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**

Выполнили:

Дормидонтов М.А.

Проверил:

Гай В.Е.

Нижний Новгород 2021

**Задание:**

Используя модуль Keras, написать алгоритм работы нейронной сети для распознавания изображений.

**Ход работы:**

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

import keras

import tensorflow as tf

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

from keras.applications.xception import Xception, preprocess\_input

from keras.models import Sequential

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

Затем нам необходимо подключиться к гугл диску, чтобы занести туда файлы, на которых будет тренироваться наша нейронная сеть, а также тестовую выборку:

from google.colab import drive

drive.mount('/content/drive')

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

# Пути к наборам данных

train\_dir = '/content/drive/MyDrive/Keras/train'

test\_dir = '/content/drive/MyDrive/Keras/test'

# Размер изображений

img\_width, img\_height = 500, 500

# Размер мини-выборки

batch\_size = 60

# Кол-во изображений для обучения

nb\_train\_samples = 1034

# Кол-во изображений для теста

nb\_test\_samples = 133

При помощи ImageDataGenerator создаем генератор изображений:

# Создание генератора изображений

datagen = ImageDataGenerator(rescale=1. / 255)

train\_generator = datagen.flow\_from\_directory(

    train\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='categorical')

test\_generator = datagen.flow\_from\_directory(

    test\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='categorical')

Создаем экземпляр модели сети Xception и также говорим, что сверточную часть сети обучать не надо и выводим информацию о слоях:

# Создание экземпляра модели сети Xception

xception = Xception(

    input\_shape=(img\_width, img\_height, 3),

    include\_top=False,

    weights="imagenet"

)

xception.trainable = False

xception.summary()

Описание всех параметров:

* input\_shape : Необязательный кортеж формы
* include\_top : логическое значение, следует ли включать полностью подключенный уровень в верхнюю часть сети.
* weights : один из None(случайная инициализация), «imagenet» (предварительное обучение в ImageNet) или путь к загружаемому файлу весов.

Создаем составную сеть и выводим информацию о слоях:

# Создание модели составной сети (Составная сеть)

model = Sequential()

# Добавляем сверточные слои

model.add(xception)

# 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(3, activation='sigmoid'))

model.summary()

Компилируем модель:

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

              optimizer='SGD',

              metrics=['accuracy'])

Обучаем составную сеть:

model.fit\_generator(

    train\_generator,

    steps\_per\_epoch=nb\_train\_samples // batch\_size,

    epochs=9,

    validation\_data=test\_generator,

    validation\_steps=nb\_test\_samples // batch\_size)

Обобщаем данные в процентах:

