jupyter Character Recognition Last Checkpoint: 5 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [2]:
```
import cv2 as cv
import numpy as np
import pandas as pd
from keras.models import load_model
import tensorflow as tf
import matplotlib.pyplot as plt
```

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
```

In [4]:
```
data = pd.read_csv('A_Z Handwritten Data.csv').astype('float32')
```

In [6]:
```
x = data.drop('0',axis = 1)
y = data['0']
```

In [7]:
```
print(x.shape)
print(y.shape)
```

```
(372450, 784)
(372450,)
```

In [8]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [9]:
```
x_train = np.reshape(x_train.values,(x_train.shape[0],28,28))
```

In [10]:
```
x_test = np.reshape(x_test.values,(x_test.shape[0],28,28))
```

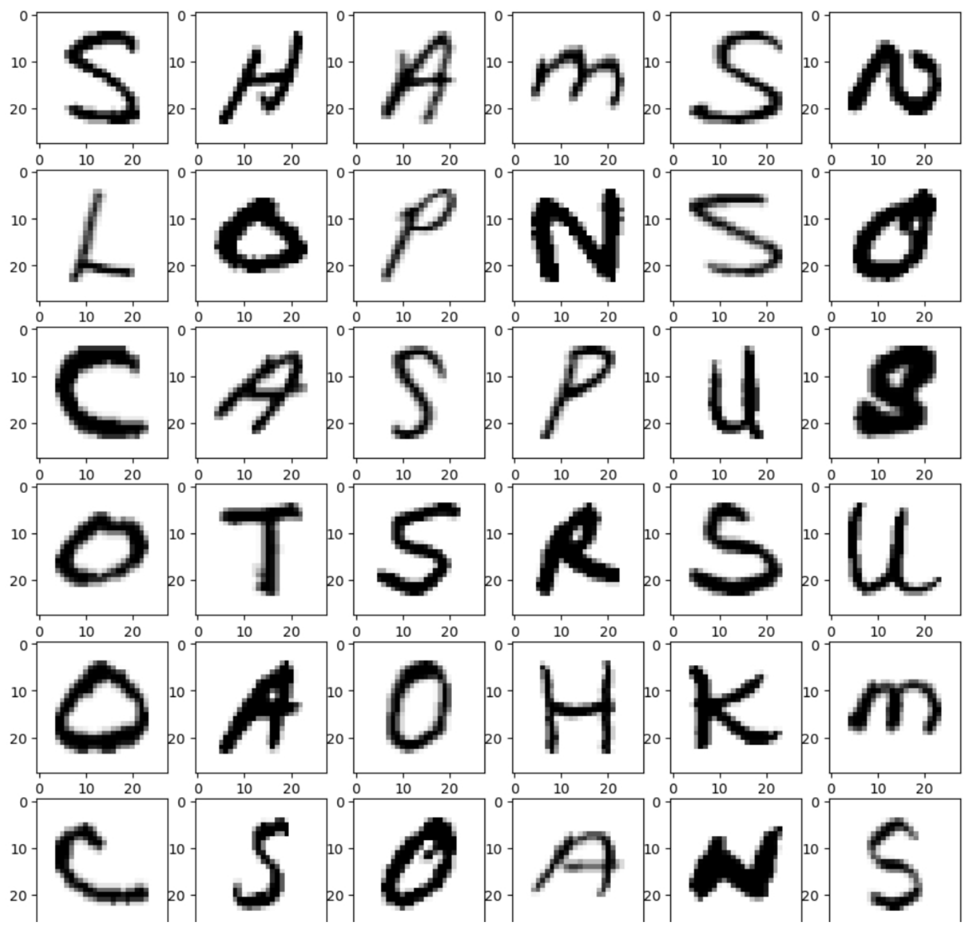
In [11]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

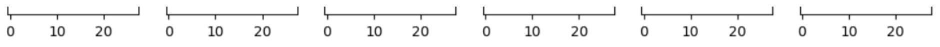
```
(297960, 28, 28)
(74490, 28, 28)
(297960,)
(74490,)
```

In [12]:
```
shuffle_data = shuffle(x_train)
```

In [13]:
```
# Visualizing our training data

_, axis = plt.subplots(6,6, figsize = (12,12))
axis = axis.flatten()
for i in range(36):
    axis[i].imshow(np.reshape(shuffle_data[i], (28,28)), cmap="Greys")
plt.show()
```





In [14]:
```
# Reshaping our model and adding channel 1 to it

x_train = x_train.reshape(x_train.shape[0],x_train.shape[1],x_train.shape[2],1)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2],1)
print("New shape of training data: ", x_train.shape)
print("New shape of testing data: ", x_test.shape)
```

```
New shape of training data:  (297960, 28, 28, 1)
New shape of testing data:  (74490, 28, 28, 1)
```

In [15]:
```
y_training = to_categorical(y_train, num_classes = 26, dtype='int')
y_testing = to_categorical(y_test, num_classes = 26, dtype='int')
print("New shape of testing data: ", y_training.shape)
print("New shape of testing data: ", y_testing.shape)
```

```
New shape of testing data:  (297960, 26)
New shape of testing data:  (74490, 26)
```

In [21]:
```
model = Sequential()

model.add(Conv2D(64 , (3, 3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(2,2))

model.add(Flatten())

model.add(Dense(128,activation ="relu"))
model.add(Dense(26,activation ="softmax"))
```

In [22]:
```
# x_train.shape,y_training.shape
x_test.shape,y_testing.shape
```

Out[22]:
```
((74490, 28, 28, 1), (74490, 26))
```

In [23]:
```
model.compile(optimizer = 'adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_training, epochs=20,  validation_data = (x_test,y_testing))
```

```
Epoch 1/20
9312/9312 [==============================] - 174s 19ms/step - loss: 0.2239 - accuracy: 0.9367 - val_loss: 0.1352 - val_accurac
y: 0.9609
Epoch 2/20
9312/9312 [==============================] - 169s 18ms/step - loss: 0.1067 - accuracy: 0.9696 - val_loss: 0.0967 - val_accurac
y: 0.9724
Epoch 3/20
9312/9312 [==============================] - 178s 19ms/step - loss: 0.0905 - accuracy: 0.9736 - val_loss: 0.1230 - val_accurac
y: 0.9644
Epoch 4/20
9312/9312 [==============================] - 179s 19ms/step - loss: 0.0827 - accuracy: 0.9762 - val_loss: 0.1016 - val_accurac
y: 0.9735
Epoch 5/20
9312/9312 [==============================] - 180s 19ms/step - loss: 0.0785 - accuracy: 0.9776 - val_loss: 0.0982 - val_accurac
y: 0.9750
Epoch 6/20
9312/9312 [==============================] - 183s 20ms/step - loss: 0.0761 - accuracy: 0.9786 - val_loss: 0.0978 - val_accurac
y: 0.9732
Epoch 7/20
9312/9312 [==============================] - 180s 19ms/step - loss: 0.0730 - accuracy: 0.9795 - val_loss: 0.1090 - val_accurac
y: 0.9750
Epoch 8/20
9312/9312 [==============================] - 183s 20ms/step - loss: 0.0725 - accuracy: 0.9804 - val_loss: 0.0962 - val_accurac
y: 0.9763
Epoch 9/20
9312/9312 [==============================] - 178s 19ms/step - loss: 0.0716 - accuracy: 0.9803 - val_loss: 0.1011 - val_accurac
y: 0.9745
Epoch 10/20
9312/9312 [==============================] - 175s 19ms/step - loss: 0.0691 - accuracy: 0.9813 - val_loss: 0.1111 - val_accurac
y: 0.9746
Epoch 11/20
9312/9312 [==============================] - 171s 18ms/step - loss: 0.0710 - accuracy: 0.9811 - val_loss: 0.1180 - val_accurac
y: 0.9729
Epoch 12/20
9312/9312 [==============================] - 176s 19ms/step - loss: 0.0691 - accuracy: 0.9816 - val_loss: 0.1047 - val_accurac
y: 0.9758
Epoch 13/20
9312/9312 [==============================] - 177s 19ms/step - loss: 0.0669 - accuracy: 0.9823 - val_loss: 0.1071 - val_accurac
y: 0.9754
Epoch 14/20
9312/9312 [==============================] - 170s 18ms/step - loss: 0.0687 - accuracy: 0.9821 - val_loss: 0.1092 - val_accurac
y: 0.9767
Epoch 15/20
9312/9312 [==============================] - 169s 18ms/step - loss: 0.0675 - accuracy: 0.9824 - val_loss: 0.1035 - val_accurac
y: 0.9776
Epoch 16/20
9312/9312 [==============================] - 165s 18ms/step - loss: 0.0669 - accuracy: 0.9827 - val_loss: 0.1020 - val_accurac
y: 0.9784
Epoch 17/20
9312/9312 [==============================] - 166s 18ms/step - loss: 0.0682 - accuracy: 0.9829 - val_loss: 0.0980 - val_accurac
y: 0.9792
Epoch 18/20
9312/9312 [==============================] - 170s 18ms/step - loss: 0.0678 - accuracy: 0.9832 - val_loss: 0.0991 - val_accurac
y: 0.9790
Epoch 19/20
9312/9312 [==============================] - 171s 18ms/step - loss: 0.0695 - accuracy: 0.9830 - val_loss: 0.1186 - val_accurac
y: 0.9756
Epoch 20/20
9312/9312 [==============================] - 172s 18ms/step - loss: 0.0688 - accuracy: 0.9834 - val_loss: 0.1040 - val_accurac
y: 0.9795
```

Out[23]:
```
<keras.callbacks.History at 0x1eb1e4ba9d0>
```

In [ ]:
```
# from ann_visualizer.visualize import ann_viz
# ann_viz(model, view=True, filename="construct_model", title="CNN — Model")
```

In [24]:
```
print("Evaluate on test data")
results = model.evaluate(x_test, y_testing, batch_size=100)
print("test loss, test acc:", results)
```

```
Evaluate on test data
745/745 [==============================] - 10s 13ms/step - loss: 0.1040 - accuracy: 0.9795
```

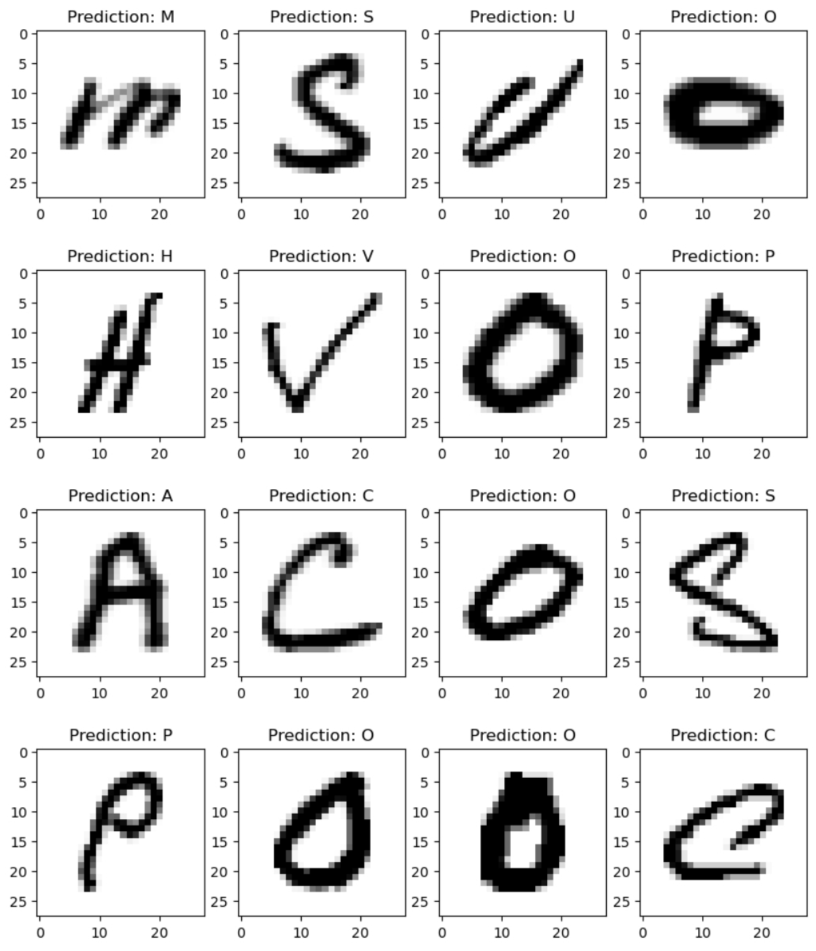
```
745/745 [==============================] - 10s 13ms/step - loss: 0.1040 - accuracy: 0.9795
test loss, test acc: [0.10400985926389694, 0.9794737696647644]
```

In [25]:
```python
model.save(r'handwritten_character_recog_model.h5')
```

In [26]:
```python
words = {0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O',15:'P',16:'Q',17:'R',18:
```

In [35]:
```python
_, axes = plt.subplots(4,4, figsize=(10,12))
axes = axes.flatten()
for i in range(16):
    image = np.reshape(x_test[i], (28,28))
    axes[i].imshow(image, cmap="Greys")
    image = np.reshape(x_test[i], (1,28,28,1))
    pred = words[np.argmax([model.predict(image)])]
    axes[i].set_title("Prediction: "+pred)
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
```


Prediction: M
Prediction: S
Prediction: U
Prediction: O
Prediction: H
Prediction: V
Prediction: O
Prediction: P
Prediction: A
Prediction: C
Prediction: O
Prediction: S
Prediction: P
Prediction: O
Prediction: O
Prediction: C