МИНИСТЕРСТВО НАУКИ И ВЫСШЕГООБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

Отчёт по предмету «Технологии программирования на Python»

Обучающийся _____ М.И. Жиляев
(*подпись*)

Преподаватель _____ Н.В.Головастиков
(*подпись*)

Самара 2025

СОДЕРЖАНИЕ

1 Введение

Основная задача работы — разработка комплексного программного решения для загрузки, анализа и хранения изображений кошек с использованием внешнего API (The Cat API). В ходе выполнения проекта были успешно освоены и применены на практике следующие современные подходы в программировании на Python:

1) Методы компьютерного зрения, включая операцию свёртки, выделение границ объектов и обнаружение углов.
2) Механизмы для эффективной параллельной обработки данных: асинхронные функции и многопроцессорные вычисления.
3) Принципы взаимодействия с внешними веб-сервисами по протоколу REST.
4) Инструменты для отладки и контроля качества: всестороннее логирование и модульные тесты.
5) Практика создания хорошо структурированного, распространённого кода, оформленного в виде пакета, готового к установке и публикации в репозитории PyPI.

Разработка велась поэтапно, в рамках пяти лабораторных работ, где каждый новый этап вносил новый функционал и совершенствовал архитектурное решение.

1.1 Первое задание

В рамках первой лабораторной работы необходимо было:

1. Реализовать консольное приложение для обработки изображений
2. Самостоятельно реализовать основные функции обработки изображений: свёртку изображения с ядром, преобразование RGB-изображения в оттенки серого, гамма-коррекцию, выделение границ оператором Собеля, детекцию углов методом Харриса
3. Выполнить визуальное сравнение с готовыми функциями из библиотеки OpenCV

4. Добавить замер времени выполнения операций с помощью декоратора

1.2 Второе задание

Необходимо:

1. Подключиться к Cat API (https://thecatapi.com/) и получить API_KEY
2. Скачивать изображения животных с информацией о породе
3. Преобразовывать изображения в numpy-массив и выполнять выделение контуров пользовательским и библиотечным методами
4. Сохранять исходные и обработанные изображения в отдельную поддиректорию с порядковыми номерами
5. Инкапсулировать функционал в двух классах:
    a. CatImage: хранит изображение и метаданные, методы обработки, перегрузка операторов сложения/вычитания/преобразования в строку
    b. CatImageProcessor: работа с API, управление обработкой и сохранением, декоратор для замера времени методов

1.3 Третье задание

Необходимо:

1. Написать скрипт, анализирующий данные в csv-файле в соответствии с вариантом.
2. Каждый отдельный этап обработки (чтение файла, извлечение данных, агрегация) должен осуществляться в отдельном генераторе. Генераторы должны быть организованы в пайплайн.
3. Вывести результаты обработки в виде графика с помощью пакета matplotlib.

1.4 Четвертое задание

Необходимо:

1. Реализовать асинхронное скачивание и сохранение изображений (библиотеки aiohttp и aiofiles)

2. Реализовать параллельную обработку изображений с помощью свёртки в отдельных процессах (multiprocessing или ProcessPoolExecutor)
3. Порядковый номер изображения определяется в начале при получении списка URL и не меняется
4. Добавить замер времени работы программы и вывод информации об этапах обработки с указанием PID процессов

1.5 Пятое задание

Необходимо:

1. Добавить логирование с использованием модуля logging:
    a. В файл app.log (уровень DEBUG с подробной информацией: время, файл, строка)
    b. В консоль (уровень INFO с краткой информацией)
2. Создать модуль тестирования с использованием unittest, реализовать минимум 2 TestCase для проверки функционала CatImage и CatImageProcessor

2 Основная часть

Приложение построено по принципам объектно-ориентированного программирования и состоит из следующих основных компонентов:

2.1 Модуль CatImage.py

2.1.1 Класс CatImage

Класс CatImage инкапсулирует изображение и связанные с ним метаданные:

1. image: исходное изображение в виде numpy-массива
2. image_url: URL, с которого было загружено изображение
3. breed: порода кошки (если доступна)
4. lib_image: изображение после обработки библиотечным методом
5. custom_image: изображение после обработки пользовательским методом

Также класс еализует перегрузку операторов сложения, вычитания и преобразования в строку.

2.2 Модуль custom_image_processing.py

2.2.1 Класс CustomImageProcessing

Класс с методами обработки изображений. Содержит реализации алгоритмов компьютерного зрения:

Содержит самостоятельно реализованные алгоритмы:

1. Собственная реализация свёртки (_convolution)
2. Преобразование RGB в grayscale (_rgb_to_grayscale)
3. Оператор Собеля для выделения границ (edge_detection)
4. Адаптивный детектор Харриса с подавлением немаксимумов (corner_detection)

2.3 Модуль CatImageProcessor.py

2.3.1 Класс CatImageProcessor

CatImageProcessor отвечает за организацию процесса обработки и сохранения изображений. Основная его функция — координация многопроцессорной обработки изображений с использованием multiprocessing.Pool.

Помимо обработки, класс управляет асинхронным сохранением результатов в файлы с использованием библиотеки aiofiles.

Особенностью CatImageProcessor является его способность работать в рамках единого пайплайна, где он получает подготовленные данные от CatClient.

2.4 Модуль CatClient.py

2.4.1 Класс CatClient

Класс CatClient обеспечивает взаимодействие с The Cat API. Он выполняет синхронный запрос для получения метаданных изображений и асинхронную загрузку бинарных данных изображений. Класс поддерживает API-ключ через переменные окружения и включает обработку ошибок сетевых запросов:

2.5 Модуль logging_config.py

Модуль logging_config.py реализует двухуровневое логирование. В консоль выводятся краткие сообщения уровня INFO о ходе выполнения, старте и завершении этапов, а также об ошибках. В файл app.log записываются подробные сообщения уровня DEBUG с временными метками, информацией о файле и строке кода, деталями выполнения каждого этапа. Модуль поддерживает настройку имени файла и директории логов через аргументы командной строки, автоматически создаёт директории и форматирует сообщения.

2.6 Покрытие тестами

Используется встроенный модуль unittest для создания автоматических тестов. Для тестирования асинхронных функций применяется unittest.IsolatedAsyncioTestCase. Mock-объекты из unittest.mock позволяют тестировать API клиент без реальных сетевых запросов

Для тестирования созданы три модуля. Модуль test_cat_image.py тестирует базовый функционал класса CatImage: инициализацию объекта, обработку границ изображения, сложение изображений, обработку ошибок при несовместимых размерах. Модуль test_cat_image_processor.py тестирует основной класс обработчика: получение изображений из API с мокированием, сохранение изображений в файлы, обработку пустых списков. Модуль test_integration.py содержит интеграционные тесты с использованием Mock-объектов для проверки полного пайплайна обработки.

2.7 Сетевое взаимодействие

Для сетевого взаимодействия используются асинхронные запросы через aiohttp для параллельной загрузки нескольких изображений и неблокирующих операций ввода-вывода. Многопроцессорная обработка через модуль multiprocessing распределяет вычисления для CPU-интенсивных операций и обходит ограничения GIL. Асинхронная работа с файлами через aiofiles обеспечивает неблокирующую запись на диск и параллельное сохранение нескольких файлов.

2.8 Основные используемые библиотеки

Здесь приведена часть библиотек, использовавшихся в ходе написания лабораторных работ.

1. aiofiles – для асинхронной записи в файл.
2. aiohttp – для асинхронного обращения к API.
3. pillow – для сохранения изображений.
4. numpy – для реализации кастомных методов работы с изображениями.
5. opencv-python – для обработки изображений и сравнения с кастомной реализацией алгоритмов.
6. asyncio – для работы с асинхронными операциями.

2.9 Инструкция по запуску

Далее приведена последовательность действий по запуску программы.

1. Клонировать репозиторий: git clone https://github.com/tssvett/6401zhilyaevmi
2. Создать виртуальное окружение: python -m venv .venv
3. Активировать виртуальное окружение: .venv\Scripts\activate
4. Установить зависимости: pip install -r requirements.txt
5. Создать файл .env в корне проекта с полями API_KEY и BASE_URL=https://api.thecatapi.com/v1/images/search
6. Запустить основной пайплайн обработки изображений: python -m lab5
7. При желании запустить тесты: python -m unittest discover lab5/tests -vv

Если все настроено правильно и сервер catapi доступен, то после запуска программы информация о ее работе будет выводиться в консоль. Пример вывода представлен на рисунке 1.

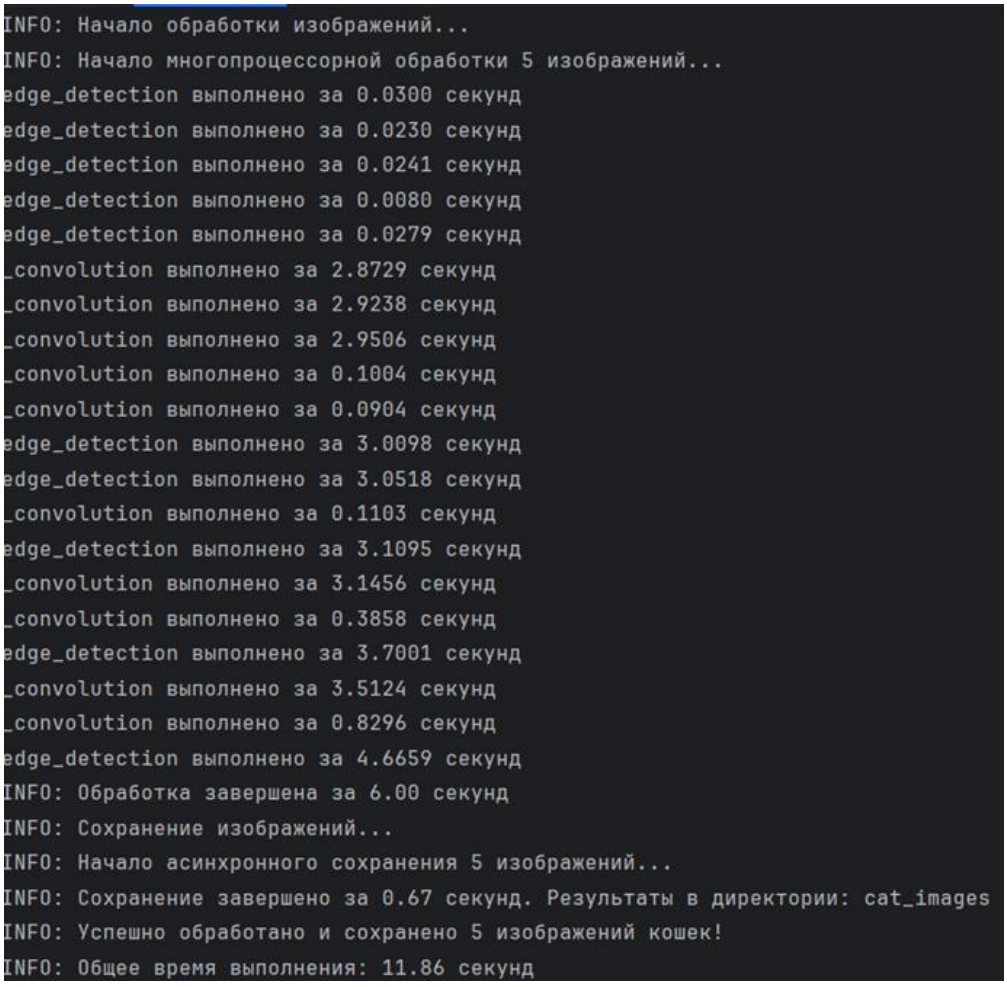

Рисунок 1 – Пример работы программы

После этого в папке data будут сохранены изображения до и после обработки. Пример представлен на рисунке 2.

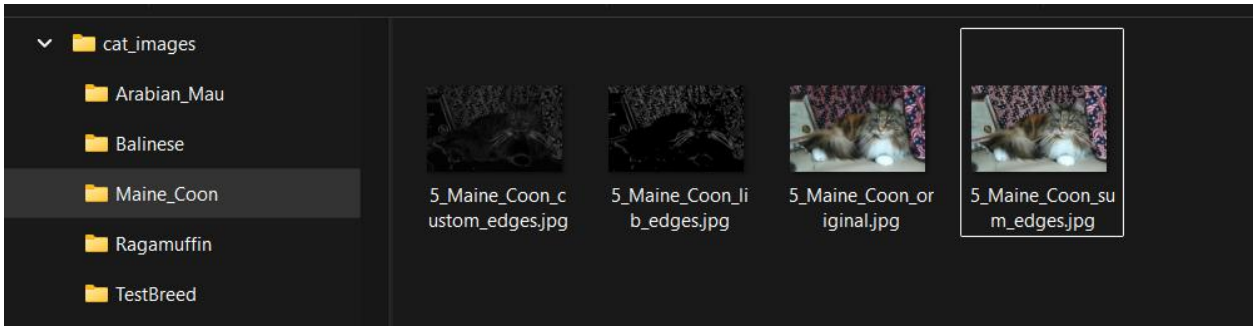

Рисунок 2 – Пример сохранения изображений

3 Заключение

В ходе выполнения лабораторных работ было создано полнофункциональное приложение для обработки изображений с использованием внешнего API. Успешно освоены следующие технологии и методы: асинхронность и многопроцессорность, алгоритмы обработки изображений, логирования в многопроцессорном коде, unit-тесты с использованием Mock-объектов.

Приложение А – ссылка на репозиторий GitHub

Приложение размещено по ссылке: https://github.com/tssvett/6401zhilyaevmi

Приложение Б – основные зависимости проекта

Далее перечислены основные зависимости проекта.

1. aiofiles>=23.0.0
2. aiohttp>=3.9.0
3. pillow>=10.0.0
4. numpy>=1.24.0
5. scipy>=1.11.0
6. opencv-python>=4.8.0
7. python-dotenv>=1.0.0