# **8.1. Использование обученных нейронных сетей для распознавания объектов (сверточная сеть)**

# 

( <https://www.asozykin.ru/courses/nnpython> )

1). Использование обученной сверточной нейронной сети для распознавания цифр

Для распознавания рукописных цифр воспользуемся [сверточной нейронной сетью](https://www.asozykin.ru/deep_learning/2017/05/08/CNN-for-MNIST.html).

Keras содержит специальный модуль для работы с изображениями, который называется image. С его помощью можно быстро загрузить изображение из файла и преобразовать его в массив numpy, который мы можем передать модели для распознавания:

import numpy **as** np

from keras.preprocessing import image

*# Загружаем изображение*

img\_path **=** '2.png'

img **=** image**.**load\_img(img\_path, target\_size**=**(28, 28), grayscale**=**True)

*# Преобразуем изображением в массив numpy*

x **=** image**.**img\_to\_array(img)

*# Инвертируем и нормализуем изображение*

x **=** 255 **-** x

x **/=** 255

x **=** np**.**expand\_dims(x, axis**=**0)

Загружаем из файла обученную [сверточную нейронную сеть](https://www.asozykin.ru/deep_learning/2017/05/08/CNN-for-MNIST.html):

json\_file **=** open("mnist\_model.json", "r")

loaded\_model\_json **=** json\_file**.**read()

json\_file**.**close()

loaded\_model **=** model\_from\_json(loaded\_model\_json)

loaded\_model**.**load\_weights("mnist\_model.h5")

Компилируем модель перед использованием

loaded\_model**.**compile(loss**=**"categorical\_crossentropy", optimizer**=**"adam", metrics**=**["accuracy"])

Теперь мы готовы к распознаванию цифры. Для распознавания вызываем метод модели predict:

prediction **=** loaded\_model**.**predict(x)

Модель выдает массив из 10 значений в формате One Hot Encoding. Выбираем индекс максимального значения и печатаем его:

**print**(np**.**argmax(prediction))

[2]

**Таким образом, мы выполнили следующие действия:**

# Загружаем сохраненную модель и компилируем с параметрами, используемыми при ее обучении

# Проверяем модель на тестовых данных

import numpy as np

from keras.preprocessing import image

# Загружаем свою картинку

img\_path = ('7.png')

img = image.load\_img(img\_path, target\_size=(28, 28), greyscale=True)

plt.imshow(img, cmap=’grey’)

plt.show()

# Преобразуем картинку в массив и нормализуем

x = image.img\_to\_array(img)

x = 255 - x

x /= 255

x = np.expand\_dims(x, axis=0)

prediction = loaded\_model.predict(x)

print(prediction)

prediction = numpy.argmax(prediction, axis=1)

print(prediction)

2). Использование обученной сверточной нейронной сети для распознавания объектов на изображениях

Давайте рассмотрим более сложную задачу - распознавание объектов на изображении. Будем использовать нейронную сеть, обученную на [наборе данных CIFAR-10](https://www.asozykin.ru/courses/nnpython-lab2). Попробуем распознать картинку самолета, которую сеть не видела в процессе обучения.

import numpy **as** np

from keras.preprocessing import image

img\_path **=** 'plane.jpg'

img **=** image**.**load\_img(img\_path, target\_size**=**(32, 32))

В отличие от рукописных цифр, в этот раз изображение цветное и его размер 32х32, в соответствии с форматом CIFAR-10. Преобразуем картинку в массив numpy:

x **=** image**.**img\_to\_array(img)

x **/=** 255

x **=** np**.**expand\_dims(x, axis**=**0)

Загружаем сеть, обученную на [наборе данных CIFAR-10](https://www.asozykin.ru/courses/nnpython-lab2), и компилируем модель:

json\_file **=** open("cifar10\_model.json", "r")

loaded\_model\_json **=** json\_file**.**read()

json\_file**.**close()

loaded\_model **=** model\_from\_json(loaded\_model\_json)

loaded\_model**.**load\_weights("cifar10\_model.h5")

loaded\_model**.**compile(loss**=**'categorical\_crossentropy', optimizer**=**'adam', metrics**=**['accuracy'])

Запускаем распознавание объекта:

prediction **=** loaded\_model**.**predict(x)

Для удобства вывода задаем список с названиями классов объектов:

classes**=**['самолет', 'автомобиль', 'птица', 'кот', 'олень', 'собака', 'лягушка', 'лошадь', 'корабль', 'грузовик']

Печатаем результат распознавания:

**print**(classes[np**.**argmax(prediction)])

[самолет]

Как видим, нейронная сеть справилась с задачей, несмотря на то, что обучение проводилось на изображениях размером 32х32. Наше изображение пришлось уменьшить до этого размера, но сеть все равно распознала на картинке самолет.

**Таким образом, мы выполнили следующие действия:**

# \*\*\*\*\* Загружаем сеть, обученную на наборе данных CIFAR-10, и компилируем модель:

import numpy

from keras.datasets import cifar10

from keras.models import Sequential

from keras.layers import Dense, Flatten, Activation

from keras.layers import Dropout

from keras.layers.convolutional import Conv2D, MaxPooling2D

from keras.utils import np\_utils

from keras.optimizers import SGD

from keras.models import model\_from\_json

json\_file = open("cifar10\_model.json", "r")

loaded\_model\_json = json\_file.read()

json\_file.close()

loaded\_model = model\_from\_json(loaded\_model\_json)

loaded\_model.load\_weights("cifar10\_model.h5")

loaded\_model.compile(loss='categorical\_crossentropy', optimizer='sgd', metrics=['accuracy'])

print ("Загрузили Model")

# \*\*\*\*\* Загружаем изображение в Keras:

import numpy as np

from keras.preprocessing import image

img\_path = 'plane.jpg'

img = image.load\_img(img\_path, target\_size=(32, 32))

# В отличие от рукописных цифр, в этот раз изображение цветное и его размер 32х32, в соответствии с форматом CIFAR-10. Преобразуем картинку в массив numpy:

x = image.img\_to\_array(img)

x /= 255

x = np.expand\_dims(x, axis=0)

# Запускаем распознавание объекта:

prediction = loaded\_model.predict(x)

print(prediction)

# Для удобства вывода задаем список с названиями классов объектов:

classes=['самолет', 'автомобиль', 'птица', 'кот', 'олень', 'собака', 'лягушка', 'лошадь', 'корабль', 'грузовик']

# Печатаем результат распознавания:

print(classes[np.argmax(prediction)])

Итоги

Мы научились применять нейронные сети для анализа любых изображений, а не только тех, которые входят в стандартные наборы данных для обучения. В Keras для этого есть класс imageиз модуля keras.preprocessing.

**Проведите эксперименты со своими изображениями.**