1. Пример распознавания цифр (сверточная сеть)

**import** numpy  
**from** keras.datasets **import** mnist  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense, Dropout, Flatten  
**from** keras.layers **import** Conv2D, MaxPooling2D  
**from** keras.utils **import** np\_utils  
  
*# Устанавливаем seed для повторяемости результатов*numpy.random.seed(42)  
  
*# Размер изображения*img\_rows, img\_cols = 28, 28  
  
*# Загружаем данные*(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()  
  
*# Преобразование размерности изображений*X\_train = X\_train.reshape(X\_train.shape[0], 1, img\_rows, img\_cols)  
X\_test = X\_test.reshape(X\_test.shape[0], 1, img\_rows, img\_cols)  
input\_shape = (1, img\_rows, img\_cols)  
  
*# Нормализация данных*X\_train = X\_train.astype(**'float32'**)  
X\_test = X\_test.astype(**'float32'**)  
X\_train /= 255  
X\_test /= 255  
  
*# Преобразуем метки в категории*Y\_train = np\_utils.to\_categorical(y\_train, 10)  
Y\_test = np\_utils.to\_categorical(y\_test, 10)  
  
*# Создаем последовательную модель*model = Sequential()  
  
model.add(Conv2D(16, kernel\_size=(3, 3),  
 activation=**'relu'**,  
 input\_shape=input\_shape))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(0.2))  
model.add(Conv2D(32, (3, 3), activation=**'relu'**))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(0.2))  
model.add(Flatten())  
model.add(Dense(500, activation=**'relu'**))  
model.add(Dropout(0.5))  
model.add(Dense(10, activation=**'softmax'**))  
  
*# Компилируем модель*model.compile(loss=**"categorical\_crossentropy"**, optimizer=**"adam"**, metrics=[**"accuracy"**])  
  
**print**(model.summary())  
  
*# Обучаем сеть*model.fit(X\_train, Y\_train, batch\_size=100, epochs=2, validation\_split=0.2, verbose=2)  
  
*# Оцениваем качество обучения сети на тестовых данных*scores = model.evaluate(X\_test, Y\_test, verbose=0)  
**print**(**"Точность работы на тестовых данных: %.2f%%"** % (scores[1]\*100))

*# Генерируем описание модели в формате json*model\_json = model.to\_json()  
*# Записываем модель в файл*json\_file = open(**"mnist\_model\_conv.json"**, **"w"**)  
json\_file.write(model\_json)  
json\_file.close()  
model.save\_weights(**"mnist\_model\_conv.h5"**)  
  
**print** (**"Сохранили Model"**)

# Загружаем данные об архитектуре сети из файла json

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

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

json\_file.close()

# Создаем модель на основе загруженных данных

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

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

# Компилируем модель

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

# Проверяем модель на тестовых данных

scores = loaded\_model.evaluate(X\_test, Y\_test, verbose=0)

print("Точность модели на тестовых данных: %.2f%%" % (scores[1]\*100))

import numpy as np

from keras.preprocessing import image

img\_path = ('7.png')

img = image.load\_img(img\_path, target\_size=(28, 28), grayscale=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. Пример распознавания объектов (сверточная сеть)

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

# Задаем seed для повторяемости результатов

numpy.random.seed(42)

# Загружаем данные

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

# Размер мини-выборки

batch\_size = 32

# Количество классов изображений

nb\_classes = 10

# Количество эпох для обучения

nb\_epoch = 1

# Размер изображений

img\_rows, img\_cols = 32, 32

# Количество каналов в изображении: RGB

img\_channels = 3

# Нормализуем данные

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train /= 255

X\_test /= 255

# Преобразуем метки в категории

Y\_train = np\_utils.to\_categorical(y\_train, nb\_classes)

Y\_test = np\_utils.to\_categorical(y\_test, nb\_classes)

# Создаем последовательную модель

model = Sequential()

# Первый сверточный слой

model.add(Conv2D(32, (3, 3), padding='same',

input\_shape=(3, 32, 32), activation='relu'))

# Второй сверточный слой

model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))

# Первый слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Слой регуляризации Dropout

model.add(Dropout(0.25))

# Третий сверточный слой

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

# Четвертый сверточный слой

model.add(Conv2D(64, (3, 3), activation='relu'))

# Второй слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Слой регуляризации Dropout

model.add(Dropout(0.25))

# Слой преобразования данных из 2D представления в плоское

model.add(Flatten())

# Полносвязный слой для классификации

model.add(Dense(512, activation='relu'))

# Слой регуляризации Dropout

model.add(Dropout(0.5))

# Выходной полносвязный слой

model.add(Dense(nb\_classes, activation='softmax'))

# Задаем параметры оптимизации

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical\_crossentropy',

optimizer=sgd,

metrics=['accuracy'])

# Обучаем модель

model.fit(X\_train, Y\_train,

batch\_size=batch\_size,

epochs=nb\_epoch,

validation\_split=0.1,

shuffle=True,

verbose=2)

# Оцениваем качество обучения модели на тестовых данных

scores = model.evaluate(X\_test, Y\_test, verbose=0)

print("Точность работы на тестовых данных: %.2f%%" % (scores[1]\*100))

# Генерируем описание модели в формате json

model\_json = model.to\_json()

# Записываем модель в файл

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

json\_file.write(model\_json)

json\_file.close()

model.save\_weights("cifar10\_model.h5")

print ("Сохранили Model")

# \*\*\*\*\* 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)])