 0.96862745 0.49803922 0.01568628 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.00784314 0.32156864 0.8980392  0.7529412
 0.6666667  0.41960785 0.01176471 0.         0.         0.01176471
 0.0627451  0.09803922 0.14117648 0.07843138 0.00392157 0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.03529412 0.54509807 0.9607843  0.50980395
 0.09803922 0.07058824 0.         0.00392157 0.08627451 0.3254902
 0.62352943 0.72156864 0.8392157  0.61960787 0.14117648 0.01176471
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.1254902  0.8        0.9764706  0.49803922
 0.01568628 0.         0.00392157 0.18431373 0.62352943 0.9098039
 0.98039216 0.9764706  0.9882353  0.9607843  0.6862745  0.30588236
 0.02745098 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.14509805 0.8509804  0.98039216 0.49803922
 0.01568628 0.         0.09019608 0.80784315 0.96862745 0.96862745
 0.8156863  0.5568628  0.6784314  0.9098039  0.98039216 0.80784315
 0.18039216 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.14509805 0.8509804  0.98039216 0.49803922
 0.01960784 0.1254902  0.5137255  0.98039216 0.9098039  0.62352943
```

```
 0.18431373 0.03921569 0.09019608 0.39215687 0.91764706 0.9607843
 0.49803922 0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.14509805 0.8509804  0.98039216 0.49803922
 0.05882353 0.49803922 0.9490196  0.85490197 0.37254903 0.08627451
 0.00392157 0.         0.         0.13333334 0.8        0.99215686
 0.79607844 0.01568628 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.14509805 0.8509804  0.98039216 0.5254902
 0.37254903 0.8627451  0.9137255  0.30980393 0.03137255 0.
 0.         0.         0.         0.03137255 0.49803922 0.9607843
 0.87058824 0.03529412 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.14509805 0.8509804  0.98039216 0.6784314
 0.827451   0.98039216 0.67058825 0.01568628 0.         0.
 0.         0.         0.         0.         0.2        0.87058824
 0.9607843  0.13333334 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.14509805 0.8509804  0.99607843 0.99215686
 0.9843137  0.6862745  0.13333334 0.         0.         0.
 0.         0.         0.         0.         0.08235294 0.6745098
 0.9843137  0.4509804  0.01568628 0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.2        0.87058824 0.99607843 0.99607843
 0.9098039  0.3254902  0.01176471 0.         0.         0.
 0.         0.         0.         0.         0.01960784 0.49803922
 0.9764706  0.6745098  0.08235294 0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.01568628 0.44705883 0.9607843  1.         0.9843137
 0.67058825 0.08627451 0.         0.         0.         0.
 0.         0.         0.         0.         0.00784314 0.32156864
 0.9098039  0.8627451  0.19607843 0.         0.         0.
 0.00392157 0.         0.         0.         0.        ]
[0.         0.01568628 0.49019608 0.9764706  0.99607843 0.91764706
 0.33333334 0.01176471 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.08627451
 0.6666667  0.9607843  0.5019608  0.03921569 0.         0.01568628
 0.18431373 0.07058824 0.         0.         0.        ]
[0.         0.01176471 0.37254903 0.93333334 0.99607843 0.9137255
```

```
 0.32156864 0.00784314 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.00784314
 0.2627451  0.9137255  0.8627451  0.3764706  0.16078432 0.32941177
 0.627451   0.25490198 0.00392157 0.        ]
[0.         0.01176471 0.40784314 0.94509804 0.99607843 0.9137255
 0.3137255  0.00784314 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.01568628 0.62352943 0.9647059  0.91764706 0.85490197 0.9019608
 0.6745098  0.1254902  0.         0.        ]
[0.         0.01176471 0.37254903 0.93333334 0.99607843 0.8
 0.13333334 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.18431373 0.80784315 0.9764706  0.9764706  0.85490197
 0.30980393 0.01176471 0.         0.        ]
[0.         0.01568628 0.49019608 0.9764706  0.9843137  0.54901963
 0.03529412 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.02745098 0.3019608  0.49019608 0.49019608 0.3019608
 0.03137255 0.         0.         0.        ]
[0.         0.01568628 0.49803922 0.98039216 0.9764706  0.49019608
 0.01568628 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.00784314 0.01568628 0.01568628 0.00784314
 0.         0.         0.         0.        ]
[0.         0.01568628 0.44705883 0.9607843  0.9137255  0.32156864
 0.00784314 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.1764706  0.7411765  0.7254902  0.12941177
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.01960784 0.1254902  0.1254902  0.01960784
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
```

```
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]]
```

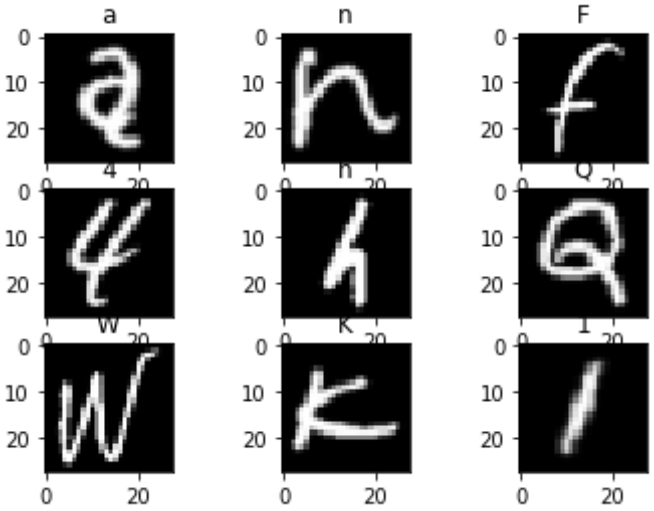
Out[11]: <matplotlib.image.AxesImage at 0x7f2ffe3346a0>




In [12]:
```
chr(mapp[train_y[1]])
```

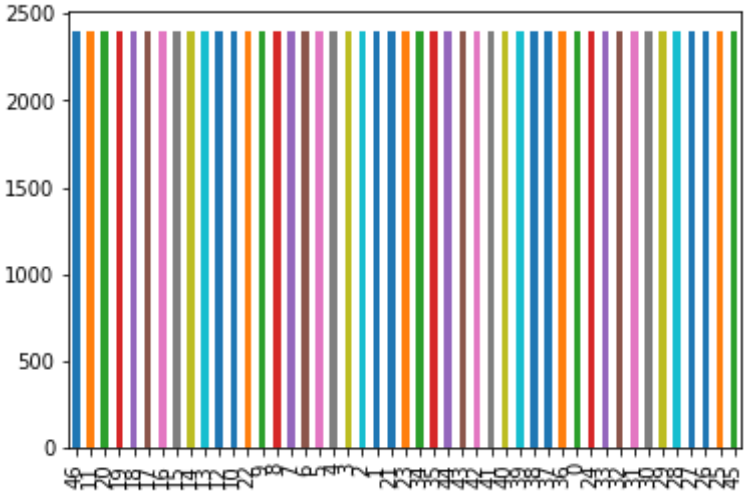
Out[12]: 'n'

In [13]:
```
# plot image
for i in range(0, 9):
    plt.subplot(330 + (i+1))
    plt.imshow(train_x[i], cmap=plt.get_cmap('gray'))
    plt.title(chr(mapp[train_y[i]]))
```




In [14]:
```
train_y.value_counts().plot(kind='bar')
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2fff1c5780>



In [15]:
```
# number of classes
num_classes = train_y.nunique()
```

In [16]:
```
# One hot encoding
train_y = np_utils.to_categorical(train_y, num_classes)
test_y = np_utils.to_categorical(test_y, num_classes)
print("train_y: ", train_y.shape)
print("test_y: ", test_y.shape)
```

```
train_y:  (112799, 47)
test_y:  (18799, 47)
```

In [17]:
```
print ("train_x:",train_x.shape)
```

```
train_x: (112799, 28, 28)
```

In [18]:
```
# Reshape image for CNN
train_x = train_x.reshape(-1, HEIGHT, WIDTH, 1)
test_x = test_x.reshape(-1, HEIGHT, WIDTH, 1)
```

In [19]:
```
print ("train_x:",train_x.shape)
```

```
train_x: (112799, 28, 28, 1)
```

In [20]:
```
# partition to train and val
train_x, val_x, train_y, val_y = train_test_split(train_x, train_y, tes
t_size= 0.10, random_state=7)
```

In [21]:
```
# Building model
# ((w - k + 2P)/S) + 1
model = Sequential()

model.add(Conv2D(filters=128, kernel_size=(5,5), padding = 'same', acti
vation='relu',\
                 input_shape=(HEIGHT, WIDTH,1)))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Conv2D(filters=64, kernel_size=(3,3) , padding = 'same', acti
vation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

```
model.add(Flatten())
model.add(Dense(units=128, activation='relu'))
model.add(Dropout(.5))
model.add(Dense(units=num_classes, activation='softmax'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 128)       3328
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        73792
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               401536
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 47)                6063
=================================================================
Total params: 484,719
Trainable params: 484,719
Non-trainable params: 0
_________________________________________________________________
```

In [22]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [ ]:
```
history = model.fit(train_x, train_y, epochs=300, batch_size=512, verbose=1, \
                    validation_data=(val_x, val_y))
```

In [ ]:
```
def show_result(epochs, acc, val_acc):
    plt.plot(epochs, acc, 'y')
    plt.plot(epochs, val_acc, 'b')
    plt.title('Model accuracy',fontsize = 18)
    plt.ylabel('Accuracy',fontsize = 18)
    plt.xlabel('Epoch',fontsize = 18)
    plt.legend(['Train', 'Val'], loc='best')
    plt.savefig('res.png')
    plt.show()
```

In [ ]:
```
history.history.keys()
```

In [ ]:
```
train = history.history['acc']
val = history.history['val_acc']
epochs = range(1,len(train)+1)
```

In [ ]:
```
show_result(epochs, train, val)
```

In [ ]:
```
#Save the model
# serialize model to JSON
model_json = model.to_json()
with open("model2.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model2.h5")
print("Saved model to disk")
```

In [ ]:
```
#from joblib import dump
```

In [ ]:
```
#dump(model,'model.pkl')
```

In [ ]:
```
#!ls ../
```

In [ ]:
```

```

In [ ]: