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:

Шатов Д.В.

Панкратьев И.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.mobilenet import MobileNet, preprocess\_input

from keras.models import Sequential

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

from google.colab import drive

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

drive.mount('/content/drive')

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

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

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

img\_width, img\_height = 500, 500

batch\_size = 79

nb\_train\_samples = 1034

nb\_test\_samples = 133

При помощи ImageDataGenerator создаем генератор 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 MobileNet и также говорим, что сверточную часть сети обучать не надо и выводим информацию о слоях:

Mobile\_Net = MobileNet(

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

    alpha=1.0,

    depth\_multiplier=1,

    dropout=0.001,

    include\_top=False,

    weights="imagenet",

    input\_tensor=None,

    pooling=None,

    classes=1000,

    classifier\_activation="softmax"

Mobile\_Net.trainable = False

Mobile\_Net.summary()

Описание всех параметров:

* input\_shape : Необязательный кортеж формы
* альфа : контролирует ширину сети.
* depth\_multiplier : множитель глубины для глубинной свертки.
* dropout : процент отсева.
* 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.
* input\_tensor : Необязательный тензор Keras (т.е. вывод layers.Input()) для использования в качестве ввода изображения для модели.
* pooling : Дополнительный режим объединения для извлечения функций, когда include\_top - False.
* classes : необязательное количество классов для классификации изображений, указывается только в том случае, если include\_top установлено значение True и weights не указан
* classifier\_activation : Функция активации для использования на «верхнем» слое.

Создаем составную сеть и выводим информацию о слоях:

# Создание модели составной сети

model = Sequential()

# 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(1, activation='sigmoid'))

model.summary()

Компилируем модель:

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

              optimizer='SGD',

              metrics=['accuracy'])

Обучаем составную сеть:

model.fit\_generator(

    train\_generator,

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

    epochs=5,

    validation\_data=test\_generator,

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

Обобщаем данные в процентах:

scores = model.evaluate\_generator(test\_generator, nb\_test\_samples // batch\_size)

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