In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```
df = pd.read_csv('../../dataset/A_Z Handwritten Data.csv (1).zip')
```

In [5]:
```
df.head()
```

Out[5]:

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.639	0.640	0.641	0.642	0.643	0.644
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0

5 rows × 785 columns

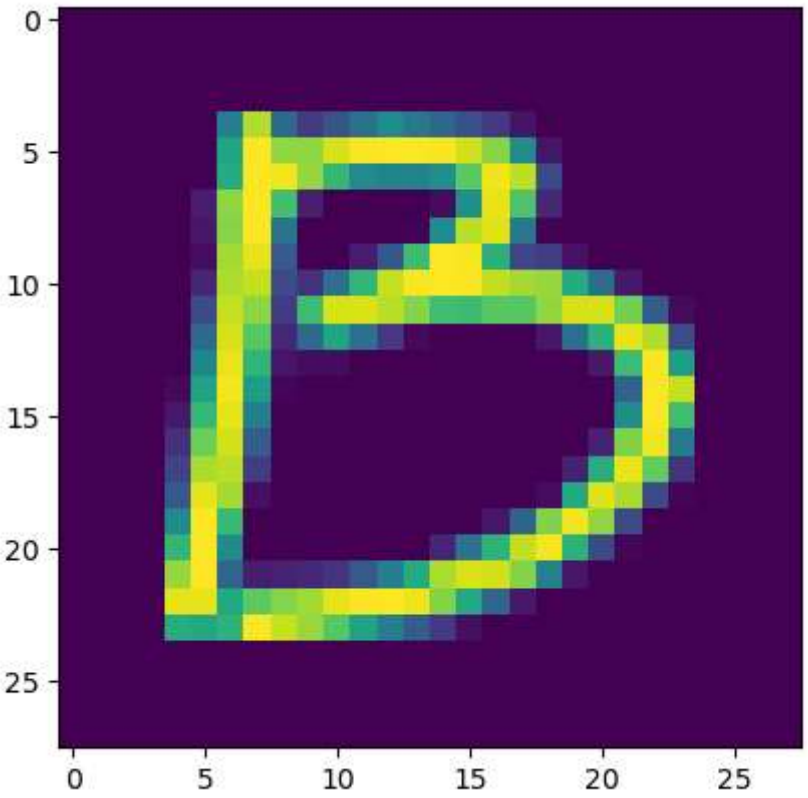
In [6]:
```
y = df['0']
X = df.drop('0',axis=1)
```

In [7]:
```
X.shape
```

Out[7]:
```
(372450, 784)
```

In [8]:
```
img = X.iloc[20000,:].values.reshape(28,28)
plt.imshow(img)
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x172a05b6650>
```



In [12]:
```
X = X/255
```

In [13]:
```
from tensorflow.keras.utils import to_categorical
```

```
WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cr
oss_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax
_cross_entropy instead.
```

In [14]:
```
ya = to_categorical(y,num_classes=26)
ya.shape
```

Out[14]:
```
(372450, 26)
```

In [15]:
```
X = X.values.reshape(372450,28,28,1)
X.shape
```

Out[15]:
```
(372450, 28, 28, 1)
```

Model building

In [16]:
```
from tensorflow.keras.layers import Dense,Conv2D,MaxPool2D,Flatten
from tensorflow.keras.models import Sequential
```

In [17]:
```
model = Sequential()
model.add(Conv2D(16,(3,3),activation = 'relu',input_shape=(28,28,1)))
model.add(MaxPool2D())
model.add(Conv2D(32,(3,3),activation = 'relu'))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(40,activation='relu'))
model.add(Dense(30,activation='relu'))
model.add(Dense(26,activation='softmax'))

model.compile(loss='categorical_crossentropy', metrics=['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\backend.py:873: The name tf.get_default_graph is dep
recated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.
max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimi
zer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

In [18]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 16)        160

 max_pooling2d (MaxPooling2  (None, 13, 13, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 32)        4640

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 32)          0
 g2D)

 flatten (Flatten)           (None, 800)               0

 dense (Dense)               (None, 40)                32040

 dense_1 (Dense)             (None, 30)                1230

 dense_2 (Dense)             (None, 26)                806

=================================================================
Total params: 38876 (151.86 KB)
Trainable params: 38876 (151.86 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [19]:
```
model.fit(X,ya,batch_size=1000,epochs=20)
```

```
Epoch 1/20
WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTens
orValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue
instead.

WARNING:tensorflow:From C:\Users\saksh\dsml27F\envs\dsml27_env1\Lib\site
-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executin
g_eagerly_outside_functions is deprecated. Please use tf.compat.v1.execu
ting_eagerly_outside_functions instead.

373/373 [==============================] - 66s 161ms/step - loss: 0.8779
- accuracy: 0.7560
Epoch 2/20
373/373 [==============================] - 60s 162ms/step - loss: 0.1948
- accuracy: 0.9446
Epoch 3/20
373/373 [==============================] - 60s 161ms/step - loss: 0.1199
- accuracy: 0.9664
Epoch 4/20
373/373 [==============================] - 60s 161ms/step - loss: 0.0915
- accuracy: 0.9742
Epoch 5/20
373/373 [==============================] - 60s 161ms/step - loss: 0.0766
- accuracy: 0.9787
Epoch 6/20
373/373 [==============================] - 60s 161ms/step - loss: 0.0672
- accuracy: 0.9813
Epoch 7/20
373/373 [==============================] - 61s 162ms/step - loss: 0.0602
- accuracy: 0.9834
Epoch 8/20
373/373 [==============================] - 61s 164ms/step - loss: 0.0549
- accuracy: 0.9847
Epoch 9/20
373/373 [==============================] - 61s 163ms/step - loss: 0.0501
- accuracy: 0.9861
Epoch 10/20
373/373 [==============================] - 60s 161ms/step - loss: 0.0460
- accuracy: 0.9872
Epoch 11/20
373/373 [==============================] - 60s 161ms/step - loss: 0.0429
- accuracy: 0.9877
Epoch 12/20
373/373 [==============================] - 62s 166ms/step - loss: 0.0394
- accuracy: 0.9887
Epoch 13/20
373/373 [==============================] - 61s 164ms/step - loss: 0.0368
- accuracy: 0.9894
Epoch 14/20
373/373 [==============================] - 61s 163ms/step - loss: 0.0344
- accuracy: 0.9900
Epoch 15/20
373/373 [==============================] - 60s 162ms/step - loss: 0.0318
- accuracy: 0.9909
Epoch 16/20
373/373 [==============================] - 61s 163ms/step - loss: 0.0299
```

```
- accuracy: 0.9914
Epoch 17/20
373/373 [==============================] - 60s 162ms/step - loss: 0.0279
- accuracy: 0.9919
Epoch 18/20
373/373 [==============================] - 61s 165ms/step - loss: 0.0263
- accuracy: 0.9924
Epoch 19/20
373/373 [==============================] - 61s 165ms/step - loss: 0.0242
- accuracy: 0.9929
Epoch 20/20
373/373 [==============================] - 61s 164ms/step - loss: 0.0234
- accuracy: 0.9930
```

Out[19]: <keras.src.callbacks.History at 0x17343240e10>

Prediction on own handwriting

In [20]:
```
y[:5]
```

Out[20]:
```
0    0
1    0
2    0
3    0
4    0
Name: 0, dtype: int64
```

In [21]:
```
img = X[0,:].reshape(1,28,28,1)
model.predict_on_batch(img).argmax()
```

Out[21]: 0

In [22]:
```
model.predict_on_batch(X[:5,:]).argmax(axis=1)
```

Out[22]: array([0, 0, 0, 0, 0], dtype=int64)

In [23]:
```
import cv2
```

In [24]:
```
def get_digit(filename):
    path = '../../dataset/Alphabet Recognization/' + filename
    A = cv2.imread(path,0)
    A = cv2.resize(A,(28,28))
    A = A/255
    A = A.reshape(1,28,28,1)
    return model.predict_on_batch(A).argmax()
```

In [25]:
```
get_digit('B.jpg')
```

Out[25]: 6

In [26]:
```
import os
```

In [29]:
```
filenames = os.listdir('../../dataset/Alphabet Recognization/my_images alp
filenames
```

Out[29]:
```
['A.jpg',
 'AA.jpg',
 'B.jpg',
 'BB.jpg',
 'C.jpg',
 'CC.jpg',
 'D.jpg',
 'DD.jpg',
 'E.jpg',
 'EE.jpg',
 'F.jpg',
 'FF.jpg',
 'G.jpg',
 'GG.jpg',
 'H.jpg',
 'HH.jpg',
 'I.jpg',
 'II.jpg',
 'J.jpg',
 'JJ.jpg',
 'K.jpg',
 'KK (1).jpg',
 'L.jpg',
 'LL.jpg',
 'M.jpg',
 'MM.jpg',
 'N.jpg',
 'NN.jpg',
 'O.jpg',
 'OO.jpg',
 'P.jpg',
 'PP.jpg',
 'Q.jpg',
 'QQ.jpg',
 'R.jpg',
 'RR.jpg',
 'S.jpg',
 'SS.jpg',
 'T.jpg',
 'TT.jpg',
 'U.jpg',
 'UU.jpg',
 'V.jpg',
 'VV.jpg',
 'W.jpg',
 'WW.jpg',
 'X.jpg',
 'XX.jpg',
 'Y.jpg',
 'YY.jpg',
 'Z.jpg',
 'ZZ.jpg']
```

In [30]:

```
mapping = { 0 :'A', 1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',
for file in filenames:
    yp = get_digit(file)
    #yp = yp.tolist()
    x = mapping.get(yp)
    print(file,'\t', x)
```

```
A.jpg    A
AA.jpg   A
B.jpg    G
BB.jpg   B
C.jpg    C
CC.jpg   C
D.jpg    D
DD.jpg   D
E.jpg    E
EE.jpg   E
F.jpg    E
FF.jpg   F
G.jpg    G
GG.jpg   G
H.jpg    H
HH.jpg   H
I.jpg    I
II.jpg   I
J.jpg    T
JJ.jpg   J
K.jpg    K
KK (1).jpg       K
L.jpg    L
LL.jpg   L
M.jpg    M
MM.jpg   Y
N.jpg    N
NN.jpg   N
O.jpg    J
OO.jpg   O
P.jpg    Y
PP.jpg   P
Q.jpg    G
QQ.jpg   Q
R.jpg    K
RR.jpg   R
S.jpg    G
SS.jpg   S
T.jpg    V
TT.jpg   T
U.jpg    U
UU.jpg   U
V.jpg    V
VV.jpg   K
W.jpg    W
WW.jpg   W
X.jpg    X
XX.jpg   X
Y.jpg    Y
YY.jpg   Y
Z.jpg    Z
ZZ.jpg   Z
```

In [34]:
```
ground_truth_labels = {
    'A.jpg': 'A', 'AA.jpg': 'A', 'B.jpg': 'B', 'BB.jpg': 'B', 'C.jpg': 'C
    'E.jpg': 'E', 'EE.jpg': 'E', 'F.jpg': 'F', 'FF.jpg': 'F', 'G.jpg': 'G
    'I.jpg': 'I', 'II.jpg': 'I', 'J.jpg': 'J', 'JJ.jpg': 'J', 'K.jpg': 'K
    'M.jpg': 'M', 'MM.jpg': 'M', 'N.jpg': 'N', 'NN.jpg': 'N', 'O.jpg': 'O
    'Q.jpg': 'Q', 'QQ.jpg': 'Q', 'R.jpg': 'R', 'RR.jpg': 'R', 'S.jpg': 'S
    'U.jpg': 'U', 'UU.jpg': 'U', 'V.jpg': 'V', 'VV.jpg': 'V', 'W.jpg': 'W
    'Y.jpg': 'Y', 'YY.jpg': 'Y', 'Z.jpg': 'Z', 'ZZ.jpg': 'Z'
}

predicted_labels = {
    'A.jpg': 'A', 'AA.jpg': 'A', 'B.jpg': 'G', 'BB.jpg': 'B', 'C.jpg': 'C
    'E.jpg': 'E', 'EE.jpg': 'E', 'F.jpg': 'E', 'FF.jpg': 'F', 'G.jpg': 'G
    'I.jpg': 'I', 'II.jpg': 'I', 'J.jpg': 'T', 'JJ.jpg': 'J', 'K.jpg': 'K
    'M.jpg': 'M', 'MM.jpg': 'Y', 'N.jpg': 'N', 'NN.jpg': 'N', 'O.jpg': 'J
    'Q.jpg': 'G', 'QQ.jpg': 'Q', 'R.jpg': 'K', 'RR.jpg': 'R', 'S.jpg': 'G
    'U.jpg': 'U', 'UU.jpg': 'U', 'V.jpg': 'V', 'VV.jpg': 'K', 'W.jpg': 'W
    'Y.jpg': 'Y', 'YY.jpg': 'Y', 'Z.jpg': 'Z', 'ZZ.jpg': 'Z'
}

# Calculate accuracy
correct_predictions = sum(1 for file, predicted_label in predicted_labels.
total_images = len(predicted_labels)
accuracy = (correct_predictions / total_images) * 100 if total_images > 0

print(f'Accuracy: {accuracy:.2f}%')
```

```
Accuracy: 78.85%
```

In [35]:
```
print(41*100/52)
```

```
78.84615384615384
```

In [ ]:
```

```