Importing Libraries

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import tensorflow as tf
from tensorflow import keras
import string
from keras.optimizers import Adam
from PIL import Image
```

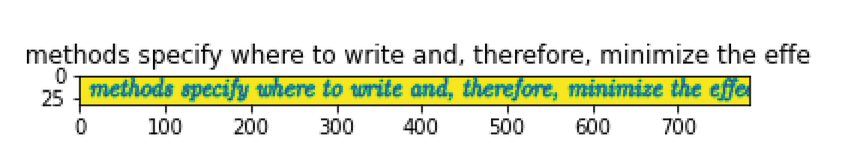
In [ ]:
```python
from google.colab import drive
drive.mount("drive")
```

Selecting the Images and excluding the directories in the folder

The Text file 'UsedSentences.txt' contains the output of the text in the Image and the name of the Image. The name of the image is used to extract the image from the directory and add it to our dataset.

In [2]:
```python
text_file=open(r"UsedSentences.txt","r")

details=[]
outputs=[]
names=[]
for line in text_file:
    a=line.split('#')
    outputs.append(a[1].strip('\n'))
    details.append(a[0])

for detail in details:
    a=detail.split(' ')
    names.append(a[0])
X=[]

for name in names:
    img=Image.open('drive/My Drive/CleanedImages/Encoder_Clean_Renamed/'+name+'.png','r')
    img = img.resize((784,32), Image.ANTIALIAS)
    img=np.asarray(img)
    img=img[:,:,0]
    X.append(img)

X=np.asarray(X)
plt.imshow(X[42])
plt.title(outputs[42])
print("No of Images :",X.shape[0])

symbols = " "+string.ascii_lowercase + string.ascii_uppercase+"0123456789.,*&!@~():`^]¢‘;|-«"
print("Characters :",symbols)
print("No of chars :",len(symbols))
```

```
No of Images : 1774
Characters :  abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.,*&!@~():`^]¢‘;|-«
No of chars : 82
```



Creating a One Hot Encoded Array

98 is taken as the maximum number of Character in the Sentences. If the letter is present in the output then it is encoded to 1.

In [4]:
```python
Y=np.zeros(shape=(len(outputs),98,len(symbols)))
for example_no,name in enumerate(outputs):
    for letter_no,letter in enumerate(name):
        try:
            Y[example_no][letter_no][symbols.index(letter)]=1
        except:
            print(letter,end=" ")
```

Reshaping the array(X) to pass it to the convolution

In [5]:
```python
X=np.reshape(X,(X.shape[0],X.shape[1],X.shape[2],1))
print("Shape of X is :",X.shape)
```

```
Shape of X is : (1774, 32, 784, 1)
```

Building the Neural Network Model

In [6]:
```python
def OCRModel():
    image=keras.layers.Input((32,784,1))
    conv1=keras.layers.Conv2D(16,(3,3),activation='relu',padding='same')(image)

    mp1=keras.layers.MaxPooling2D((2,2),padding='same')(conv1)
    conv2=keras.layers.Conv2D(32,(3,3),activation='relu',padding='same')(mp1)

    mp2=keras.layers.MaxPooling2D((2,2),padding='same')(conv2)
    conv3=keras.layers.Conv2D(64,(3,3),activation='relu',padding='same')(mp2)

    mp3=keras.layers.MaxPooling2D((2,2),padding='same')(conv3)
    conv4=keras.layers.Conv2D(128,(3,3),activation='relu',padding='same')(mp3)

    mp4=keras.layers.MaxPooling2D((2,1),padding='same')(conv4)
    conv5=keras.layers.Conv2D(256,(3,3),activation='relu',padding='same')(mp4)

    mp5=keras.layers.MaxPooling2D((2,1),padding='same')(conv5)
    conv6=keras.layers.Conv2D(256,(3,3),activation='relu',padding='same')(mp5)


    bn=keras.layers.BatchNormalization()(conv6)
    sq=keras.backend.squeeze(bn,axis=1)

    rn1=keras.layers.Bidirectional(keras.layers.LSTM(256,return_sequences=True))(sq)
    rn2=keras.layers.Bidirectional(keras.layers.LSTM(256,return_sequences=True))(rn1)

    exd=keras.backend.expand_dims(rn2,axis=2)
    maping=keras.layers.Conv2D(len(symbols),(2,2),activation='relu',padding='same')(exd)
    maping=keras.backend.squeeze(maping,axis=2)
    maping = tf.keras.layers.Softmax()(maping)

    model=keras.Model(image,maping)
    model.compile(loss='categorical_crossentropy', optimizer='adam')

    return model
```

In [7]:
```python
from keras.utils.vis_utils import plot_model
OCR=OCRModel()
OCR.fit(X,Y,epochs=50)
```

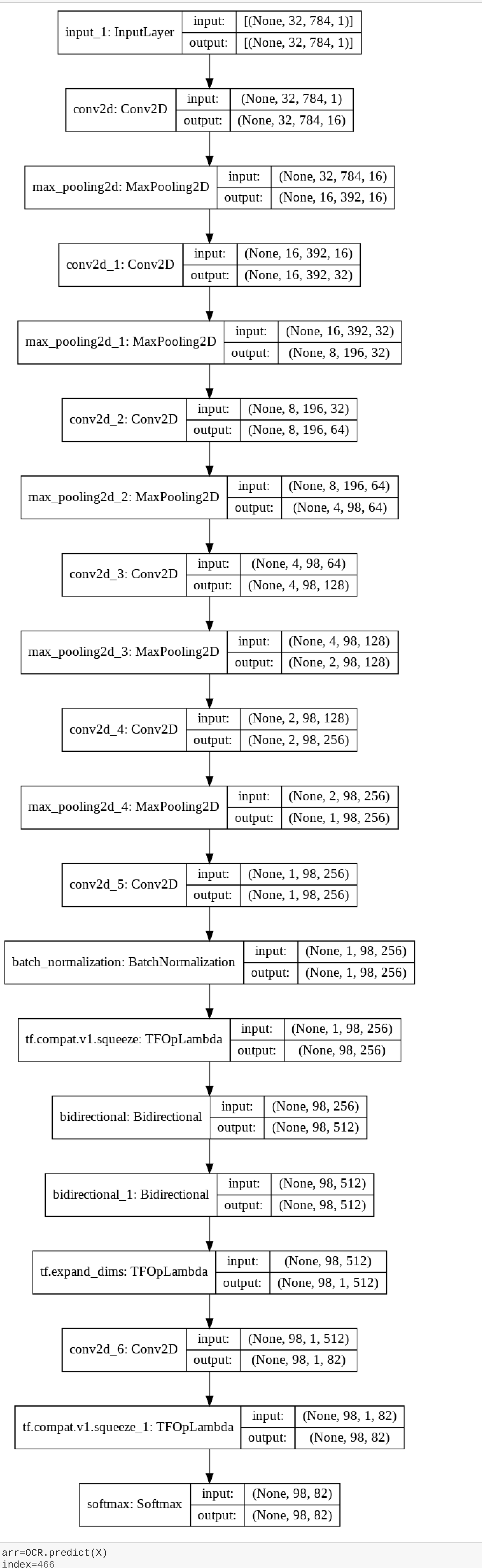
```
Epoch 1/50
56/56 [==============================] - 218s 4s/step - loss: 1.6218
Epoch 2/50
56/56 [==============================] - 208s 4s/step - loss: 1.5169
Epoch 3/50
56/56 [==============================] - 207s 4s/step - loss: 1.4908
Epoch 4/50
56/56 [==============================] - 209s 4s/step - loss: 1.4694
Epoch 5/50
56/56 [==============================] - 208s 4s/step - loss: 1.4412
Epoch 6/50
56/56 [==============================] - 206s 4s/step - loss: 1.3892
Epoch 7/50
56/56 [==============================] - 206s 4s/step - loss: 1.3187
Epoch 8/50
56/56 [==============================] - 211s 4s/step - loss: 1.2096
Epoch 9/50
56/56 [==============================] - 211s 4s/step - loss: 1.0823
Epoch 10/50
56/56 [==============================] - 208s 4s/step - loss: 0.8999
Epoch 11/50
56/56 [==============================] - 208s 4s/step - loss: 0.7549
Epoch 12/50
56/56 [==============================] - 207s 4s/step - loss: 0.5958
Epoch 13/50
56/56 [==============================] - 207s 4s/step - loss: 0.4634
Epoch 14/50
56/56 [==============================] - 209s 4s/step - loss: 0.3595
Epoch 15/50
56/56 [==============================] - 206s 4s/step - loss: 0.2563
Epoch 16/50
56/56 [==============================] - 207s 4s/step - loss: 0.2230
Epoch 17/50
56/56 [==============================] - 209s 4s/step - loss: 0.1799
Epoch 18/50
56/56 [==============================] - 208s 4s/step - loss: 0.1402
Epoch 19/50
56/56 [==============================] - 208s 4s/step - loss: 0.1027
Epoch 20/50
56/56 [==============================] - 206s 4s/step - loss: 0.0694
Epoch 21/50
56/56 [==============================] - 208s 4s/step - loss: 0.0891
Epoch 22/50
56/56 [==============================] - 208s 4s/step - loss: 0.0693
Epoch 23/50
56/56 [==============================] - 208s 4s/step - loss: 0.0399
Epoch 24/50
56/56 [==============================] - 209s 4s/step - loss: 0.0281
Epoch 25/50
56/56 [==============================] - 213s 4s/step - loss: 0.0226
Epoch 26/50
56/56 [==============================] - 212s 4s/step - loss: 0.0201
Epoch 27/50
56/56 [==============================] - 208s 4s/step - loss: 0.0213
Epoch 28/50
56/56 [==============================] - 207s 4s/step - loss: 0.0165
Epoch 29/50
56/56 [==============================] - 210s 4s/step - loss: 0.0147
Epoch 30/50
56/56 [==============================] - 211s 4s/step - loss: 0.0136
Epoch 31/50
56/56 [==============================] - 208s 4s/step - loss: 0.0132
Epoch 32/50
56/56 [==============================] - 207s 4s/step - loss: 0.0123
Epoch 33/50
56/56 [==============================] - 212s 4s/step - loss: 0.0117
Epoch 34/50
56/56 [==============================] - 216s 4s/step - loss: 0.0115
Epoch 35/50
56/56 [==============================] - 214s 4s/step - loss: 0.0110
Epoch 36/50
56/56 [==============================] - 207s 4s/step - loss: 0.0105
Epoch 37/50
56/56 [==============================] - 212s 4s/step - loss: 0.0100
Epoch 38/50
56/56 [==============================] - 207s 4s/step - loss: 0.0111
Epoch 39/50
56/56 [==============================] - 208s 4s/step - loss: 0.1953
Epoch 40/50
56/56 [==============================] - 210s 4s/step - loss: 0.3640
Epoch 41/50
56/56 [==============================] - 207s 4s/step - loss: 0.0709
Epoch 42/50
56/56 [==============================] - 213s 4s/step - loss: 0.0237
Epoch 43/50
56/56 [==============================] - 209s 4s/step - loss: 0.0186
Epoch 44/50
56/56 [==============================] - 211s 4s/step - loss: 0.0133
Epoch 45/50
56/56 [==============================] - 210s 4s/step - loss: 0.0139
Epoch 46/50
56/56 [==============================] - 207s 4s/step - loss: 0.0107
Epoch 47/50
56/56 [==============================] - 211s 4s/step - loss: 0.0102
Epoch 48/50
56/56 [==============================] - 208s 4s/step - loss: 0.0097
Epoch 49/50
56/56 [==============================] - 208s 4s/step - loss: 0.0092
Epoch 50/50
56/56 [==============================] - 211s 4s/step - loss: 0.0088
```

Out[7]: `<tensorflow.python.keras.callbacks.History at 0x7fd704ba6f10>`

Plotting the Model

In [8]:
```python
plot_model(OCR, to_file='gan_plot.png', show_shapes=True, show_layer_names=True)
```

Out[8]:



input_1: InputLayer
input: [(None, 32, 784, 1)]
output: [(None, 32, 784, 1)]
conv2d: Conv2D
input: (None, 32, 784, 1)
output: (None, 32, 784, 16)
max_pooling2d: MaxPooling2D
input: (None, 32, 784, 16)
output: (None, 16, 392, 16)
conv2d_1: Conv2D
input: (None, 16, 392, 16)
output: (None, 16, 392, 32)
max_pooling2d_1: MaxPooling2D
input: (None, 16, 392, 32)
output: (None, 8, 196, 32)
conv2d_2: Conv2D
input: (None, 8, 196, 32)
output: (None, 8, 196, 64)
max_pooling2d_2: MaxPooling2D
input: (None, 8, 196, 64)
output: (None, 4, 98, 64)
conv2d_3: Conv2D
input: (None, 4, 98, 64)
output: (None, 4, 98, 128)
max_pooling2d_3: MaxPooling2D
input: (None, 4, 98, 128)
output: (None, 2, 98, 128)
conv2d_4: Conv2D
input: (None, 2, 98, 128)
output: (None, 2, 98, 256)
max_pooling2d_4: MaxPooling2D
input: (None, 2, 98, 256)
output: (None, 1, 98, 256)
conv2d_5: Conv2D
input: (None, 1, 98, 256)
output: (None, 1, 98, 256)
batch_normalization: BatchNormalization
input: (None, 1, 98, 256)
output: (None, 1, 98, 256)
tf.compat.v1.squeeze: TFOpLambda
input: (None, 1, 98, 256)
output: (None, 98, 256)
bidirectional: Bidirectional
input: (None, 98, 256)
output: (None, 98, 512)
bidirectional_1: Bidirectional
input: (None, 98, 512)
output: (None, 98, 512)
tf.expand_dims: TFOpLambda
input: (None, 98, 512)
output: (None, 98, 1, 512)
conv2d_6: Conv2D
input: (None, 98, 1, 512)
output: (None, 98, 1, 82)
tf.compat.v1.squeeze_1: TFOpLambda
input: (None, 98, 1, 82)
output: (None, 98, 82)
softmax: Softmax
input: (None, 98, 82)
output: (None, 98, 82)

In [9]:
```python
arr=OCR.predict(X)
index=466
c=""
print(len(arr[0]))
for i in range(len(arr[0])):
    c=c+(symbols[np.argmax(arr[index][i])])
```

```
98
```

Seeing the Output of the Model and comparing it with the original output.

In [15]:
```python
print("predicted:",c.strip())
print("\nOrignal:",outputs[index])
plt.imshow(X[index][:,:,0])
```

```
predicted: fileer haat repacsss the poxe vvalss  wwihtttenniigh

Orignal:  filter that replaces the pixel values with the neighb
```

Out[15]: `<matplotlib.image.AxesImage at 0x7fd6ff7e0210>`