1. Подключаем нужные библиотеки

```
import os
import cv2 #load images
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf #machine learning
```

2. Загружаем датасет из библиотеки

```
mnist = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

3. Извлекаем данные из полученного датасета

```
(x_train, y_train), (x_test, y_test) = mnist
```

```
print(y_test)
```

```
[7 2 1 ... 4 5 6]
```

4. Создаем последовательную модель нейронной сети, добавляя слои и вид функции активации:

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
model.add(tf.keras.layers.Dense(512, activation='sigmoid'))
model.add(tf.keras.layers.Dense(192, activation='sigmoid'))
model.add(tf.keras.layers.Dense(128, activation='sigmoid'))
model.add(tf.keras.layers.Dropout(0.25))
model.add(tf.keras.layers.Dense(512, activation='sigmoid'))
model.add(tf.keras.layers.Dense(192, activation='sigmoid'))
model.add(tf.keras.layers.Dense(128, activation='sigmoid'))
model.add(tf.keras.layers.Dropout(0.25))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

5. Компилируем и обучаем нейросеть на 5 эпохах

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))
```

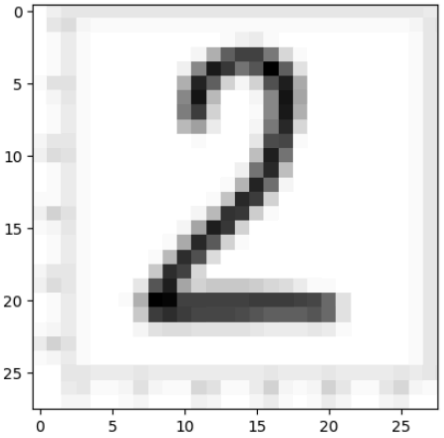
```
Epoch 1/5
1875/1875 [==============================] - 22s 11ms/step - loss: 0.9867 - accuracy: 0.6472 - val_loss: 0.6449 - val_accuracy: 0.7823
Epoch 2/5
1875/1875 [==============================] - 21s 11ms/step - loss: 0.5120 - accuracy: 0.8429 - val_loss: 0.3914 - val_accuracy: 0.8845
Epoch 3/5
1875/1875 [==============================] - 21s 11ms/step - loss: 0.4052 - accuracy: 0.8816 - val_loss: 0.3370 - val_accuracy: 0.9022
Epoch 4/5
1875/1875 [==============================] - 21s 11ms/step - loss: 0.3474 - accuracy: 0.8986 - val_loss: 0.3121 - val_accuracy: 0.9111
Epoch 5/5
1875/1875 [==============================] - 21s 11ms/step - loss: 0.3061 - accuracy: 0.9107 - val_loss: 0.2702 - val_accuracy: 0.9209
<keras.callbacks.History at 0x7f68cc5b7010>
```

6. Загружаем изображения (2.png, 4.png, 44.png, 6.png, 7.png, 9.png), преобразуем их в формат, пригодный для использования моделью нейронной сети, и затем используем обученную модель для классификации изображений:

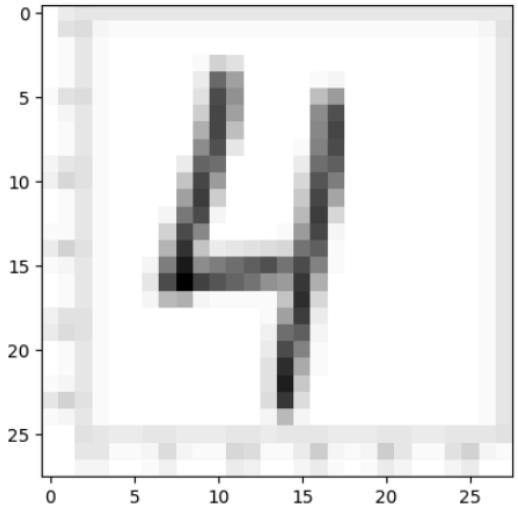
```
np.set_printoptions(suppress=True)
list = ['2.png', '4.png', '44.png', '6.png', '7.png', '9.png']
for img in list:
  img = cv2.imread(img)[:,:,0]
  img = np.invert(np.array([img]))
  prediction = model.predict(img)
  np.set_printoptions(suppress=True)
  print(prediction)
  print(np.argmax(prediction))
  print("")
  plt.imshow(img[0], cmap=plt.cm.binary)
  plt.show()
```

7. Получаем результат распознанных изображений

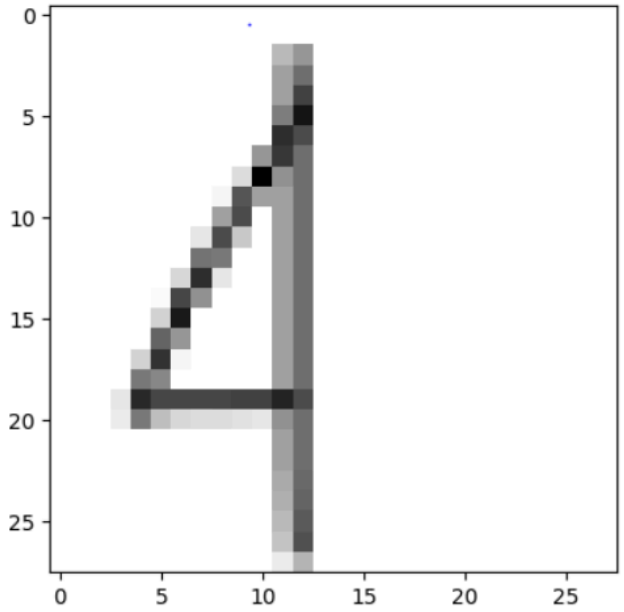
```
1/1 [==============================] - 0s 21ms/step
[[0.00012114 0.01224995 0.9777123  0.00791066 0.00006727 0.00006217
  0.00027242 0.00095008 0.00064402 0.00001002]]
2
```



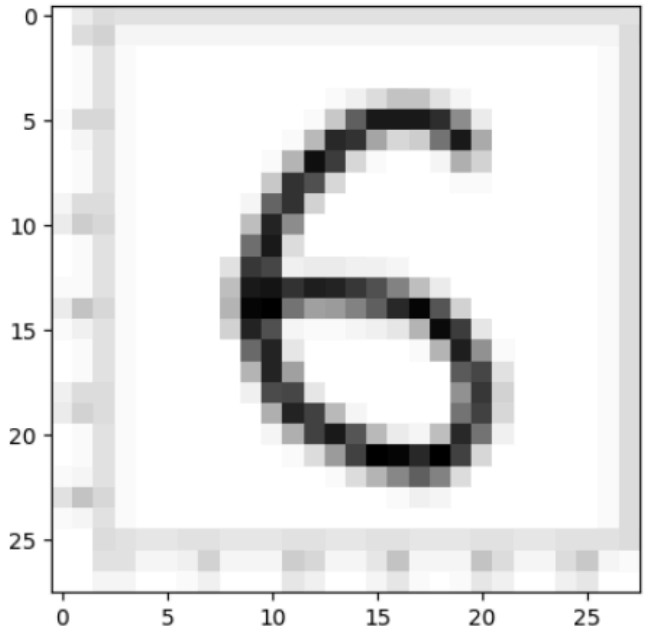

```
1/1 [==============================] - 0s 31ms/step
[[0.00042579 0.00009086 0.00337806 0.00014088 0.95575565 0.0003805
  0.00039667 0.01134276 0.0005306  0.02755828]]
4
```



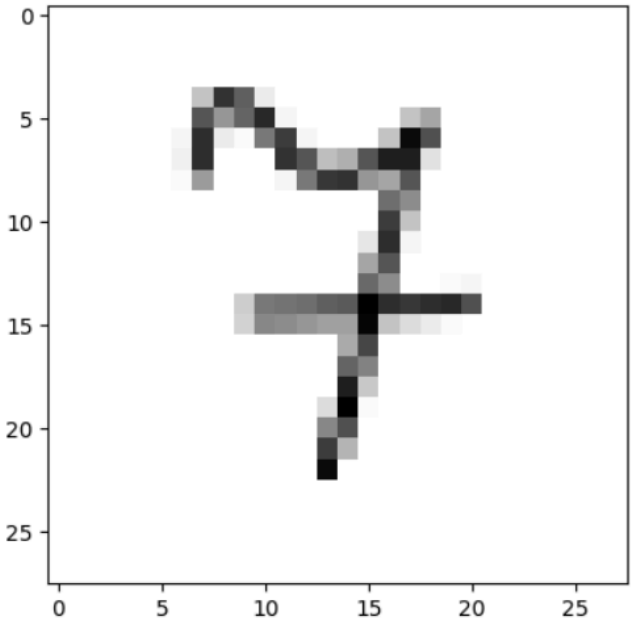
```
1/1 [==============================] - 0s 30ms/step
[[0.12151208 0.00183297 0.00048865 0.00122067 0.4990972  0.00851228
  0.31714988 0.00341984 0.00362473 0.04314167]]
4
```



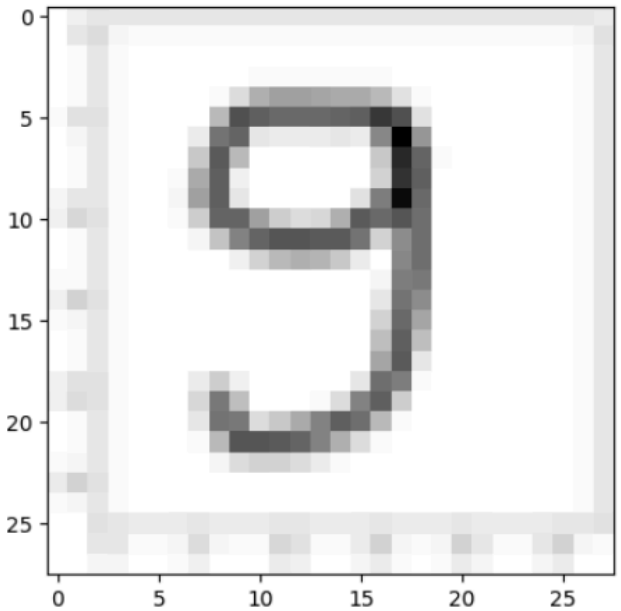

```
1/1 [==============================] - 0s 28ms/step
[[0.00530894 0.00047738 0.00065188 0.0028057  0.01042578 0.49987534
  0.43800634 0.0001596  0.04130796 0.00098114]]
5
```



```
1/1 [==============================] - 0s 16ms/step
[[0.00004451 0.00592084 0.32255873 0.09355085 0.00559549 0.00054621
  0.00003532 0.5690527  0.00268974 0.00000569]]
7
```




```
1/1 [==============================] - 0s 29ms/step
[[0.00097146 0.00112701 0.00349811 0.9285838  0.00005219 0.0561353
  0.00004571 0.00034048 0.0046571  0.00458883]]
3
```



Успешно распознаны были 4 из 6 изображений, процент успеха составил 66%.