Котолевский Максим Николаевич, группа 19

Лабораторная работа № 1

Вариант № 1

Цель работы

Необходимо реализовать классификацию черно-белых изображений, которые представляют из себя рукописные цифры. Эти рукописные цифры представлены изображениями размерностью 28x28 пикселей по 10-ти категориям.

Задание

1. Разработать ПО для реализации задачи распознавания рукописных цифр.
2. Создать примеры, показывающую работу сети (сеть распознает верно).
3. Создать примеры, показывающую работу сети (сеть распознает не верно).

Код программы (внесённые изменения в шаблон кода выделены)

```
import cv2
import numpy as np
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data
()
```

```
train_images.shape
len(train_labels)
train_labels
test_images.shape
len(test_labels)
test_labels
```

```
from keras import models
from keras import layers
network = models.Sequential() //последовательная модель данных
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
// первый слой, модуль обработки данных
network.add(layers.Dense(10, activation='softmax')) //второй слой
```

```
network.compile(optimizer='rmsprop',
 loss='categorical_crossentropy',
 metrics=['accuracy'])
//компилируем сеть (оптимизатор, функция потерь и метрика)
```

```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255
//масштабируем в нужный диапазон от 0 до 1
```

```
from tensorflow.keras.utils import to_categorical
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
//кодируем метки категорий
```

```
network.fit(train_images, train_labels, epochs=5, batch_size=128)
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_acc:', test_acc)
//подаем обучающие данные
```

```
digit = train_images[100]
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```

```
digit = cv2.imread('1_1.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```

```
digit = cv2.imread('1_2.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```

```
digit = cv2.imread('3_2.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```

```
digit = cv2.imread('3_1.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```

```
digit = cv2.imread('4_1.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
```

```
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```

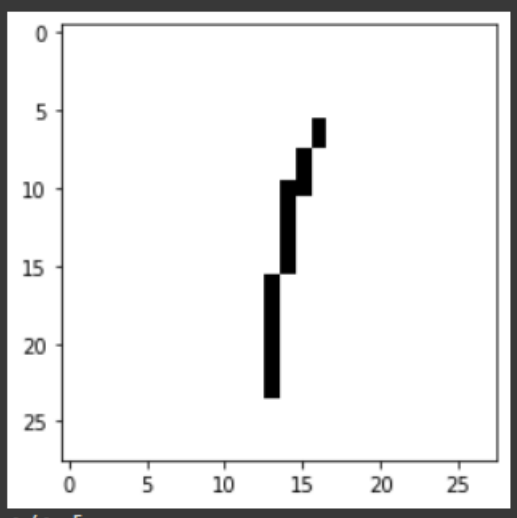
```
digit = cv2.imread('4_2.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```

Результаты выполнения задания

Было разработано ПО для реализации задачи распознования рукописных цифр. Были созданы примеры, показывающие работу сети.

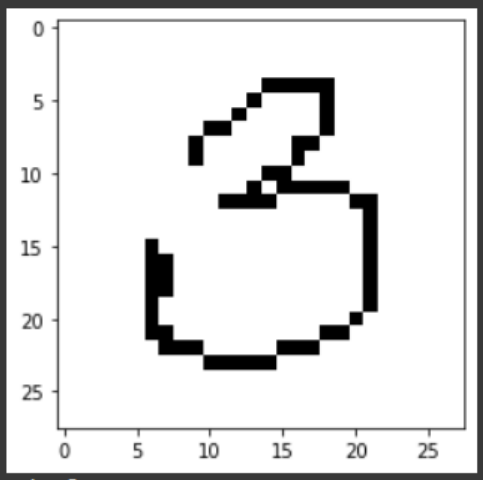
Сеть распознает верно:

```
digit = cv2.imread('1_2.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```



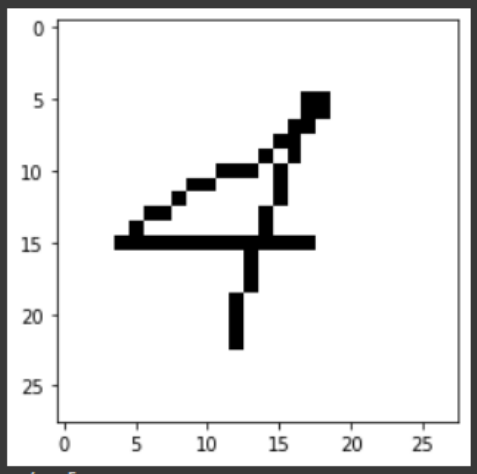

```
1/1 [==============================] - 0s 18ms/step
Digit:  1
```

```
digit = cv2.imread('3_1.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```




```
1/1 [==============================] - 0s 13ms/step
Digit:  3
```

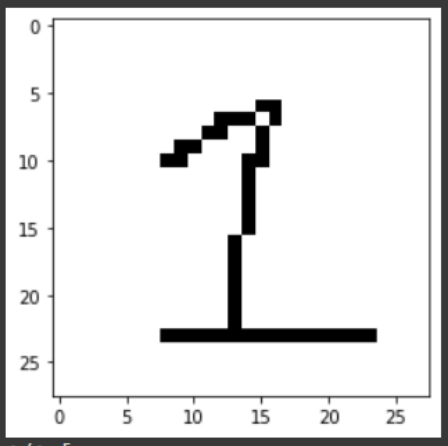
```
digit = cv2.imread('4_2.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```




```
1/1 [==============================] - 0s 21ms/step
Digit:  4
```

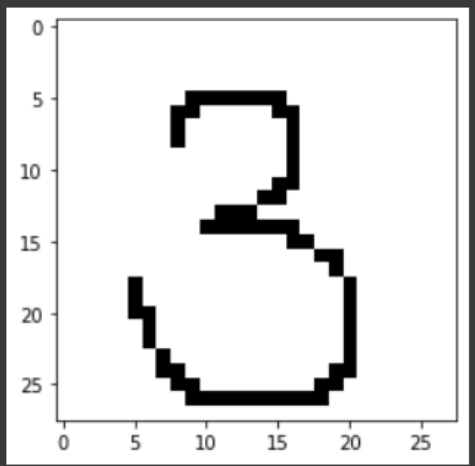
Сеть распознает неверно:

```
digit = cv2.imread('1_1.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```



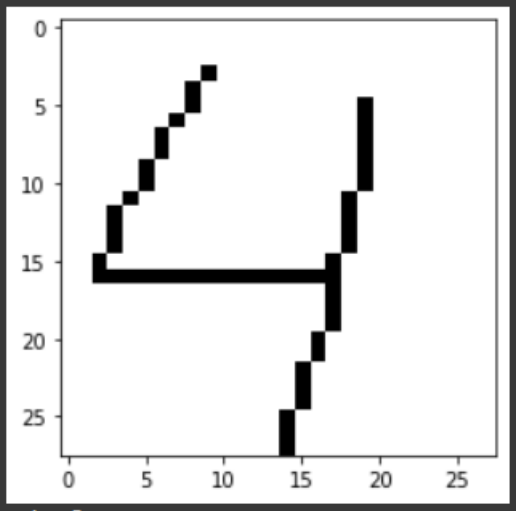
```
1/1 [==============================] - 0s 33ms/step
Digit:  2
```

```
digit = cv2.imread('3_2.png', cv2.IMREAD_GRAYSCALE)
digit = cv2.bitwise_not(digit)
import matplotlib.pyplot as plt
digit = digit.reshape(28, 28)
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
pred = network.predict(digit.reshape(1,28*28))
print('Digit: ', pred.argmax())
```



```
1/1 [==============================] - 0s 21ms/step
Digit:  9
```

```
[47] digit = cv2.imread('4_1.png', cv2.IMREAD_GRAYSCALE)
     digit = cv2.bitwise_not(digit)
     import matplotlib.pyplot as plt
     digit = digit.reshape(28, 28)
     plt.imshow(digit, cmap=plt.cm.binary)
     plt.show()
     pred = network.predict(digit.reshape(1,28*28))
     print('Digit: ', pred.argmax())
```




```
1/1 [==============================] - 0s 16ms/step
Digit:  7
```