Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01– «Информатика и вычислительная техника»

**проект**

**по дисциплине**

**«Искусственный интеллект»**

**на тему**

**Разработка программы распознавания животного по изображению**

Выполнили студент гр. КС-19-1б

Ладыгин Владимир Андреевич

Иванов Илья Борисович

Минин Валентин Валерьевич

Еремеева Полина Алексеевна

Проверил:

стар. преп. каф. ИТАС

Яруллин Денис Владимирович

(оценка) (подпись)

(дата)

Пермь 2022

Содержание

[Введение 3](#_Toc103188419)

[Реализация программы 4](#_Toc103188420)

[Пример работы программы 7](#_Toc103188421)

[ПРИЛОЖЕНИЕ А – Код программы 8](#_Toc103188422)

Введение

Нейронные сети — одно из направлений в разработке систем искусственного интеллекта. Идея заключается в том, чтобы максимально близко смоделировать работу человеческой нервной системы — а именно, её способности к обучению и исправлению ошибок. В этом состоит главная особенность любой нейронной сети — она способна самостоятельно обучаться и действовать на основании предыдущего опыта, с каждым разом делая всё меньше ошибок.

Распознавание изображений — классический пример использования нейронных сетей. В нашем случае проект будет по изображению распознавать какое на картинке животное: кошка или собака. Примерную работу ИИ можно увидеть на рисунке 1.

|  |
| --- |
|  |
| Рисунок 1 – Работа распознавания животного |

Наш искусственный интеллект не будет распознавать какие-либо предметы такие как: машины, люди, другие животные и так далее. Все дело в том, что мы будем использовать для тренировки ИИ датасеты от компании Microsoft, где расположены более 25000 различных фотографий кошек и собак, что даст нам возможность правильно настроить весы для распознавания того, что находится на изображении.

Реализация программы

**Необходимые библиотеки:**

1. NumPy

NumPy – библиотека с открытым исходным кодом для языка программирования Python. Возможности:

1. поддержка многомерных массивов (включая матрицы);
2. поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами.
3. TensorFlow

TensorFlow – открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия. Данная библиотека помогает в построении и тренировки нейронной сети.

1. Matplotlib

Matplotlib – библиотека на языке программирования Python для визуализации данных двумерной (2D) графикой, на ее основе можно построить различные графики, изображения, диаграммы и другие визуальные данные, которые человеком воспринимаются гораздо проще нежели текст.

**Среда разработки**

Основой для среды разработки был Google Colab – это бесплатный облачный сервис на основе Jupyter Notebook. Google Colab предоставляет всё необходимое для машинного обучения, анализа данных и обучения в целом прямо в браузере, даёт бесплатный доступ к невероятно быстрым GPU и TPU, программа бесплатная и позволять выполнять код блоками.

**Алгоритм проекта**

Подключение необходимых библиотек NumPy, TensorFlow, Matplotlib, датасетов от tensorflow\_datasets, методы для работы с изображениями и создания классов.

Далее подключение датасетов от Microsoft, которое 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.

Следующим шагом уменьшаем все фотографии до размера 224 на 224 пикселя для уменьшения времени ожидания тренировки, так как лучше давать примерно одинаковые данные.

После каждое изображение датасета обрезаем до необходимого размера, а полученное изображение перемешиваем и преобразовываем их пачками по 16 единиц.

Теперь создаем модель нейронной сети и основной слой на основе класса MobileNetV2.

Далее реализовываем объект на основе класса Sequential - обучение и выводы для модели, параметры для модели будут передаваться: основной слой, глобальная средняя операция объединения для пространственных данных, на сколько изменяются весы каждый раз, 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.

Затем последует 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.

После необходимо загрузить фото в google colab.

Последним шагом будет создание списка из всех загруженных фото и цикла, где будем подгружать каждое фото, преобразовывать его в массив, уменьшать фото через метод resize\_image, позже давать возможность модели решить, что находиться на фото в зависимости от вычисленного prediction. Наконец отображаем фото вместе с надписью над ними, чтоб увидеть работу программы.

Пример работы программы

|  |
| --- |
|  |
| Рисунок 2 – Набор данных |

ПРИЛОЖЕНИЕ А – Код программы

Отдельные компоненты кода в Google Colab:

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))