|  |
| --- |
|  |
|  |
| |  | | --- | |  | | МИНОБРНАУКИ РОССИИ | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет****"**  **РТУ МИРЭА** | |  | |  | |
| Институт информационных технологий (ИТ) |
| Кафедра инструментального и прикладного программного обеспечения (ИиППО) |

|  |
| --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 4** |
| **по дисциплине** |
| **«Программное обеспечение интеллектуальных систем**»  *Нейронные сети* |

|  |  |
| --- | --- |
|  |  |
| Выполнил студент группы ИКБО-02-17 | *Крамаренко З.В* |
| Приняла | *Зорина Н.В.* |

|  |  |  |
| --- | --- | --- |
| Работа выполнена | «\_\_\_» \_\_\_\_\_\_\_\_2020 |  |
|  |  |  |
| «Зачтено» | «\_\_\_» \_\_\_\_\_\_\_\_2020 |  |

**Цель:** ознакомиться с работой нейронных сетей на примере создания приложений со сторонними библиотеками Python.

**Задание:**

Реализуем свёрточную нейронную сеть при помощи библиотеки Keras, которая является надстройкой над TensorFlow. СНС будет классифицировать 2 типа изображений: кот и собака.

**Ход работы**

Покажем, как реализовать модель свёрточной нейронной сети с использованием фреймворка для глубокого обучения Keras. Данный фреймворк является надстройкой над TensorFlow. Будем решать задачу классификации на датасет cats\_vs\_dogs от Microsoft. Все данные (цветные изображения) подразделяются на 2 класса: кошки и собаки.

Импортируем необходимые библиотеки:

- numpy – библиотека для работы c многомерными массивами и матрицами,

- scikit-learn – библиотека алгоритмов для решения задач классического машинного обучения,

- tensorflow – библиотека для решения задач построения и тренировки нейронной сети,

- keras – библиотека для быстрой реализации нейронных сетей, является надстройкой над TensorFlow.

import numpy as np

import tensorflow as tf

import tensorflow\_datasets as tfds

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout

import matplotlib.pyplot as plt

from google.colab import files

Далее загрузим наш Microsoft dataset (цветные изображения кошек и собак) с 25 тыс. картинок. И разобьём данные на выборки. Мною было принято решение все картинки отдать на обучение НС (чтобы повысить точность модели), а тестовый набор данных будет собран самостоятельно

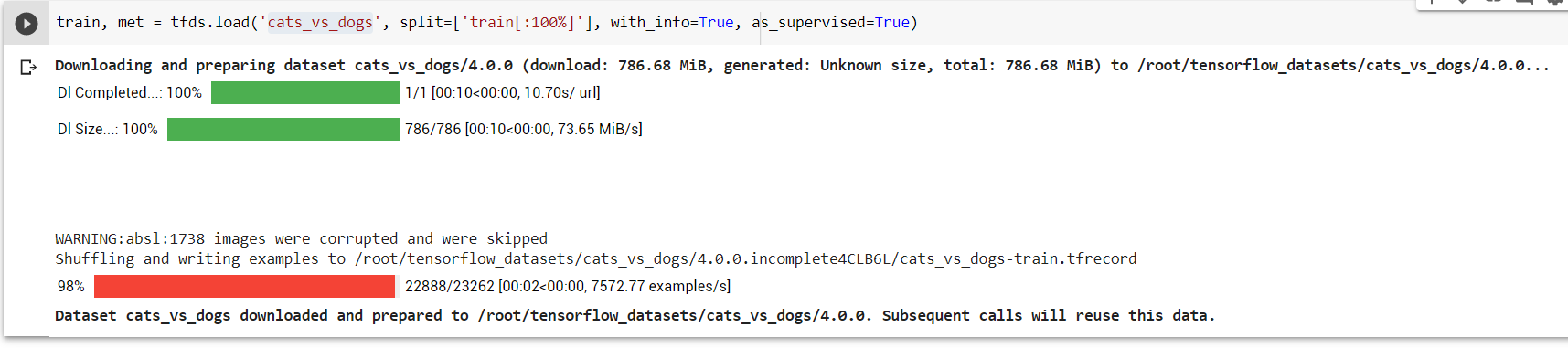


Рисунок 1 – Загрузка dataset

Как можно заметить, то Microsoft решил вставить в dataset 1738 поврежденных картинок.

Если вывести само изображение и подписи обучающей выборки, то можно заметить, что 0 – коты, а 1 – собаки. Поэтому напишем функцию, которая будет преобразовывать наши картинки в более понятный для сети вид. Как нам известно, что НС работает с числами. Поэтому нам необходимо преобразовать наши картинки в числовой вектор, а точнее матрицу.

Значения пикселей в картинках храниться в диапазоне от 0 до 255, а НС гораздо проще распознавать данные, которые находятся в промежутке от 0 до 1. Поэтому поделим пиксели на 255 и сменим разрешение на 224 на 224.

SIZE = 224

def resize\_image(img, label):

  img = tf.cast(img, tf.float32)

  img = tf.image.resize(img, (SIZE, SIZE))

  img = img / 255.0

  return img, label

Далее применим нашу функцию resize\_image на данные для обучения. И разобьем выборку на части, чтобы поместилась в память видеокарты GC.

train\_resized = train[0].map(resize\_image)

train\_batches = train\_resized.shuffle(1000).batch(16)

Далее напишем нашу НС. У нас будет свёрточная нейронная сеть. Для этого возьмем обученные слои из библиотеки Keras, т.к принцип работы у СНС одинаков для всех картинок. СНС будет сканировать нашу картинку по частям и пытаться распознать там определенные фичи (уши, хвост и т.д). 1 слой собирает линии, 2 слой собирает из этих линий углы, 3 слой – более сложные изгибы и т.д, пока не дойдем до нужной нам фичи.

base\_layers = tf.keras.applications.MobileNetV2(input\_shape=(SIZE, SIZE, 3), include\_top=False)

base\_layers.trainable = False

После соберём нашу модель из начальных keras слоев и финальных, которые и будет отвечать за вывод. Сначала делаем базовые слои, потом глобальный пуллинг, Dropout(0.2), чтобы избежать заучивания (переобучение). В качестве оптимизатора будем использовать не обычный стохастический градиентный спуск, а его модификацию ADAM – Adaptive Moment Estimation.

model = tf.keras.Sequential([

                             base\_layers,

                             GlobalAveragePooling2D(),

                             Dropout(0.2),

                             Dense(1)

])

model.compile(optimizer='adam', loss=tf.keras.losses.BinaryCrossentropy(from\_logits=True), metrics=['accuracy'])

Далее обучаем нашу модель в 2 эпохи, чтобы повысить точность.

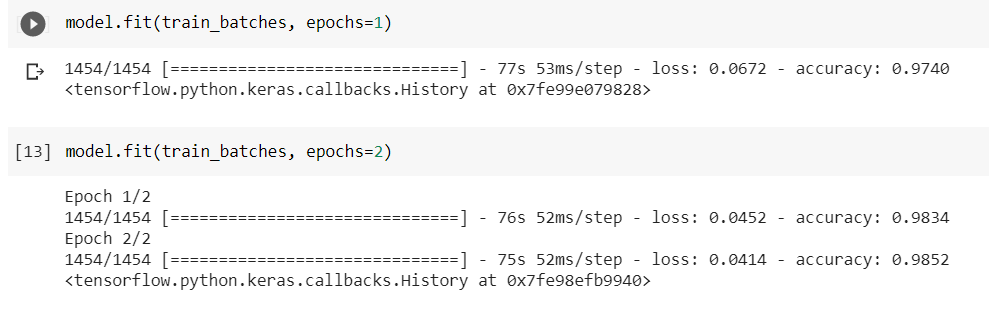


Рисунок 2 – Обучение НС

Как можно заметить после 2 эпох обучения точность модели составляет 98,52%. Это очень хороший результат.

Далее мы скачали 6 картинок в формате jpg с кошками и собаками. Загрузим их для дальнейшего тестирования.

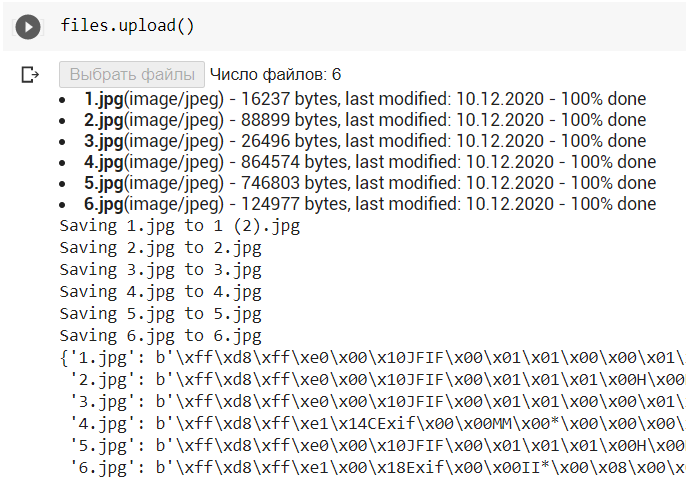


Рисунок 3 – Загрузка изображений для тестирования

Далее напишем функцию, которая будет последовательно брать картинки из нашего массива, применять к ним ранее написанную функцию resize\_image и выдавать результат. Как мы помним, наши картинки имели подписи 0 – коты, а 1 – собаки. Ровно посередине это 0.5, значит, чем больше мы идем в минус (в сторону 0 и далее), тем больше это кот, чем больше в плюс – собака.

Запустим наш код и проверим результаты на нашем массиве картинок.

for i in range(6):

  img = load\_img(f'{i+1}.jpg')

  img\_array = img\_to\_array(img)

  img\_resized, \_ = resize\_image(img\_array, \_)

  img\_expended = np.expand\_dims(img\_resized, axis=0)

  prediction = model.predict(img\_expended)[0][0]

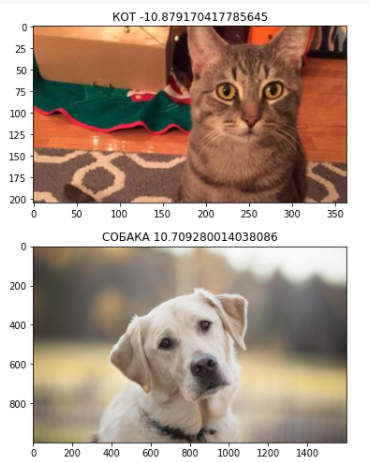
  pred\_label = 'КОТ' if prediction < 0.5 else 'СОБАКА'

  plt.figure()

  plt.imshow(img)

  plt.title(f'{pred\_label} {prediction}')

***Результаты работы программы***





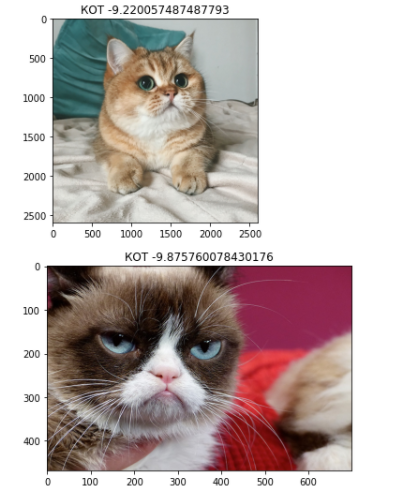


Рисунок 4 – Результаты работы НС на тестовой выборке

Ссылка на Google Colab (можно запустить код через браузер): <https://colab.research.google.com/drive/1uw3hu63wpzctiWVyDf68fYdnX8F2YgaR?usp=sharing>

Ссылка на GitHube: <https://github.com/ZlataKr/POIS/blob/master/Lab4_ipynb_.ipynb>

**Вывод**

В ходе выполнения лабораторной работы мы ознакомились с работой нейронных сетей на примере создания приложений со сторонними библиотеками Python. Реализовали свёрточную нейронную сеть при помощи библиотеки Keras, которая является надстройкой над TensorFlow. СНС классифицирует 2 типа изображений: кот и собака. Точность модели составляет 98.52%.

**Список литературы**

1. Себастьян Рашка. Python и машинное обучение / - М.: Изд-во Вильямс, 2019

2. Петер Флах. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / - М.: Изд-во ДМК Пресс, 2019

3. Keras [Электронный ресурс] - Режим доступа: <https://pypi.org/project/Keras/>

4. LinuxBlog.РФ, Сверточная нейронная сеть на Python и Keras [Электронный ресурс] - Режим доступа: <https://xn--90aeniddllys.xn--p1ai/svertochnaya-nejronnaya-set-na-python-i-keres/>

5. Материалы лекций