

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторная работа № 4

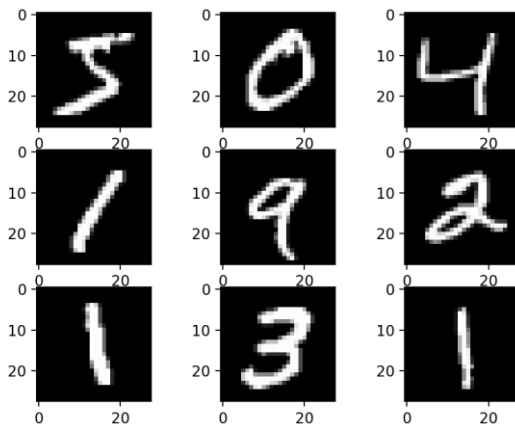
тематика

Распознавание рукописных символов

Москва 2021

Цель

Реализовать классификацию черно-белых изображений рукописных цифр (28x28) по 10 категориям (от 0 до 9).



Набор данных содержит 60,000 изображений для обучения и 10,000 изображений для тестирования.

Подробнее: <https://keras.io/datasets/> <http://yann.lecun.com/exdb/mnist/>

Задачи

- Ознакомиться с представлением графических данных
- Ознакомиться с простейшим способом передачи графических данных нейронной сети
- Создать модель
- Настроить параметры обучения
- Написать функцию, позволяющую загружать изображение пользователя и классифицировать его

Выполнение работы

Набор данных MNIST уже входит в состав Keras в форме набора из четырех массивов Numpy.

Листинг 1:

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

Здесь `train_images` и `train_labels` — это тренировочный набор, то есть данные, необходимые для обучения. После обучения модель будет проверяться тестовым (или контрольным) набором, `test_images` и `test_labels`. Изображения хранятся в массивах Numpy, а метки — в массиве цифр от 0 до 9. Изображения и метки находятся в прямом соответствии, один к одному.

Для проверки корректности загрузки достаточно сравнить тестовое изображение с его меткой.

Листинг 2:

```
import matplotlib.pyplot as plt
plt.imshow(train_images[0], cmap=plt.cm.binary)
plt.show()
```

```
print(train_labels[0])
```

Исходные изображения представлены в виде массивов чисел в интервале [0, 255]. Перед обучением их необходимо преобразовать так, чтобы все значения оказались в интервале [0, 1].

Листинг 3:

```
train_images = train_images / 255.0  
test_images = test_images / 255.0
```

Также необходимо закодировать метки категорий. В данном случае прямое кодирование меток заключается в конструировании вектора с нулевыми элементами со значением 1 в элементе, индекс которого соответствует индексу метки.

Листинг 4:

```
from keras.utils import to_categorical  
train_labels = to_categorical(train_labels)  
test_labels = to_categorical(test_labels)
```

Теперь можно задать базовую архитектуру сети.

Листинг 5:

```
from tensorflow.keras.layers import Dense, Activation, Flatten  
from tensorflow.keras.models import Sequential  
model = Sequential()  
model.add(Flatten())  
model.add(Dense(256, activation='relu'))  
model.add(Dense(10, activation='softmax'))
```

Чтобы подготовить сеть к обучению, нужно настроить еще три параметра для этапа компиляции:

1. функцию потерь, которая определяет, как сеть должна оценивать качество своей работы на обучающих данных и, соответственно, как корректировать ее в правильном направлении;
2. оптимизатор — механизм, с помощью которого сеть будет обновлять себя, опираясь на наблюдаемые данные и функцию потерь;
3. метрики для мониторинга на этапах обучения и тестирования — здесь нас будет интересовать только точность (доля правильно классифицированных изображений).

Листинг 6:

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

Теперь можно начинать обучение сети, для чего в случае использования библиотеки Keras достаточно вызвать метод `fit` сети — он пытается адаптировать (fit) модель под обучающие данные.

Листинг 7:

```
model.fit(train_images, train_labels, epochs=5, batch_size=128)
```

В процессе обучения отображаются две величины: потери сети на обучающих данных и точность сети на обучающих данных.

Теперь проверим, как модель распознает контрольный набор:

Листинг 8:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)  
print('test_acc:', test_acc)
```

Требования

1. Найти архитектуру сети, при которой точность классификации будет не менее 95%
2. Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения
3. Написать функцию, которая позволит загружать пользовательское изображение не из датасета