Никонов Иван, группа АиСИС

Лабораторная работа № 4

В данной лабораторной необходимо продемонстрировать работу с библиотеками языка Python Tensorflow и Keras, предназначенные для машинного обучения. Для чего нужно построить и обучить модель для распознавания цифр. После чего убедится в ее работе на примере созданных файлов с корректными и некорректными изображениями.

Было подготовлен датасет из десяти корректных изображений (по одному на каждую цифру) и пяти некорректных.

Вывод программы:

Test loss: 0.05498858541250229

Test accuracy: 0.9811999797821045

Корректные картинки

Предсказана цифра: 0 с вероятностью 1.0000, На самом деле: 0

Предсказана цифра: 1 с вероятностью 0.9991, На самом деле: 1

Предсказана цифра: 2 с вероятностью 1.0000, На самом деле: 2

Предсказана цифра: 3 с вероятностью 1.0000, На самом деле: 3

Предсказана цифра: 4 с вероятностью 0.9999, На самом деле: 4

Предсказана цифра: 5 с вероятностью 1.0000, На самом деле: 5

Предсказана цифра: 6 с вероятностью 0.9227, На самом деле: 6

Предсказана цифра: 2 с вероятностью 0.6178, На самом деле: 7

Предсказана цифра: 8 с вероятностью 0.9500, На самом деле: 8

Предсказана цифра: 3 с вероятностью 0.7281, На самом деле: 9

Не корректные картинки

Предсказана цифра: 2 с вероятностью 0.9657, На самом деле: Вопрос

Предсказана цифра: 1 с вероятностью 0.9903, На самом деле: Восклицание

Предсказана цифра: 8 с вероятностью 0.4635, На самом деле: Звезда

Предсказана цифра: 1 с вероятностью 0.9368, На самом деле: Крестик

Предсказана цифра: 2 с вероятностью 0.9985, На самом деле: Улыбка

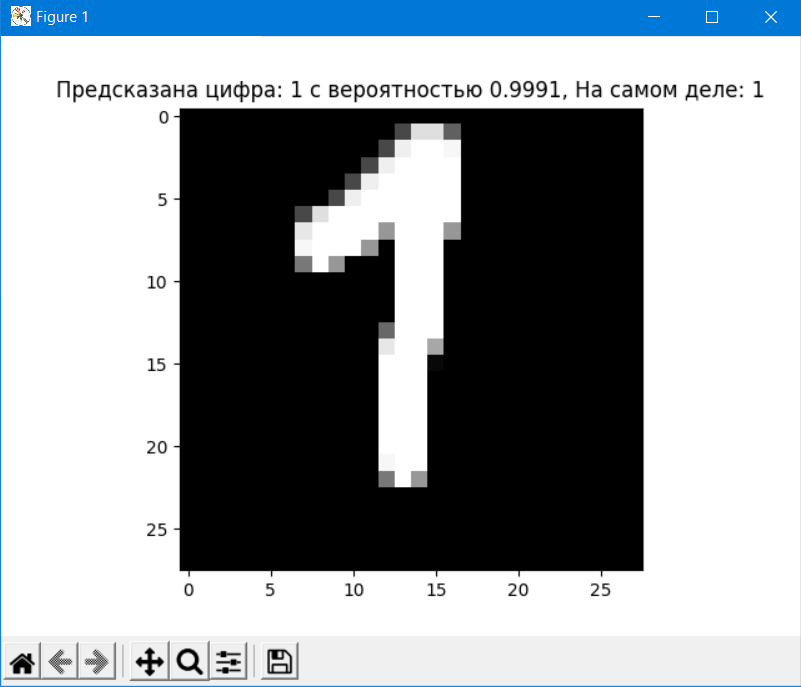
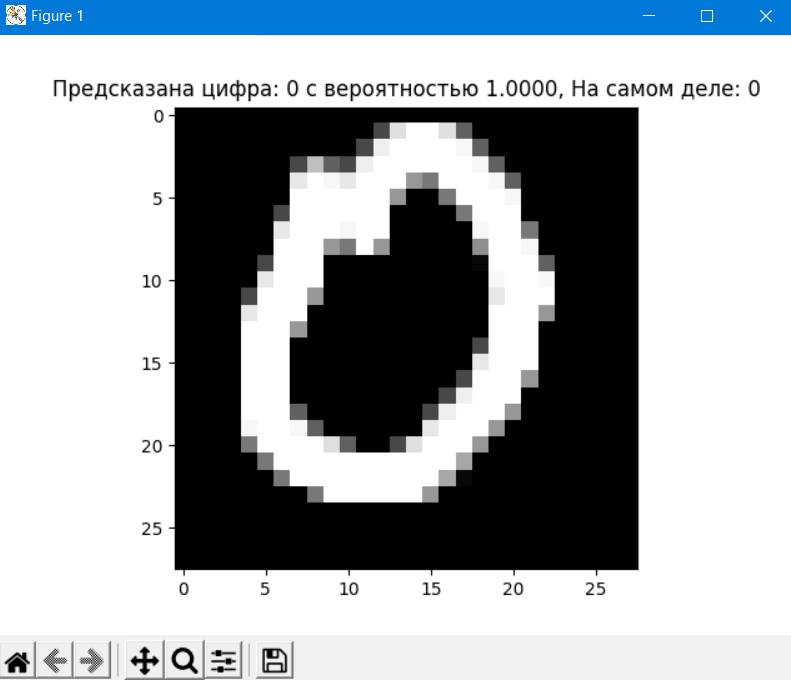
Можно сделать следующие выводы: цифра 7 похожа на 2, и в связи с этим, модель определила первую как вторую в большой вероятностью. Цифра 9 похожа на цифру 3, но меньше чем 7 на 2.

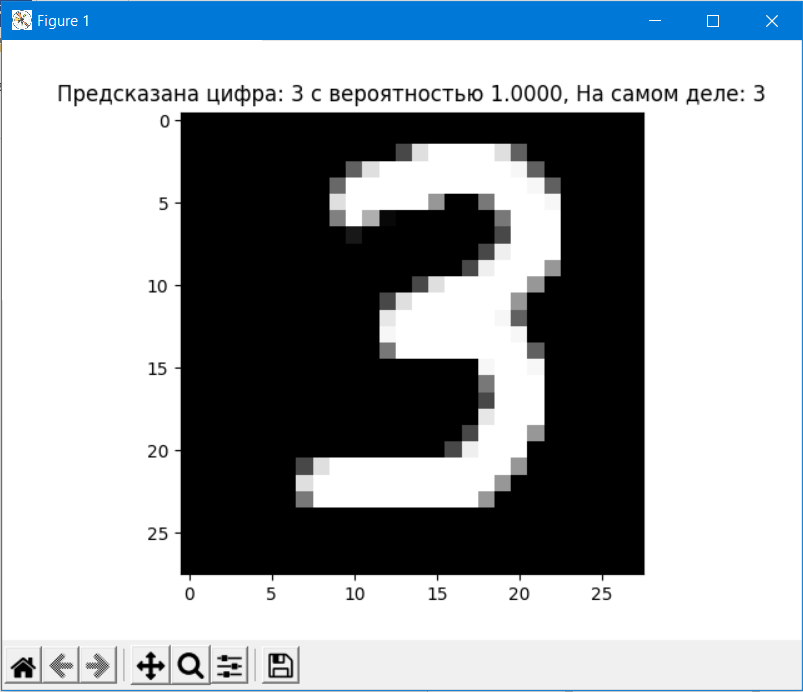
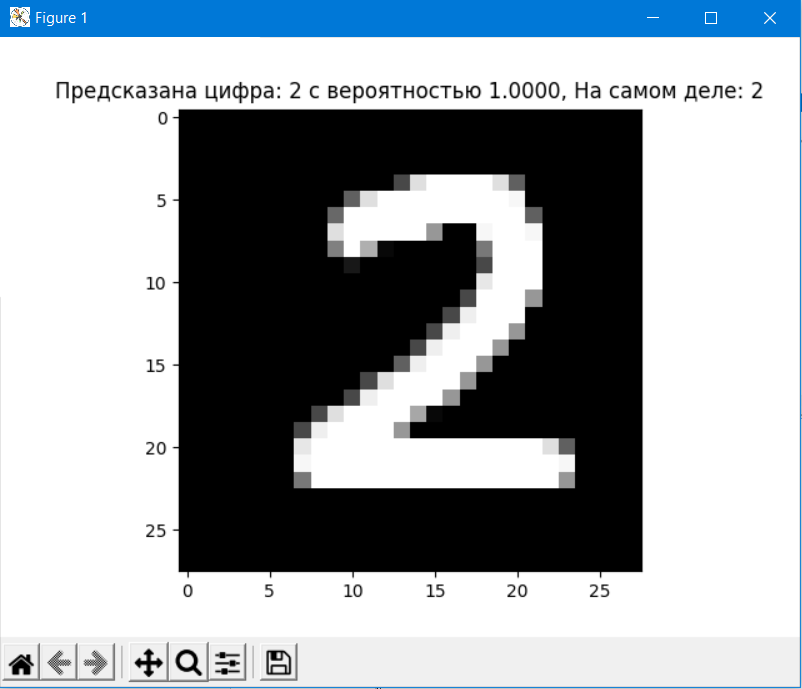
Символ вопросительного и восклицательного знака похожи на цифры 2 и 1 соответственно, в связи с этим модель выдала большую вероятность нахождения на изображениях именно их.

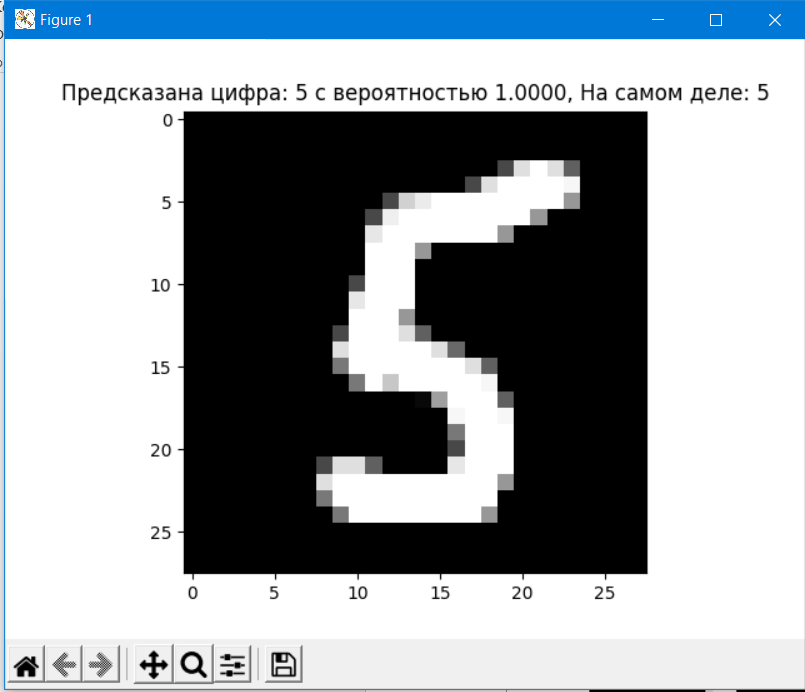
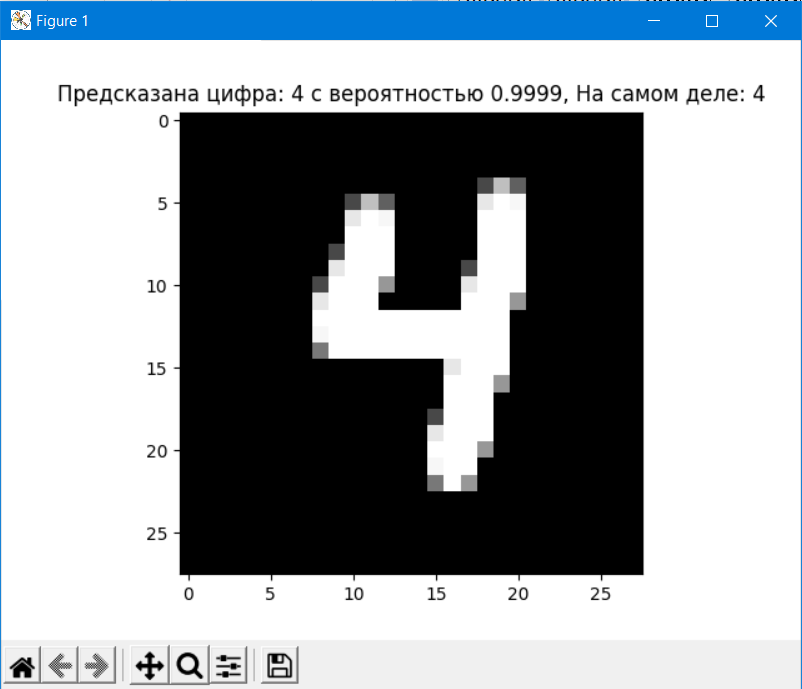
Рисунок «звезда» определился как цифра 8, но с весьма малой вероятностью.

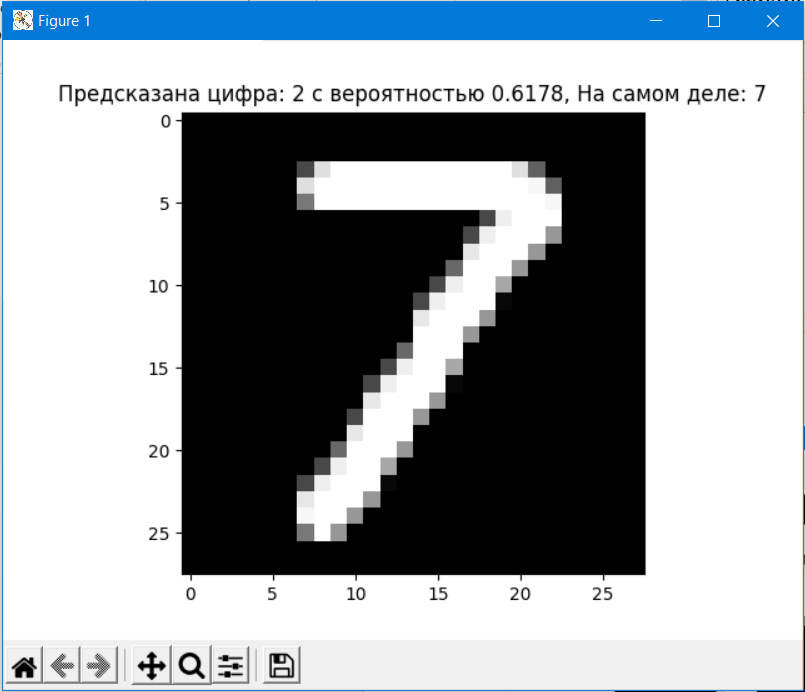
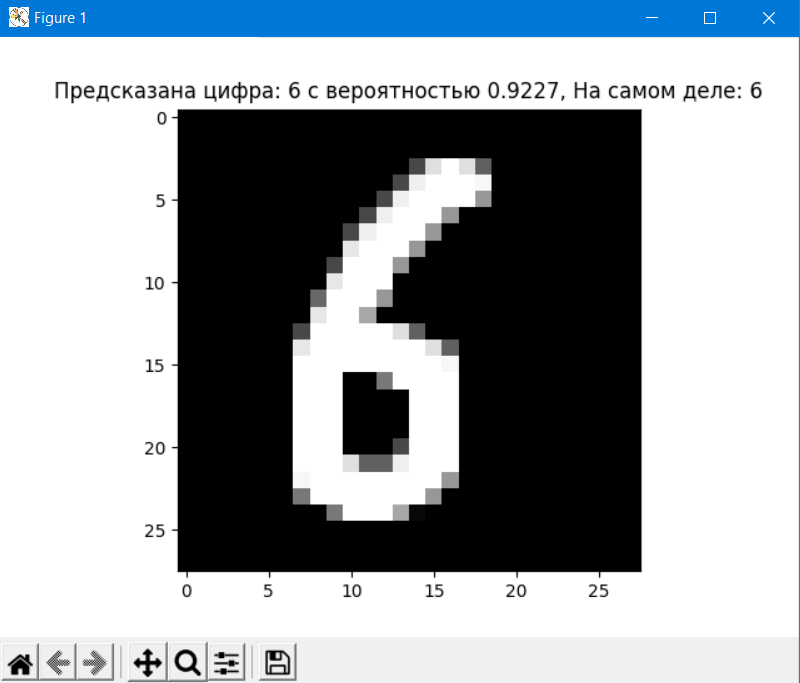
Но рисунки «крестик» и «улыбка» определились с большой вероятностью как цифры 1 и 2 соответственно, что вызывает вопросы, потому как они не сильно похожи на определяемые цифры.

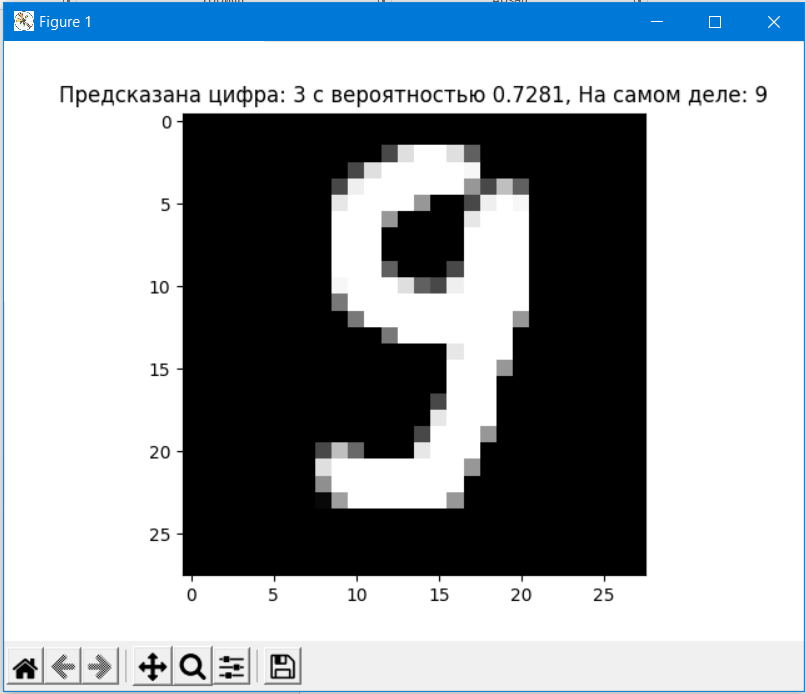
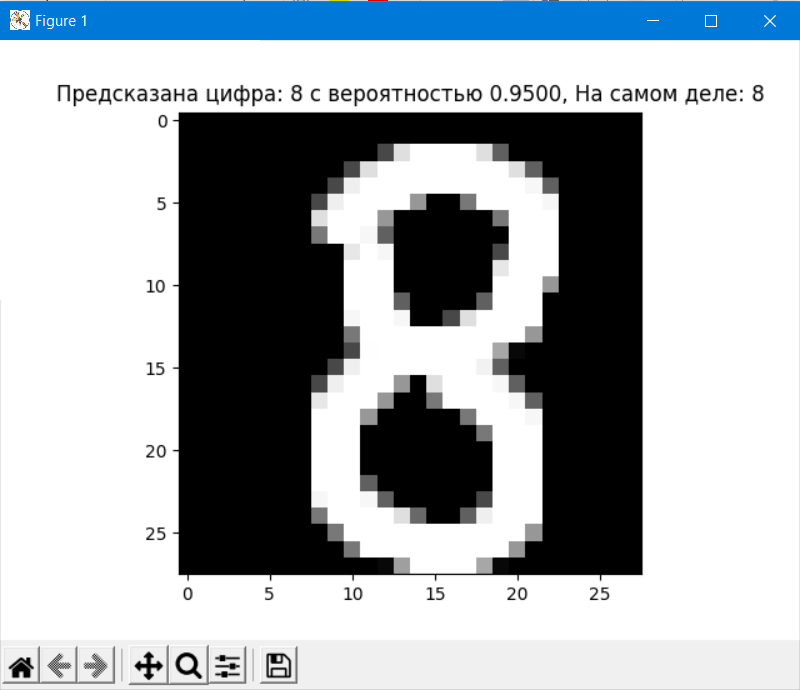
Ниже представлен вывод программы с картинками и вероятностями определения.

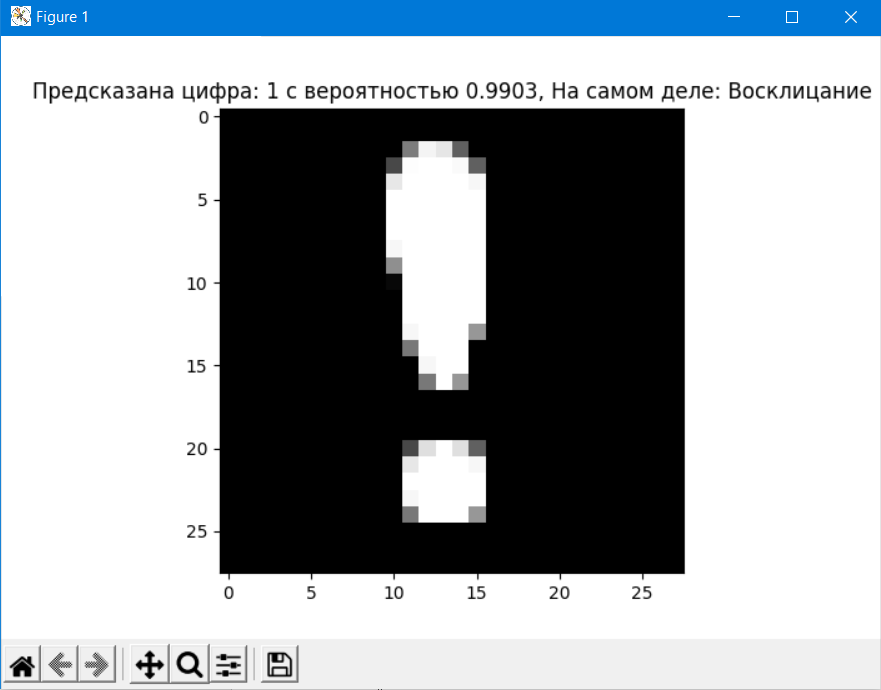
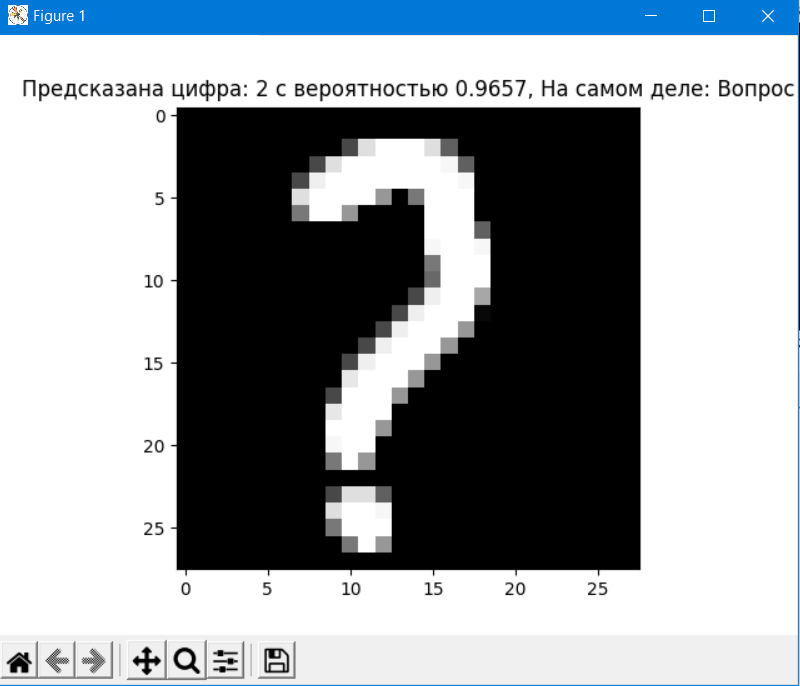


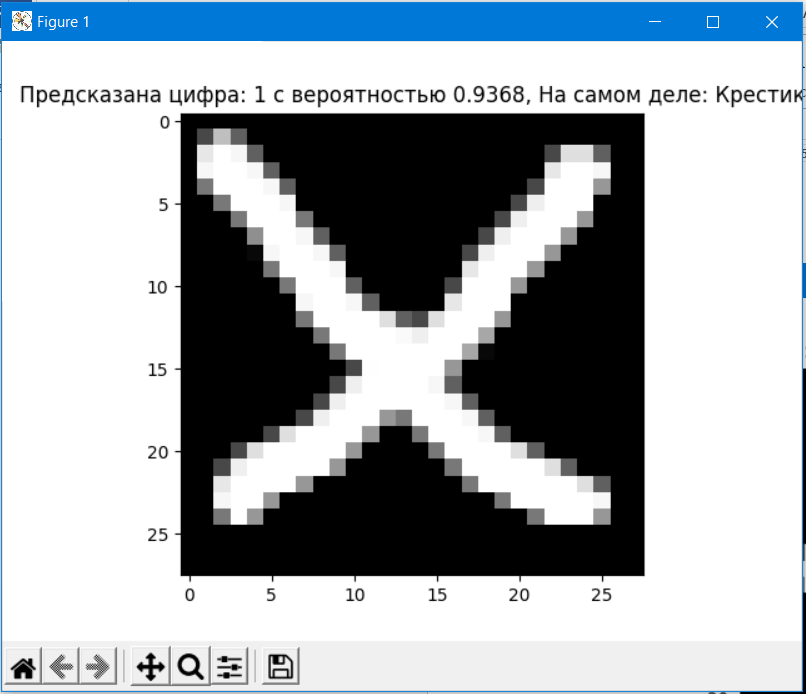
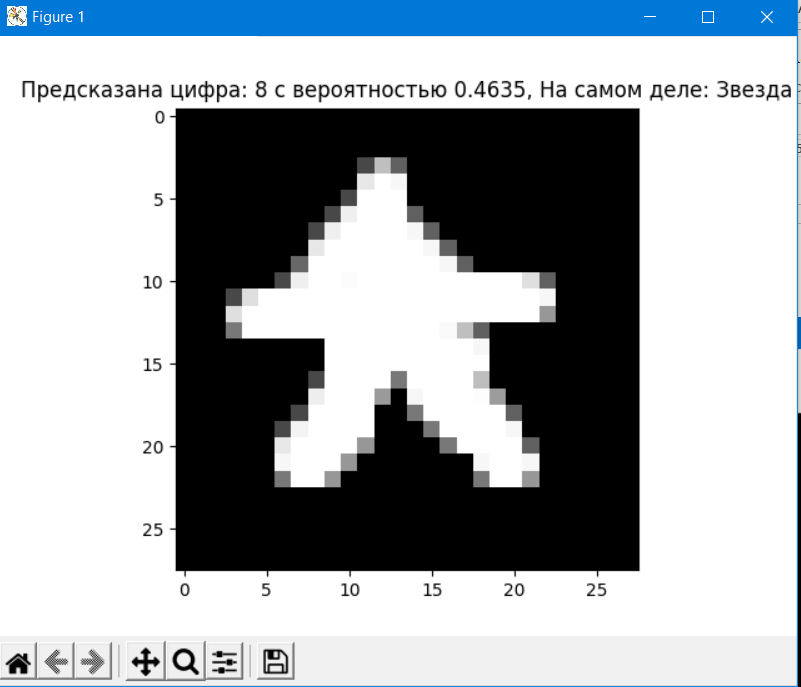


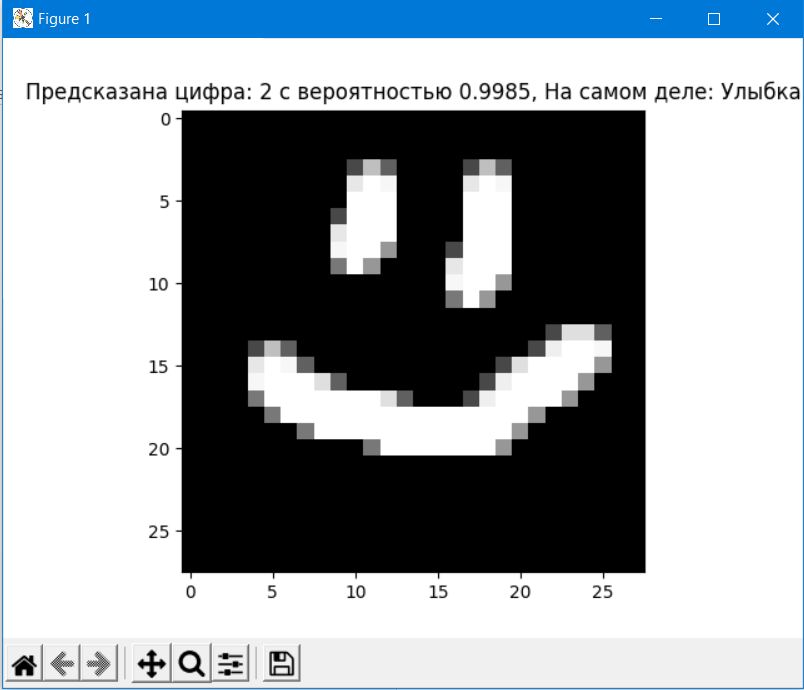












Исходный код основного исполняемого файла:

import numpy as np  
from tensorflow import keras  
from tensorflow.keras import layers  
from img\_loader import load\_correct\_images, load\_incorrect\_images  
import matplotlib.pyplot as plt  
  
*# Model / data parameters*num\_classes = 10  
input\_shape = (28, 28, 1)  
  
*# the data, split between train and test sets*(x\_train, y\_train), (x\_test, y\_test) = keras.datasets.mnist.load\_data()  
  
*# Scale images to the [0, 1] range*x\_train = x\_train.astype(**"float32"**) / 255  
x\_test = x\_test.astype(**"float32"**) / 255  
*# Make sure images have shape (28, 28, 1)*x\_train = np.expand\_dims(x\_train, -1)  
x\_test = np.expand\_dims(x\_test, -1)  
print(**"x\_train shape:"**, x\_train.shape)  
print(x\_train.shape[0], **"train samples"**)  
print(x\_test.shape[0], **"test samples"**)  
  
  
*# convert class vectors to binary class matrices*y\_train = keras.utils.to\_categorical(y\_train, num\_classes)  
y\_test = keras.utils.to\_categorical(y\_test, num\_classes)  
  
*# Build the model*model = keras.Sequential(  
 [  
 keras.Input(shape=input\_shape),  
 layers.Conv2D(32, kernel\_size=(3, 3), activation=**"relu"**),  
 layers.MaxPooling2D(pool\_size=(2, 2)),  
 layers.Conv2D(64, kernel\_size=(3, 3), activation=**"relu"**),  
 layers.MaxPooling2D(pool\_size=(2, 2)),  
 layers.Flatten(),  
 layers.Dropout(0.5),  
 layers.Dense(num\_classes, activation=**"softmax"**),  
 ]  
)  
  
*# train*batch\_size = 128  
epochs = 3  
  
model.compile(loss=**"categorical\_crossentropy"**, optimizer=**"adam"**, metrics=[**"accuracy"**])  
  
model.fit(x\_train, y\_train, batch\_size=batch\_size, epochs=epochs, validation\_split=0.1)  
model.summary()  
  
*# evaluate*score = model.evaluate(x\_test, y\_test, verbose=0)  
print(**"Test loss:"**, score[0])  
print(**"Test accuracy:"**, score[1])  
  
print(**"Корректные картинки"**)  
for img\_entry in load\_correct\_images():  
 img = img\_entry[**'img'**]  
 pred = model.predict(img)  
 max\_val = 0  
 max\_idx = 0  
 for idx, val in enumerate(pred[0]):  
 if val > max\_val:  
 max\_val = val  
 max\_idx = idx  
  
 max\_val\_str = **"%.4f"** % max\_val  
 info = **f'Предсказана цифра:** {max\_idx} **с вероятностью** {max\_val\_str}**, На самом деле:** {img\_entry[**"label"**]}**'** print(info)  
 plt.title(info)  
 plt.imshow(img\_entry[**'raw'**], cmap=plt.cm.binary)  
 plt.show()  
  
print(**"Не корректные картинки"**)  
for img\_entry in load\_incorrect\_images():  
 img = img\_entry[**'img'**]  
 pred = model.predict(img)  
 max\_val = 0  
 max\_idx = 0  
 for idx, val in enumerate(pred[0]):  
 if val > max\_val:  
 max\_val = val  
 max\_idx = idx  
  
 max\_val\_str = **"%.4f"** % max\_val  
 info = **f'Предсказана цифра:** {max\_idx} **с вероятностью** {max\_val\_str}**, На самом деле:** {img\_entry[**"label"**]}**'** print(info)  
 plt.title(info)  
 plt.imshow(img\_entry[**'raw'**], cmap=plt.cm.binary)  
 plt.show()

Исходный код файла img\_loader.py предназначенный для загрузки изображений:

import os  
from PIL import Image  
import numpy as np  
import matplotlib.pyplot as plt  
  
correct\_images\_path = os.path.join(os.path.curdir, **'imgs\_correct'**)  
incorrect\_images\_path = os.path.join(os.path.curdir, **'imgs\_incorrect'**)  
  
  
def load\_correct\_images():  
 result = []  
 for img\_name in get\_file\_list(correct\_images\_path):  
 img\_path = os.path.join(correct\_images\_path, img\_name)  
 img = Image.open(img\_path).convert(**'L'**)  
 im2arr = convert\_image(img)  
 result.append({**'raw'**: img, **'img'**: im2arr, **'label'**: img\_name[:1]})  
  
 return result  
  
  
def load\_incorrect\_images():  
 result = []  
 for img\_name in get\_file\_list(incorrect\_images\_path):  
 img\_path = os.path.join(incorrect\_images\_path, img\_name)  
 img = Image.open(img\_path).convert(**'L'**)  
 im2arr = convert\_image(img)  
 result.append({**'raw'**: img, **'img'**: im2arr, **'label'**: img\_name[:-4]})  
  
 return result  
  
  
def convert\_image(image):  
 resized\_img = np.resize(image, (28, 28, 1))  
 im2arr = np.array(resized\_img)  
 im2arr = im2arr.reshape(1, 28, 28, 1)  
 im2arr = (255 - im2arr.astype(**'float32'**)) / 255  
 return im2arr  
  
  
def get\_file\_list(dir\_path):  
 return [f for f in os.listdir(dir\_path) if os.path.isfile(os.path.join(dir\_path, f)) and f[-4:] == **'.png'**]