

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Автоматизированные системы обработки информации и управления | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Искусственный интеллект | | |
| Курс | IV | | | Группа | 413 |

Отчёт по контрольной работе № 1

**Применение технологий искусственных нейронных сетей в задаче распознавания видов цветов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 413 |  |  |  | Беналлал А-М.Н  Мезенцова А.О |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**ЦЕЛИ И ЗАДАЧИ**

Целью данной работы является создание системы распознавания видов цветов с использованием ИНС.

**Установка и настройка**

Для работы с TensorFlow требуется предварительная установка Python и подходящего менеджера пакетов (например, pip). Основные шаги установки:

1. Устанавливаем Python 3.10.
2. Переходим на официальный сайт TensorFlow и следуем инструкцию установки библиотеки (рисунок 1).
3. Используя терминал копируем команду из официального сайта и вставляем в терминал и ждем установку библиотеки (рисунок 1).

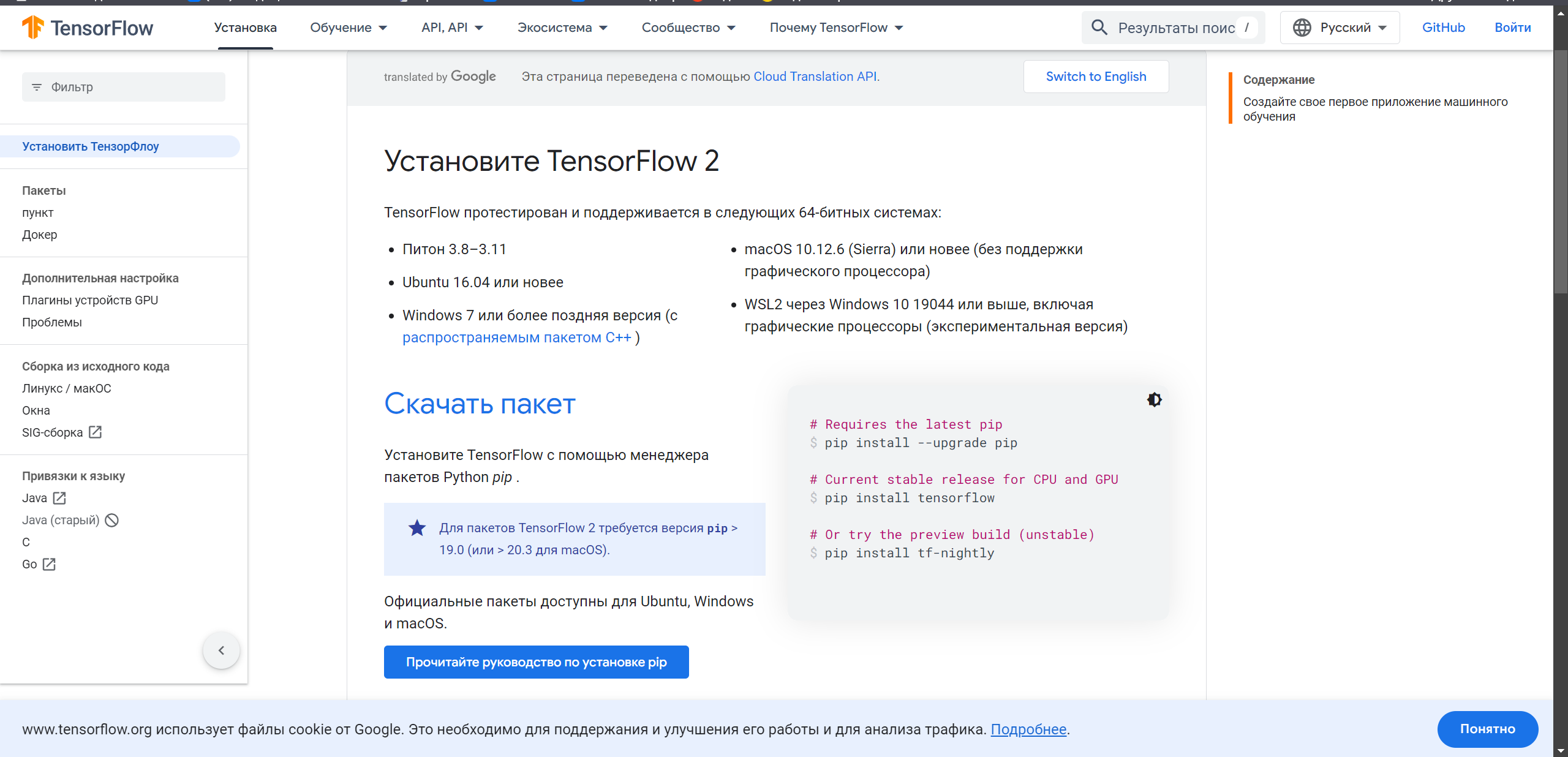


Рисунок 1 – официальный сайт TensorFlow

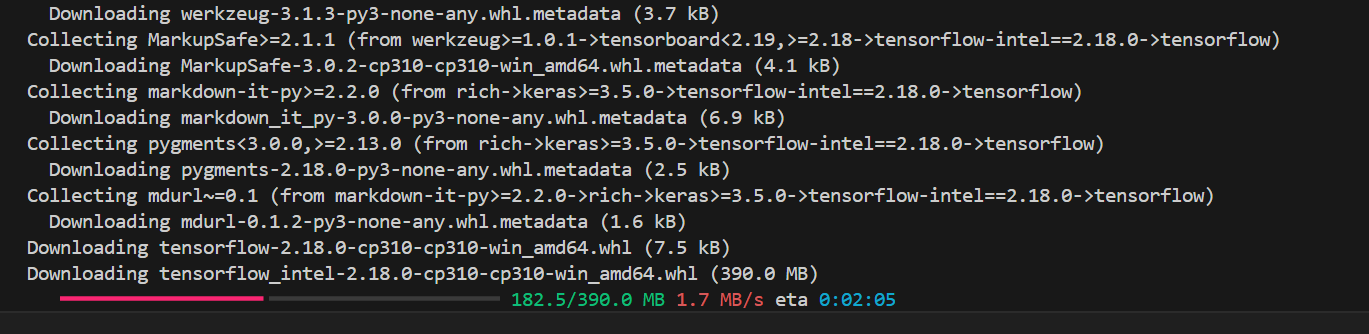


Рисунок 2 – установка библиотеки TensorFlow

# **Структура данных и категории цветов в датасете**

Для создания программу для распознавания видов цветов было использовано набор данных, загруженный из интернета, который содержит изображения пяти различных видов растений: подсолнух, ромашка, тюльпан, одуванчик и роза. датасет включает в себя изображения растений в различных ракурсах и яркости, метки (категории), которые указывают на вид растения. Данные разделяются на обучающую, валидационную и тестовую выборки.

Подготовка данных включает их масштабирование, нормализацию и преобразование в формат, совместимый с TensorFlow. Каждая категория данных содержит около шестисот изображений, что позволяет эффективно обучить модель нейронной сети (рисунок 3-7).

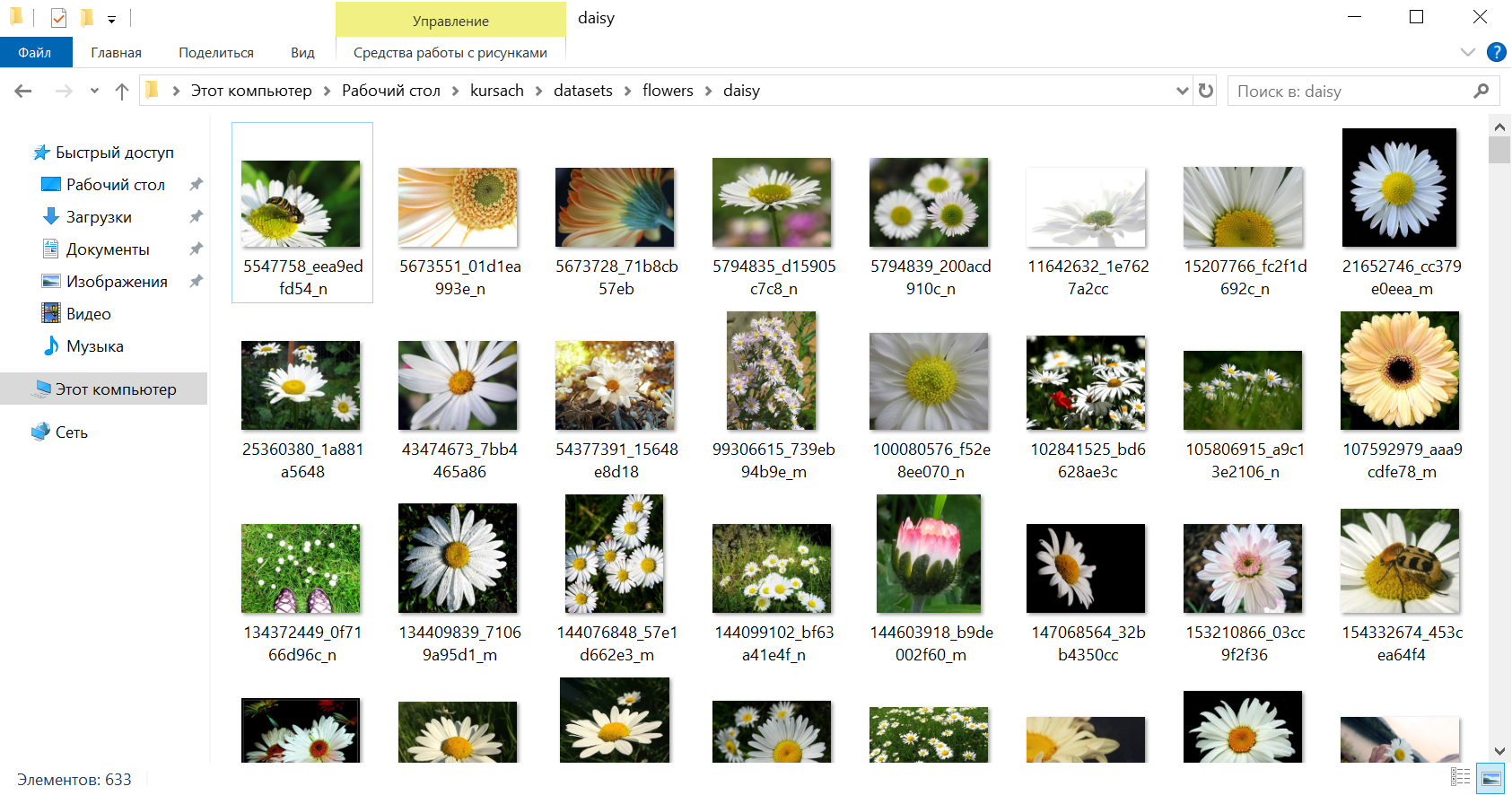


Рисунок 3 – изображений ромашек из набора данных для классификации видов цветов

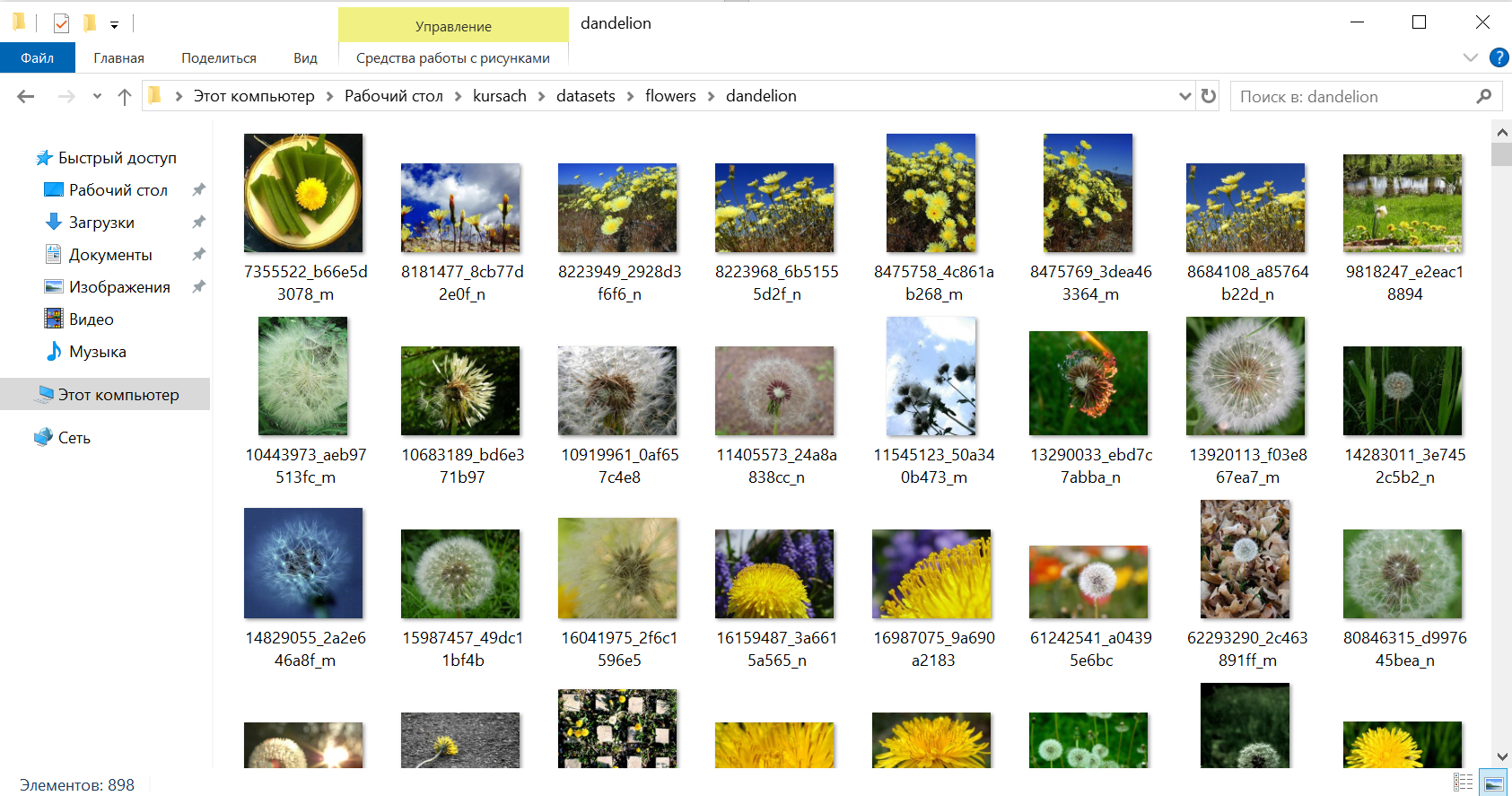


Рисунок 4 – изображений одуванчиков из набора данных для классификации видов цветов



Рисунок 5 – изображений роз из набора данных для классификации видов цветов

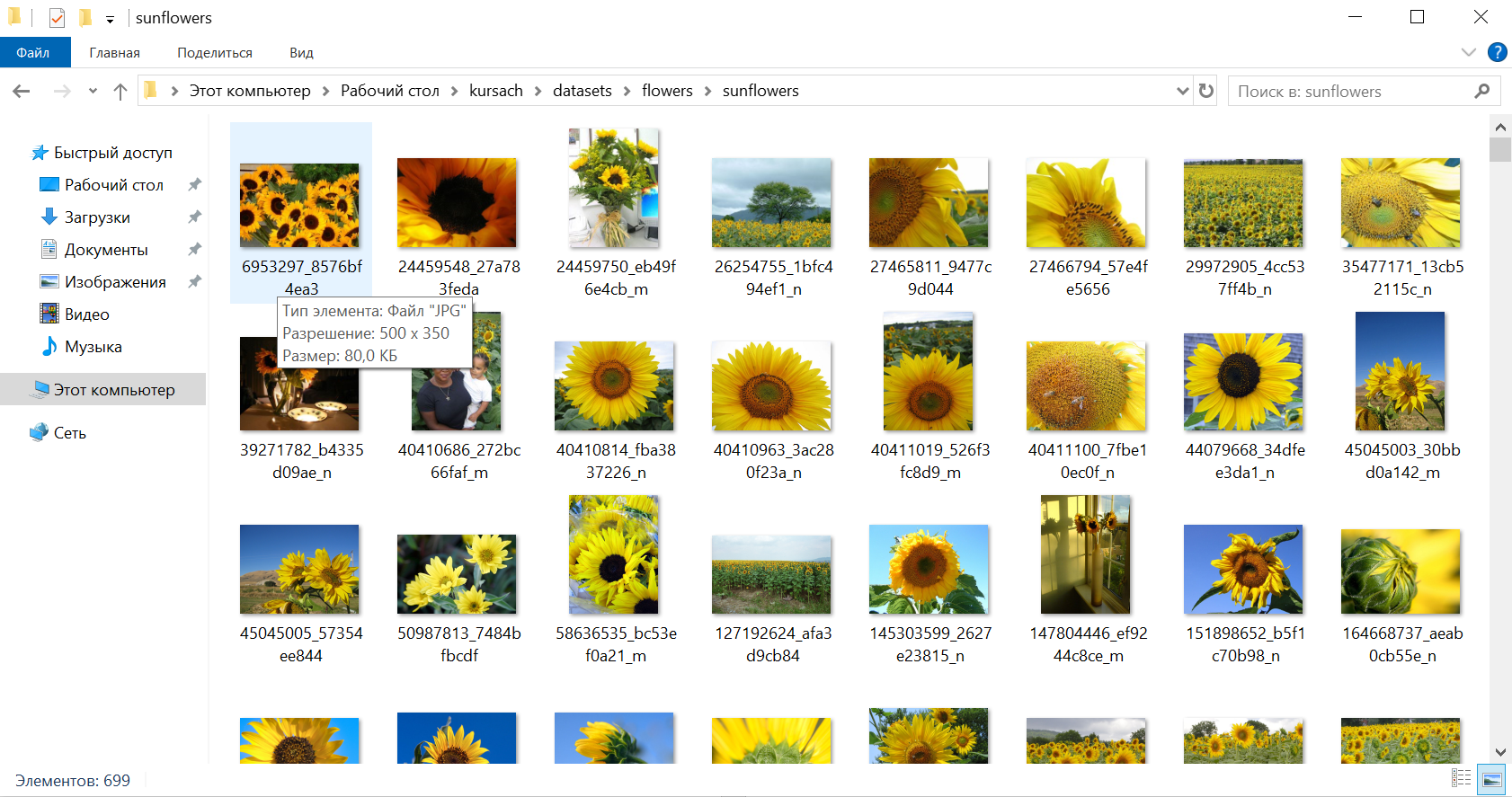


Рисунок 6 – изображений подсолнуха из набора данных для классификации видов цветов

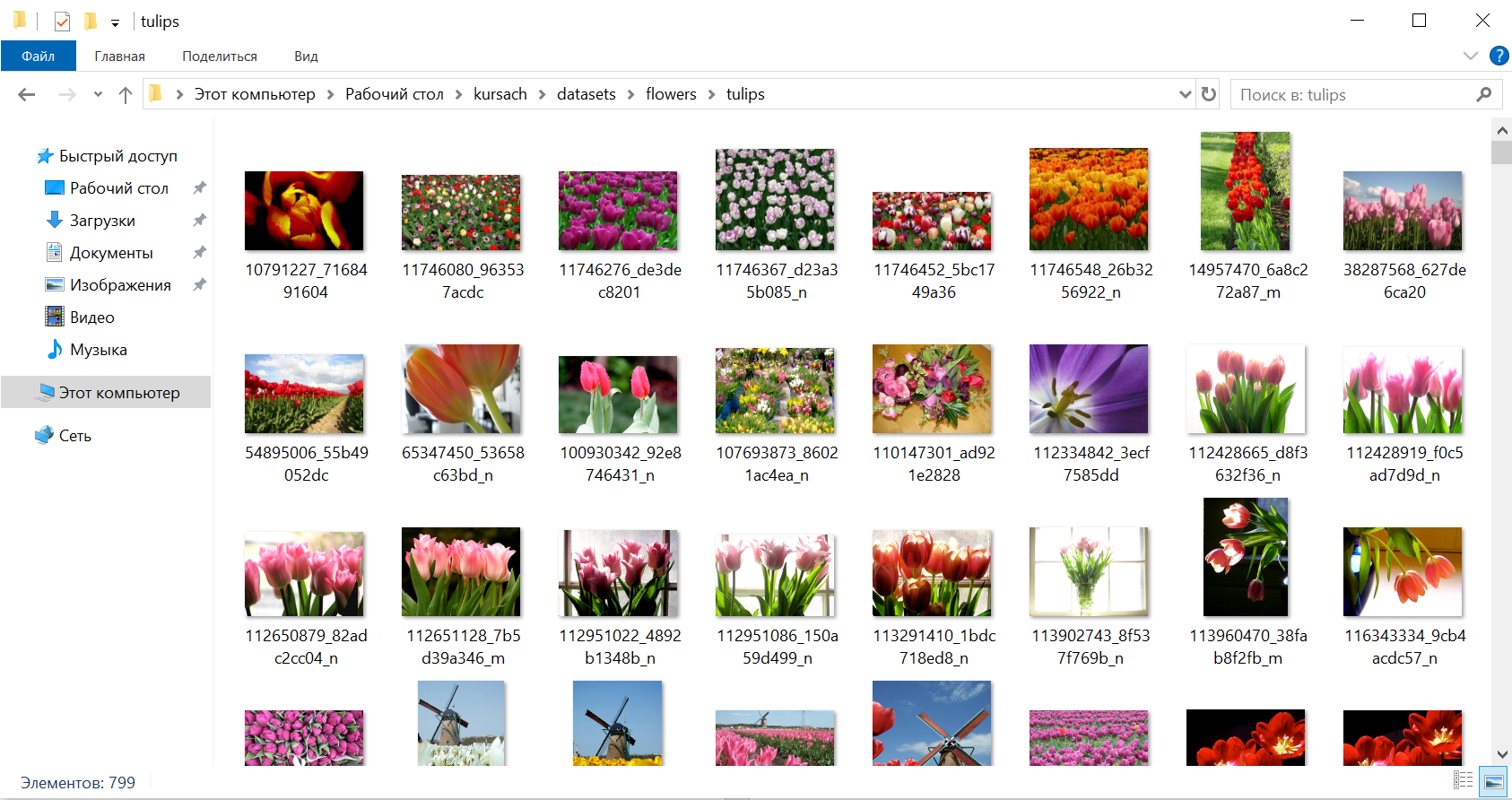


Рисунок 7 – изображений тюльпанов из набора данных для классификации видов цветов

# **АЛГОРИТМ РАБОТЫ модулей из БИБЛИОТЕКИ tensorflow**

Для создания программы были использованы модули layers и Sequential, которые реализуют обработку данных в нейронной сети через последовательное применение математических операций на каждом этапе.

Модуль layers предоставляет собой набор строительных блоков для создания нейронных сетей. Каждый из слоев выполняет определённые математические операции над входными данными. Например, слой Rescaling нормализует значения пикселей изображений, преобразуя их из диапазона [0, 255] в [0, 1] по формуле:

где I(x,y) исходное значение пикселя, а I′(x,y) нормализованное значение. Слой Conv2D выполняет свёртку изображения с фильтром (ядром) для выделения признаков, таких как текстуры и формы. Выходное значение для пикселя рассчитывается по формуле:

где I входное изображение, K ядро (фильтр), а kh,kw ​ размеры ядра. Слой MaxPooling2D уменьшает размер данных, сохраняя наиболее значимые признаки, вычисляя максимум в заданных областях:

где k размер окна выборки. Для предотвращения переобучения слой Dropout случайным образом отключает нейроны, используя вероятностное распределение:

где — входное значение нейрона. Слой Flatten преобразует многомерные данные, например , в одномерный массив длины . На завершающем этапе используются полносвязные слои (Dense). Каждый слой вычисляет линейную комбинацию входных данных:

где W — матрица весов, b — вектор смещений, а f — функция активации. Для скрытого слоя используется функция активации ReLU:

а для последнего слоя классификации — функция Softmax, которая преобразует выходы в вероятности:

Модуль Sequential объединяет все эти слои в единую цепочку, упрощая построение модели. Его работа математически описывается как последовательная композиция функций. Если сеть состоит из n слоёв, то для входных данных выход всей сети определяется как:

Каждый слой выполняет своё преобразование данных, передавая результат следующему слою.

**Метрики Presision recall**

Для оценки метрик **precision**, **recall**, и **F1-score** используются следующие формулы:

***1. Confusion Matrix (Матрица ошибок)***

Для каждого класса рассчитывается матрица ошибок:

|  |  |  |
| --- | --- | --- |
|  | **Предсказано Положительное** | **Предсказано Отрицательное** |
| **Истинное Положительное (TP)** | TP | FN |
| **Истинное Отрицательное (TN)** | FP | TN |

**1) TP** (True Positive): Количество правильно классифицированных объектов данного класса.

**2) FP** (False Positive): Количество объектов других классов, ошибочно отнесённых к данному классу.

**3) FN** (False Negative): Количество объектов данного класса, ошибочно отнесённых к другим классам.

**4) TN** (True Negative): Количество объектов других классов, которые правильно классифицированы как не принадлежащие данному классу.

***2****.* ***Precision (Точность)***

* **Precision** показывает, какая доля предсказаний "положительного" класса оказалась правильной.
* Значение лежит в интервале [0,1] [0, 1], где 1 означает идеальную точность.

***3****.* ***Recall (Полнота)***

* **Recall** показывает, какая доля объектов данного класса была правильно классифицирована.
* Значение лежит в интервале [0,1] [0, 1], где 1 означает, что модель нашла все объекты данного класса.

1. ***F1-Score (Гармоническое среднее)***

* **F1-Score** — это гармоническое среднее между Precision и Recall.
* Значение лежит в интервале [0,1][0, 1], где 1 указывает на идеальный баланс между Precision и Recall.

***5****.* ***Macro, Micro, and Weighted Averages***

Если вы хотите усреднить метрики по всем классам, можно использовать следующие подходы:

***Macro Average****:*

Простое усреднение по всем классам:

***Micro Average****:*

Общая оценка по всем объектам, без выделения отдельных классов:

***Weighted Average****:*

Взвешенное среднее, учитывающее количество объектов каждого класса:

где

— количество объектов в классе .

*Пример использования:*

Если для класса "Цветы" мы имеем:

* **TP = 50**
* **FP = 10**
* **FN = 5**

Тогда:

1. **Precision**:
2. **Recall**:
3. **F1-Score**:

# **Предобработка изображения**

Предобработка изображений является важным этапом подготовки данных для классификации видов цветов. Она улучшает качество данных, стандартизирует их и помогает повысить точность модели. Из основных предобработки можно учесть:

Разделение данных на выборки: Датасет делится на тренировочную и валидационную выборки. Тренировочная выборка используется для обучения модели, а валидационная для проверки её точности на новых данных.

-Приведение изображений к единому размеру: Все изображения стандартизируются до фиксированного размера, чтобы модель могла обрабатывать данные в едином формате, в нашем случае 180\*180.

-Масштабирование данных: Значения пикселей нормализуются в диапазоне от 0 до 1 что улучшает стабильность модели и ускоряет её обучение.

-Аугментация данных: чтобы увеличить объём данных и повысить устойчивость модели к изменениям в исходных изображениях нужно применить аугментацию. Она включает вращение изображений, изменение масштаба, зеркальное отражение и изменение контраста.

# **Реализация алгоритма**

Для реализации алгоритма классификации изображений цветов была разработана программа на операционной системе Windows 10 x64. В качестве языка программирования было использовано Python 3.10, а для написания и редактирования кода применяем редактор Visual Studio Code (рисунок 8).

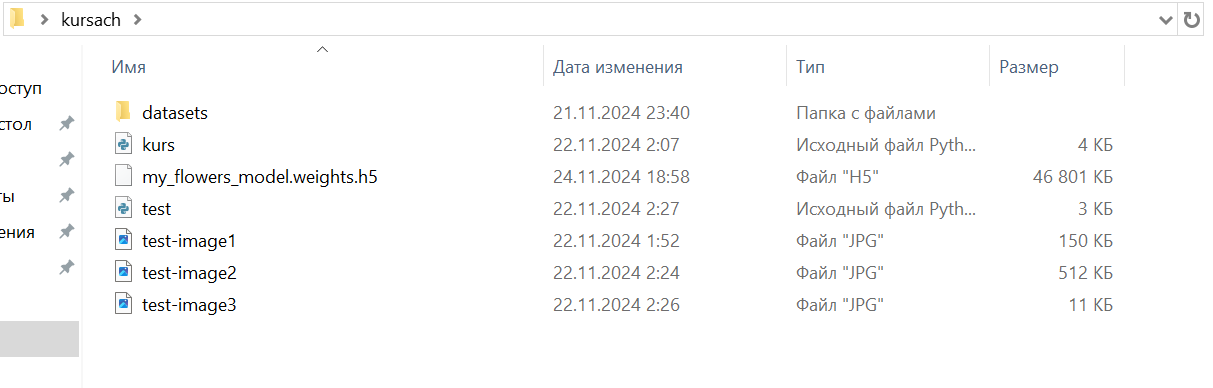


Рисунок 8 – папка программы для классификации изображении цветов

В программе было импортировано несколько библиотек (рисунок 9):

-Pathlib: для работы с путями к файлам директориям.

-NumPy: для работы с массивами и математическими вычислениями.

-Matlotlib: для построения графиков.

-Tensorflow: для построения и обучения моделей машинного обучения.

-Tensorflow.Keras: для создания, настройки и обучения нейронной сети.

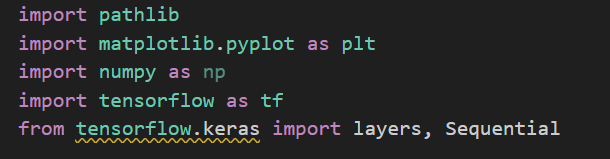


Рисунок 9 – импортирование библиотек

указываем путь к папки, где находится датасет с картинками цветов, после этого надо определить количество картинок, которые будут обрабатываться за 1 шаг обучения, далее фиксируем размер 180\*180 пикселей чтобы соответствовать совместимости с моделью. делим набор данных на 2 части, 80 процентов берет обучающая модель, а остальные (20 процентов) берет проверочная, это делается для воспроизводимости результатов (рисунок 10-11).

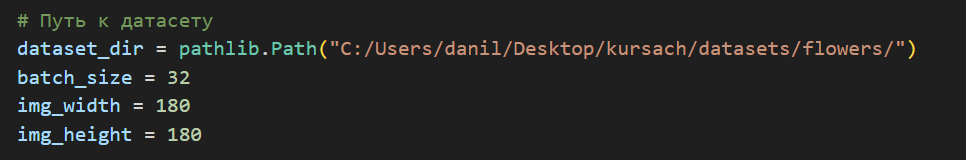


Рисунок 10 – путь к датасету и базовые настройки для обработки изображений.

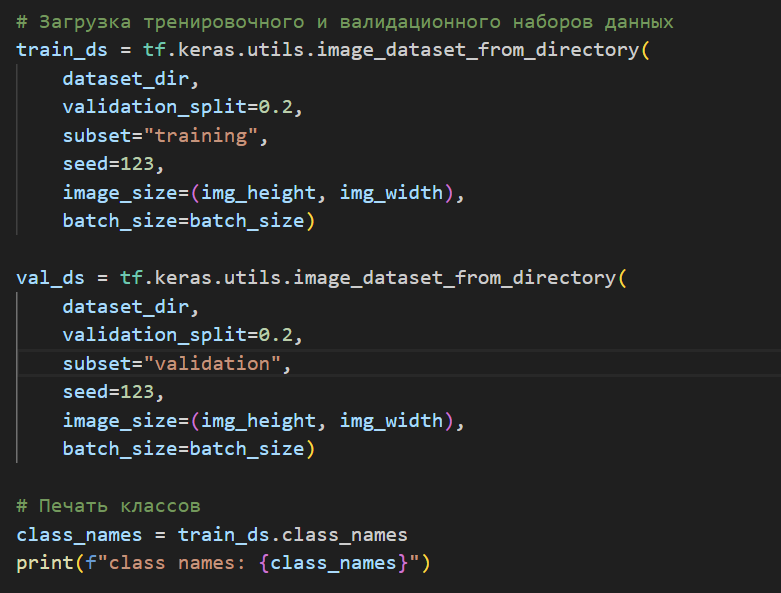


Рисунок 11 – Загрузка и разделение датасета на тренировочную и валидационную выборки

затем информация кэшируется и загружается одновременно, это делается для того, чтобы дать возможность стимулировать тренировку и минимизировать замедления. далее используем несколько методов аугментации такие как зеркальное отражения, вращения и повышения или уменьшения контрастности, это нам дает разнообразие и устойчивость модели(рисунок 12).

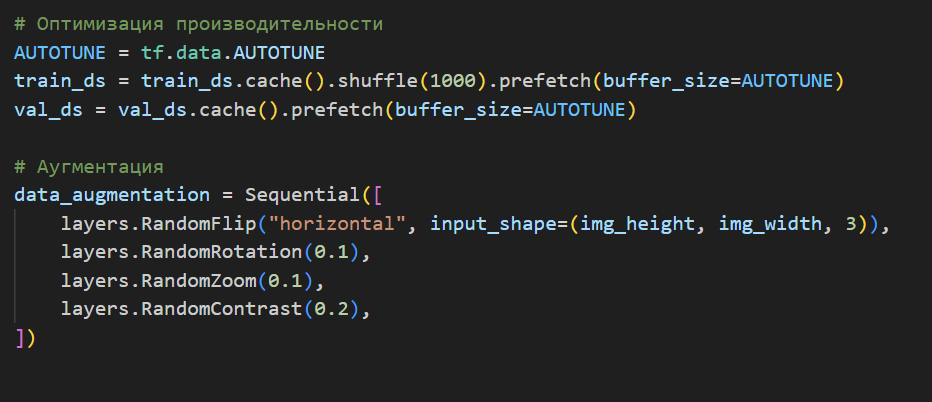


Рисунок 12 – Оптимизация производительности и аугментация данных для повышения устойчивости модели

вслед за этим создаем архитектуру с Sequential, который включает в себя несколько сверточных блока, один слой свертки, второй это слой подвыборки, они используются чтобы извлекать основные характеристики к примеру края, формы и текстуры. с каждым уровнем начинаем увеличивать количество фильтров на 16, 32 и 64. в дальнейшем модель может не обучатся, а только запоминать данные, это называется переобучения, и чтобы избежать этого мы должны использовать еще один слой, называется слой регуляризации. данные разворачиваются в одномерный массив перед подачей в полносвязные слои (рисунок 13).

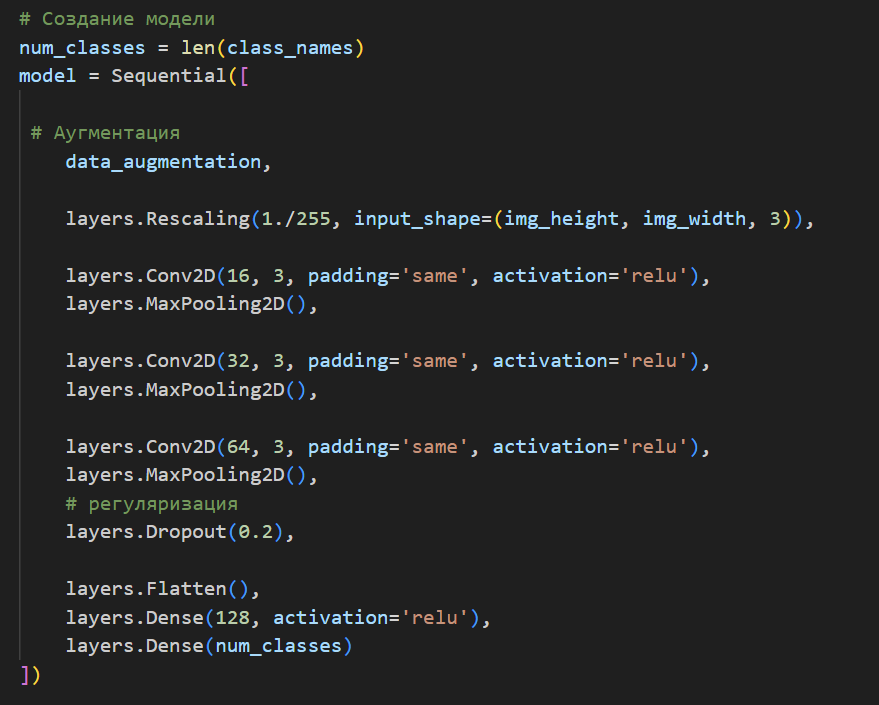


Рисунок 13 – Архитектура нейронной сети для классификации изображений цветов

дальше для компиляции модели назначаем оптимизатор, который будет задействован в процессе минимизации ошибки и корректировку модели, потом вводим функцию потерь, которая предназначен для многоклассовой классификации, для оценки соответственности прогноза модели истинным меткам. после того как произвели компиляцию вызываем метод для визуализации архитектуру модели, количество параметров и размерности входных и выходных данных для каждого слоя. после этого идет процесс обучения, в этом процессе мы используем обучающая выборку и валидационную, это делается для подбора параметров и оценки производимости модели. процесс обучения повторяется несколько раз, это называется эпоха, это для того чтобы стабильно улучшать свои прогнозы.

за этим идет сбор и анализ данных, в этом этапе данные о точности и ошибке сбираются на обучающих и проверочных данных, они позволяют анализировать процесс обучения и если результаты плохо себя показали, то можно корректировать для более лучшего результата. после того как нашли наилучшую обученную модель, оно сохраняется чтобы не повторять процесс обучения, так как это занимает немало времени (рисунок 14-15).

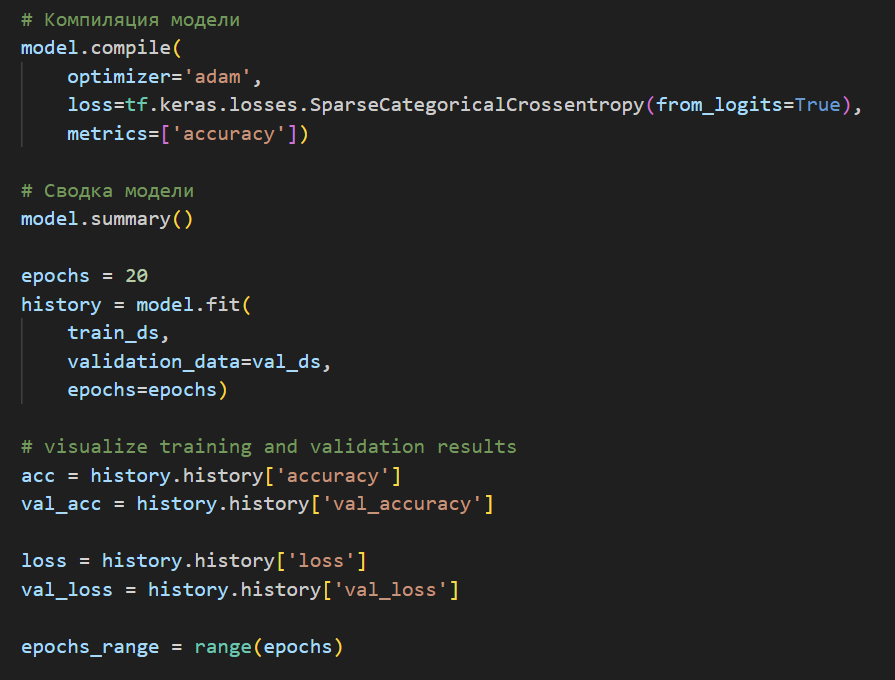


Рисунок 14 – Компиляция, обучение и визуализация результатов модели

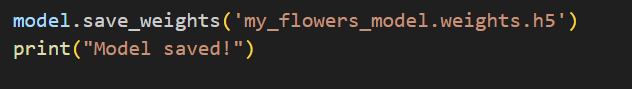


Рисунок 15 – сохранения модели

визуализируем результаты обучения модели. Создаются два графика, которые помогают понять, как модель обучалась и насколько она эффективно обобщает данные. было использовано библиотека Matplotlib, которые рисует графики. интерфейс графиков включает в себя два графика с заголовками, чтобы можно было их различать тренировочной и валидационную выборку, первый график показывает зависимость изменения точности на тренировочной и валидационной выборках от количества эпох. Второй график показывает тоже самое, но только демонстрирует динамику ошибки (рисунок 16).

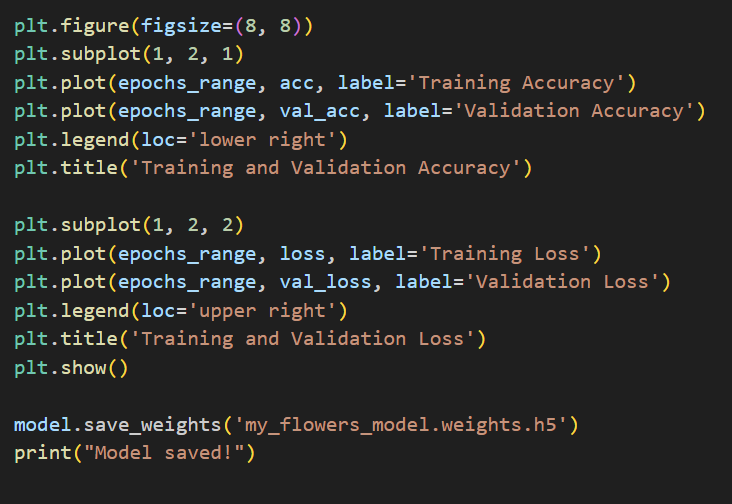


Рисунок 16 – Графики точности и ошибки на тренировочной и валидационной выборках

Для нахождения наилучшую обученную модель были созданы две модели, первая модель на 10 эпох без аугментации и регуляризации, вторая модель на 20 эпох с аугментацией и регуляризацией. на первом графике демонстрируется переобучения, который способствует запоминания а не обобщать. причине этому было низкая точность на валидационной выборке, при этом она практический не улучшается, хотя на обучающей выборке она быстро растет с каждой эпохи. а вот на втором графике показала более лучшие значения выборок, так как мы добавили метод аугментации и регуляризации и это снизило риск обучения, и это означает, что модель повысила свою устойчивость и дала положительные результаты(рисунок 17-18)

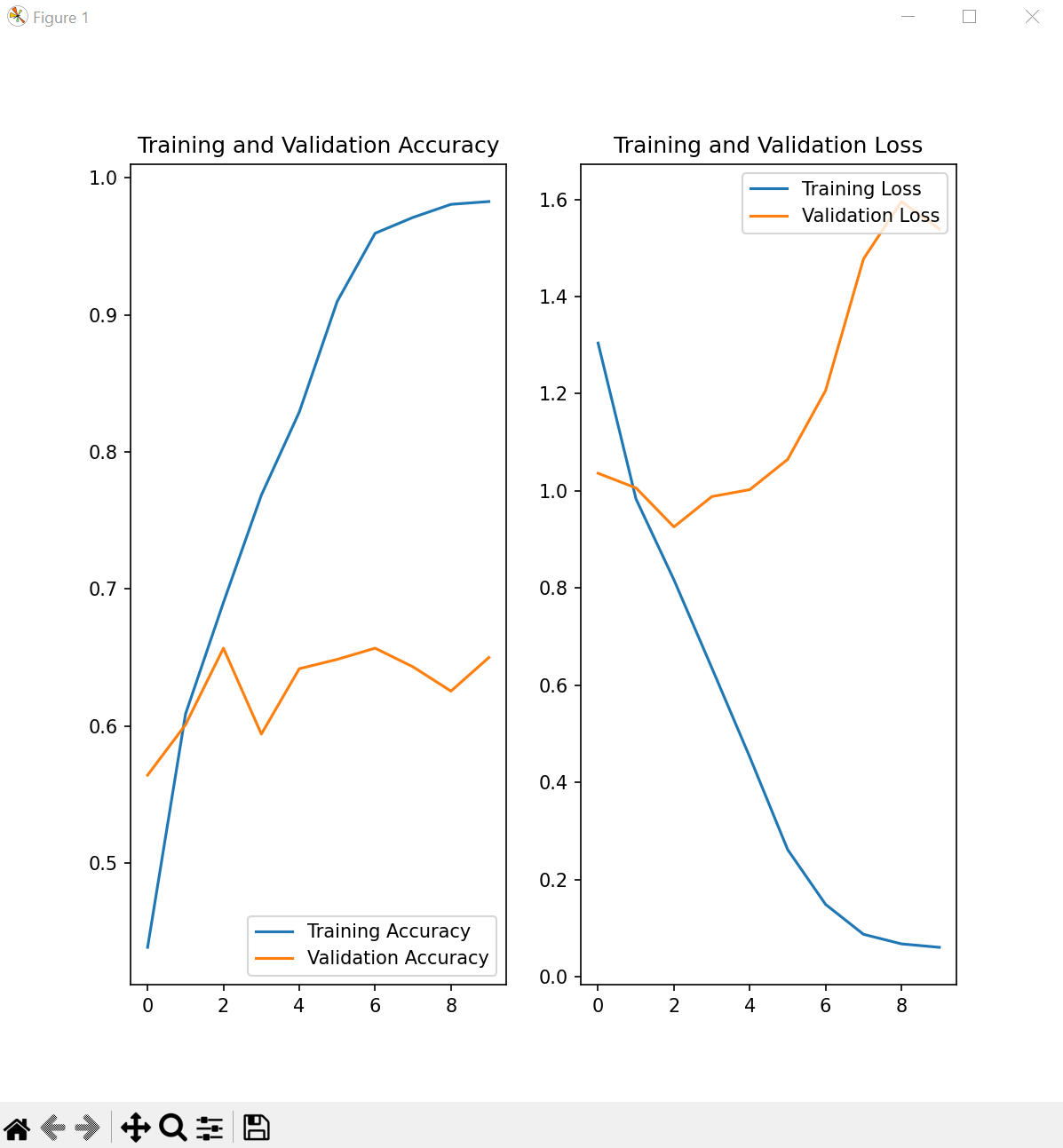


Рисунок 17 – Результаты обучения модели без аугментации данных и регуляризации (10 эпох)

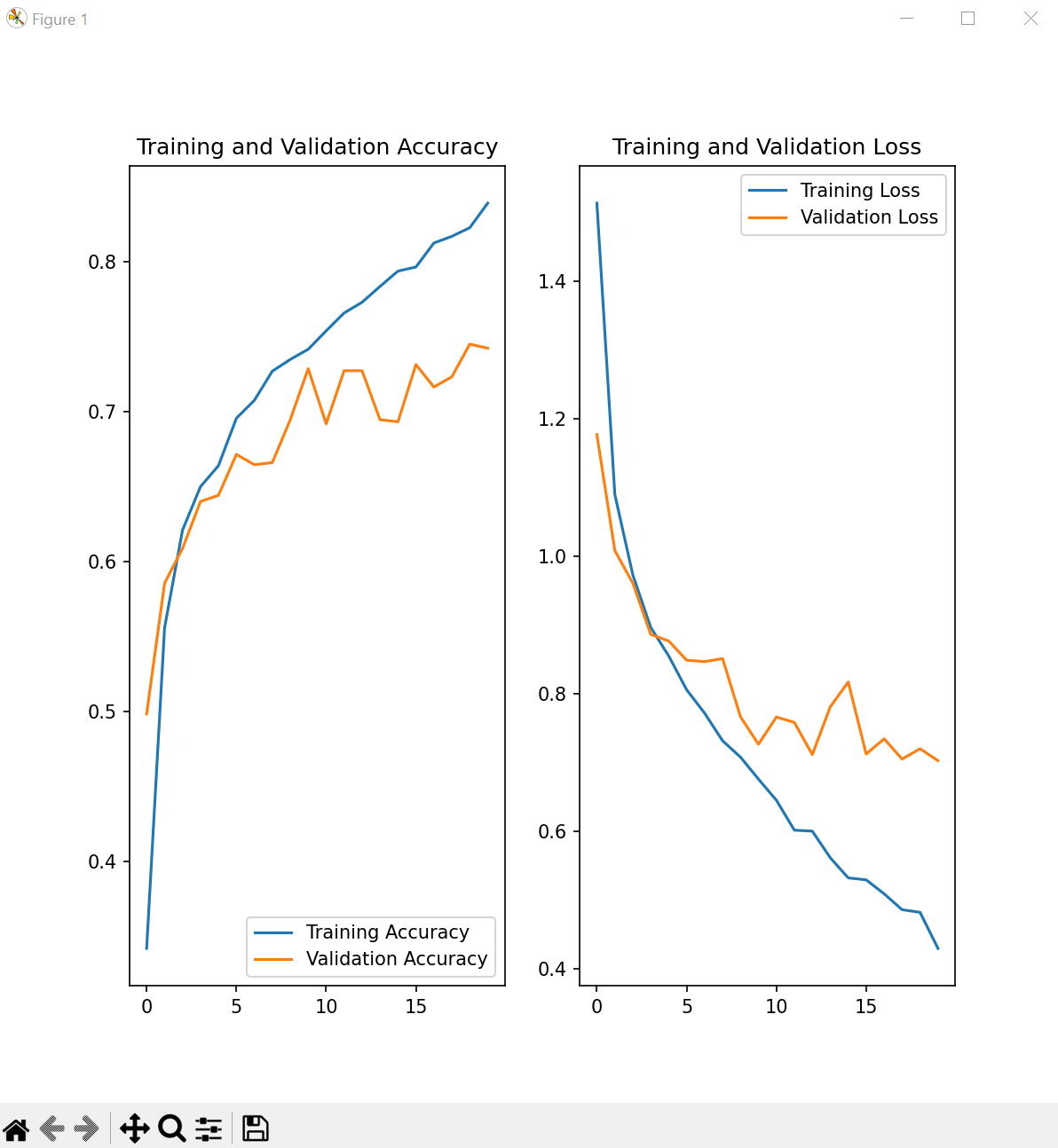


Рисунок 18 – Результаты обучения модели с аугментацией данных и регуляризацией (20 эпох)

После нахождения более лучших результаты модели сохраняем ее (рисунок 19).



Рисунок 19 – сохранённая модель

после того как мы сохранили наилучшую модель, можно перейти к основному этап, был создан файл PY, который будет загружать сохраненную модель и оценивать точность модели в процентах. для подготовки модель для тестирование, надо загрузить изображения цветка, который надо определить его, он преобразовывается в массив и выполняет предсказания, используем функцию для получения вероятности и определяем класс с максимальной вероятностью (рисунок 20).

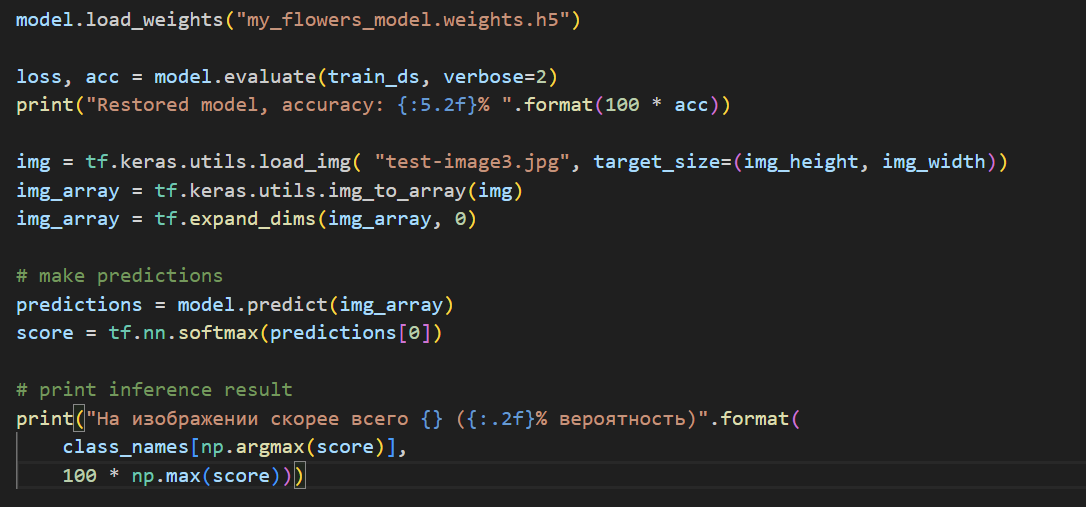


Рисунок 20 - Загрузка весов модели и предсказание на новом изображении

Для тестирования было использовано две картинки, картина с ромашкой и подсолнуха (рисунки 21-22).



Рисунок 21- картина подсолнуха для тестирования



Рисунок 22 – картина ромашки для тестирования

Результатами работы модели на тестовых данных продемонстрировала, что в изображении подсолнуха модель показывает точность 89.58 процентов что свидетельствует о хорошем уровне обучения. хотя модель определила, что изображение скорее всего относится к подсолнухи с ве-роятностью 69.83% это говорит о том, что изображение показала признаки, которые ассоциируется с подсолнухами. Однако в изображении, которое изображено ромашка вероятность показывает 99 процентов. Это говорить о высокой точности модели. оно успешно выделила ключевые показатели, типичный для ромашек, что подтверждает её способность к точной классификации (рисунок 23-24).

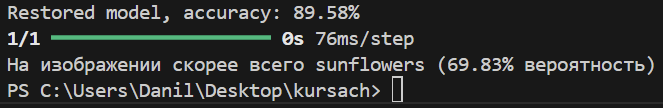


Рисунок 23 – Результат предсказания: подсолнухи (69.83% вероятность)

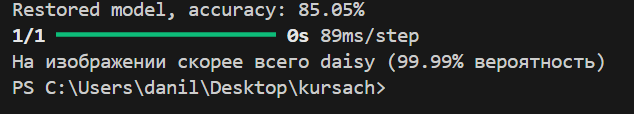


Рисунок 24 – Результат предсказания: ромашки (99.99% вероятность)

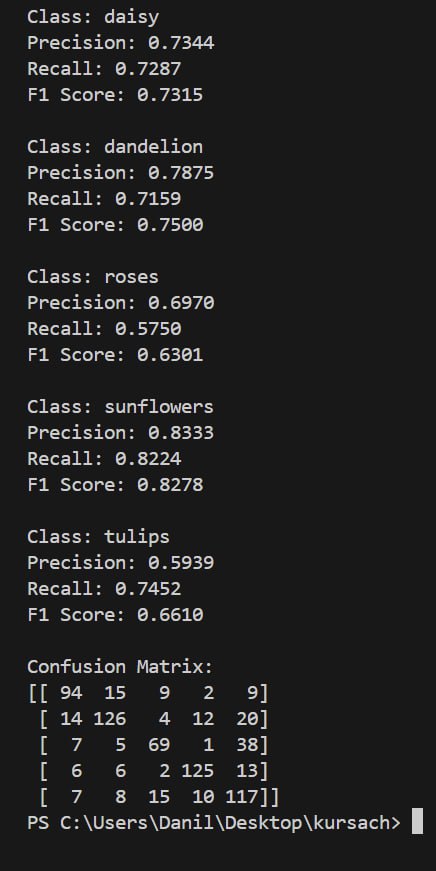


Рисунок 25 – Результат метрики Precision и Recall

# **ВЫВОД**

в ходе работы было создано модель на основе ИНС для распознавания цветов. Модель показала хорошую производимость в задаче классификации и точность в 89% . исследование показали хорошую точность, например для "ромашки" модель показала точность в 99%, и время обработки занимает в среднем 77 миллисекунд, что доказывает отличную производимость модели. но при тестирование модель для вида цветка "подсолнуха" показали некоторые недостатки, потому что точность показала 69%. это говорить от том, что в нашем датасете есть схожесть в некоторых видов цветов, что уменьшает процент точности. еще может быть причина из-за того, что датасет маленький и ограниченный виды цветков, что ограничивает обобщающую способность модели.

эту модель можно улучшить еще лучшее, увеличивая объем датасета и видов цветов, желательно выбирать не однотипные изображения, а добавлять изображения разных ракурсов, яркости и фоном для большего разнообразия датасета и точности модели. еще можно улучшить метод аугментации, такие как добавления шума и изменение контрастности, это делает модель более устойчивым. так же можно использовать сложные архитектуры нейронные сети для улучшения способности модели выделать сложные признаки.

|  |  |  |
| --- | --- | --- |
| Цветок | Точность | Причина |
| ромашки | 99% | Отличная производимость модели |
| подсолнух | 69% | * датасете есть схожесть в некоторых видов цветов * датасет маленький и ограниченный виды цветков |