1

# Библиотеки

# Поддержка функций работы с массивами, высокоуровневых математических функций, предназначенных для работы с многомерными массивами

import numpy as np

# Построение и тренировка нейронной сети

import tensorflow as tf

# Датасеты

import tensorflow\_datasets as tfds

# Методы для работы с изображениями - подгрузка и преобразование в массивы данных

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

# Создание классов - помощь в создании нейронной сети

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

# Визуализация данных в двухмерной графике

import matplotlib.pyplot as plt

# Загрузка изображений через редактор google colab

from google.colab import files

2

# Подключение датасетов от Microsoft - содержит набор фотографий около 25000 с изображением кошек и собак -

# это позволит натренировать нейросеть и создать весы, по которым будет пониматься, какое животное изображено на рисунке

# Работаем с tensorflow\_datasets и функцией load указываем, что данные будут подгружены с дополнительной информацией

train, \_ = tfds.load('cats\_vs\_dogs', split=['train[:100%]'], with\_info=True, as\_supervised=True)

3

# Уменьшает все фотографии до размера 224 на 224 пикселя для уменьшения времени ожидания тренировки (лучше давать примерно одинаковые данные)

SIZE = (224, 224)

def resize\_image(img, label):

img = tf.cast(img, tf.float32)

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

img /= 255.0

return img, label

4

# Каждое изображение датасета обрезаем до необходимого размера, полученное изображение перемешиваем и преобразовываем их пачками по 16 единиц

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

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

5

# Создаем модель нейронной сети

# Создаем основной слой на основе класса MobileNetV2

base\_layers = tf.keras.applications.MobileNetV2(input\_shape=(SIZE[0], SIZE[1], 3), include\_top=False)

6

# Реализовываем объект на основе класса Sequential - обучение и выводы для модели

model = tf.keras.Sequential ([

base\_layers, # Основной слой

GlobalAveragePooling2D(), # Глобальная средняя операция объединения для пространственных данных

Dropout(0.2), # На сколько изменяются весы каждый раз

Dense(1) # Плотность связный слой с нейронной сетью (первый слой)

])

# В конце создания используется метод compile, где указывается формат optimizer и метод BinaryCrossentropy, который вычисляет потерю между метками и предсказаниями

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

7

# Тренировка модели - для этого необходимо использовать функцию fit, куда передается набор фото, взятых из датасета и позже обрезанных, а также

# указывается количество тренировок, так как у нас 25000 картинок, то хватит одного обучения

model.fit(train\_batches, epochs=1)

8

# Загрузка фото в google colab

files.upload()

9

# Создание списка из всех загруженных фото

images = ['1.jpg', '2.jpg', '3.jpg', '4.jpg', '5.jpg', '6.jpg']

# Цикл, где будем подгружать каждое фото, преобразовывать его в массив, уменьшать фото через метод resize\_image,

# далее давать возможность модели решить, что находиться на фото в зависимости от вычисленного prediction

# После отображаем фото вместе с надписью над ними

for i in images:

img = load\_img(i)

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)

plt.figure()

plt.imshow(img)

label = 'Собака' if prediction > 0 else 'Кошка'

plt.title('{}'.format(label))