Минобрнауки России

Федеральное государственное бюджетное образовательное

учреждение высшего образования

**НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ИМ. Р.Е. АЛЕКСЕЕВА**

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

|  |  |  |
| --- | --- | --- |
|  |  |  |

Курс “Аппаратное и программное обеспечение роботизированных систем”

**Отчет по лабораторной работе №3**

Выполнил:

Грачев Д.С.

Проверил:

Гай В.Е.

Нижний Новгород 2021

Задание:

Используя модуль Keras, написать алгоритм работы нейронной сети для распознавания изображений. Используемая модель нейронной сети: MobileNet. Набор данных: MNIST.

Ход работы:

Для начала мы подключаем все модели, которые будут необходимы для реализации нейронной сети:

from keras.models import Sequential, Model

from keras.layers import Lambda, Input

from keras.backend import tf as ktf

from keras.layers.core import Activation, Flatten, Dense, Dropout

from keras.layers.convolutional import Convolution2D, MaxPooling2D, ZeroPadding2D

from keras.optimizers import SGD, Adam

from tensorflow.python.keras.preprocessing.image import ImageDataGenerator

from keras.applications.mobilenet import MobileNet

from keras.datasets import mnist

from keras.utils import np\_utils

from keras.preprocessing.image import ImageDataGenerator

import cv2, numpy as np

import keras

Затем загружаем данные:

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

Дальше соответственно задаем размер изображения, размер подвыборки, количество изображений для обучения и для теста. Все эти значения нам понадобятся в дальнейшем:

img\_width, img\_height = 28, 28

batch\_size = 100

nb\_train\_samples = 60000

nb\_test\_samples = 10000

Приведем загруженные данные к виду, воспринимаемому нейронной сетью. Изменяем их таким образом, чтобы значения варьировались в пределах от 0.0 до 1.0, после чего утраиваем каждый черно-белый пиксель, чтобы сымитировать цветное изображение, так как MobileNet работает с цветными изображениями. Набор ответов также приводим к виду, воспринимаемому нейронной сетью (one hot encoding):

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

x\_train = x\_train / 255.0

x\_test = x\_test / 255.0

x\_train = np.stack((x\_train,)\*3, axis=-1)

x\_test = np.stack((x\_test,)\*3, axis=-1)

y\_train = np\_utils.to\_categorical(y\_train)

y\_test = np\_utils.to\_categorical(y\_test)

num\_classes = y\_test.shape[1]

Создаем экземпляр модели сети MobileNet с отсеченной квалификационной частью и определяем, какую часть сверточный сети обучать не надо, и выводим информацию о слоях:

mobile\_net = MobileNet(weights='imagenet', include\_top=False, input\_shape=(128, 128, 3))

mobile\_net.trainable = True

trainable = False

for layer in mobile\_net.layers:

  if layer.name == 'conv\_dw\_13':

    trainable = True

  layer.trainable = trainable

mobile\_net.summary()

Создаем дополнительный слой, преобразующий изображение к размеру 128х128, такой слой необходим по тому что сеть MobileNet не воспринимает изображения меньше чем 32х32:

inp = Input(shape=(None, None, 3))

out = Lambda(lambda image: ktf.image.resize(image, (128, 128)))(inp)

inputLayer = Model(inputs=inp, outputs=out, name="resizer")

inputLayer.summary()

Создаем составную сеть и выводим информацию о слоях:

# Создание модели составной сети

model = Sequential()

# Добавляем слой для преобразования размера изображения

model.add(inputLayer)

# Добавляем сверточные слои

model.add(mobile\_net)

# Преобразуем двумерный массив MobileNet в одномерный

model.add(Flatten())

# Полносвязный слой

model.add(Dense(256, activation='relu'))

# Слой регуляризации (для предотвращения переобучения)

model.add(Dropout(0.5))

# Кол-во классов

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

model.summary()

Компилируем модель:

epochs = 5

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

              optimizer=Adam(lr=1e-5),

              metrics=['accuracy'])

Обучаем составную сеть:

model.fit(x\_train, y\_train, validation\_data=(x\_test, y\_test), epochs = , batch\_size=batch\_size)

Обобщаем данные в процентах:

scores = model.evaluate(x\_test, y\_test, verbose=0)

print("Accuracy: %.2f%%" % (scores[1]\*100))