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:
Гора К.А.
Проверил:
Гай В.Е.

Нижний Новгород 2021

Задание:
Используя модуль Keras, написать алгоритм работы нейронной сети для распознавания изображений. Используемая модель нейронной сети: **ResNet50**. Набор данных: Horse_or_humans.

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.resnet50  import ResNet50, preprocess_input
from keras.models import Sequential
from keras.layers.core import Flatten, Dense, Dropout
```

Затем загружаем данные, 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/datalab3/train'
test_dir = '/content/drive/MyDrive/datalab3/val'
# Размер изображений
img_width, img_height = 300, 300
# Размер мини-выборки
batch_size = 80
# Кол-во изображений для обучения
nb_train_samples = 1027
# Кол-во изображений для теста
nb_test_samples = 256
```

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='binary')

test_generator = datagen.flow_from_directory(
    test_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')
```

Создаем экземпляр модели сети ResNet50 и также говорим, что сверточную часть сети обучать не надо и выводим информацию о слоях:

```
# Создание экземпляра модели сети ResNet50
resnet = ResNet50(
    input_shape=(img_width, img_height, 3),
    include_top=False,
    weights="imagenet"
)

resnet.trainable = False

resnet.summary()
```

Описание всех параметров:

- input_shape: Кортеж формы
- include_top: логическое значение, следует ли включать полностью подключенный уровень в верхнюю часть сети.
- weights: «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(resnet)
# 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.build(input_shape=(None, img_width, img_height, 3))
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=7,
    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))
```

```
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:1948: UserWarning: `Model.evaluate_generator`
  warnings.warn('`Model.evaluate_generator` is deprecated and '
Точность работы на тестовых данных: 73.75%
```