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:

Йылдирим Мустафа

Проверил:

Гай В.Е.

Нижний Новгород 2021

**Задание:**

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

**Вариант:**

**Данные: Horse-or-human**

**Сеть: Xception**

**Листинг:**

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

# Подключение модулей

from keras.models import Sequential

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

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

from keras.optimizers import SGD

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

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

import cv2, numpy as np

Так как нужные нам данные находятся на сайте, мы их скачиваем и загружаем на Гугл диск

from google.colab import drive

drive.mount('/content/drive')

Дальше задаем размер изображения, размер мини-выборки, количество изображений для обучения и для теста

# Пути к наборам данных

train\_dir = '/content/drive/MyDrive/Keras/horse-or-human/Test'

test\_dir = '/content/drive/MyDrive/Keras/horse-or-human/Validation'

# Размер изображений

img\_width, img\_height = 300, 300

# Размер мини-выборки

batch\_size = 64

# Кол-во изображений для обучения

nb\_train\_samples = 1027

# Кол-во изображений для теста

nb\_test\_samples = 256

При помощи ImageDataGenerator создаем генератор изображений:

Указываем класс Binary, так как у нас только 2 вида картинок

# Создание генератора изображений

datagen = ImageDataGenerator(rescale=1. / 255)

# Генератор для обучения

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

    train\_dir,

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

    batch\_size=batch\_size,

    class\_mode='binary')

# Генератор для теста

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 Xception

# Создание экземпляра модели сети Xception

xception = Xception(

    weights='imagenet',

    include\_top=False,

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

# weights - веса предварительно обученной сети

# include\_top = false означает, что мы загружаем только сверточную часть сети, без квалификационной

# input\_shape - размер тензора

# Сверточную часть сети обучать не надо

xception.trainable = False

xception.summary()

Создаем составную сеть и выводим информацию о слоях:

# Создание модели составной сети

model = Sequential()

# Добавляем сверточные слои

model.add(xception)

# Преобразуем двумерный массив Xception 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:

# Компилируем составную сеть

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

              optimizer='SGD',

              metrics=['accuracy'])

Далее обучаем составную сеть:

Запускаем 7 эпох

# Обучение

model.fit\_generator(

    train\_generator,

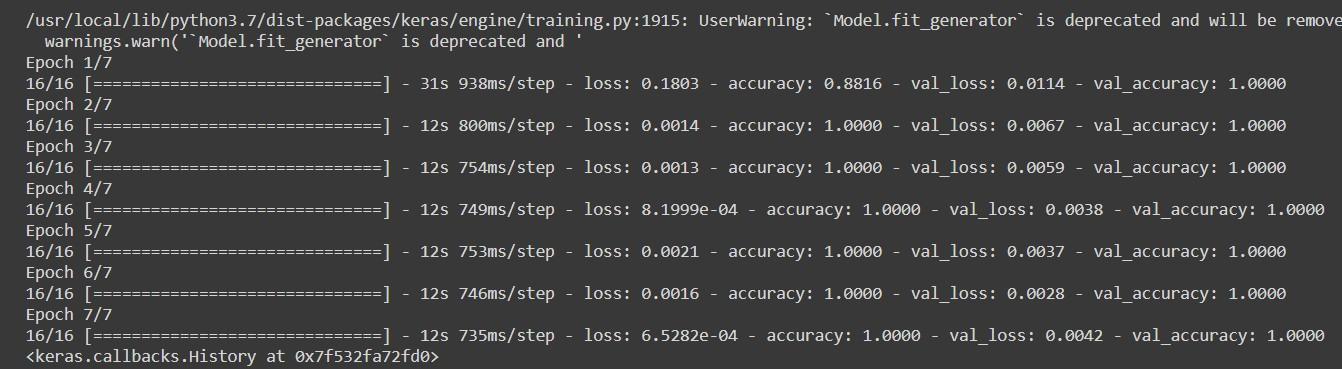
    steps\_per\_epoch=nb\_train\_samples // batch\_size,

    epochs=7,

    validation\_data=test\_generator,

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

Вот что получилось после обучения:



Результат в процентах

