Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПО ПРОГРАММЕ БАКАЛАВРИАТА

09.03.04 Программная инженерия

(код, наименование ОПОП ВО: направление подготовки, направленность (профиль))

«Разработка программно-информационных систем»

Интеллектуальная система распознавания и классификации возгораний, полученных с БПЛА

(название темы)

Дипломный проект

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР _____ Д. А. Каракчиев

(подпись, дата) (инициалы, фамилия)

Группа ПО-02б

Руководитель ВКР _____ Р. А. Томакова

(подпись, дата) (инициалы, фамилия)

Нормоконтроль _____ А. А. Чаплыгин

(подпись, дата) (инициалы, фамилия)

ВКР допущена к защите:

Заведующий кафедрой _____ А. В. Малышев

(подпись, дата) (инициалы, фамилия)

Курск 2024 г.

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

УТВЕРЖДАЮ:

Заведующий кафедрой

_____

(подпись, инициалы, фамилия)

«____» _____ 20____ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ ПО ПРОГРАММЕ БАКАЛАВРИАТА

Студента Каракчиева Д.А., шифр 20-06-0391, группа ПО-02б

1. Тема «Интеллектуальная система распознавания и классификации возгораний, полученных с БПЛА» утверждена приказом ректора ЮЗГУ от «04» апреля 2024 г. № 1616-с.

2. Срок предоставления работы к защите «11» июня 2024 г.

3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

1) определить функциональные и технические требования разрабатываемого приложения;

2) разработать концептуальную модель системы поиска и классификации возгораний на данных, полученных с БПЛА;

3) спроектировать программную систему поиска и классификации возгораний;

4) сконструировать и протестировать программную систему поиска и классификации возгораний.

3.2. Входные данные и требуемые результаты для программы:

1) Входными данными для программной системы являются: данные справочников комплектующих, конфигураций, ПО ; данные изображения,

полученные с БПЛА, на которых предположительно могут быть зафиксированы участки возгорания; данные изображения, отформатированные в отдельных приложениях; модель нейронной сети по локализации возгораний.

2) Выходными данными для программной системы являются: изображения, с выделенными рамками вокруг участков возгорания и вероятность их наличия; сведения о выполненных анализах изображений, включая вероятность и точность обнаружения возгораний; выходные отчёты (инфографика) – по результатам анализа изображений, по эффективности обнаружения возгораний, по работе системы в целом; отчет с информацией о изображении; сведения о точности ошибки в тензоре нейронной сети.

4. Содержание работы (по разделам):

4.1. Введение

4.1. Анализ предметной области

4.2. Техническое задание: основание для разработки, назначение разработки, требования к программной системе, требования к оформлению документации.

4.3. Технический проект: общие сведения о программной системе, проект данных программной системы, проектирование архитектуры программной системы, проектирование пользовательского интерфейса программной системы.

4.4. Рабочий проект: спецификация компонентов и классов программной системы, тестирование программной системы, сборка компонентов программной системы.

4.5. Заключение

4.6. Список использованных источников

5. Перечень графического материала:

Лист 1. Сведения о ВКРБ

Лист 2. Цель и задачи разработки

Лист 3. Диаграммы компонентов

Лист 4. Архитектура нейронной сети

Лист 5. Функция активации ReLU

Лист 6. Функция вычисления ошибки IoU Loss

Лист 7. Интерфейс приложения

Лист 8. Демонстрация работы программы

Лист 9. Заключение

Руководитель ВКР		Р. А. Томакова
	(подпись, дата)	(инициалы, фамилия)
Задание принял к исполнению		Д. А. Каракчиев
	(подпись, дата)	(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 110 страницам. Работа содержит 36 иллюстраций, 6 таблиц, 50 библиографических источников и 9 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: БПЛА, система машинного зрения, TensorFlow, TFRecord, распознавание образов, классификация возгораний, обработка изображений, нейронные сети, обучение с учителем, автоматизация, информационные технологии, данные с дронов, предобработка данных, моделирование, анализ данных, безопасность, аварийные ситуации, компьютерное зрение, алгоритмы, модуль, сущность, информационный блок, метод, разработчик, администратор, пользователь, приложение.

Объектом разработки является приложение, предназначенное для распознавания и классификации возгораний на изображениях, полученных с БПЛА.

Целью выпускной квалификационной работы является разработка интеллектуальной системы, предназначенной оперативно и точно идентифицировать очаги возгорания для предотвращения и минимизации ущерба от пожаров.

В процессе создания приложения были выделены основные сущности, разработаны алгоритмы для обработки и анализа изображений, использованы методы машинного обучения для обучения модели распознавания, а также создан пользовательский интерфейс для взаимодействия с системой.

При разработке приложения использовалась платформа TensorFlow для создания и обучения нейронных сетей, а также стороннее приложение LabelImg для удобной работы с изображениями.

Разработанное приложение было успешно протестировано и готово к внедрению для использования в чрезвычайных ситуациях.

ABSTRACT

The volume of work is 110 pages. The work contains 36 illustrations, 6 tables, 50 bibliographic sources and 9 sheets of graphic material. The number of applications is 2. The graphic material is presented in annex A. The layout of the site, including the connection of components, is presented in annex B.

List of keywords: UAV, computer vision system, TensorFlow, TFRecord, pattern recognition, fire classification, image processing, neural networks, supervised learning, automation, information technology, data from drones, data preprocessing, modeling, data analysis, security, emergency situations, computer vision, algorithms, module, entity, information block, method, developer, administrator, user, application.

The object of development is an application designed to recognize and classify fires in images obtained from a UAV.

The goal of the final qualifying work is to develop an intelligent system designed to quickly and accurately identify fire sources to prevent and minimize damage from fires.

In the process of creating the application, the main entities were identified, algorithms were developed for image processing and analysis, Machine learning methods were used to train the recognition model, and a user interface was created to interact with the system.

When developing the application, the TensorFlow platform was used to create and train neural networks, as well as a third-party application LabelImg for convenient work with images.

The developed application has been successfully tested and is ready for implementation for use in emergency situations.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

НС – нейронная сеть.

ИНС – искуственная нейронная сеть.

ИС – интеллектуальная система.

ИТ – информационные технологии.

ПО – программное обеспечение.

РП – рабочий проект.

БПЛА – беспилотный летательный аппарат.

ТЗ – техническое задание.

ТП – технический проект.

UML (Unified Modelling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения.

ВВЕДЕНИЕ

В условиях активного развития информационных технологий и увеличения объемов данных, особенно актуальной становится задача автоматизации процессов анализа визуальной информации. Прогресс в области машинного зрения и искусственного интеллекта позволяет создавать системы, способные эффективно распознавать и классифицировать объекты на изображениях, полученных с беспилотных летательных аппаратов (БПЛА). Использование глубоких нейронных сетей и алгоритмов компьютерного зрения открывает новые возможности для мониторинга и реагирования на чрезвычайные ситуации, такие как возгорания.

Современные исследования в области искусственного интеллекта направлены на создание алгоритмов, способных анализировать сложные и разнообразные данные с высокой степенью точности. Разработка интеллектуальной системы распознавания и классификации возгораний является одним из таких направлений. Эта система предназначена для оперативного обнаружения и точной классификации возгораний, что может значительно повысить эффективность принятия решений при борьбе с огненными стихиями.

Целью данной работы является разработка интеллектуальной системы, использующей данные с БПЛА для распознавания и классификации возгораний. Для достижения цели были поставлены и решены *следующие задачи*:

- исследовать существующие методы и подходы в области машинного зрения для распознавания возгораний;
- разработать алгоритмы для обработки и анализа изображений, полученных с БПЛА;
- создать модель глубокой нейронной сети для классификации типов возгораний;
- провести экспериментальное тестирование разработанной системы.

Структура и объем работы. Отчет состоит из введения, 4 разделов основной части, заключения, списка использованных источников, 2 приложений. Текст выпускной квалификационной работы равен 11 страницам.

Во введении сформулирована цель работы, поставлены задачи разработки, описана структура работы, приведено краткое содержание каждого из разделов.

В первом разделе на стадии описания технической характеристики предметной области приводится сбор информации и файлов для включающей выборки при обучении, на которой осуществляется обучение нейронной сети.

Во втором разделе на стадии технического задания приводятся требования к разрабатываемому приложению.

В третьем разделе на стадии технического проектирования представлены проектные решения для приложения по классификации возгораний.

В четвертом разделе приводится список классов и их методов, использованных при разработке приложения, производится тестирование разработанной интеллектуальной системы.

В заключении излагаются основные результаты работы, полученные в ходе разработки.

В приложении А представлен графический материал. В приложении Б представлены фрагменты исходного кода.

1 Анализ предметной области

1.1 Структура и Технические Характеристики БПЛА

1.1.1 Техническое устройство БПЛА

Беспилотные летательные аппараты (БПЛА), известные также как дроны, представляют собой высокотехнологичные устройства, которые находят разнообразное применение от военных операций до сельскохозяйственного мониторинга. Они могут быть сконструированы в различных формах, включая фиксированные крылья для дальних и высотных полетов или многороторные системы для более гибкого управления и вертикального взлета и посадки.

Корпус БПЛА обычно изготавливается из легких, но прочных материалов, таких как углеродное волокно, что обеспечивает оптимальное сочетание прочности и веса. Аэродинамический дизайн корпуса способствует уменьшению сопротивления воздуха, что увеличивает эффективность полета.

Силовая установка БПЛА может варьироваться от электрических моторов, которые обеспечивают тихий и экологичный полет, до бензиновых двигателей, предлагающих большую мощность и продолжительность полета. Современные БПЛА оснащены сложными системами управления, которые включают в себя автопилот, различные датчики для навигации и стабилизации, а также GPS для точного позиционирования.

Полезная нагрузка БПЛА может включать высококачественные камеры для фотографии и видеосъемки, различные датчики для сбора данных и специализированное оборудование для выполнения конкретных задач. Источники питания, такие как аккумуляторы или солнечные панели, обеспечивают энергией все системы БПЛА.

Коммуникационные системы играют ключевую роль в безопасности и эффективности полетов БПЛА, обеспечивая надежную связь между дроном и оператором. Системы безопасности, включая аварийное возвращение на

базу и парашюты, гарантируют, что БПЛА может быть безопасно возвращен в случае нештатных ситуаций.

1.1.2 Компоненты и подсистемы БПЛА

Компоненты БПЛА играют критически важную роль в их функционировании и эффективности. Здесь представлены наиболее значимые компоненты дронов:

1. Автопилот и системы управления. Эти системы являются мозгом БПЛА, обеспечивая автоматическое управление полетом. Они включают в себя микропроцессоры, программное обеспечение для управления полетом, а также датчики для стабилизации и навигации. Автопилот может выполнять задачи, такие как взлет, полет по заданным точкам, обход препятствий и посадка.

2. Системы наблюдения и датчики. Эти системы собирают данные с помощью различных датчиков, таких как камеры, инфракрасные датчики, лидары и радары. Они используются для картографирования, сбора геоданных, наблюдения и других задач, требующих визуализации или измерения.

3. Коммуникационное оборудование. Включает в себя радиопередатчики и приемники для обмена данными между БПЛА и оператором. Это обеспечивает передачу телеметрии, видео и управляющих команд в реальном времени. Также могут использоваться спутниковые системы связи для дальних полетов и обеспечения связи вне зоны прямой видимости.

1.1.3 Разнообразие моделей БПЛА и их применение в обнаружении возгораний

Беспилотные аппараты представляют собой широкий спектр аэронавтических систем, каждая из которых обладает уникальными характеристиками и способностями, делающими их подходящими для различных задач и миссий. В контексте обнаружения возгораний, БПЛА становятся неоценимым инструментом благодаря их способности быстро и эффективно собирать данные с высокой точностью.

Модели БПЛА варьируются от небольших квадрокоптеров до крупных фиксированных крыльев, каждая из которых имеет свои преимущества. Например, квадрокоптеры могут зависать в воздухе и маневрировать в ограниченных пространствах, что делает их идеальными для детального изучения определенных участков. С другой стороны, БПЛА с фиксированными крыльями способны на длительные полеты на большие расстояния, что позволяет им покрывать обширные территории при поиске признаков возгорания.

Примеры разных видов БПЛА, широко используемых для тушения пожаров и возгораний:

1. DJI Phantom 4 Pro. Этот квадрокоптер широко используется для фотографии и видеосъемки благодаря своей стабильности в полете и высококачественной камере. Он также может быть адаптирован для мониторинга и обнаружения пожаров с помощью дополнительных датчиков.

2. Parrot Anafi. Этот компактный дрон оснащен тепловизионной камерой, что делает его подходящим для поиска тепловых подписей в рамках задач по обнаружению пожаров. Его легкость и мобильность позволяют быстро развертывать его на местности.

3. General Atomics MQ-9 Reaper. Это беспилотное летательное средство с фиксированным крылом, которое первоначально разрабатывалось для выполнения военных задач, но также может использоваться для длительного мониторинга больших территорий, что делает его полезным для обнаружения пожаров.

4. Insitu ScanEagle. Этот небольшой, но мощный БПЛА с фиксированным крылом способен проводить полеты продолжительностью до 24 часов, что делает его идеальным для непрерывного наблюдения за территориями в целях предотвращения пожаров.

5. Firefly6 Pro. Этот БПЛА оснащен инфракрасными камерами для обнаружения очагов возгорания и системами для доставки огнетушащих средств. Он способен к вертикальному взлету и посадке, что позволяет ему оперативно действовать в сложных условиях и на ограниченных площадках, где традиционные средства не могут быть использованы.

Использование БПЛА в обнаружении возгораний включает в себя не только непосредственный поиск огня, но и анализ температурных аномалий, изменений в растительности и других индикаторов, которые могут указывать на потенциальную угрозу пожара. Современные БПЛА оснащены различными датчиками, включая тепловизионные камеры и датчики для анализа спектральных данных, что позволяет им обнаруживать возгорания на ранних стадиях, когда они еще могут быть локализованы и потушены с минимальными усилиями.

1.1.4 Работа камер и сенсоров на БПЛА для обнаружения возгораний

Типы камер и их основные характеристики:

1. Оптические камеры. Обеспечивают высокое разрешение и детализацию изображений, что позволяет операторам видеть мелкие детали на земле. Они могут быть оснащены зумом для увеличения участков интереса.

2. Инфракрасные камеры. Позволяют обнаруживать тепло, исходящее от объектов, что особенно полезно в условиях низкой видимости или ночью. Они могут выявлять тепловые подписи возгораний, даже если они не видны в оптическом диапазоне.

3. Мультиспектральные камеры. Сочетают в себе несколько типов датчиков для сбора данных в различных диапазонах спектра. Это позволяет анализировать растительность и обнаруживать изменения, которые могут указывать на риск возгорания.

Использование датчиков тепла и газов:

1. Тепловые датчики. Обнаруживают повышенные температуры, что может быть признаком начинающегося пожара. Они могут быть настроены на определенные пороговые значения для автоматического оповещения.

2. Газовые датчики. Способны обнаруживать наличие газов, таких как углекислый газ или метан, которые могут выделяться при горении. Это помогает в раннем обнаружении пожаров.

Обработка данных в реальном времени:

1. Аналитическое программное обеспечение. Интегрировано с БПЛА для анализа собранных данных на лету. Это позволяет операторам быстро реагировать на изменения и принимать решения.

2. Коммуникационные системы. Обеспечивают передачу данных с БПЛА на землю в реальном времени, что позволяет командам на земле координировать действия по борьбе с пожарами.

1.2 Виды пожаров и методы обнаружения

1.2.1 Типы возгораний и их особенности

Лесные пожары, пожары на промышленных объектах и городские пожары – это три основных типа возгораний, каждый из которых имеет свои уникальные характеристики и требует специфического подхода к тушению и предотвращению.

Лесные пожары часто возникают в результате естественных процессов, таких как удары молний, но также могут быть вызваны человеческой деятельностью. Они распространяются быстро, усугубляемые сухой растительностью, ветром и топографическими условиями. Лесные пожары могут охватывать огромные территории и вызывать значительный экологический и экономический ущерб. Они также могут привести к потере биоразнообразия и эрозии почвы.

Пожары на промышленных объектах представляют собой особую опасность из-за наличия взрывоопасных и токсичных материалов. Такие пожары могут возникать в результате технологических нарушений, несоблюдения правил безопасности или аварий. Они требуют быстрого и профессионального реагирования, поскольку последствия могут включать не только уничтожение имущества, но и серьезные риски для здоровья и безопасности людей, а также для окружающей среды.

Городские пожары могут возникать в жилых и коммерческих зданиях и часто связаны с неисправной электропроводкой, неосторожным обращением с огнем или умышленными поджогами. Они могут быстро распространяться

между зданиями, особенно в плотно застроенных районах, и требуют немедленного вмешательства пожарных служб. Городские пожары также представляют угрозу для жизни людей и могут привести к значительным материальным потерям.

1.2.2 Сравнительный анализ методов обнаружения очагов возгораний

Сравнительный анализ методов обнаружения пожаров включает в себя оценку различных технологий и подходов, используемых для раннего выявления и предупреждения о пожарах. Основные методы включают использование дымовых датчиков, тепловых датчиков, инфракрасных камер и систем видеонаблюдения.

Дымовые датчики являются наиболее распространенным и доступным средством, обнаруживающим частицы дыма в воздухе. Тепловые датчики реагируют на повышение температуры, что может указывать на наличие пожара. Инфракрасные камеры и системы видеонаблюдения позволяют оперативно обнаруживать источники тепла и пламени, особенно в условиях плохой видимости.

Каждый из этих методов имеет свои преимущества и недостатки. Например, дымовые датчики могут быстро срабатывать на дым, но они не всегда эффективны в открытых или хорошо проветриваемых пространствах. Тепловые датчики могут не сработать, если пожар возник вне их диапазона действия. Инфракрасные камеры и системы видеонаблюдения требуют сложной калибровки и могут быть дорогими в установке и обслуживании.

В итоге, выбор метода обнаружения пожаров зависит от конкретных условий и требований к безопасности. Важно провести тщательный анализ потенциальных рисков и определить наиболее подходящую систему для каждого конкретного случая. Современные технологии также предлагают интегрированные решения, сочетающие различные методы обнаружения для повышения надежности и эффективности системы предупреждения о пожарах.

1.2.3 Технические проблемы и сложности при обнаружении возгораний с помощью БПЛА

Одной из основных задач является обеспечение стабильности и точности полета БПЛА в различных погодных условиях. Сильный ветер, дождь и другие атмосферные явления могут существенно повлиять на управляемость и эффективность работы БПЛА.

Кроме того, необходимо точно калибровать оптические и инфракрасные камеры, чтобы они могли эффективно обнаруживать признаки возгорания на больших расстояниях и в различных условиях освещенности. Это требует сложных алгоритмов обработки изображений и может быть затруднено в случае, если на местности присутствуют другие источники тепла.

Дальность и время полета БПЛА также ограничены их энергетическими возможностями. Необходимо регулярно подзаряжать или менять аккумуляторы, что может быть проблематично в удаленных или труднодоступных районах.

Обеспечение безопасности полетов БПЛА и предотвращение столкновений с другими летательными аппаратами является еще одной важной задачей. Для этого требуется интеграция с системами воздушного контроля и соблюдение строгих правил использования воздушного пространства.

Наконец, обработка и анализ большого объема данных, собранных БПЛА, требует мощных вычислительных ресурсов и специализированного программного обеспечения, что также может быть сложностью, особенно в условиях реального времени.

1.3 Применение технологий для предотвращения пожаров

1.3.1 Использование данных об обнаруженных пожарах для оперативного реагирования и предотвращения катастроф.

С помощью беспилотных летательных аппаратов возможно быстро оценить масштабы возгорания, определить его точное местоположение и направление распространения огня. Это позволяет спасательным службам эф-

фективно распределять ресурсы и направлять пожарные команды туда, где они наиболее нужны.

Инфракрасные камеры на БПЛА способны выявлять очаги возгорания, которые невидимы для человеческого глаза, особенно в условиях сильного задымления или ночью. Это дает возможность предотвратить распространение огня на ранней стадии и снизить вероятность возникновения крупномасштабных катастроф.

Кроме того, данные с БПЛА используются для создания точных карт распространения огня, что необходимо для планирования эвакуации населения и определения безопасных маршрутов. Также они помогают в координации действий различных служб, участвующих в тушении пожаров и оказании помощи пострадавшим.

Важным аспектом является и использование данных для анализа причин возникновения пожаров и разработки мер по их предотвращению в будущем. Анализируя информацию о прошлых пожарах, можно выявить наиболее уязвимые участки территории и принять необходимые меры для уменьшения риска возгорания.

1.3.2 Роль БПЛА в операциях по тушению пожаров и организации спасательных мероприятий

Благодаря возможности сбора информации в реальном времени, БПЛА обеспечивают командам быстрый доступ к актуальным данным о ситуации на месте пожара. Это позволяет оперативно принимать решения и адаптировать стратегии тушения в соответствии с меняющимися условиями.

Координация действий различных служб спасения является ключевым элементом успешного тушения пожаров. БПЛА предоставляют командирам на местах и центрам управления операциями точные данные о распространении огня, плотности дыма и возможных опасностях. Это позволяет спасательным службам эффективно распределять ресурсы, направлять пожарные бригады в наиболее нужные точки и обеспечивать безопасность персонала.

Предоставление данных для составления планов тушения также является важной функцией БПЛА. С их помощью можно создавать детализированные карты местности, отслеживать изменения в распространении огня и определять оптимальные маршруты для подхода к очагам возгорания. Эти данные необходимы для разработки стратегий тушения, которые максимально сокращают время на борьбу с огнем и минимизируют риски для жизни и здоровья людей.

1.3.3 Влияние автоматизированных систем обнаружения на эффективность противопожарных операций

Автоматизированные системы обнаружения пожаров оказывают значительное влияние на эффективность противопожарных операций. Они сокращают время, необходимое для обнаружения пожаров, что критически важно для предотвращения их распространения. Благодаря быстрому реагированию на возгорания, возможности для локализации огня и предотвращения его распространения значительно увеличиваются.

Системы автоматического обнаружения обеспечивают увеличение точности определения местоположения пожара, что позволяет спасательным службам быстрее и точнее реагировать на чрезвычайные ситуации. Это приводит к более оперативному принятию решений и эффективному распределению ресурсов, что способствует снижению ущерба от пожаров и сохранению жизней.

1.4 Инновации в Технологиях Пожарного Дронирования

1.4.1 Прогрессивные методы классификации и локализации возгораний с применением нейронных сетей

Современные системы используют сложные алгоритмы для анализа данных с дронов и спутников, что позволяет с высокой точностью определять местоположение и характеристики возгораний. Нейронные сети, обученные

на больших объемах данных, способны распознавать различные типы пожаров и предсказывать их поведение.

Как нейронные сети могут быть использованы в этой области:

1. Сбор данных. Нейронные сети начинают с анализа больших объемов данных о пожарах, включая изображения и видео, полученные с дронов и спутников.

2. Обучение модели. Данные используются для обучения нейронных сетей распознавать различные типы пожаров и их характеристики.

3. Классификация пожаров. Обученные модели способны классифицировать пожары по типу, размеру и интенсивности.

4. Локализация пожаров. Нейронные сети анализируют геопространственные данные для точного определения местоположения пожаров.

5. Прогнозирование поведения огня. С помощью алгоритмов нейронные сети могут предсказывать направление и скорость распространения огня.

6. Оценка ущерба. Нейронные сети могут анализировать потенциальный ущерб от пожара, помогая планировать эвакуацию и ресурсное обеспечение.

1.4.2 Перспективы применения беспилотных массивов дронов для комплексного контроля за пожарами и оценки ущерба

Беспилотные массивы дронов открывают новые горизонты в области контроля за пожарами и оценки ущерба. Эти технологии предлагают революционный подход к мониторингу и реагированию на чрезвычайные ситуации, обеспечивая беспрецедентную оперативность и точность.

Системы дронов способны оперативно собирать данные с различных уголков зоны бедствия, предоставляя операторам полную картину происходящего. Использование множества дронов одновременно позволяет получать объемные данные о температуре, скорости ветра и влажности воздуха, что критически важно для оценки ситуации и принятия решений.

В будущем можно ожидать следующие инновации:

1. Автономные дроны-пожарные. Разработка дронов, способных не только обнаруживать пожары, но и самостоятельно проводить первичное тушение, например, с помощью воды или огнетушащих веществ.

2. Интеграция с Интернетом вещей (IoT). Соединение дронов с датчиками на зданиях и в лесах для создания единой сети раннего реагирования на пожары.

3. Усовершенствованные алгоритмы прогнозирования. Использование глубокого обучения для анализа данных и создания более точных моделей поведения огня, что позволит предсказывать пожары за дни и недели до их возникновения.

4. Роботизированные пожарные станции. Автоматизация пожарных станций с помощью ИИ, которые будут координировать действия дронов и наземных роботов-пожарных.

5. Системы виртуальной и дополненной реальности. Обучение пожарных с помощью VR и AR, позволяющее имитировать различные сценарии пожаров для повышения эффективности и безопасности тренировок.

6. Сетевые операции. Разработка протоколов для координации множества дронов и роботов, работающих вместе в условиях пожара, для оптимизации процесса тушения и снижения рисков для человеческих пожарных.

Эти инновации не только улучшат реагирование на пожары, но и помогут в предотвращении их возникновения, а также в минимизации ущерба и ускорении процесса восстановления после пожаров.

1.4.3 Непосредственное использование БПЛА для тушения пожаров

БПЛА могут быть оснащены датчиками для обнаружения пожаров и системами доставки огнетушащих средств, таких как вода или пена. Они могут быстро достигать труднодоступных мест и выполнять тушение на ранних стадиях пожара, что снижает риски для пожарных и повышает шансы на предотвращение распространения огня.

Преимущества:

- БПЛА могут быть запущены немедленно и достигать места пожара быстрее, чем наземные команды;

- они могут летать в районы, недоступные для пожарных машин, например, в горных или заболоченных районах;

- снижение риска для жизни пожарных, поскольку БПЛА могут выполнять опасные задачи;

- возможность сбора ценной информации о пожаре для анализа и планирования тушения.

Недостатки:

- БПЛА могут нести только ограниченное количество огнетушащего средства;

- ограниченное время полета из-за емкости аккумуляторов;

- плохие погодные условия могут ограничивать использование БПЛА;

- необходимость соблюдения авиационных правил и регуляций.

Перспективы: Развитие технологий может привести к увеличению грузоподъемности и времени полета БПЛА, а также к улучшению их устойчивости к погодным условиям. Интеграция с искусственным интеллектом может улучшить способность БПЛА к самостоятельному обнаружению пожаров и принятию решений о тушении.

Выгодно ли это: Определенно, использование БПЛА выгодно, особенно в регионах с частыми лесными пожарами и труднодоступными территориями. Они могут сократить время реагирования и уменьшить ущерб от пожаров, что в долгосрочной перспективе может быть экономически оправданным, несмотря на начальные затраты на разработку и внедрение системы в ту или иную область.

2 Техническое задание

2.1 Основание для разработки

Основанием для разработки является задание на выпускную квалификационную работу бакалавра «Интеллектуальная система распознавания и классификации возгораний, полученных с БПЛА».

2.2 Цель и назначение разработки

Программно-информационная система предназначена для автоматизации процесса обнаружения и классификации возгораний, повышения эффективности мониторинга пожароопасных зон и поддержки принятия информированных решений при реагировании на потенциальные угрозы.

Посредством создания интеллектуальной системы мы стремимся революционизировать процесс обнаружения и реагирования на пожары, а также позволить специалистам быстро и эффективно определять возгорания без лишних затрат.

Задачами данной разработки являются:

- создание информационной базы для выбора нескольких изображений для последовательной классификации;
- предоставление предварительной обработки изображения для распознавания;
- реализация классификации возгораний по типу;
- выявление оценки степени опасности возгорания;
- обучение нейронной сети на подготовленных данных;
- оптимизация параметров сети для достижения максимальной точности распознавания;
- создание удобного и эффективного пользовтельского интерфейса.

2.3 Требования пользователя к интерфейсу приложения

Приложение должно включать в себя:

- графический интерфейс пользователя;

- возможность загрузки изображений с БПЛА для их распознавания;

- отображение результата распознавания с указанием класса возгорания и степень опасности, а также уверенность в распознавании;

- возможность загрузки данных обучения для улучшения качества распознавания;

- возможность вывода полученных изображений с распознанным возгоранием для дальнейшего использования.

Композиции шаблонов программы представлены на рисунках 2.1 и 2.2.

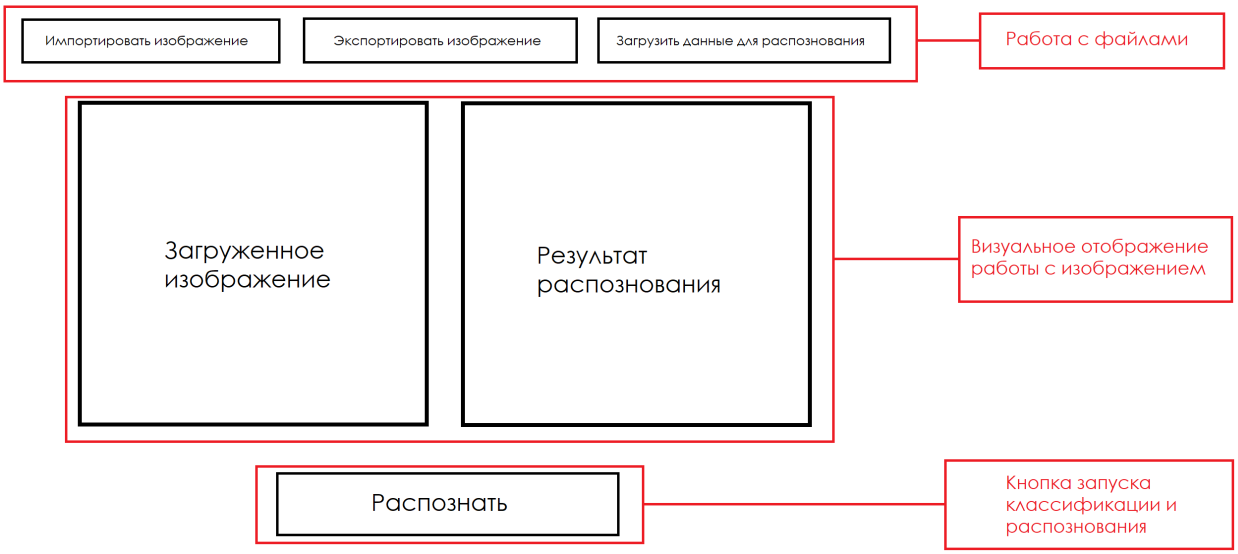


Рисунок 2.1 – Композиция шаблона программы. Окно №1.

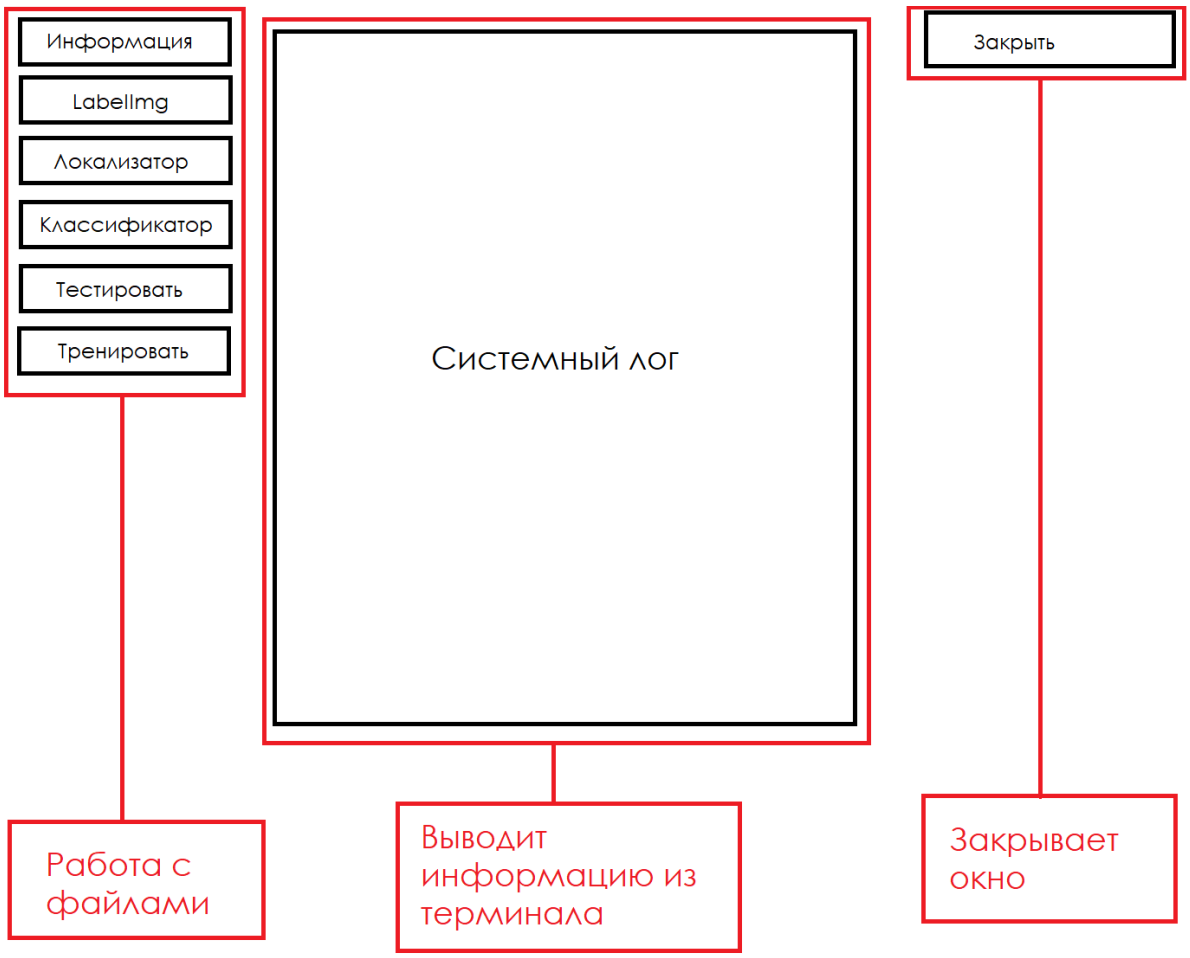


Рисунок 2.2 – Композиция шаблона программы. Окно №2.

2.4 Моделирование вариантов использования

Для разрабатываемого приложения была реализована модель, которая обеспечивает наглядное представление вариантов использования приложения.

Она помогает в физической разработке и детальном анализе взаимосвязей объектов. При построении диаграммы вариантов использования применяется унифицированный язык визуального моделирования UML.

Диаграмма вариантов использования описывает функциональность разрабатываемой системы. Она отражает взаимодействие системы с актерами, такими как операторы БПЛА, службы реагирования и специалисты. Каждый прецедент на диаграмме описывает действия системы для актеров: загрузка изображения, загрузка данных для распознавания, само распозна-

вание и вывод результирующего изображения. Диаграмма обеспечивает понимание целей системы, выявляя пробелы и соответствие потребностям заинтересованных сторон. Прецедент служит для описания набора действий, которые система предоставляет пользователю.

Диаграмма предоставлена на рисунке 2.3

Рисунок 2.3 – Диаграмма вариантов использования

На основании анализа предметной области в программе должны быть реализованы следующие прецеденты:

1. Распознание объекта (возгорания) на изображении с помощью нейронной сети.
2. Сохранение результатов обучения для дальнейшего использования.
3. Загрузка результатов обучения для дальнейшего использования.
4. Обучение и переобучение нейронной сети.
5. Сохранение записей для обучения нейронной сети.

2.5 Требования к оформлению документации

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Целью этого проекта является спроектировать и разработать приложение, которое поможет специализированным службам по тушению всеразличных возгораний вести более эффективную и проработанную деятельность.

Это приложение представляет собой интеллектуальную систему, предназначеную для автоматического обнаружения и классификации очагов возгораний. Эта система способна, при помощи нейронной сети, распознавать объекты, такие как возгорания и пожары, и определять их класс и уровень опасности на основе цветовых характеристик и наличиии задымленности на изображении, а также обучаться при помощи пользователя.

3.2 Обоснование выбора технологии проектирования

Для задачи классификации возгораний, полученных с БПЛА, используется две сверточных нейроных сетей. Такие сети хорошо подходят для задачи обработки изображений и классификации объектов, что делает их эффективным выбором для анализа данных с камер БПЛА. Она способна извлекать характерные признаки из изображений, таких как формы, текстуры и цвета, что позволяет эффективно классифицировать возгорания.

3.2.1 Описание используемых технологий и языков программирования

В процессе разработки приложения используются программные средства и языки программирования. Каждое программное средство и каждый язык программирования применяется для круга задач, при решении которых они необходимы.

3.2.2 Сверточные нейронные сети

Convolutional neural network(CNN) или Сверточная нейронная сеть - это тип искусственной нейронной сети, который широко используется для задач

обработки изображений и компьютерного зрения. Ключевой особенностью CNN является использование сверточных слоев, которые извлекают характерные признаки из входных изображений. Эти признаки могут включать в себя формы, текстуры, края или цвета.

Основная идея CNN заключается в применении сверточных фильтров к входному изображению, что позволяет выявить повторяющиеся шаблоны или признаки. Эти фильтры скользят по изображению, извлекая информацию на разных уровнях абстракции. Затем эта информация проходит через дополнительные слои, такие как подвыборка или пулинг, которые уменьшают пространственные размеры данных, и полностью подключенные слои, которые выполняют классификацию или регрессию.

CNN показали впечатляющие результаты в задачах классификации изображений, обнаружения объектов, сегментации и даже в анализе медицинских изображений. Они эффективно обрабатывают большие объемы данных, обучаясь выявлять сложные зависимости и характерные признаки.

3.2.3 Машинное обучение

Машинное обучение - это раздел искусственного интеллекта, который фокусируется на разработке алгоритмов и моделей, позволяющих компьютерам обучаться и улучшать свои задачи без явного программирования.

Применение алгоритмов машинного обучения лежит в основе нашей системы анализа и классификации данных. Мы обучаем нашу модель на обширной базе изображений, позволяя системе эффективно распознавать и классифицировать объекты в реальном времени. Этот процесс включает в себя использование сложных алгоритмов, которые могут извлекать и интерпретировать характерные особенности из данных изображений.

3.2.4 Язык программирования Python

Python является одним из самых популярных и широко используемых языков программирования для разработки приложений искусственного ин-

теллекта и машинного обучения. Он имеет простой и понятный синтаксис, что ускоряет процесс разработки и делает код более читаемым.

3.2.5 Библиотеки Python

Библиотеки, использованные в нашей программе:

1. NumPy. Фундаментальная библиотека для научных расчетов в Python. Она обеспечивает эффективную работу с многомерными массивами и матричными вычислениями, что критически важно для обработки и манипуляции данными.

2. Matplotlib. Библиотека для визуализации данных. Она позволяет создавать настраиваемые и интуитивно понятные графики, диаграммы и изображения, что облегчает визуальный анализ данных и представление результатов.

3. OpenCV (Open Source Computer Vision Library). Библиотека компьютерного зрения, которая предлагает широкий спектр алгоритмов для обработки изображений и видео. Она идеально подходит для задач обработки изображений, обнаружения объектов и анализа видео, полученных с камер БПЛА.

4. Pandas. Библиотека для анализа и манипуляции данными. Она предоставляет удобные структуры данных, такие как DataFrame, и богатый набор инструментов для обработки, фильтрации и агрегации данных, упрощая подготовку и анализ больших наборов данных.

5. TensorFlow. Это открытая платформа машинного обучения с масштабируемыми инструментами для обучения и развертывания моделей. Она обеспечивает гибкую и эффективную инфраструктуру для создания сложных нейронных сетей. Keras - это высокоуровневый API, построенный на TensorFlow, который упрощает процесс создания и обучения нейронных сетей. Он предлагает простой и интуитивно понятный интерфейс, позволяя быстро разрабатывать и экспериментировать с различными архитектурами моделей.

3.2.6 Архитектура сверточной нейронной сети

Архитектура сверточной нейронной сети включает в себя 14 слоев с различными функциями.

Схема архитектуры нейронной сети представлена на рисунке 3.1.

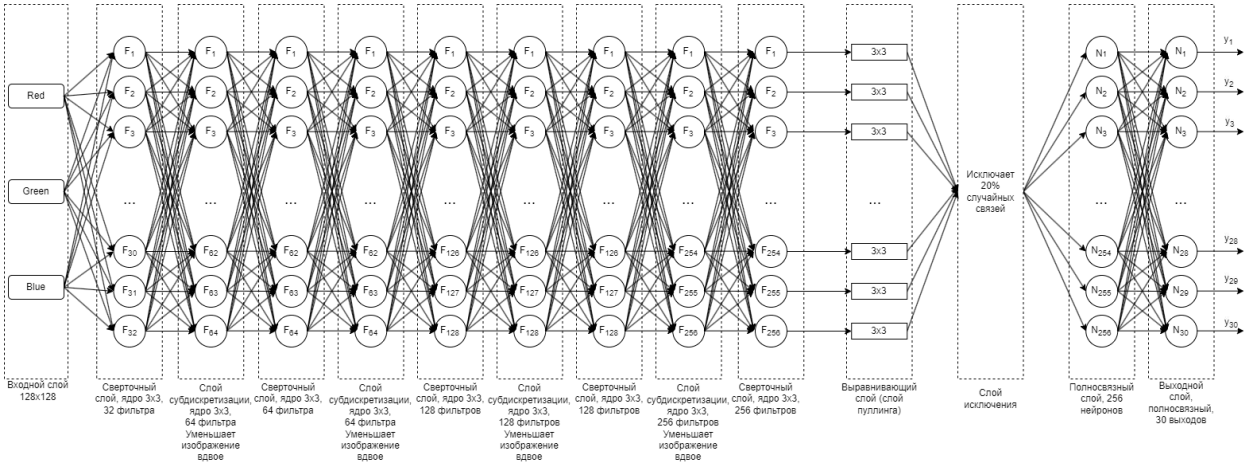


Рисунок 3.1 – Архитектура нейронной сети

Слои нейронной сети классификации описаны далее:

1. Input Layer (Входной слой).

Функция Input определяет входной слой с размером изображения 128×128 пикселей и 3 цветовыми каналами (RGB).

2. Convolutional Layers (Сверточные слои, 9).

Первый свёрточный слой Conv2D имеет 32 фильтра размером 3×3, функцию активации ReLU и параметр padding=’same’, который сохраняет размерность входного изображения.

Второй свёрточный слой Conv2D увеличивает количество фильтров до 64 и применяет шаг (strides) равный 2, что уменьшает размерность изображения вдвое.

Следующие два слоя Conv2D также имеют 64 фильтра и один из них снова применяет шаг 2 для уменьшения размерности.

Пятый и шестой свёрточные слои Conv2D содержат 128 фильтров каждый, с шагом 2 на шестом слое.

Седьмой свёрточный слой Conv2D сохраняет 128 фильтров и размерность.

Восьмой и девятый свёрточные слои Conv2D увеличивают количество фильтров до 256, с шагом 2 на восьмом слое.

3. Flatten Layer (Выравнивающий слой).

Слой Flatten преобразует многомерный тензор свёрточных слоёв в одномерный, чтобы его можно было подать на полносвязные слои.

4. Dropout Layer (Слой исключения).

Слой Dropout с параметром 0.2 предотвращает переобучение, случайным образом ”выключая”20% нейронов на каждом шаге обучения.

5. Dense Layers (Полносвязные слои, 2).

Первый полносвязный слой Dense имеет 256 нейронов и функцию активации ReLU.

Второй полносвязный слой Dense формирует выходной слой с 30 нейронами, количество которых соответствует количеству выходных значений, которые должна предсказывать модель.

3.2.7 Функция активации ReLU

ReLU, или Rectified Linear Unit, — это функция активации, которая используется в нейронных сетях для увеличения нелинейности. Формула ReLU предоставлена формулой 1.

$$f(x) = max(0, x) \tag{1}$$

Это означает, что если вход x положительный, то функция возвращает x, а если x отрицательный, то функция возвращает 0. ReLU популярна потому, что она ускоряет обучение нейронных сетей без значительной потери точности. Она также помогает решить проблему исчезающего градиента, так как производная для положительных значений всегда равна 1, что обеспечивает более быстрое и эффективное обучение.

График функции ReLu предоставлен на изображении 3.2.

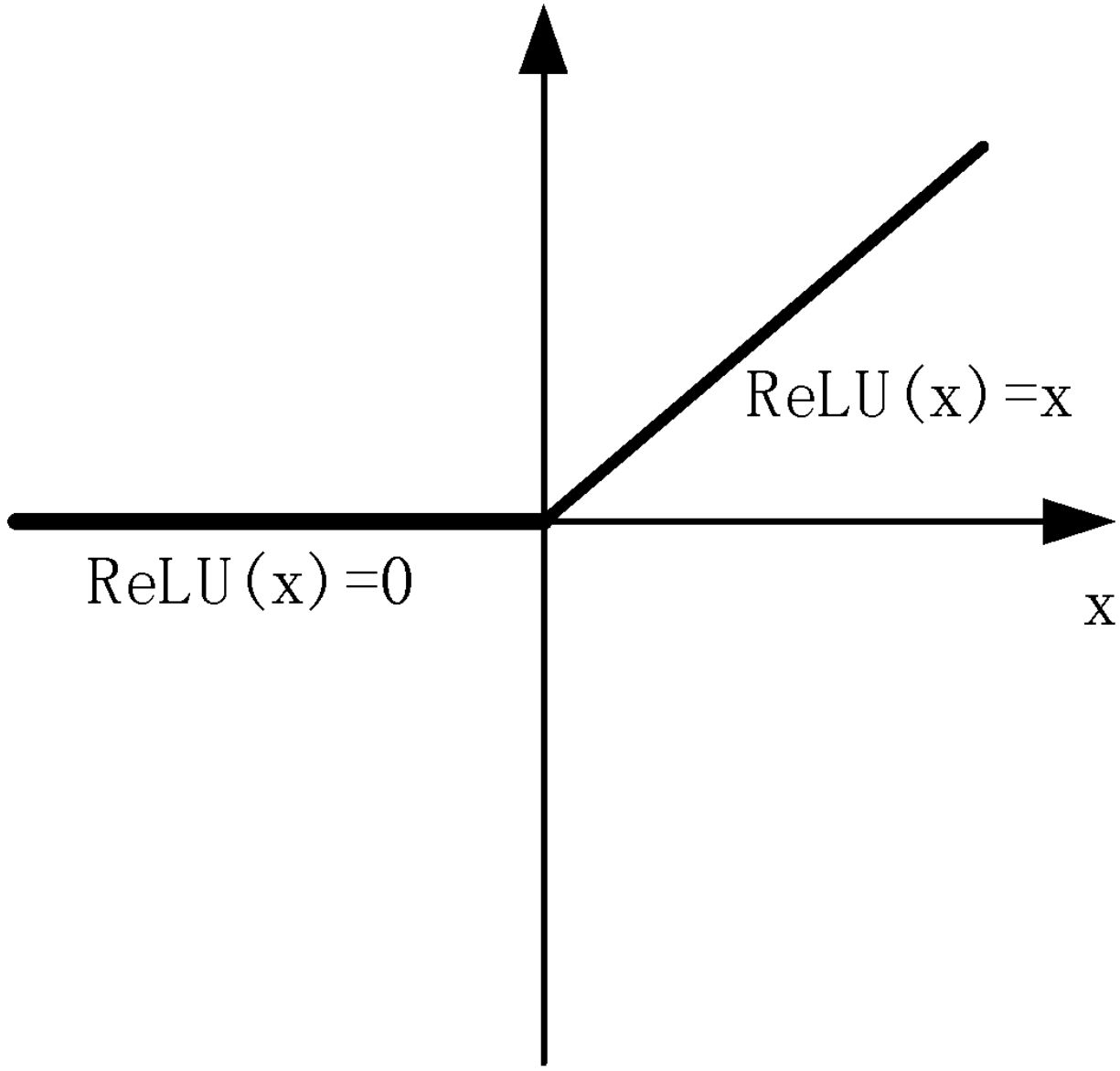


Рисунок 3.2 – График функции ReLU

3.2.8 Функция IoU Loss

Intersection over Union (IoU) является метрикой, используемой для измерения точности объектного детектора на определенном наборе данных. Если мы работаем с задачами компьютерного зрения, такими как сегментация изображений или обнаружение объектов, IoU может помочь оценить, насколько предсказанные границы объекта соответствуют истинным границам.

Наглядным образом функцию можно увидеть на изображении 3.3.

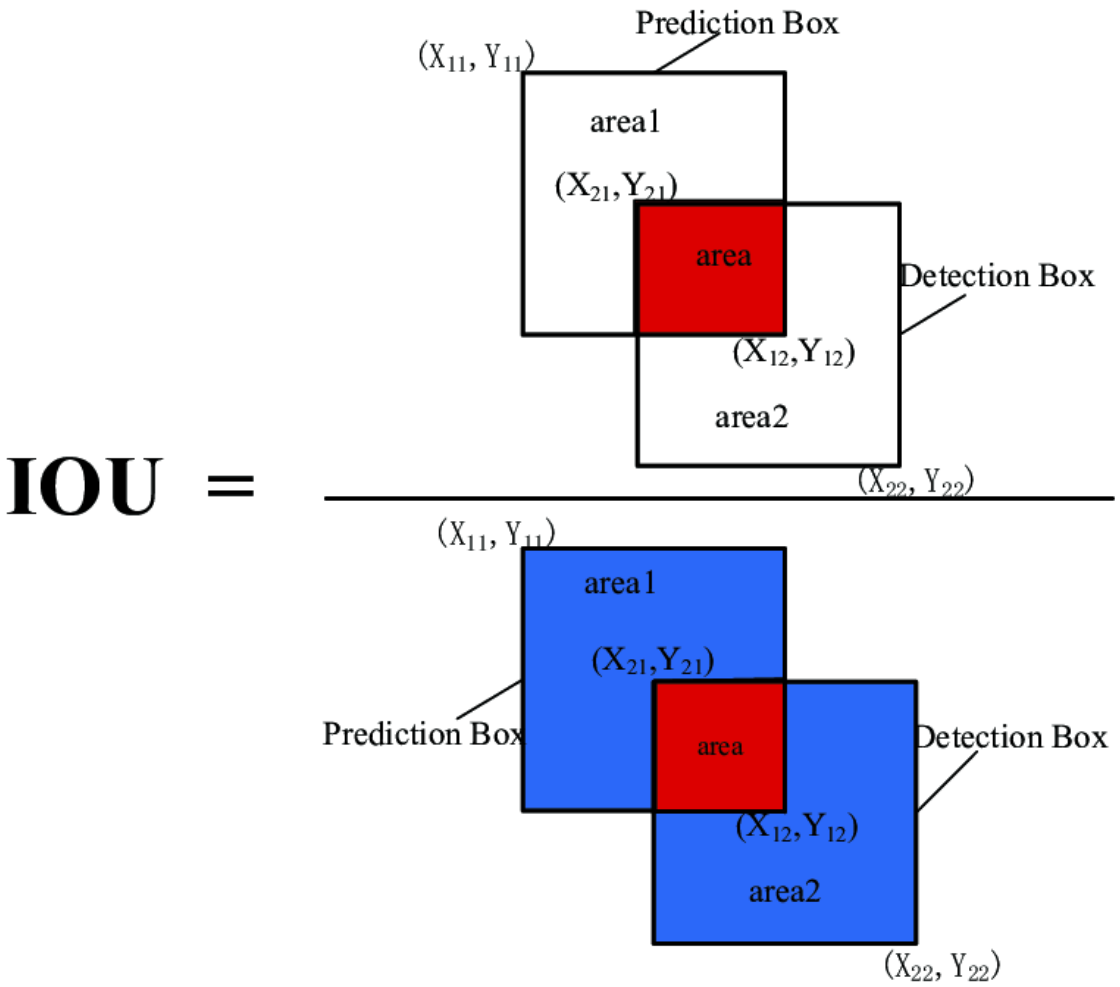


Рисунок 3.3 – График функции IoU

IoU рассчитывается по формуле 2.

$$IoU = \frac{\text{площадь пересечения}}{\text{площадь объединения}} \quad (2)$$

Где:

- площадь пересечения - это область, где предсказанная граница и истинная граница объекта перекрываются;

- площадь объединения - это область, покрытая как предсказанной границей, так и истинной границей, вместе взятых.

IoU loss - это функция потерь, которая используется для обучения моделей, выполняющих задачи, связанные с локализацией объектов. Вместо того чтобы использовать стандартные функции потерь, такие как кросс-энтропия, которые могут не полностью отражать точность локализации, IoU loss напря-

мую оптимизирует метрику IoU, стремясь увеличить площадь пересечения и уменьшить площадь объединения.

Функция потерь IoU показана в формуле 3.

$$IoUloss = 1 - IoU \tag{3}$$

Таким образом, минимизация IoU loss в процессе обучения приводит к увеличению IoU между предсказанными и истинными границами, что помогает в задаеч локализации объектов нашего проекта.

3.3 Диаграмма компонентов

Диаграмма компонентов представляет структуру системы в виде набора компонентов и их взаимосвязей. Каждый компонент отвечает за определенную функцию в рамках системы и может включать в себя подсистемы или модули. Компоненты программы:

1. Графический интерфейс. Отвечает за создание интерфейса, в котором пользователь наглядно видит результат распознования в сравнении с оригиналом.

2. Обработка изображения. Включает в себя методы улучшения качества изображений, такие как коррекция освещения, удаление шумов и извлечение важных признаков.

3. Сверточная нейронная сеть (CNN). Ядро этой системы, реализующее алгоритмы обучения и распознавания объектов.

4. Данные параметров нейронной сети. Представляют собой обученную модель, которая хранит в себе веса и параметры, извлеченные из данных во время процесса обучения.

5. Классификация объекта. Этот модуль используется для классификации возгорания.

6. Генерация отчета. Отвечает за создание отчета для выведения класса возгорания и оценки его опасности.

3.3.1 Взаимодействие компонентов

Взаимодействие компонентов проходит по такой цепочке:

1. Пользователь загружает изображение через графический интерфейс пользователя.

2. Интерфейс передает изображение в модуль обработки изображения.

3. После обработки данные передаются в модуль сверточной нейронной сети для распознавания объекта.

4. Модуль данных параметров передает веса и параметры и корректирует работу нейронной сети.

5. Результаты распознавания классифицируются в модуле классификации объекта.

6. Модуль генерации отчета выводит данные о возгорании в графическом интерфейсе.

Диаграмма компонентов представленна на рисунке 3.4.

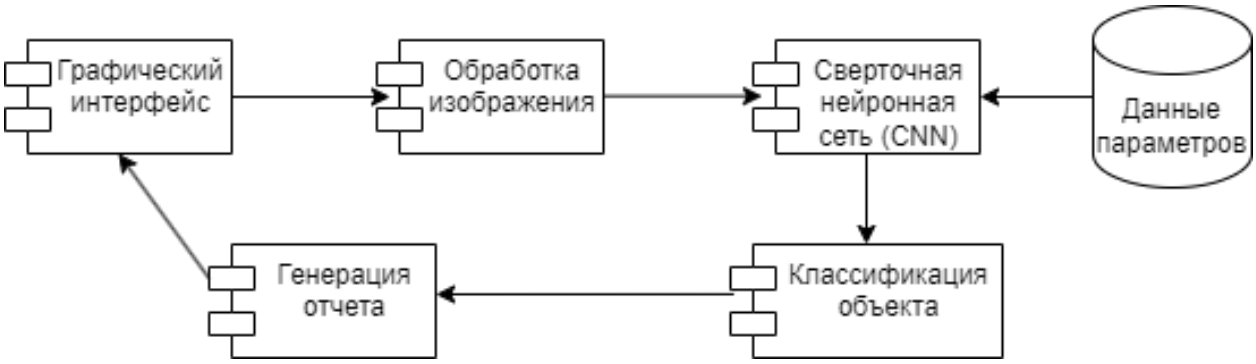


Рисунок 3.4 – Диаграмма компонентов

3.3.2 Диаграмма программных классов

Точкой входа в программу является класс main. В этом классе осуществляется запуск и инициализация основных компонентов программы:

- dataprocessing – компонент, отвечающий за обработку данных и изображений.

- creatingtfrecordclassifier – компонент, отвечающий за создание и загрузку изображений и создания их записи в формате .tfrecord.

- creatingtfrecordlocalizer – компонент, отвечающий за создание и загрузку изображений и создания их записи в формате .tfrecord.

- classifier – компонент, отвечающий за работу с нейронной сетью по классификации данных, её обучение и тестирование.

- training – компонент, отвечающий за работу с нейронной сетью по распознанию данных, её обучение и тестирование.

Диаграмма классов предоставлена на рисунке 3.5.

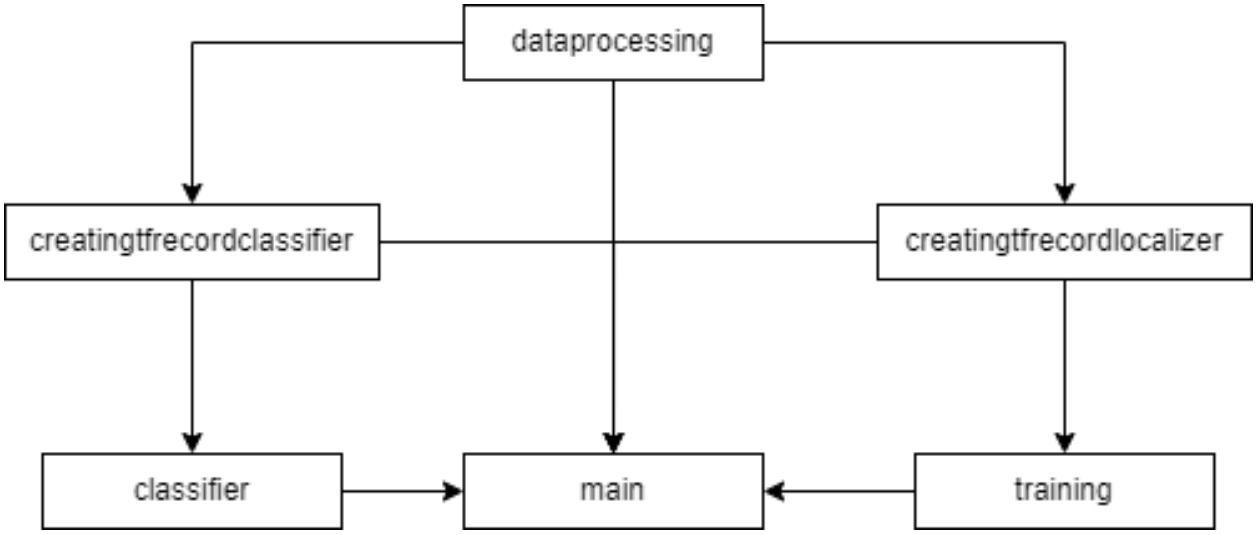


Рисунок 3.5 – Диаграмма классов и их связей

На диаграмме показаны данные связи:

1. Связь dataprocessing - main. Класс main использует основные функции класса dataprocessing, такие как отображение информации о изображении, загрузка изображения и нормализация координат изображения.

2. Связь dataprocessing - creatingtfrecordclassifier. Класс creatingtfrecordclassifier использует основные функции класса dataprocessing, такие как загрузка изображения и нормализация координат изображения. Для создания записи tfrecord необходимо создать запись со всеми изображениями и файлами формата .xml.

3. Связь dataprocessing - creatingtfrecordlocalizer. Класс creatingtfrecordlocalizer использует основные функции класса dataprocessing, такие как загрузка изображения и нормализация координат изображе-

ния. Для создания записи tfrecord необходимо создать запись со всеми изображениями и файлами формата .xml.

4. Связь creatingtfrecordclassifier - classifier. Для работы классу classifier нужен созданный в классе creatingtfrecordclassifier файл tfrecord.

5. Связь creatingtfrecordclassifier - main. Класс main использует основные функции класса creatingtfrecordclassifier, такие как создание файла tfrecord для классификатора.

6. Связь creatingtfrecordlocalizer - training. Для работы классу training нужен созданный в классе creatingtfrecordlocalizer файл tfrecord.

7. Связь creatingtfrecordlocalizer - main. Класс main использует основные функции класса creatingtfrecordlocalizer, такие как создание файла tfrecord для локализатора.

8. Связь classifier - main. Класс main использует основные функции класса classifier, такие как тестирование, обучение, загрузка и сохранение сети.

9. Связь localizer - main. Класс main использует основные функции класса localizer, такие как тестирование, обучение, загрузка и сохранение сети.

3.4 Проектирование пользовательского интерфейса

На основании требований к пользовательскому интерфейсу, представленных в пункте 2.3 технического задания, был разработан графический интерфейс приложения. Для создания пользовательского интерфейса используется библеотека tkinter и matlibplot.

На рисунке 3.6 представлен макет интерфейса окон для распознания и обучения нейронной сети. Данный макет содержит следующие элементы:

1. Загрузка изображения из системы.
2. Загрузка своей модели нейронной сети.
3. Запустить распознание изображения
4. Открыть окно тренировки модели нейронной сети.
5. Поле загруженного изображения.

6. Поле изображения с распознанным возгоранием.

7. Закрывает окно тренировки модели нейронной сети.

8. Показывает информацию о первой картинке в папке с изображениями для обучения.

9. Запускает стороннюю программу labelimg.

10. Показывает в поле для вывода информацию о наличии файлов.

11. Создает новую запись tfrecord для локализатора.

12. Создает новую запись tfrecord для классификатора.

13. Запускает функцию тестирования классификатора.

14. Запускает другую функцию тестриования классификатора с точными значениями.

15. Запускает функцию обучения классификатора.

16. Сохраняет модель нейронной сети классификатора.

17. Сохраняет модель нейронной сети локализатора.

18. Загружает модель нейронной сети локализатора.

19. Запускает функцию тестирования нейронной сети локализатора.

20. Запускает функции для обучения нейронной сети локализатора.

21. Поле вывода информации из терминала после выполнения любых операций.

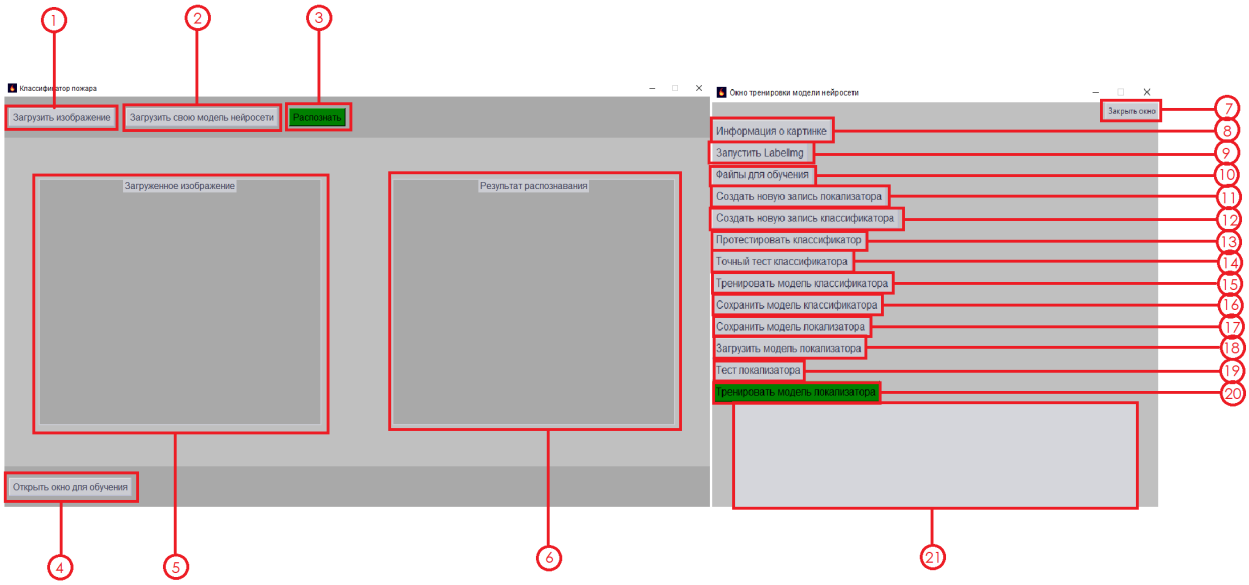


Рисунок 3.6 – Макет интерфейса окна распознавания и окна обучения нейронной сети

4 Рабочий проект

4.1 Классы, используемые при разработке приложения

Список классов и методов, которые были использованы при создании приложения представлены далее.

4.1.1 Класс main

Класс main являяется точкой входа в приложение и используется для реализации идентификации объектов с помощью нейронной сети. Здесь происходит инициализация основных компонентов программы в качестве кнопок на интерфейсе приложения. Описание полей и методов данного класса представлено в таблице 4.1.

Таблица 4.1 – Спецификация полей класса «main»

Наименование	Метод доступа	Тип данных	Описание
1	2	3	4
path_for_one	public	String	Хранит путь к выбранному изображению
localizator	public	tf.keras.Model	Модель локализатора, загруженная из файла. Файл хранится в корневой папке проекта и создается в классе training.
classifier	public	tf.keras.Model	Модель классификатора, загруженная из файла. Файл хранится в корневой папке проекта и создается в классе cllassifier.

Продолжение таблицы 4.1

1	2	3	4
window	private	Tk	Главное окно приложения. Окно настроено и имеет несколько вложений соответствующие визуальной составляющей. Содержит в себе кнопки и поля Canvas.
image_field_raw	private	Canvas	Холст для отображения исходного изображения. Содержится в окне window.
image_field_ready	private	Canvas	Холст для отображения результата распознавания. Содержится в окне window.
ConsoleRedirector	public	Class	Класс для перенаправления вывода консоли в текстовое поле text с помощью функции sys.stdout = ConsoleRedirector(text).
image_field_ready	private	Canvas	Холст для отображения результата распознавания. Содержится в окне window.

Продолжение таблицы 4.1

1	2	3	4
detect_objects	private	Function	Функция для обнаружения объектов на изображении с использованием модели локализатора. Здесь происходит подготовка изображения, локализация, нормализация, разделение на элементы, нарезание изображений и сбор в массив (10, 32, 32, 3). Далее происходит классификация, счет метки класса и сбор координат в нормальный вид.
namespace	public	Dictionary	Словарь для сопоставления индексов классов с их названиями. В нашем варианте 0 - Ничего (nothing), 1- Возгорание (fire).
visualize	private	Function	Функция для визуализации результатов детекции с помощью OpenCV. Рисует текст и квадраты на изображении в соответствии с распознанными объектами (возгораниями)

Продолжение таблицы 4.1

1	2	3	4
prettify	private	Function	Функция для улучшения результатов детекции путём объединения перекрывающихся рамок. Здесь вычисляем IoU, самый надежный способ определить совпадение. И если ни с чем не обьединили, так и оставляем результат.
detect_fire_in_image	private	Function	Функция для обнаружения огня на изображении и его визуализации. Считывает изображение, переводит его в matplotlib и выводит на Canvas поле результата.
loadimage	public	Function	Функция для загрузки изображения через диалоговое окно. Также помещает его в Canvas поле загруженного изображения.
detect	private	Function	Функция для запуска процесса обнаружения огня на загруженном изображении. Вызывает detect_fire_in_image().

Продолжение таблицы 4.1

1	2	3	4
check_and_install_packages	private	Function	Функция для проверки и установки необходимых пакетов PyQt5 и sip. Вызывается для предварительного устранения ошибок с пакетами при запуске labelimg.
run_labelimg	private	Function	Функция для запуска приложения LabelImg для аннотации изображений. Labelimg установлен заранее, но требует для запуска дополнительные пакеты в системе.
create_tfrec_classifier	private	Function	Функция для создания TFRecord для классификатора. Вызывает функцию из другого класса.
start_train_localizer	private	Function	Функция для начала обучения модели локализатора. Вызывает функцию train() и loadmodel() из другого класса.
train_window_open	private	Function	Открывает новое окно для обучения нейронной сети. Здесь реализованы функции для удобного обучения, тестирования и работы с файлами нейронной сети.
load_model_data	private	Function	Функция для загрузки пользовательской модели нейросети.

4.1.2 Класс dataprocessing

Класс dataprocessing отвечает за обработку данных и изображений. Класс предназначен для обработки данных из XML файла, который содержит аннотации для изображений, используемых в наших задачах. В классе парсится XML файл, извлекается информацию о размеченных объектах и их координатах, загружается соответствующее изображение и визуализирует его. Описание полей и методов данного класса представлено в таблице 4.2.

Таблица 4.2 – Спецификация полей класса «dataprocessing»

Наименование	Метод доступа	Тип данных	Описание
1	2	3	4
tree	public	xml.etree.ElementTree	Объект дерева XML, содержащий структурированные данные из файла XML. Значением является адрес файла для информирования.
root	public	xml.etree.ElementTree	Корневой элемент XML файла, предоставляющий доступ к дочерним элементам. Парсинг.
num_objects	public	int	Количество объектов, размеченных в XML файле. Количество размеченных объектов (6 - кол-во служебных элементов, таких как размер, название и т.д)
load_img	public	Function	Функция для загрузки и предобработки изображения с использованием TensorFlow.

Продолжение таблицы 4.2

1	2	3	4
cords	public	list	Список для хранения нормализованных координат объектов.
w	public	int	Ширина изображения, извлеченная из XML файла.
h	public	int	Высота изображения, извлеченная из XML файла.
object_cords	public	list	Список для хранения нормализованных координат одного объекта. Тут мы также нормализуем координаты от -1 до 1, опираясь на исходные координаты.
img	public	Tensor	Тензор изображения после загрузки и предобработки.
plt.figure	public	Function	Функция для создания новой фигуры в Matplotlib.
plt.subplot	public	Function	Функция для добавления подграфика в текущую фигуру.
plt.imshow	public	Function	Функция для отображения изображения в подграфике.
plt.axis	public	Function	Функция для управления отображением осей графика.
plt.show	public	Function	Функция для отображения всей фигуры с подграфиками. Открывает окно с исходным изображением.

4.1.3 Класс creatingtfrecordclassifier

Класс creatingtfrecordclassifier отвечает за создание и загрузку изображений и создания их записи в формате .tfrecord. В этом классе подготавливаются данные для дальнейшего использования в других классах и главное - создание файла classifier_dataset.tfrecord . Описание полей и методов данного класса представлено в таблице 4.3.

Таблица 4.3 – Спецификация полей класса «creatingtfrecordclassifie»

Наименование	Метод доступа	Тип данных	Описание
1	2	3	4
fn	public	str	Путь к папке с изображениями. Значением является адрес папки с нашими изображениями и xml файлами.
check_xml_list	public	function	Функция для создания списка XML файлов в указанной папке. Формируем список всех xml файлов в папке.
load_img	public	function	Функция для загрузки и предобработки изображения.
create_load_tfrec_for_classifier	public	function	Функция для создания TFRecord для классификатора. Здесь мы преобразуем папку в tfrecord для классификарора

Продолжение таблицы 4.3

1	2	3	4
namespace	public	dictionary	Словарь для сопоставления названий с метками классов. В нашем случае 0 - Ничего (nothing), 1- Возгорание (fire).
saveinrecord	public	function	Внутренняя функция для сохранения обработанного изображения и метки в TFRecord. Создается запись writer, данные предоставляются в байтовом виде и собирается экземляр, который потом записывается в запись writer.
parse_record	public	function	Функция для разбора записей TFRecord. Имена элементов остаются как при записи

4.1.4 Класс creatingtfrecordlocalizer

Класс creatingtfrecordlocalizer отвечает за создание и загрузку изображений и создания их записи в формате .tfrecord. В этом классе подготавливаются данные для дальнейшего использования в других классах и главное - создание файла localizer_dataset.tfrecord . Описание полей и методов данного класса представлено в таблице 4.4.

Таблица 4.4 – Спецификация полей класса «creatingtfrecordlocalizer»

Наименование	Метод доступа	Тип данных	Описание
1	2	3	4
fn	public	string	Путь к папке с изображениями. Значением является адрес папки с нашими изображениями и xml файлами.
load_img	private	function	Загружает изображение, декодирует, нормализует и изменяет размер.
create_load_tfrec_for_localizer	private	function	Создает TFRecord для локализатора из XML файлов.
p	private	list	Формируем список всех xml файлов в папке.
writer	private	TFRecordWriter	Записывает данные в TFRecord. Создание самой записи
tree	private	ElementTree	Адрес файла
root	private	Element	Парсит XML файлы.
num_objects	private	int	Количество объектов в XML файле.
cords	private	list	Список координат объектов.
w	private	int	Ширина изображения.
h	private	int	Высота изображения.

Продолжение таблицы 4.4

1	2	3	4
object_cords	private	list	Координаты одного объекта. Тут мы также нормализуем координаты от -1 до 1, опираясь на исходные координаты.
img	private	Tensor	Тензор изображения.
serialized_img	private	bytes	Сериализованное изображение. Готовим данные, представляем в байтовом виде.
serialized_cords	private	bytes	Сериализованные координаты. Готовим данные, представляем в байтовом виде.
example	private	Example	Пример данных для TFRecord. Собираем экземпляр
dataset	private	TFRecordDataset	Набор данных из TFRecord. Сразу после создания проверка чтения записи.
parse_record	private	function	Разбирает запись из TFRecord. Имена элементов как при записи
feature_description	private	dict	Описание приходящего экземпляра.
parsed_record	private	dict	Разобранный экземпляр.

4.1.5 Класс classifier

Класс classifier отвечает за работу с нейронной сетью по классификации данных, её обучение и тестирование. В этом классе реализован парсинг элементов из tfrecord, загрузка и создание модели нейросети, а также построение её архитектуры. Описание полей и методов данного класса представлено в таблице 4.5.

Таблица 4.5 – Спецификация полей класса «classifier»

Наименование	Метод доступа	Тип данных	Описание
1	2	3	4
parse_record	public	function	Разбирает запись TFRecord, возвращая изображение и имя.
shuffle, cache, prefetch, batch	public	method	Подготавливает датасет к обучению: перемешивание, кэширование, предварительная загрузка, батчинг.
test_classifier	public	function	Визуализирует примеры из датасета и их метки.
Model	public	class	Определяет модель нейросети с методами для обучения.
training_step	public	method	Выполняет шаг обучения, возвращая среднее значение потерь.
trainclass	public	function	Обучает классификатор и визуализирует диаграмму потерь.
imshow_and_pred	public	function	Визуализирует изображения и предсказания модели с помощью matplotlib.

Продолжение таблицы 4.5

1	2	3	4
saveclassifier	public	function	Сохраняет обученную модель классификатора.

4.1.6 Класс training

Класс training отвечает за работу с нейронной сетью по локализации данных, её обучение и тестирование. В этом классе реализован парсинг элементов из tfrecord, загрузка и создание модели нейросети, а также построение её архитектуры и явное обучение. Описание полей и методов данного класса представлено в таблице 4.6.

Таблица 4.6 – Спецификация полей класса «training»

Наименование	Метод доступа	Тип данных	Описание
1	2	3	4
parse_record	public	function	Разбирает запись TFRecord, возвращая изображение и координаты.
IoU_Loss	public	function	Вычисляет потери IoU между истинными и предсказанными рамками.
Model	public	class	Определяет модель нейросети с методами для обучения.
training_step	public	method	Выполняет шаг обучения, возвращая значение потерь.
savemodel	public	function	Сохраняет обученную модель.
loadmodel	public	function	Загружает веса модели.

Продолжение таблицы 4.6

1	2	3	4
testing	public	function	Тестирует модель, визуализируя результаты.
train	public	function	Обучает модель и визуализирует историю потерь.

4.2 Модульное тестирование разработанного приложения

Модульные тесты для класса main из модели данных представлены на рисунках 4.1-4.3.

```python
import unittest
from main import detect_objects, visualize, prettify

class TestFireDetection(unittest.TestCase):

    def setUp(self):
        self.test_image = np.zeros((1024, 1024, 3), dtype=np.uint8)
        self.test_cords = np.array([[10, 20, 30, 40] for _ in range(10)])
        self.test_classes = np.array([1 for _ in range(10)])
        self.test_probs = np.array([0.9 for _ in range(10)])

    def test_detect_objects(self):
        cords, classes, probs = detect_objects(self.test_image)
        self.assertEqual(len(cords), 10)
        self.assertEqual(len(classes), 10)
        self.assertEqual(len(probs), 10)
        self.assertTrue((cords >= 0).all() and (cords <= 1024).all())
        self.assertTrue((classes >= 0).all() and (classes <= 1).all())
        self.assertTrue((probs >= 0).all() and (probs <= 1).all())

    def test_visualize(self):
        result_image = visualize(self.test_image, self.test_cords, self.
            test_classes, self.test_probs)
        self.assertIsNotNone(result_image)
        self.assertEqual(result_image.shape, self.test_image.shape)

    def test_prettify(self):
        new_cords, new_classes, new_probs = prettify(self.test_cords, self.
            test_classes, self.test_probs)
        self.assertIsNotNone(new_cords)
        self.assertIsNotNone(new_classes)
        self.assertIsNotNone(new_probs)
        self.assertTrue(len(new_cords) <= len(self.test_cords))
```

Рисунок 4.1 – Модульный тест основных методов распознавания, визуализации и объединения класса main

```python
import unittest
from unittest.mock import patch
from main import detect_fire_in_image, loadimage, detect

class TestFireDetection(unittest.TestCase):

    def setUp(self):
        self.test_path = 'D:/NeuroPractice/NeuroPractice/src/images/fire1.jpg
            '
        self.test_image = np.zeros((1024, 1024, 3), dtype=np.uint8)

     def test_detect_fire_in_image(self):
        result = detect_fire_in_image(self.test_path)
        self.assertIsNotNone(result)
        self.assertEqual(result.shape, (1024, 1024, 3))
    def test_loadimage(self, mock_open):
        mock_open.return_value.__enter__.return_value = 'fake file content'
        image = loadimage('fake/path/to/image.jpg')
        mock_open.assert_called_with('fake/path/to/image.jpg', 'rb')
        self.assertIsNotNone(image)

    @patch('main.detect')
    def test_detect(self, mock_detect):
        mock_detect.return_value = True
        result = detect('fake/path/to/image.jpg')
        mock_detect.assert_called_once()
        self.assertTrue(result)
```

Рисунок 4.2 – Модульный тест методов распознавания возгораний, загрузки изображения и вложенного метода detect класса main

```python
import unittest
from your_module import detect_objects
import tensorflow as tf
import numpy as np

class TestObjectDetection(unittest.TestCase):
    def test_detect_objects(self):
        test_image = np.random.randint(0, 256, (128, 128, 3), dtype=np.uint8)
        test_image = tf.convert_to_tensor(test_image, dtype=tf.float32)

        cords, classes, probs = detect_objects(test_image)

        self.assertIsInstance(cords, np.ndarray, "Координаты должны быть массивом NumPy")
        self.assertIsInstance(classes, np.ndarray, "Классы должны быть массивом NumPy")
        self.assertIsInstance(probs, list, "Вероятности должны быть списком")

        self.assertEqual(cords.shape, (10, 4), "Массив координат должен иметь форму (10, 4)")
        self.assertEqual(len(classes), 10, "Массив классов должен содержать 10 элементов")
        self.assertEqual(len(probs), 10, "Список вероятностей должен содержать 10 элементов")

        for prob in probs:
            self.assertGreaterEqual(prob, 0, "Вероятность должна быть больше или равна 0")
            self.assertLessEqual(prob, 1, "Вероятность должна быть меньше или равна 1")
```

Рисунок 4.3 – Отдельный модульный тест метода распознавания класса main

4.3 Системное тестирование разработанного приложения

В целях проверки работоспособности программно-информационной системы было проведено системное тестирование. Описание тестов, их результаты и скриншоты экрана представлены в данном разделе.

На рисунке 4.4 - 4.5 представлен интерфейс программы.

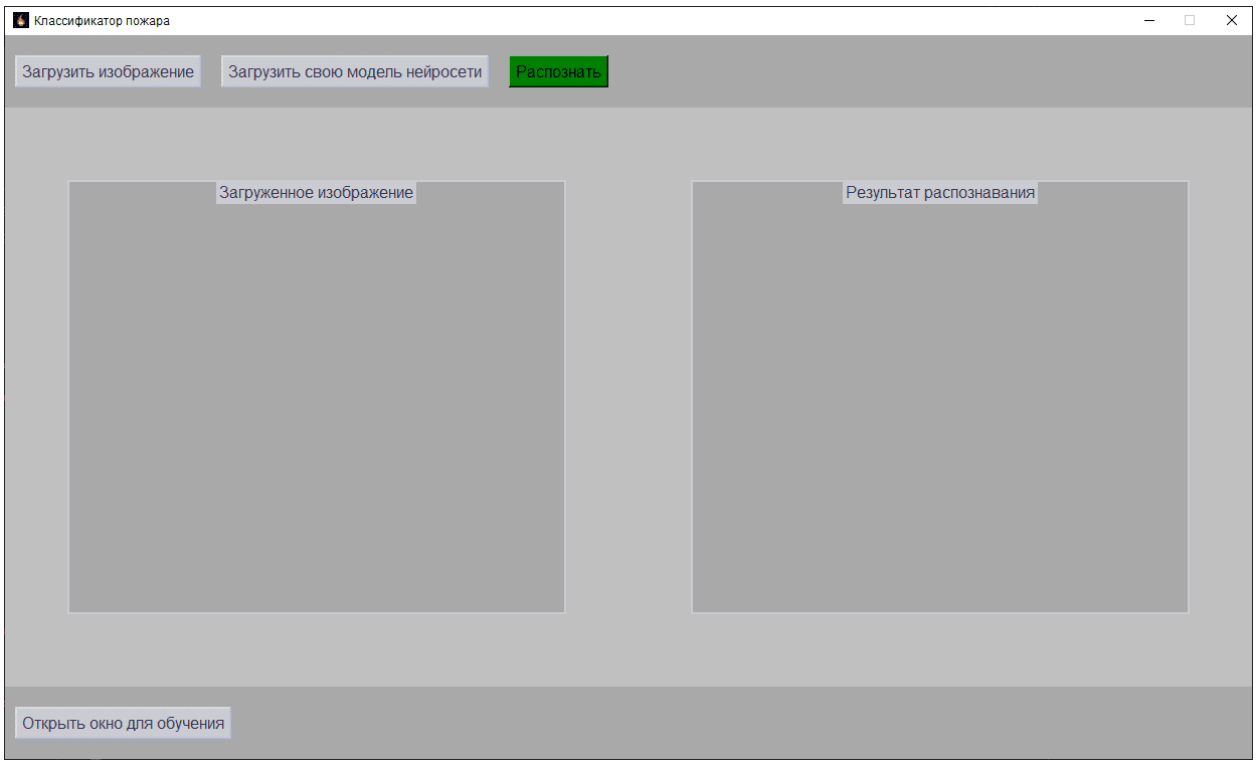


Рисунок 4.4 – Интерфейс программы. Основное окно

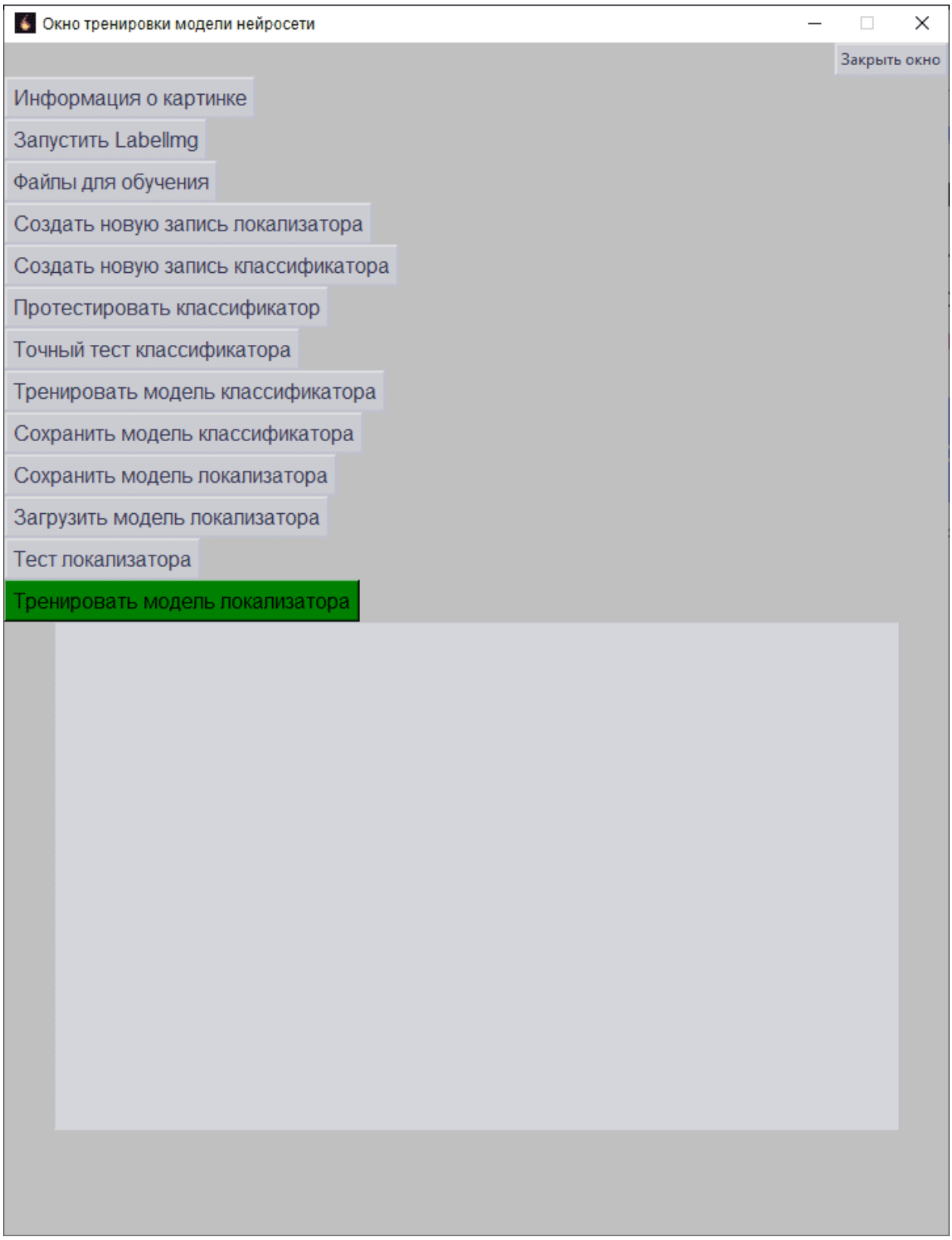


Рисунок 4.5 – Интерфейс программы. Окно обучения.

На рисунке 4.6 предоставлен вариант с загрузкой изображения.

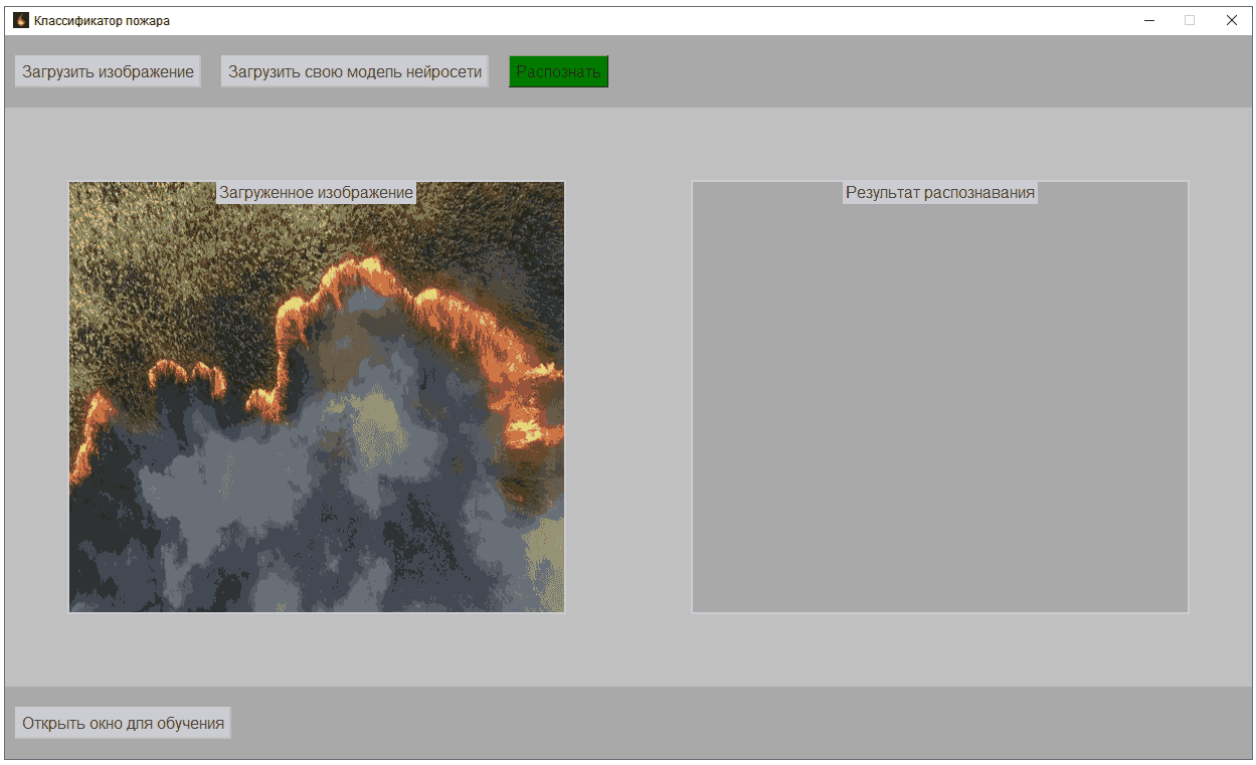


Рисунок 4.6 – Интерфейс программы с загруженным изображением.

На рисунке 4.7 предоставлен вариант с загрузкой пользовательской моделью нейронной сети. Здесь пользователь выбирает путь до файла с весами формата .keras.

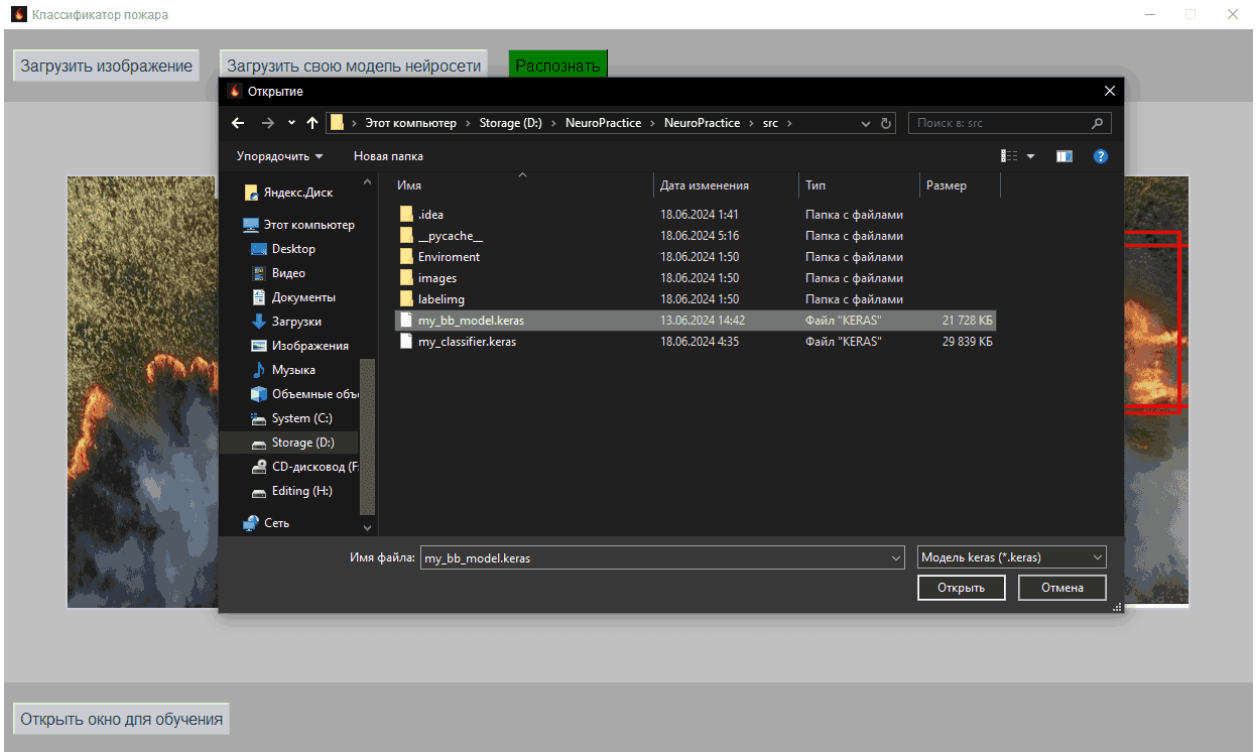


Рисунок 4.7 – Функция загрузки пользовательских весов. Выбор файла в диалоговом окне

На рисунках 4.8-4.16 представлены варианты распознования возгораний на изображениях.

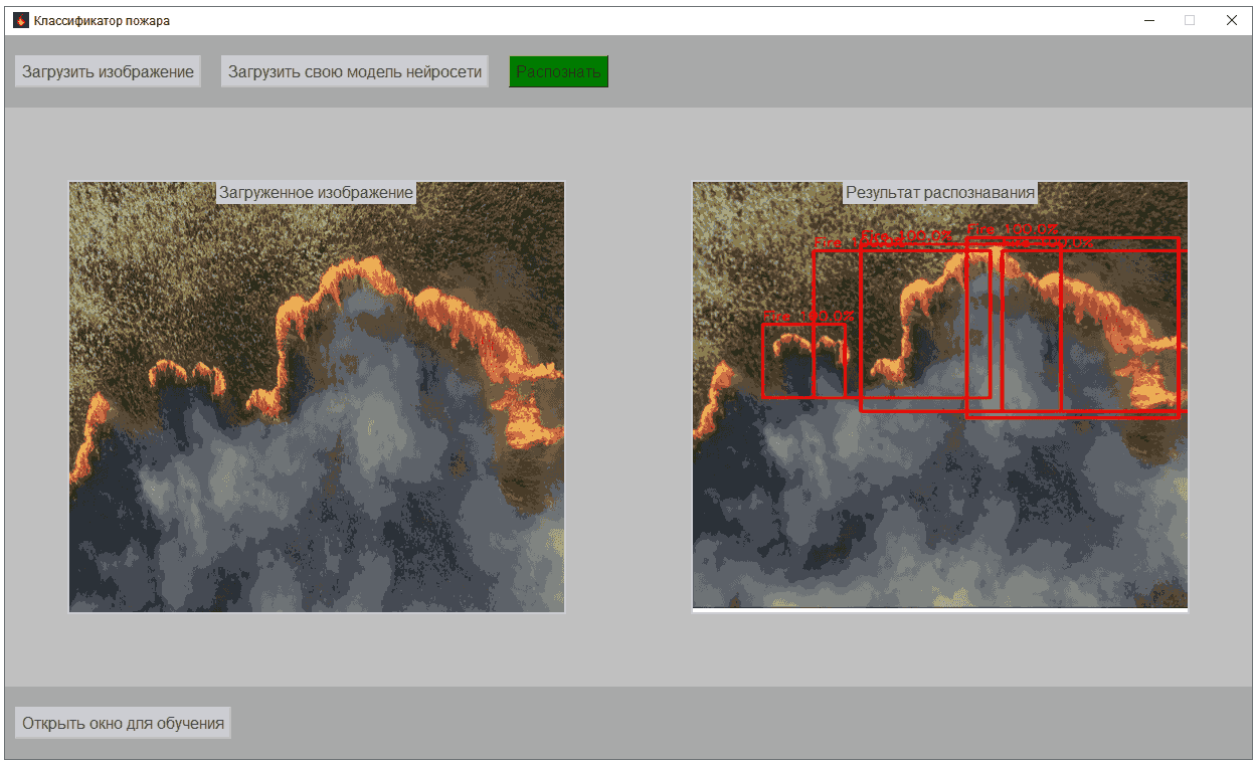


Рисунок 4.8 – Интерфейс с распознанным файлом fire1.png

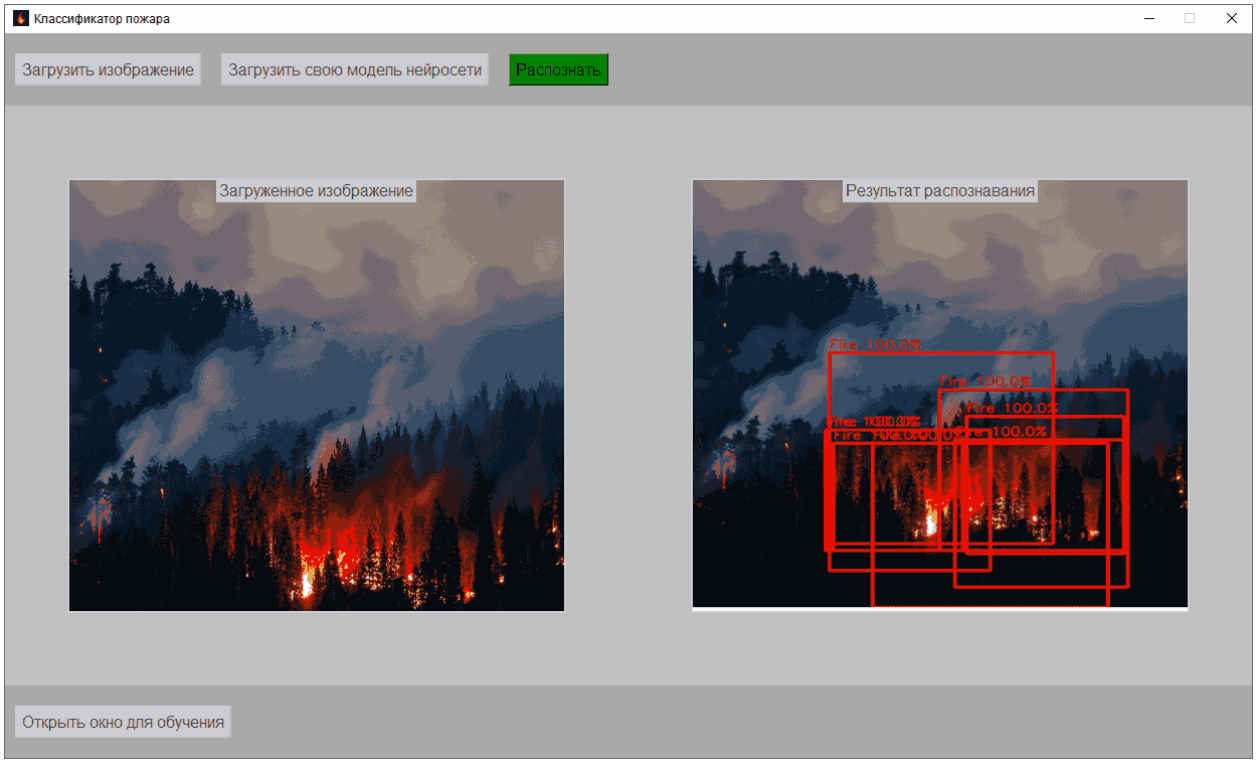


Рисунок 4.9 – Интерфейс с распознанным файлом fire2.png

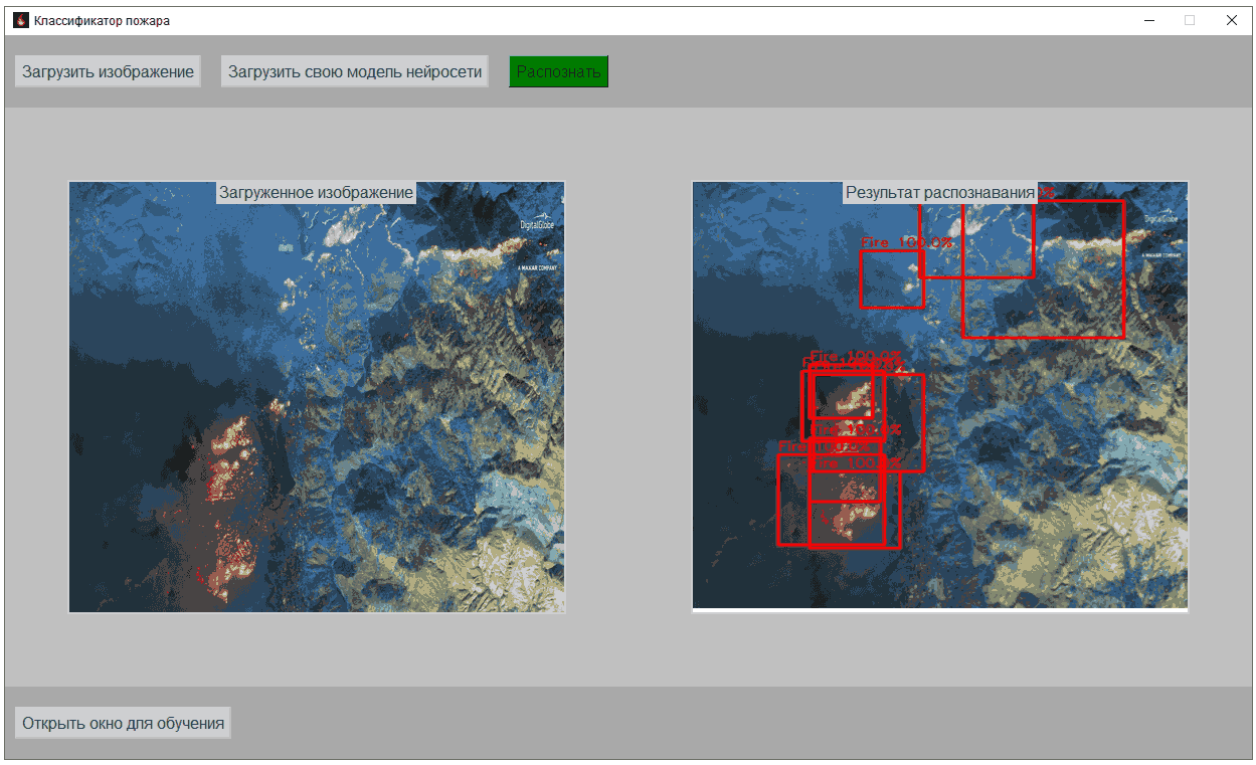


Рисунок 4.10 – Интерфейс с распознанным файлом fire3.png

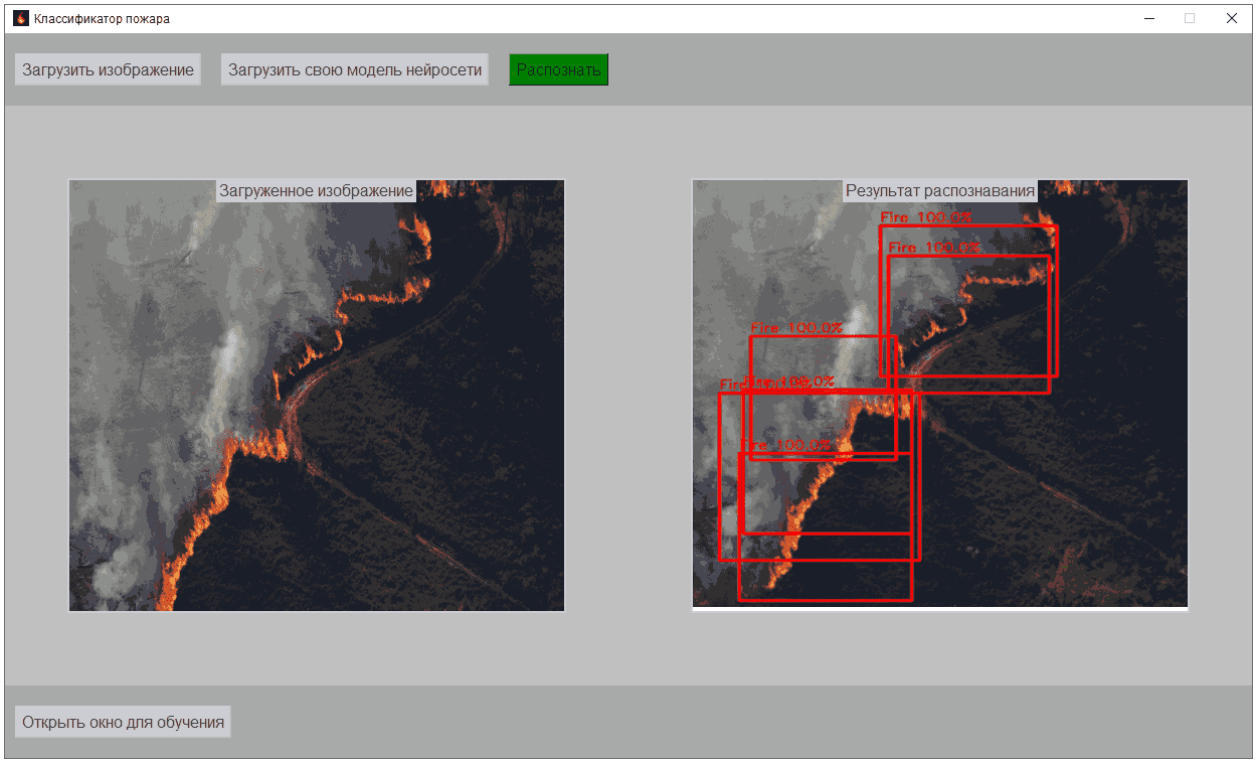


Рисунок 4.11 – Интерфейс с распознанным файлом fire4.png

Рисунок 4.12 – Интерфейс с распознанным файлом fire5.png

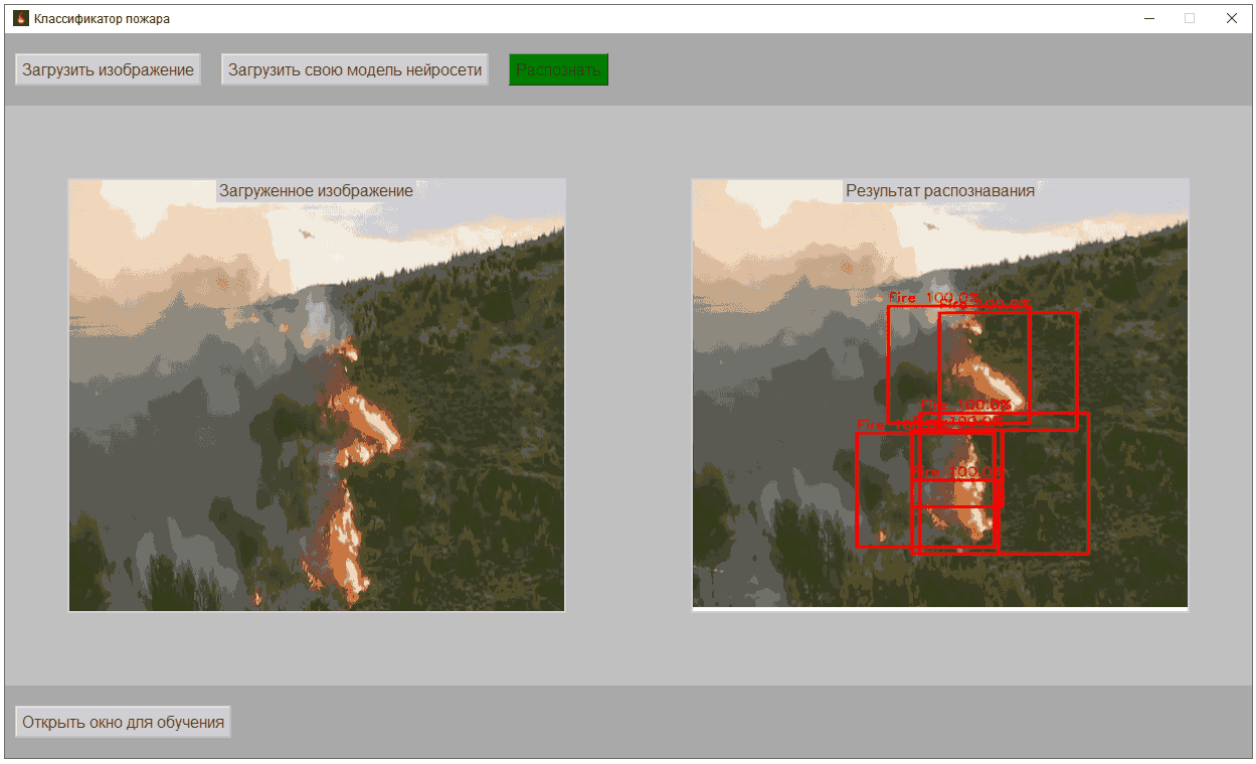


Рисунок 4.13 – Интерфейс с распознанным файлом fire6.png

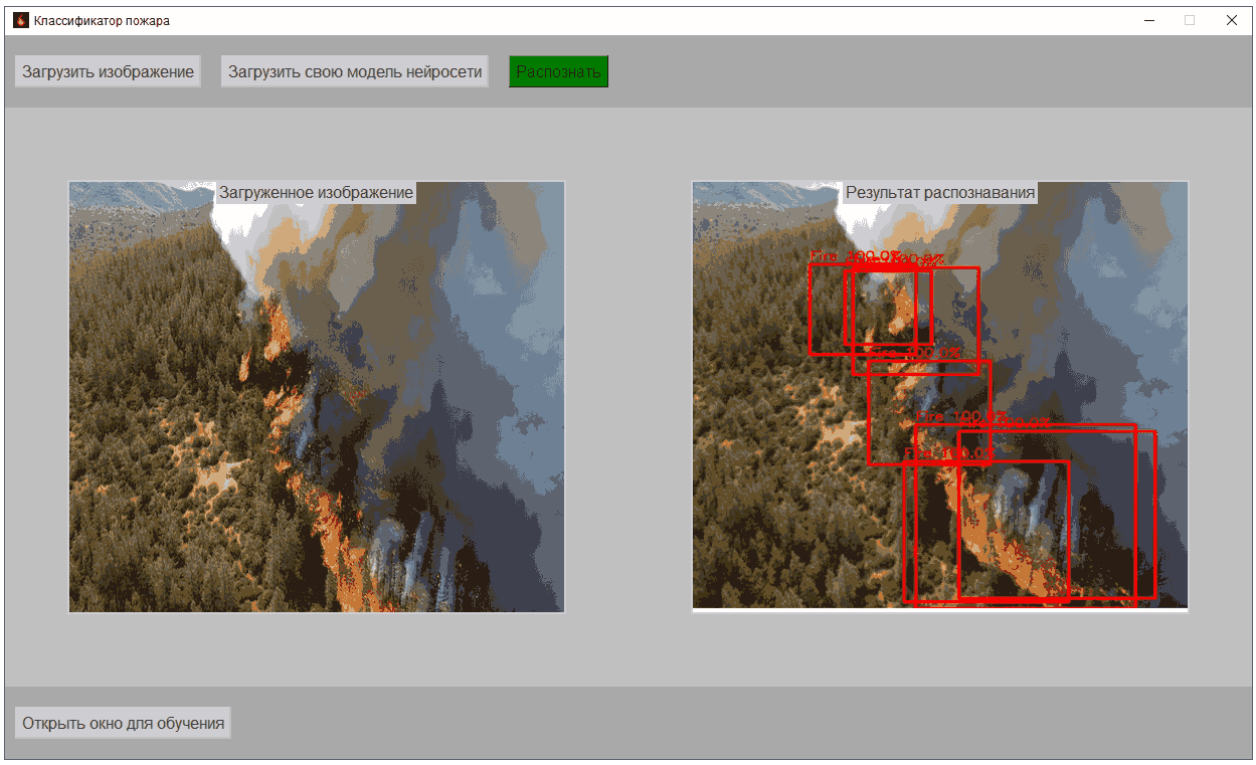


Рисунок 4.14 – Интерфейс с распознанным файлом fire7.png

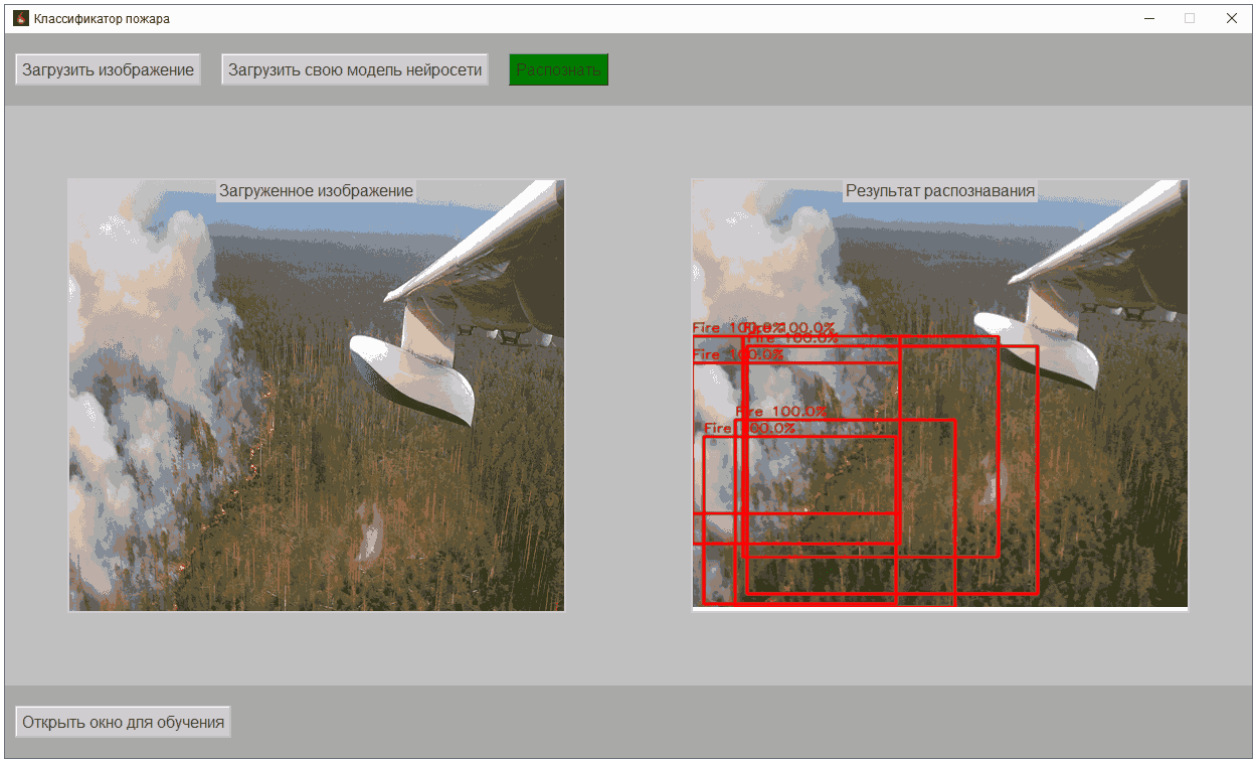


Рисунок 4.15 – Интерфейс с распознанным файлом fire8.png

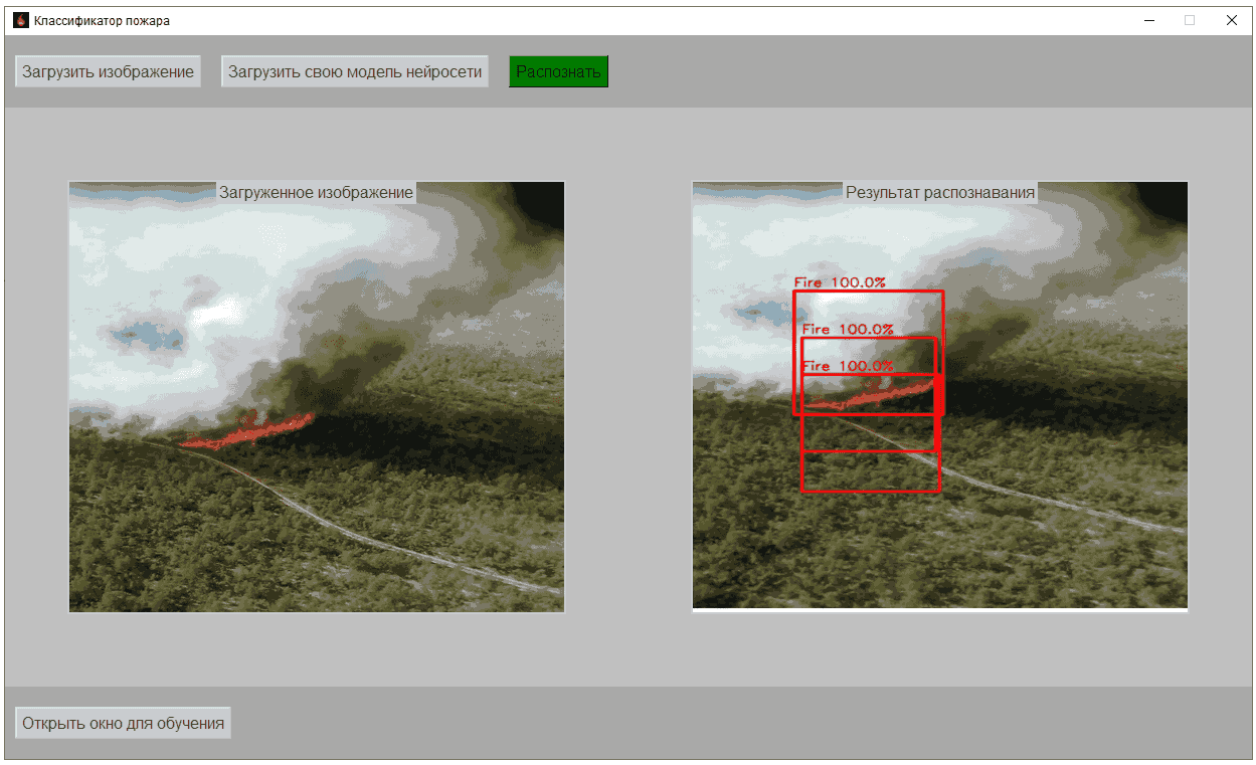


Рисунок 4.16 – Интерфейс с распознанным файлом fire9.png

На рисунке 4.17 была нажата кнопка “Информация о картинке”, после чего открылось окно с картинкой и вывелась информация по ней.

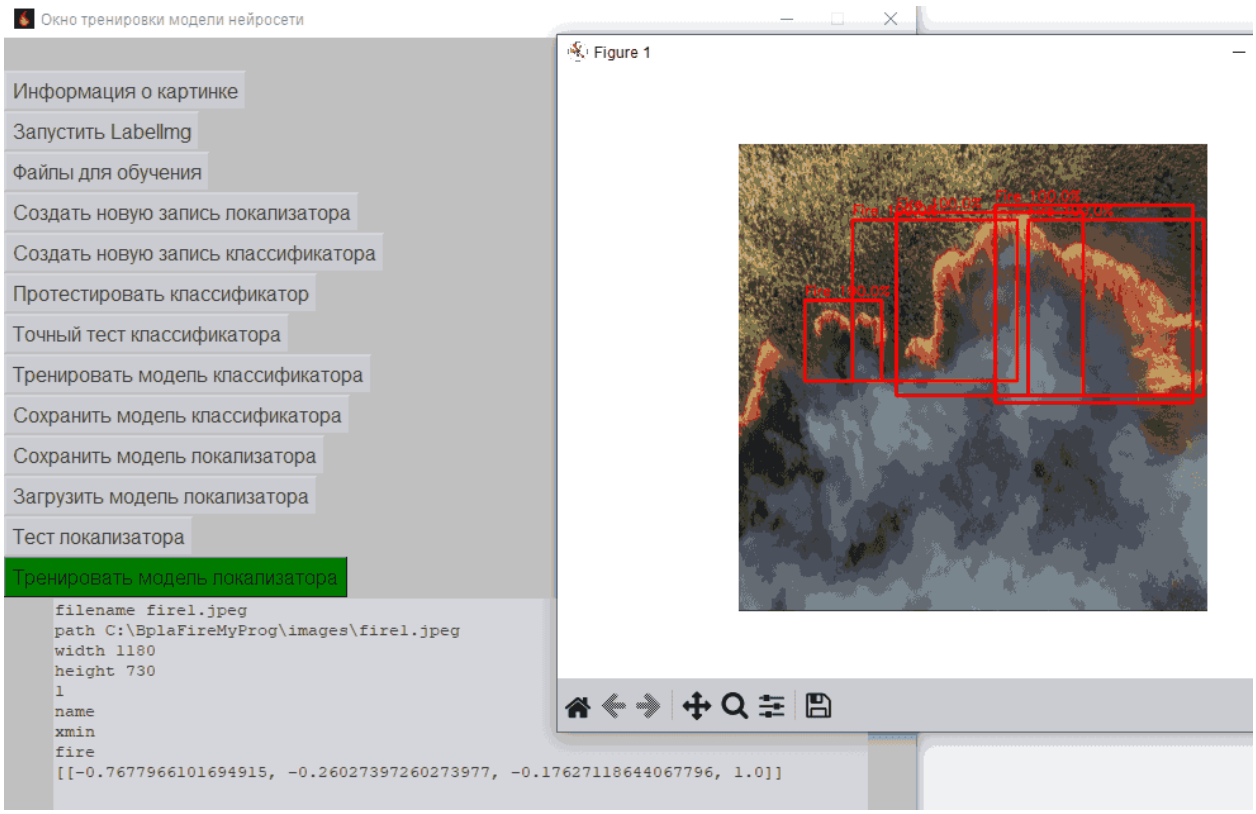


Рисунок 4.17 – Интерфейс дополнительного окна с информацией о картинке и сама картинка

На рисунке 4.18 была нажата кнопка “Файлы для обучения”, после чего в поле вывода отобразились все файлы, доступные для обучения.

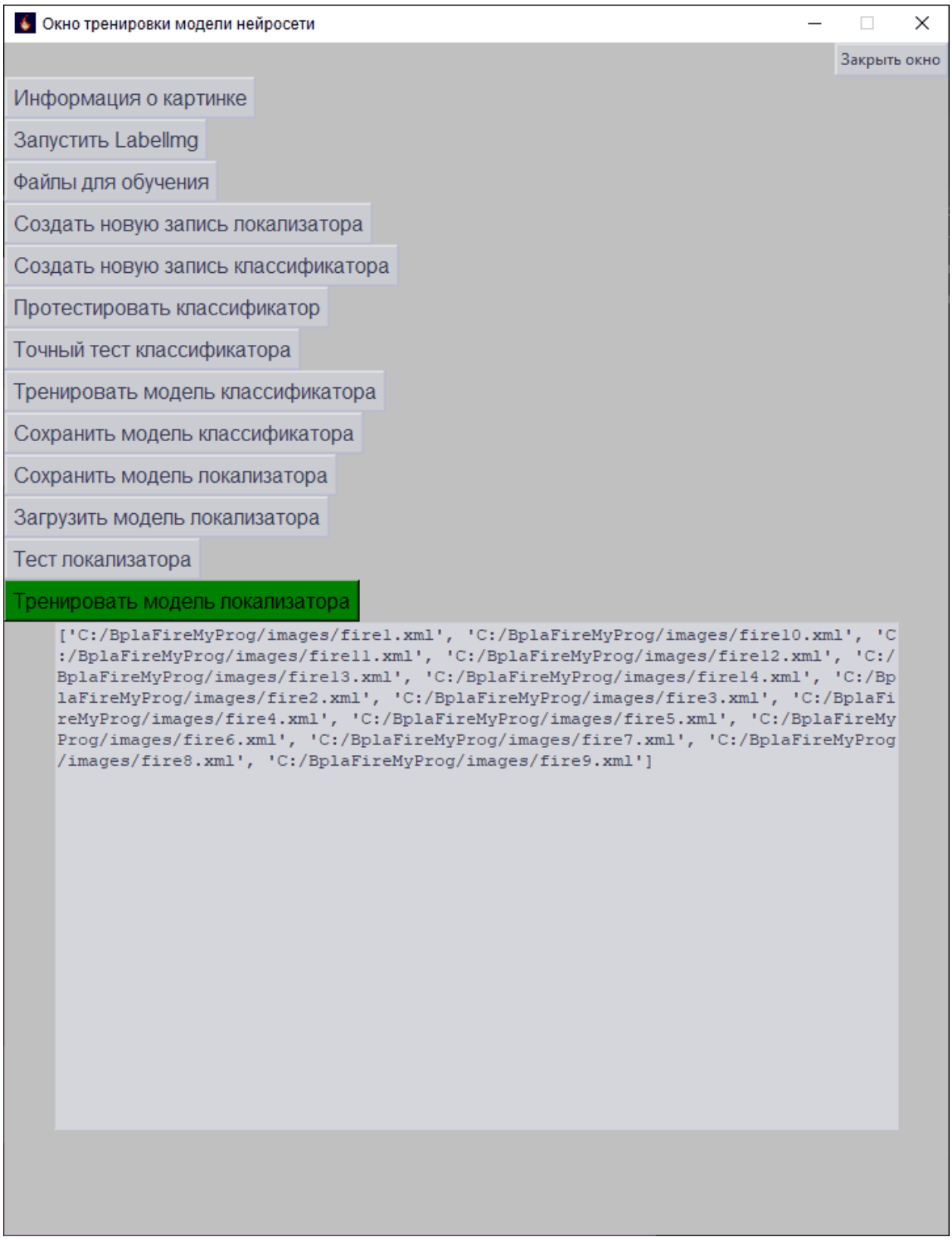


Рисунок 4.18 – Интерфейс дополнительного окна с информацией файлах для обучения

На рисунке 4.19 была нажата кнопка “Протестировать классификатор”, после чего открылось новое окно, где показаны обрезанные фрагменты изоб-

ражений, которые сеть классифицировала (0 - нет возгорания, 1 - есть возгорание).

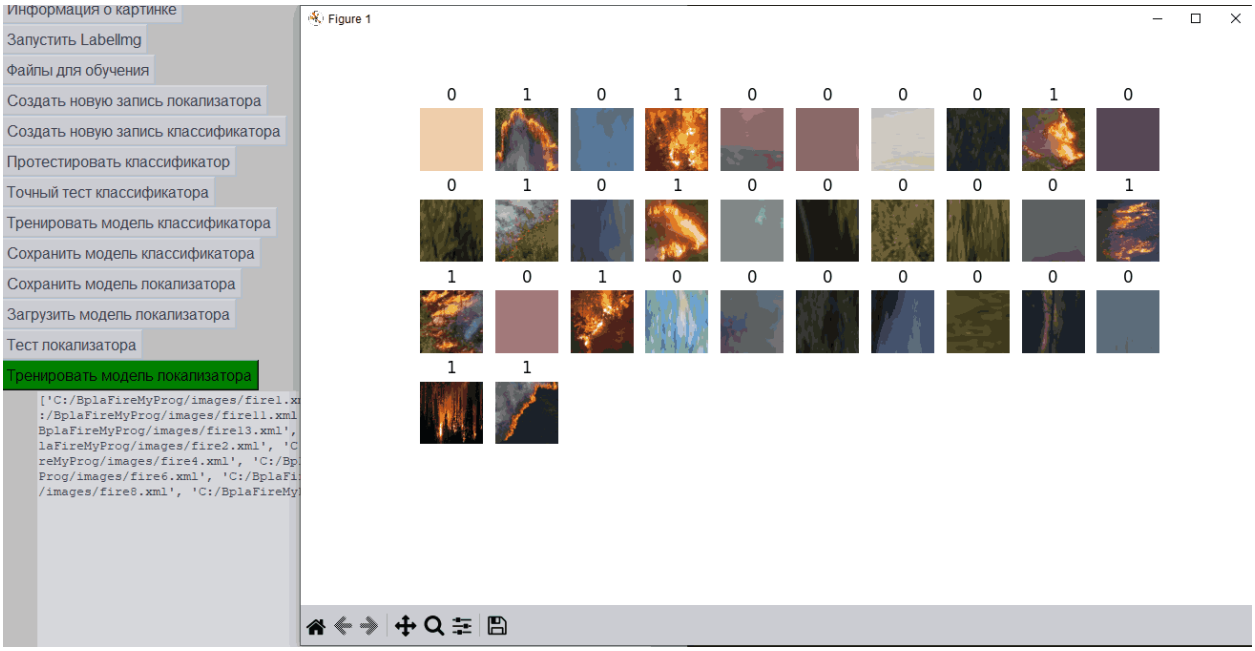


Рисунок 4.19 – Интерфейс дополнительного окна и окно с классификацией фрагментов изображений

На рисунке 4.20 была нажата кнопка “Точный тест классификатора”, после чего открылось новое окно, где показаны обрезанные фрагменты изображений, которые сеть классифицировала и придала степень принадежности с уровнем ошибки.

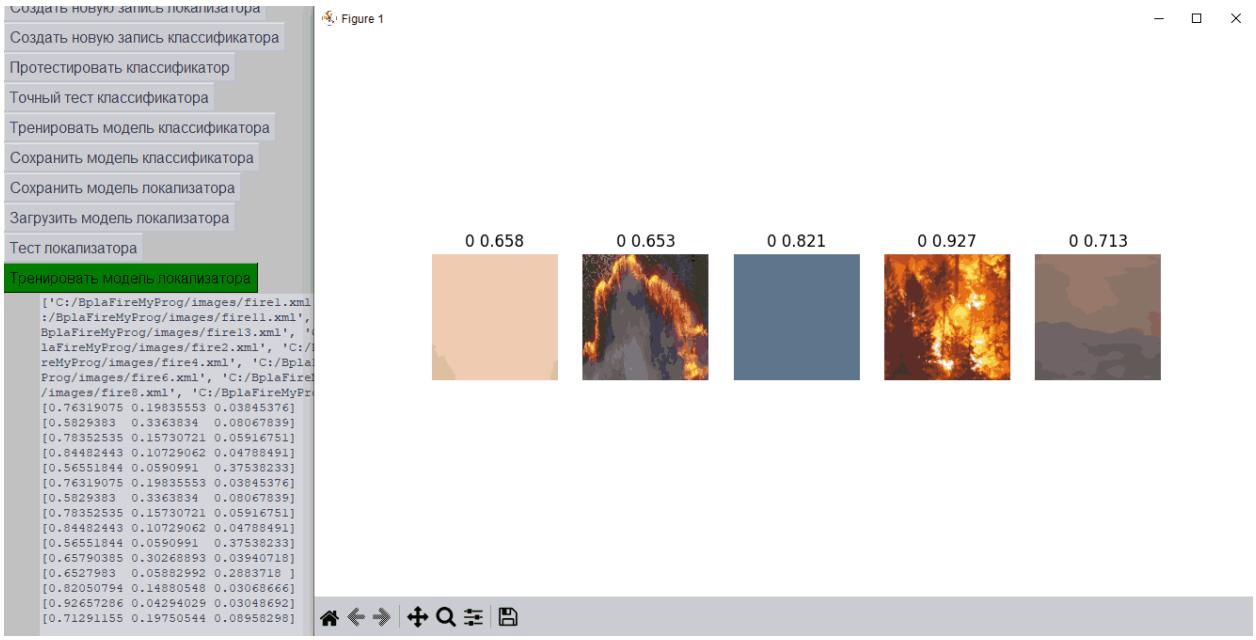


Рисунок 4.20 – Интерфейс дополнительного окна и окно с классификацией фрагментов изображений и уровнем ошибки

На рисунке 4.21 была нажата кнопка “Тренировать модель классификатора”, после чего открылось новое окно с диаграммой, где наглядно видно, как функция стремится к 0, тем самым показывая точность распознования классификатора.

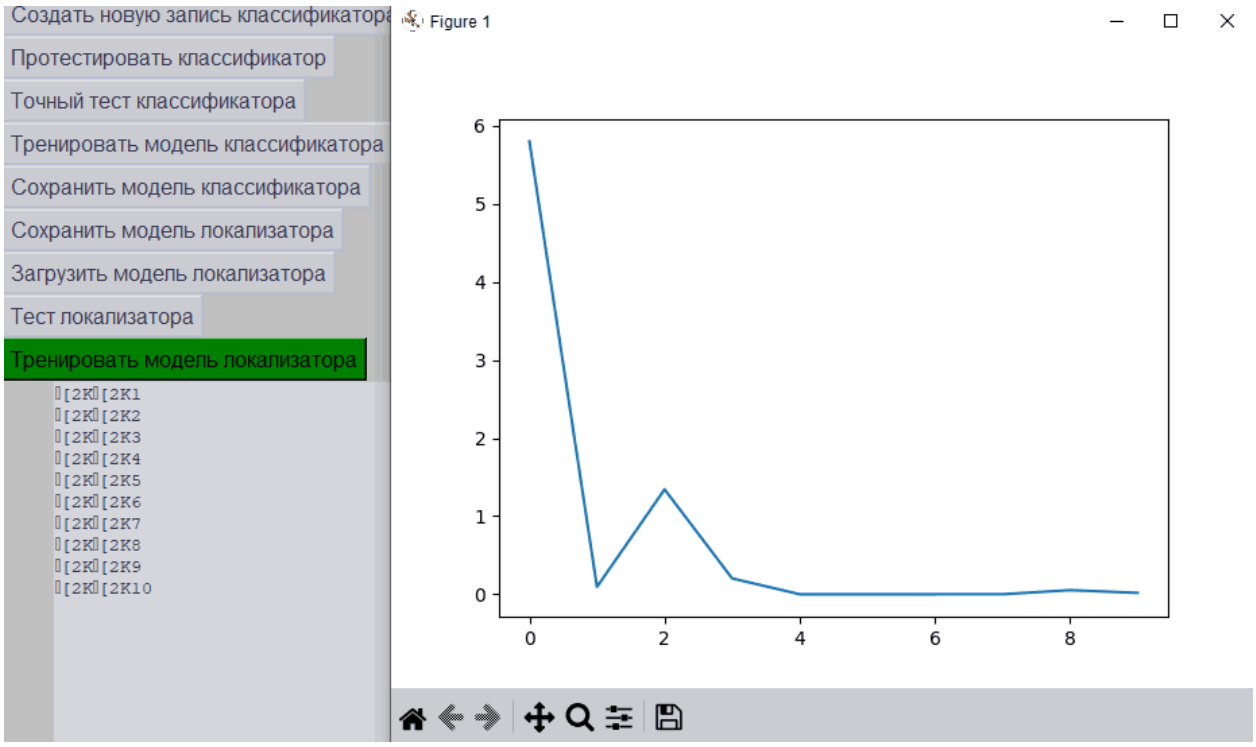


Рисунок 4.21 – Интерфейс дополнительного окна и окно с диаграммой уровня ошибки нейросети классификатора

На рисунке 4.22 была нажата кнопка “Тест локализатора”, но не была загружена модель НС локализатора, после чего изображения были распознанны некорректно.

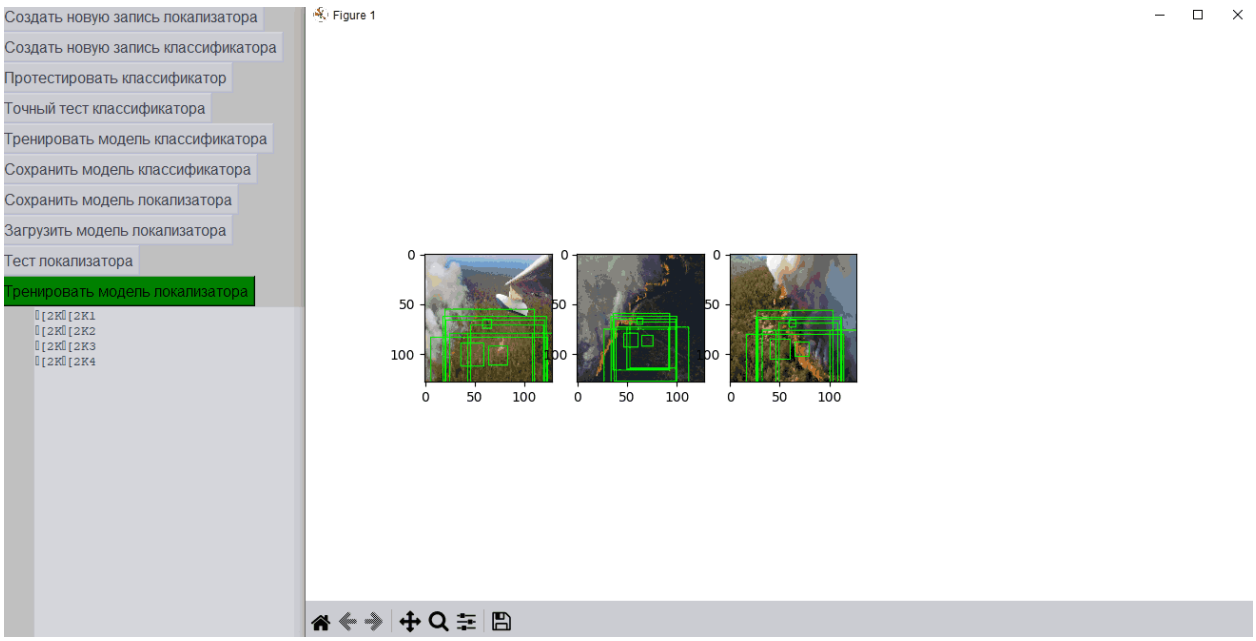


Рисунок 4.22 – Интерфейс дополнительного окна и окно с некорректно распознанными изображениями

На рисунке 4.23 была нажата кнопка “Тест локализатора” и была загружена модель НС локализатора, после чего изображения были распознанны корректно.

Рисунок 4.23 – Интерфейс дополнительного окна и окно с корректно распознанными изображениями

На рисунках 4.24 - 4.25 была нажата кнопка “Тренировать модель локализатора” и была загружена модель НС локализатора, после чего, спустя несколько эпох, изображения были распознанны. На диаграмме можно наглядно увидеть, что значения близки к 0 и даже выходят за абстрактные значения в минус.

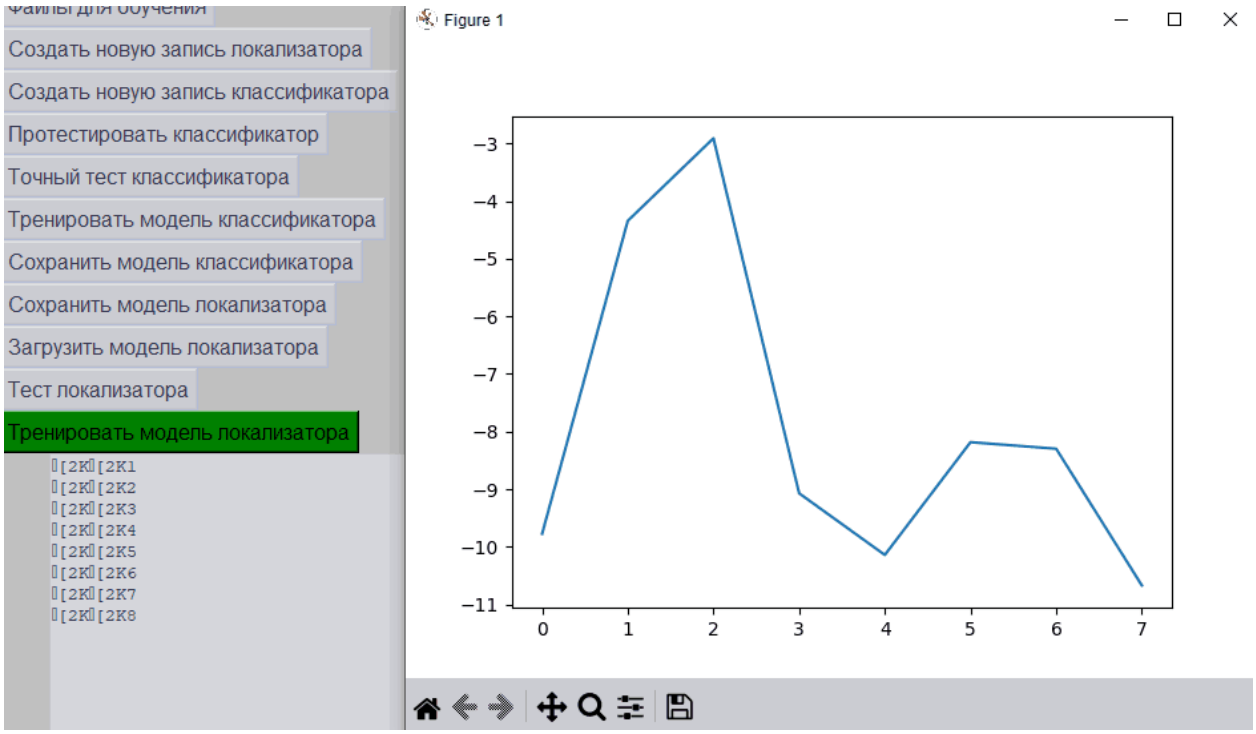


Рисунок 4.24 – Интерфейс дополнительного окна и окно с диаграммой ошибки локализатора

Рисунок 4.25 – Интерфейс дополнительного окна и окно с распознанными изображениями после нескольких эпох

На рисунках 4.26 - 4.27 была нажата кнопка “Запустить LabelImg”, после чего появилось окно с предупреждением о необходимости дополнительных пакетов. После подтверждения открылось окно со сторонним приложе-

нием, которое поможет выделить объекты на изображениях для дальнейшего сохранения их данных в формате xml.

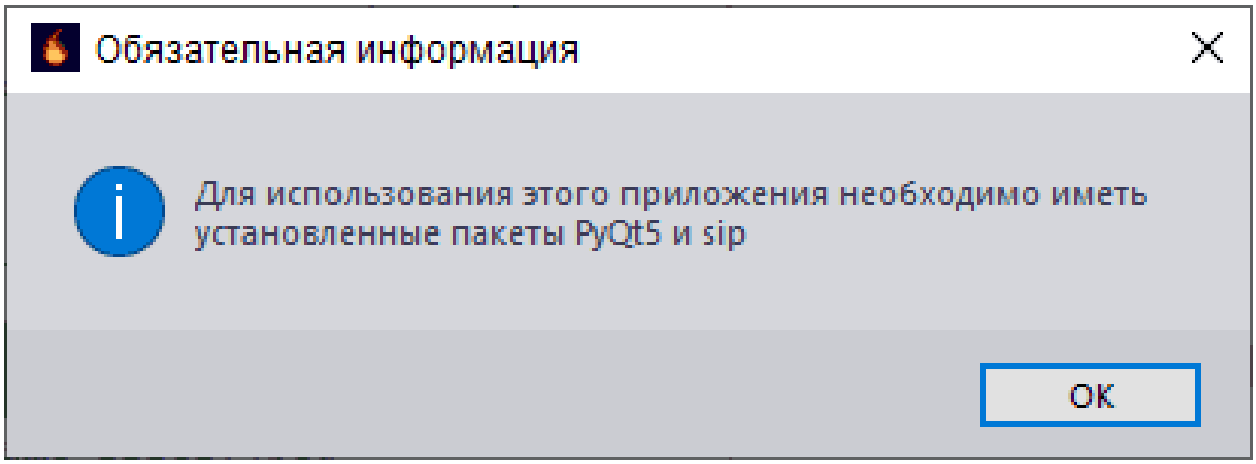


Рисунок 4.26 – Интерфейс дополнительного окна и окно с распознанными изображениями после нескольких эпох

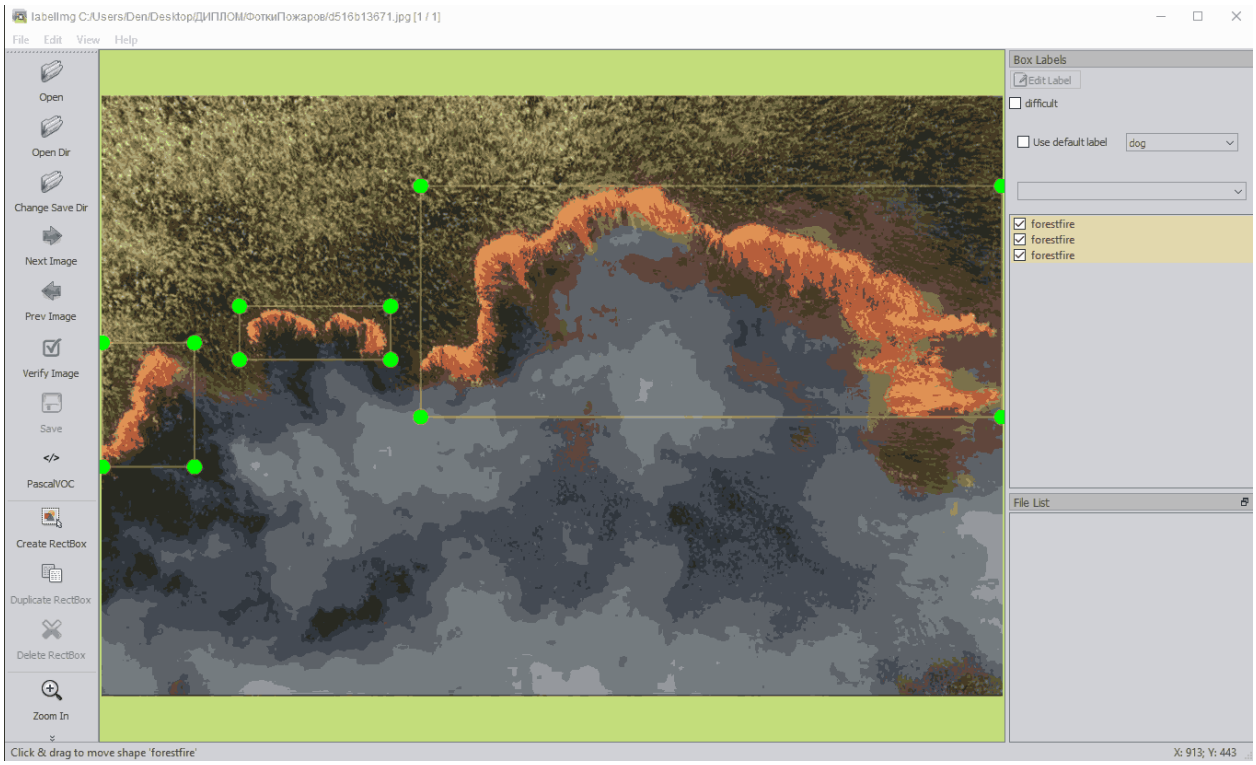


Рисунок 4.27 – Интерфейс дополнительного окна и окно с распознанными изображениями после нескольких эпох

ЗАКЛЮЧЕНИЕ

Преимущества разработки интеллектуальных систем, таких как система распознавания и классификации возгораний с БПЛА, заключаются в повышении точности и скорости обработки данных. Основным ограничением является сложность обработки больших объемов информации в реальном времени.

Компании, стремящиеся к инновациям, активно внедряют передовые технологии для повышения эффективности своей деятельности. Разработка мобильного приложения позволяет оперативно реагировать на возгорания, обнаруженные с помощью БПЛА, и предоставлять данные для принятия решений.

Основные результаты работы:

1. Проведен анализ предметной области. Выявлены ключевые требования к системе распознавания и классификации возгораний.

2. Разработана концептуальная модель приложения. Создана модель данных для обработки и анализа изображений с БПЛА.

3. Осуществлено проектирование архитектуры приложения. Разработан пользовательский интерфейс для взаимодействия с данными возгораний.

4. Реализовано и протестировано приложение. Проведено модульное и интеграционное тестирование.

Все заявленные требования были удовлетворены, все цели, поставленные на начальном этапе, достигнуты.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Интеллектуальная система обработки изображений, получаемых с беспилотных летательных аппаратов / Томакова Р.А., Филист С. А., Нефедов Н. Г. [и др.]. – Текст : непосредственный // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. – 2022. Т. 12(4): – № 4. – С. 64-85.

2. Хайкин, С. Нейронные сети: полный курс : учебное пособие / С. Хайкин. – Москва : Вильямс, 2018. – 1104 с. – ISBN 978-5-8459-2101-0. – Текст : непосредственный.

3. Лутц, М. Изучаем Python : учебное пособие / М. Лутц. – 5-е издание – Санкт-Петербург : Питер, 2019. – 1584 с. – ISBN 978-5-4461-0705-9. – Текст : непосредственный.

4. Гудфеллоу, И., Бенджио, Ю., Курвилль, А. Глубокое обучение : учебное пособие / И. Гудфеллоу, Ю. Бенджио, А. Курвилль. – Москва : ДМК Пресс, 2017. – 652 с. – ISBN 978-5-97060-487-9. – Текст : непосредственный.

5. Мерфи, К. Машинное обучение: вероятностный подход : учебное пособие / К. Мерфи. – Москва : ДМК Пресс, 2018. – 704 с. – ISBN 978-5-97060-212-7. – Текст : непосредственный.

6. Рашка, С., Мирджалили, В. Python и машинное обучение : практическое пособие / С. Рашка, В. Мирджалили. – Москва : ДМК Пресс, 2018. – 418 с. – ISBN 978-5-97060-310-0. – Текст : непосредственный.

7. Жолковский, Е. К. TensorFlow для профессионалов : учебное пособие / Е. К. Жолковский. – Москва : ДМК Пресс, 2019. – 480 с. – ISBN 978-5-97060-746-7. – Текст : непосредственный.

8. Чоллет, Ф. Глубокое обучение на Python : учебное пособие : в 5 томах / Ф. Чоллет. – Москва : ДМК Пресс, 2018. – 304 с. – ISBN 978-5-97060-409-1. – Текст : непосредственный.

9. Клейн, Р. Нечеткие системы в Python : учебное пособие / Р. Клейн. – Москва : ДМК Пресс, 2020. – 320 с. – ISBN 978-5-97060-758-0. – Текст : непосредственный.

10. Бейдер, Д. Python Tricks: A Buffet of Awesome Python Features : учебное пособие / Д. Бейдер. – Москва : ДМК Пресс, 2021. – 300 с. – ISBN 978-5-97060-999-7. – Текст : непосредственный.

11. Герон, О. Практическое машинное обучение с Scikit-Learn и TensorFlow : учебное пособие / О. Герон. – Москва : ДМК Пресс, 2019. – 572 с. – ISBN 978-5-97060-524-1. – Текст : непосредственный.

12. Нильсен, М. Нейронные сети и глубокое обучение : учебное пособие / М. Нильсен. – Москва : ДМК Пресс, 2021. – 250 с. – ISBN 978-5-97060-777-1. – Текст : непосредственный.

13. Буч, Г. Введение в UML от создателей языка : учебное пособие / Г. Буч, И. Якобсон, Д. Рамбо. – Москва : ДМК Пресс, 2015. – 498 с. – ISBN 978-5-457-43379-3. – Текст : непосредственный.

14. Джеймс, Р. UML 2.0. Объектно-ориентированное моделирование и разработка : практическое пособие / Р. Джеймс, Б. Майкл. – 2-е изд. – Санкт-Петербург : Питер, 2021. – 542 с. – ISBN 978-5-4461-9428-5. – Текст : непосредственный.

15. Зайцев, М. Г. Объектно-ориентированный анализ и программирование : учебное пособие / М. Г. Зайцев. – Новосибирск : изд-во НГТУ, 2017. – 84 с. – ISBN 978-5-04-112962-0. – Текст : непосредственный.

16. Мандел, Т. Разработка пользовательского интерфейса : учебное пособие / Т. Мандел. – ДМК Пресс, 2019. – 420 с. – ISBN 978-5-04-195060-6. – Текст : непосредственный.

17. Метод и алгоритм автономного планирования траектории полета беспилотного летательного аппарата при мониторинге пожарной обстановки в целях раннего обнаружения источника возгорания / Томакова Р.А., Филист С. А., Брежнева А. Н. [и др.] – Текст : непосредственный // Известия Юго-Западного государственного университета. Серия: Управление, вычис-

лительная техника, информатика. Медицинское приборостроение. 2023. Т. 13(1): № 1. С. 93-111.

18. Информационная система мониторинга на основе интеллектуальной классификации изображений видеопотоков / Томакова Р.А. ,Брежнев А.В., Брежнева А.Н. - Текст : непосредственный // Информационное общество. .2023. №5. С. 134-143.

19. Томакова Р.А. Методы и алгоритмы цифровой обработки изображений : учебное пособие / Р. А. Томакова, Е. А. Петрик ; Юго-Зап. гос. ун-т. - Курск : Университетская книга, 2020. - 310 с. - Библиогр.: с. 297-309. - ISBN 978-5-907270-19-0. - Текст : непосредственный.

20. Томакова Р.А. Основы теории нейрокомпьютерных систем : учебное пособие / Р. А. Томакова ; Юго-Зап. гос. ун-т. - Курск : [б. и.], 2021. - 135 с. - ISBN 978-5-907555-36-5 : Б. ц. - Текст : непосредственный.

21. Томакова Р.А. Методологические основы научных исследований : учебное пособие / Р. А. Томакова, В. И. Томаков ; Юго-Зап. гос. ун-т. - Курск : ЮЗГУ, 2017. - 204 с. - Библиогр.: с. 199-203. - ISBN 978-5-7681-1210-3. - Текст : непосредственный.

22. Чаплыгин А.А. Программирование на языке Python : учебное пособие / Юго-Зап. гос. ун-т ; сост. А. А. Чаплыгин. - Электрон. текстовые дан. (229 КБ). - Курск : ЮЗГУ, 2021. - 15 с. - Загл. с титул. экрана. - Б. ц. - Текст : электронный.

23. Северенс, Ч. Введение в программирование на Python : учебник / Ч. Северенс. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 231 с. - URL: https://biblioclub.ru/index.php?page=book&id=429184 (дата обращения: 24.08.2023) . - Б. ц. - Текст : электронный.

24. Хахаев, И. А. Практикум по алгоритмизации и программированию на Python: курс : учебное пособие / И. А. Хахаев. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 179 с. - URL: https://biblioclub.ru/index.php?page=book&id=429256 (дата обращения: 24.08.2023) . - Библиогр. в кн. - Б. ц. - Текст : электронный.

25. Программные системы статистического анализа: обнаружение закономерностей в данных с использованием системы R и языка Python : учебное пособие / В. М. Волкова, М. А. Семенова, Е. С. Четвертакова, С. С. Вожов. - Новосибирск : Новосибирский государственный технический университет, 2017. - 74 с. : ил., табл. - URL: https://biblioclub.ru/index.php?page=book&id=576496 (дата обращения: 02.03.2022) . - Режим доступа: по подписке. - Библиогр.: с. 48. - ISBN 978-5-7782-3183-2 : Б. ц. - Текст : электронный.

26. Шелудько, В. М. Язык программирования высокого уровня Python: функции, структуры данных, дополнительные модули : учебное пособие / В. М. Шелудько ; Министерство науки и высшего образования РФ ; Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет» ; Институт компьютерных технологий и информационной безопасности. - Ростов-на-Дону|Таганрог : Издательство Южного федерального университета, 2017. - 108 с. : ил. - URL: http://biblioclub.ru/index.php?page=book&id=500060 (дата обращения: 24.08.2023) . - Режим доступа: по подписке. - Библиогр. в кн. - ISBN 978-5-9275-2648-2 : Б. ц. - Текст : электронный.

27. Емельянов С.Г. Интеллектуальные системы на основе нечеткой логики и мягких арифметических операций : учебник / С. Г. Емельянов, В. С. Титов, М. В. Бобырь. - Москва : Аргамак-Медиа, 2014. - 338, [7] с. : табл., граф. - Библиогр.: с. 325-336. - 300 экз. - ISBN 978-5-00024-035-9 – Текст : непосредственный.

28. Демидова Л.А. Принятие решений в условиях неопределенности : монография / Л. А. Демидова. - 2-е изд., перераб. - Москва : Горячая линия - Телеком, 2016. - 289 с. - ISBN 978-5-9912-0513-9 : 368.68 р. - Текст : непосредственный.

29. Яхъяева, Г. Э. Основы теории нейронных сетей : учебное пособие / Г. Э. Яхъяева. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 200 с. - (Основы информационных технологий). - URL:

http://biblioclub.ru/index.php?page=book&id=429110. - ISBN 978-5-94774-818-5 : Б. ц. - Текст : электронный.

30. Лубенцова, Е. В. Системы управления с динамическим выбором структуры, нечеткой логикой и нейросетевыми моделями : монография / Е. В. Лубенцова. - Ставрополь : СКФУ, 2014. - 248 с. - URL: http://biblioclub.ru/index.php?page=book&id=457413 (дата обращения: 28.04.2022) . - Режим доступа: по подписке. - ISBN 978-5-88648-902-6 : Б. ц. - Текст : электронный.

31. Гелиг, А. Х. Введение в математическую теорию обучаемых распознающих систем и нейтронных сетей : учебное пособие / А. Х. Гелиг, А. С. Матвеев. - Санкт-Петербург : Издательство Санкт-Петербургского Государственного Университета, 2014. - 224 с. - (Прикладная математика и информатика). - URL: http://biblioclub.ru/index.php?page=book&id=457945. - ISBN 978-5-288-05551-5 : Б. ц. - Текст : электронный.

32. Келлехер, Д. Наука о данных: базовый курс : учебное пособие / Д. Келлехер, Б. Тирни ; науч. ред. З. Мамедьяров ; пер. с англ. М. Белоголовский. - Москва : Альпина Паблишер, 2020. - 224 с. - URL: http://biblioclub.ru/index.php?page=book&id=598235 (дата обращения: 09.09.2022) . - Режим доступа: по подписке. - ISBN 978-5-9614-3170-4 : Б. ц. - Текст : электронный.

33. Кассим К.Д.А. Моделирование систем искусственного интеллекта в среде Matlab и Fuzzytech : учебное пособие / К. Д. А. Кассим, С. А. Филист, О. В. Шаталова. - Курск : Деловая полиграфия, 2016. - 186 с. - Библиогр.: с. 185. - ISBN 978-5-9908582-8-2. - Текст : непосредственный.

34. Математические методы и инновационные научно-технические разработки : сборник научных трудов / Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования ”Юго-Западный государственный университет”; редкол.: В. В. Серебровский (отв. ред.) [и др.]. - Курск : ЮЗГУ, 2014. - 282 с. ; 20. - Библиогр. в конце ст. - 100 экз. - ISBN 978-5-7681-0930-1 : 380.00 р. - Текст : непосредственный.

35. Интеллектуальное планирование траекторий подвижных объектов в средах с препятствиями / Д. А. Белоглазов [и др.] ; под ред. В. Х. Пшихопова. - Москва : Физматлит, 2014. - 295 с. : ил. - Авт. указ. на обороте тит. л. - Библиогр.: с. 276-295 (314 назв.). - ISBN 978-5-9221-1595-7. - Текст : непосредственный.

36. Волков Д.А. Модель, метод и нейросетевое оптико-электронное вычислительное устройство распознавания изображений : специальность 05.13.05 ”Элементы и устройства вычислительной техники и систем управления”: автореферат диссертации на соискание ученой степени кандидата технических наук / Волков Денис Андреевич ; Юго-Западный государственный университет. - Курск, 2020. - 17 с. - Место защиты: ФГБОУ ВО ”Юго-Западный государственный университет”(Курск). - Текст : непосредственный.

37. Программирование, тестирование, проектирование, нейросети, технологии аппаратно-программных средств (практические задания и способы их решения) : учебник / С. В. Веретехина, К. С. Кармицкий, Д. Д. Лукашин [и др.]. - Москва : Директ-Медиа, 2022. - 144 с. - URL: https://biblioclub.ru/index.php?page=book&id=694782 (дата обращения: 10.01.2023) . - Режим доступа: по подписке. - Библиогр. в кн. - ISBN 978-5-4499-3321-8 : Б. ц. - Текст : электронный.

38. Интеллектуальные системы и технологии : учебное пособие / С. П. Ющенко [и др.] ; Юго-Зап. гос. ун-т. - Курск : Университетская книга, 2018. - 226 с. : ил. - Библиогр.: с. 238-240 (32 назв.). - ISBN 978-5-907138-22-3 : 560.00 р. - Текст : непосредственный.

39. Системная инженерия. Принципы и практика = Systems engineering principles and practice : учебник / А. Косяков [и др.] ; пер. с англ. под ред. В. К. Батоврин. - 2-е изд. - Москва : ДМК Пресс, 2014. - 624 с. : ил. - Указ.: с. 610-619. - 400 экз. - ISBN 978-5-97060-122-8 (в пер.). - Текст : непосредственный.

40. Сидоркина И.Г.. Системы искусственного интеллекта : учебное пособие / И. Г. Сидоркина. - Москва : КНОРУС, 2016. - 246 с. : рис. - Библиогр.: с. 244-245. - ISBN 978-5-406-04876-4 . - Текст : непосредственный.

41. Бабенко Л.К. Параллельные алгоритмы для решения задач защиты информации : монография / Л. К. Бабенко, Е. А. Ищукова, И. Д. Сидоров. - Москва : Горячая линия-Телеком, 2014. - 304 с. : ил. - Библиогр.: с. 222-224. - ISBN 978-5-9912-0426-2 : 340.00 р. - Текст : непосредственный.

42. Кузнецов, А. С. Теория вычислительных процессов : учебник / А. С. Кузнецов, Р. Ю. Царев, А. Н. Князьков. - Красноярск : Сибирский федеральный университет, 2015. - 184 с. - URL: http://biblioclub.ru/index.php?page=book&id=435696. - ISBN 978-5-7638-3193-1 : Б. ц. - Текст : электронный.

43. Исакова, А. И. Основы информационных технологий : учебное пособие / А. И. Исакова. - Томск : ТУСУР, 2016. - 206 с. : ил. - URL: http://biblioclub.ru/index.php?page=book&id=480808 (дата обращения: 18.02.2022) . - Режим доступа: по подписке. - Библиогр.: с. 197-198. - Б. ц. - Текст : электронный.

44. Гунько, А. В. Программирование (в среде Windows) : учебное пособие / А. В. Гунько ; Новосибирский государственный технический университет. - Новосибирск : Новосибирский государственный технический университет, 2019. - 155 с. - URL: https://biblioclub.ru/index.php?page=book&id=575417 (дата обращения: 03.05.2024) . - Режим доступа: по подписке. - Библиогр. в кн. - ISBN 978-5-7782-3890-9 : Б. ц. - Текст : электронный.

45. Малоразмерные беспилотные летательные аппараты: задачи обнаружения и пути их решения : монография / И. И. Олейник, А. А. Черноморец, В. Г. Андронов [и др.] ; под ред. В. Г. Андронова ; Юго-Зап. гос. ун-т. - Курск : ЮЗГУ, 2021. - 171 с. - Библиогр.: с. 149-170. - ISBN 978-5-7681-1518-0. - Текст : непосредственный.

46. Методологические основы обнаружения малоразмерных беспилотных летательных аппаратов на основе комплексной субполосной обработки

сверхкороткоимпульсных радиолокационных и оптических сигналов : монография / И. И. Олейник, А. А. Черноморец, Д. С. Коптев [и др.] ; под общ. ред. В. Г. Андронов ; Юго-Зап. гос. ун-т. - Курск : ЮЗГУ, 2021. - 204 с. - Библиогр.: с. 198-203. - ISBN 978-5-7681-1526-5 . - Текст : непосредственный.

47. Шевцов, Максим Викторович. Система мониторинга пожарной и медико-экологической безопасности с использованием анализа видеоданных с беспилотных летательных аппаратов : специальность 2.3.1 ”Системный анализ, управление и обработка информации (технические науки)”: автореферат диссертации на соискание ученой степени кандидата технических наук / Шевцов Максим Викторович ; Академия Государственной противопожарной службы МЧС России (Москва). - Электрон. текстовые дан. (476 КБ). - Москва, 2022. - 20 с. - Место защиты: Юго-Западный государственный университет (Курск). - Текст : непосредственный.

48. Аверченков, В. И. Основы математического моделирования технических систем : учебное пособие / В. И. Аверченков, В. П. Федоров, М. Л. Хейфец. - 4-е изд., стер. - Москва : Флинта, 2021. - 271 с. - URL: http://biblioclub.ru/index.php?page=book&id=93344 (дата обращения: 16.02.2024) . - Режим доступа: по подписке. - ISBN 978-5-9765-1278-8 : Б. ц. - Текст : электронный.

49. Зыков, С. В. Введение в теорию программирования. Объектно-ориентированный подход : курс лекций / С. В. Зыков. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 189 с. - (Основы информационных технологий). - URL: http://biblioclub.ru/index.php?page=book&id=429073 (дата обращения: 13.05.2024) . - Режим доступа: по подписке. - ISBN 5-9556-0009-4 : Б. ц. - Текст : электронный.

50. Сазонов С.Ю. Системный подход к моделированию процессов возникновения и развития пожаров : монография / С. Ю. Сазонов ; Юго-Зап. гос. ун-т. - Курск : Деловая полиграфия, 2016. - 218 с. - Библиогр.: с. 213-219. - ISBN 978-5-9907910-9-1. - Текст : непосредственный.

ПРИЛОЖЕНИЕ А

Представление графического материала

Графический материал, выполненный на отдельных листах, изображен на рисунках А.1–А.9.

ВКРБ 20060391.09.03.04.24.006

1

Сведения о ВКРБ

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА ПО ПРОГРАММЕ БАКАЛАВРИАТА

«Интеллектуальная система распознавания и классификации возгораний, полученных с БПЛА»

Руководитель ВКРБ
д.т.н, профессор
Томакова Римма Александровна

Автор ВКРБ
студент группы ПО-02б
Каракчиев Даниил Александрович

				ВКРБ 20060391.09.03.04.24.006	
	Фамилия И. О.	Подпись	Дата	Сведения о ВКРБ	Лит. / Масса / Масштаб
Автор работы	Каракчиев Д.А.				
Руководитель	Томакова Р.А.				Лист 1 / Листов 9
Нормоконтроль	Чаплыгин А.А.			Выпускная квалификационная работа бакалавра	ЮЗГУ ПО-02б

Рисунок А.1 – Сведения о ВКРБ

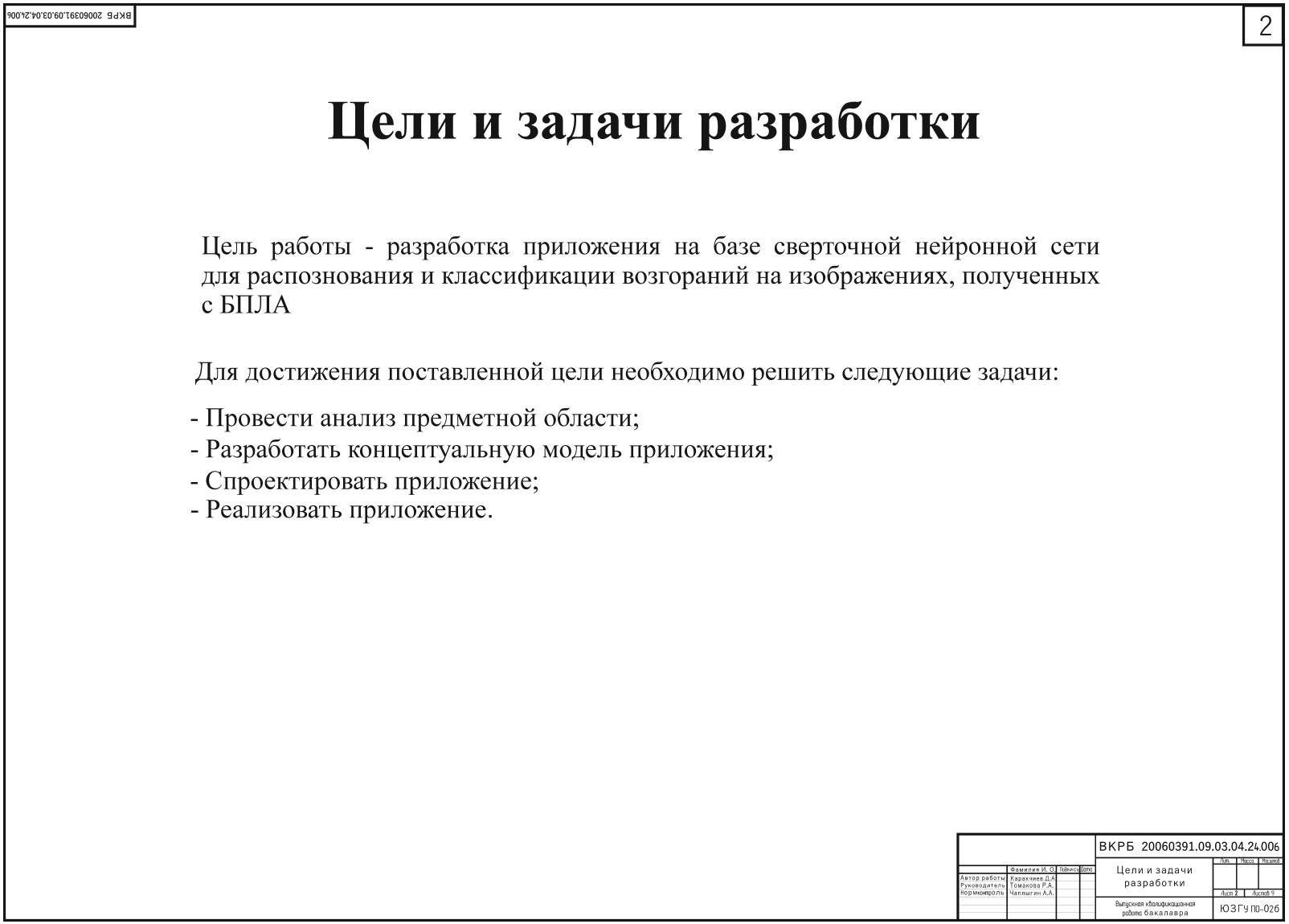


Рисунок А.2 – Цель и задачи разработки

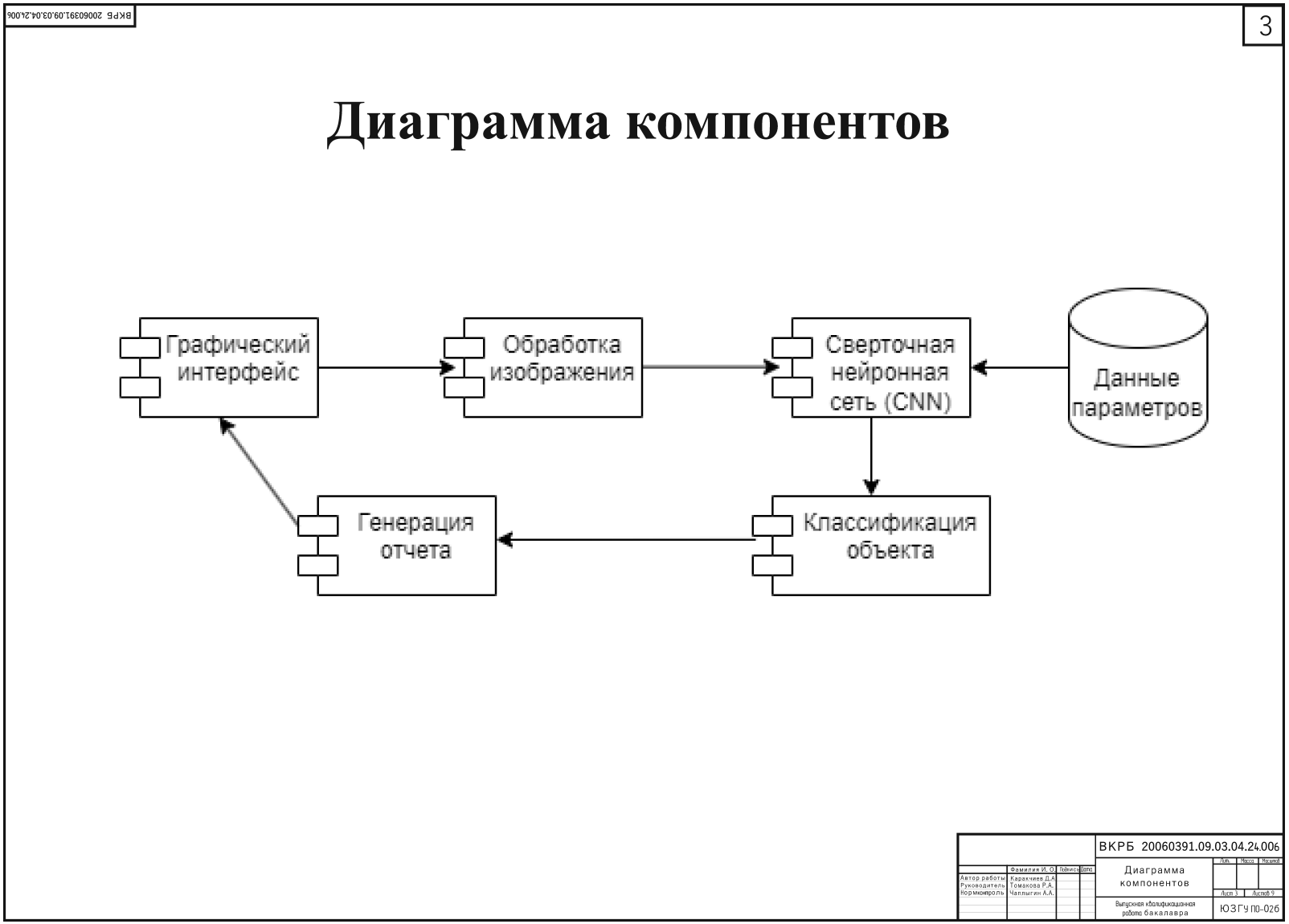
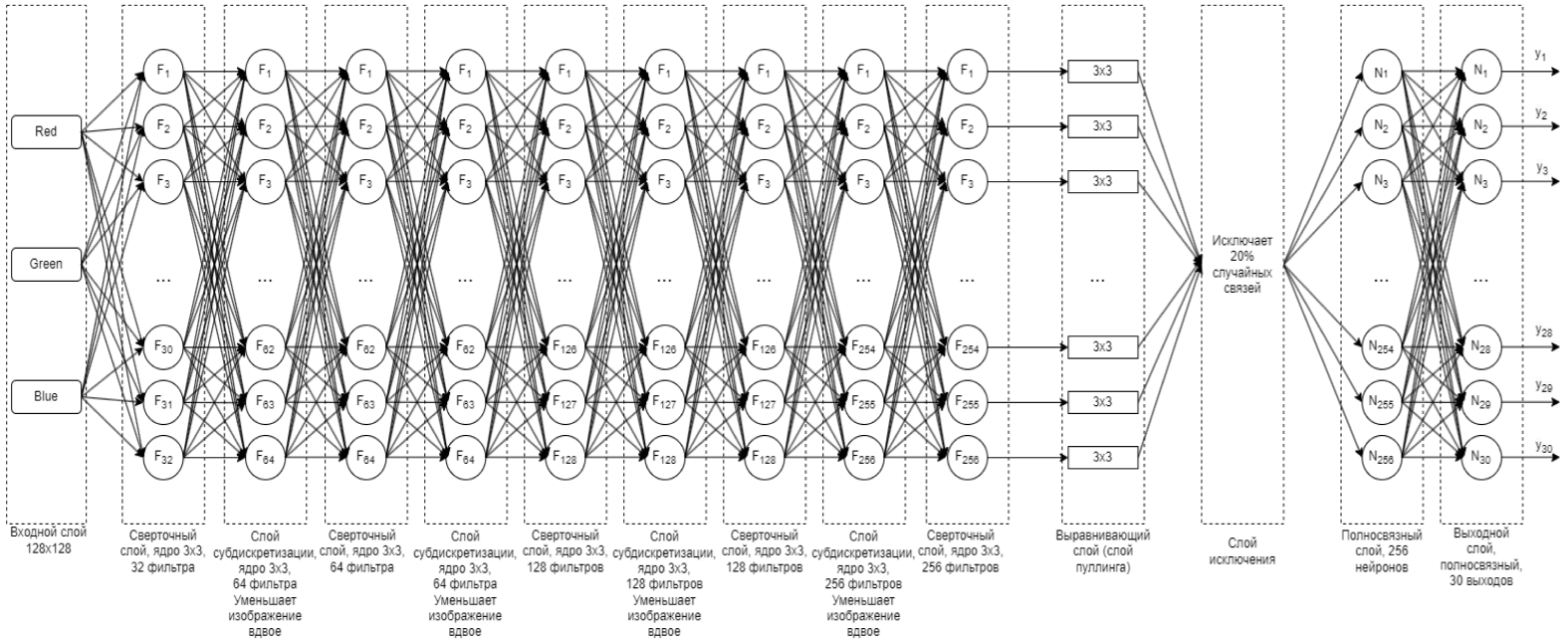


Рисунок А.3 – Диаграммы компонентов

ВКРБ 20060391.09.03.04.24.006

4

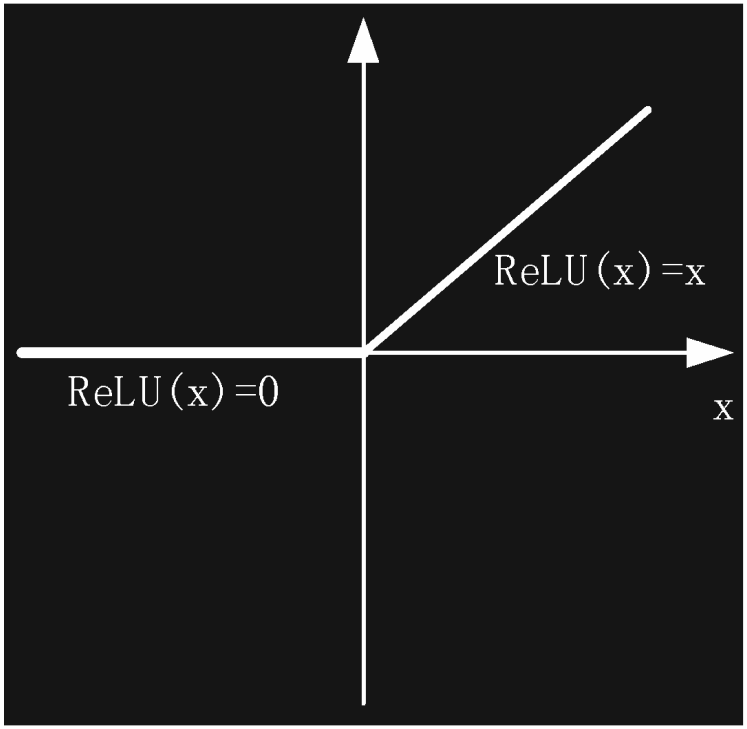
Архитектура нейронной сети



				ВКРБ 20060391.09.03.04.24.006	
	Фамилия И. О.	Подпись	Дата		
Автор работы	Карякчиев Д.А.			Архитектура нейронной сети	Лит. / Масса / Масштаб
Руководитель	Томакова Р.А.				
Нормоконтроль	Чаплыгин А.А.				Лист 4 / Листов 9
				Выпускная квалификационная работа бакалавра	ЮЗГУ ПО-026

Рисунок А.4 – Архитектура нейронной сети

Функция активации ReLU



ВКРБ 20060391.09.03.04.24.006

	Фамилия И. О.	Подпись	Дата
Автор работы	Каракчиев Д.А.		
Руководитель	Томакова Р.А.		
Нормоконтроль	Чаплыгин А.А.		

Функция активации ReLU

Лит.	Масса	Масштаб

Лист 5 Листов 9

Выпускная квалификационная работа бакалавра

ЮЗГУ ПО-026

Рисунок А.5 – Функция активации ReLU

Функция вычисления ошибки IoU Loss

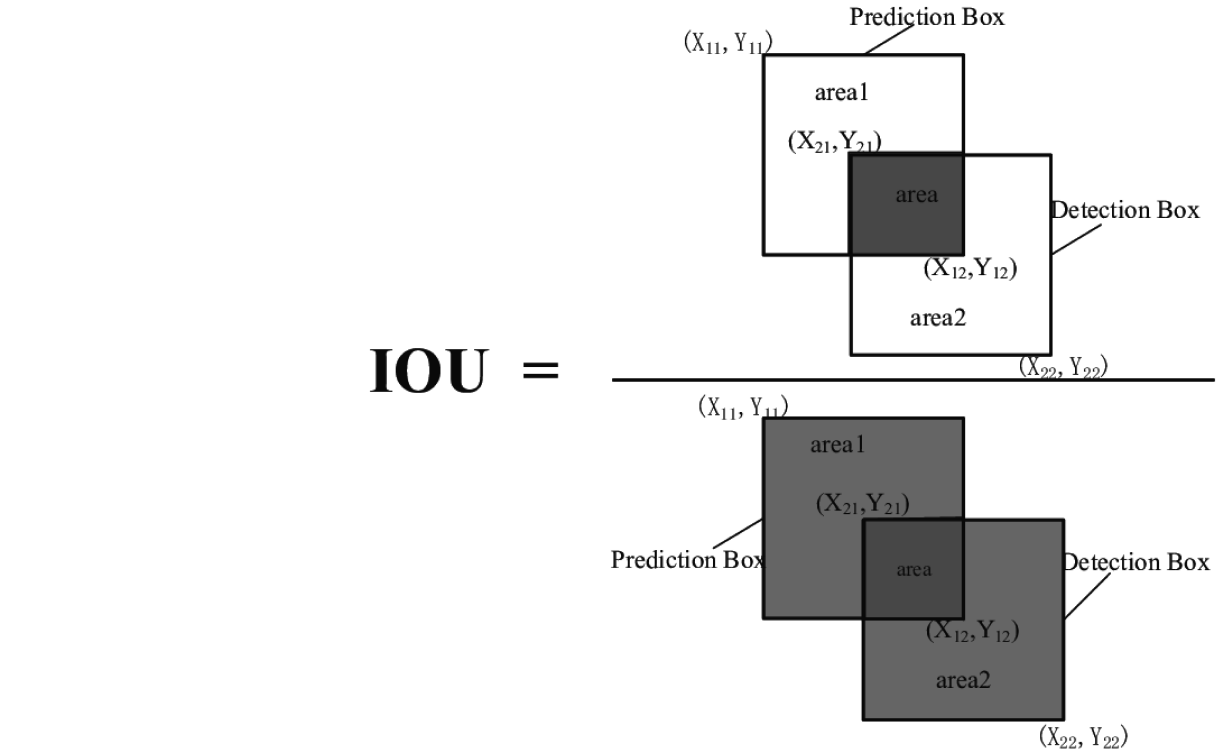


Рисунок А.6 – Функция вычисления ошибки IoU Loss

Интерфейс программы

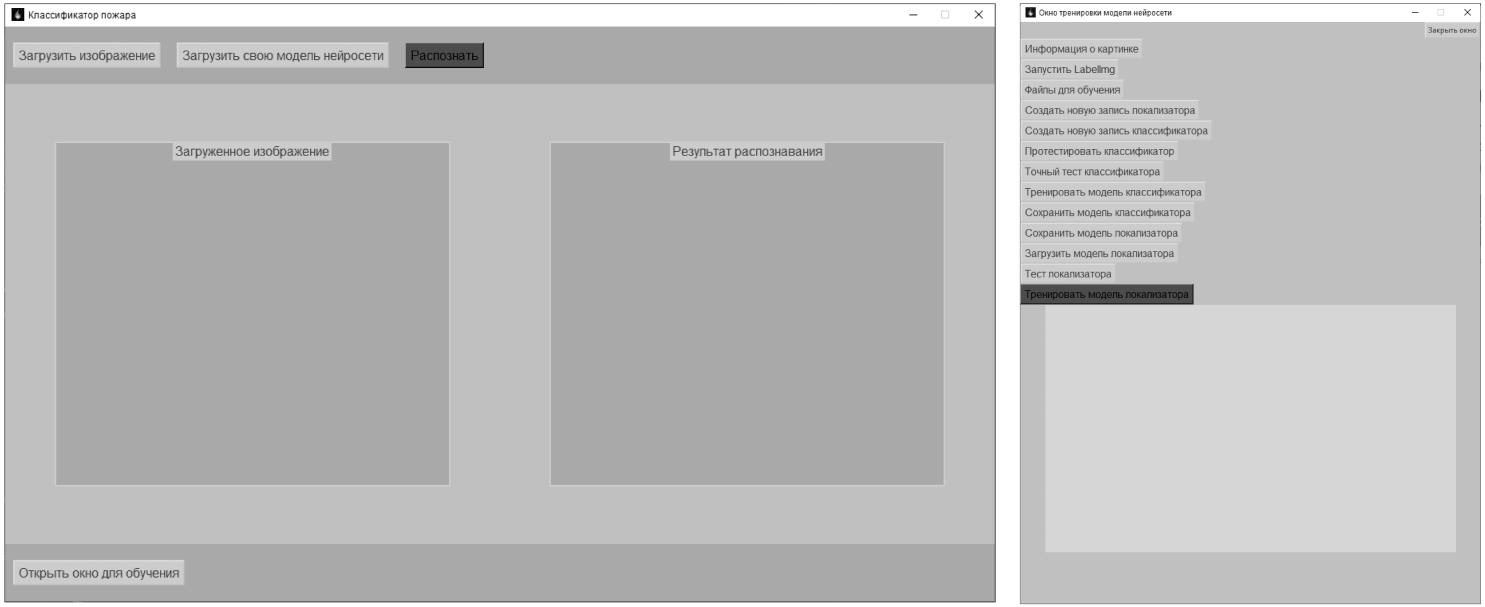


Рисунок А.7 – Интерфейс приложения

Демонстрация работы программы

Рисунок А.8 – Демонстрация работы программы

Заключение

Основные результаты работы:

Проведен анализ предметной области. Выявлена необходимость использовать язык Python и библиотеки TensorFlow, Keras и OpenCV;

Разработана концептуальная модель приложения. Разработана модель данных системы. Определены требования к системе;

Осуществлено проектирование приложения. Разработана архитектура нейронной сети. Разработан пользовательский интерфейс приложения. Выполнено обучение сверточной нейронной сети;

Реализовано и протестировано приложение. Проведено модульное и системное тестирование.

Рисунок А.9 – Заключение

ПРИЛОЖЕНИЕ Б

Фрагменты исходного кода программы

main.py

```
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import tensorflow as tf
import cv2
import numpy as np
import sys
import matplotlib.pyplot as plt
from tkinter import *
from tkinter import filedialog
# from tkinter.ttk import Combobox
from tkinter.messagebox import showwarning, showinfo
import tkinter as tk
from PIL import Image, ImageTk
import io
from dataprocessing import dataprocess
from creatingtfrecordclassifier import check_xml_list
from creatingtfrecordlocalizer import create_load_tfrec_for_localizer
from classifier import test_classifier, imshow_and_pred, trainclass,
    saveclassifier
from training import train, savemodel, loadmodel, testing
import subprocess
import pkg_resources

gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

global path_for_one
localizator = tf.keras.models.load_model('my_bb_model.keras')
classifier = tf.keras.models.load_model('my_classifier.keras')

window = Tk()
window.geometry("1240x720")
window.resizable(False, False)
window.title("Классификатор пожара")
window.iconbitmap(default="bplaicon.ico")
window.configure(bg='silver')
image_field_raw = Canvas(bg="darkgray", height=500, width=500)
image_field_raw.place(relx=0.05, rely=0.2, relheight=0.6, relwidth=0.4)
title1 = Label(image_field_raw, font=30, text=f"Загруженное изображение")
title1.pack(anchor=N)
image_field_ready = Canvas(bg="darkgray", height=500, width=500)
image_field_ready.place(relx=0.55, rely=0.2, relheight=0.6, relwidth=0.4)
title2 = Label(image_field_ready, font=30, text=f"Результат распознавания")
title2.pack(anchor=N)
frame = Frame(window, bg='darkgray')
frame.place(relheight=0.1, relwidth=1)
frame1 = Frame(window, bg='darkgray')
```

```
49 frame1.place(relheight=0.1, relwidth=1, rely=0.9)
50
51
52 class ConsoleRedirector(object):
53     def __init__(self, text_widget):
54         self.text_widget = text_widget
55
56     def write(self, string):
57         self.text_widget.insert(tk.END, string)
58         self.text_widget.see(tk.END)
59
60     def flush(self):
61         pass
62
63
64 # функция работы с нейросетями: подготовка изображения, детекция
65 # на выходе три массива: координаты (10,4) , классы (10) , вероятности (10)
66 def detect_objects(image):
67     # подготовка картинки
68     image = tf.cast(image, dtype=tf.float32) / 256
69     small_image = tf.image.resize(image, (128, 128))
70     big_image = tf.image.resize(image, (1024, 1024))
71     image_exp = tf.expand_dims(small_image, axis=0)
72
73     # локализация
74     bb_cords = localizator(image_exp)
75     bb_cords = tf.squeeze(bb_cords, axis=0)
76
77     # нормализация по размеру картинки
78     bb_cords = (bb_cords + 1) / 2 * 128
79     bb_cords = tf.reshape(bb_cords, [10, 3])
80
81     # разделяем на элементы
82     fxmin, ymin, fxmax = tf.split(bb_cords, 3, axis=1)
83
84     # нормализуем
85     xmin = tf.minimum(fxmin, fxmax)
86     xmax = tf.maximum(fxmin, fxmax)
87
88     xmin = tf.clip_by_value(xmin, 0, 128)
89     ymin = tf.clip_by_value(ymin, 0, 128)
90
91     size = xmax - xmin
92
93     # сумма координаты и размера должны быть <= 128
94     xsize = tf.clip_by_value(size, 1, 128 - xmin)
95     ysize = tf.clip_by_value(size, 1, 128 - ymin)
96
97     # нарезаем и собираем в массив (10, 32, 32, 3)
98     ymin *= 8
99     xmin *= 8
100    ysize *= 8
101    xsize *= 8
102    for n in range(10):
```

```
103         ii = tf.image.crop_to_bounding_box(big_image, int(ymin[n][0]), int(
                xmin[n][0]), int(ysize[n][0]),
104                                            int(xsize[n][0]))
105         ii = tf.image.resize(ii, (128, 128))
106         ii = tf.expand_dims(ii, axis=0)
107         if n == 0:
108             cropped = ii
109         else:
110             cropped = tf.concat([cropped, ii], axis=0)
111 
112     # классифицируем
113     probs = classifier(cropped)
114 
115     probs = probs.numpy()
116 
117     # считаем метки класса (индекс наибольшего среди вероятностей)
118     ma = np.amax(probs, axis=1)
119     ma = np.expand_dims(ma, axis=1)
120     _, classes = np.where(probs == ma)
121 
122     # берем ту вероятность, которая наибольшая
123     res_probs = []
124     for a in range(10):
125         res_probs.append(probs[a][classes[a]])
126 
127     # собираем координаты в нормальный вид
128     cords = tf.concat([xmin / 8, ymin / 8, xmin / 8 + xsize / 8, ymin / 8 +
            ysize / 8], axis=1)
129     cords = cords.numpy()
130 
131     return cords, classes, res_probs
132 
133 
134 # th - вероятность, ниже которой рамки не отображаются
135 namespace = {0: 'NOTHING', 1: 'Fire'}
136 
137 
138 def visualize(in_image, cords, classes, probs, th=0.5):
139     big_image = tf.image.resize(in_image, (1024, 1024)).numpy() / 256
140 
141     font = cv2.FONT_HERSHEY_SIMPLEX
142     fontScale = 1
143     thickness = 2
144 
145     for i in range(len(cords)):
146         if classes[i] != 0 and probs[i] >= th:
147             # введем цвета для всех обьектов
148             if classes[i] == 1:
149                 color = (1, 0, 0)
150             if classes[i] == 2:
151                 color = (0, 1, 0)
152             text = namespace[classes[i]] + ' ' + str(probs[i] * 100) + '%'
153 
154             org = (int(cords[i][0]) * 8, int(cords[i][1]) * 8 - 10)
```

```
155             # рисуем текст и квадраты
156             big_image = cv2.putText(big_image, text, org, font, fontScale,
                    color, thickness, cv2.LINE_AA)
157             big_image = cv2.rectangle(big_image, (int(cords[i][0]) * 8, int(
                    cords[i][1]) * 8),
158                                       (int(cords[i][2]) * 8, int(cords[i][3])
                                            * 8), color, 5)
159
160     return big_image
161
162
163 # функция обьединяет две рамки одного класса в одну, если они пересекаются
164 # tau - порог IoU этих рамок чтобы их обьединить (0.1 - все подряд, 0.9 -
        только очень близкие)
165 # функцию можно применять несколько раз, результат улучшится
166
167 def prettify(cords, classes, probs, tau=0.2):
168     newcords = []
169     newclasses = []
170     newprobs = []
171
172     for i1 in range(len(classes)):
173
174         if classes[i1] != 0:
175             found = False
176             for i2 in range(len(classes)):
177                 if classes[i2] != 0 and i1 != i2:
178
179                     # вычислим IoU, самый надежный способ определить
                            совпадение
180                     x_overlap = max(0, min(cords[i1][2], cords[i2][2]) - max(
                            cords[i1][0], cords[i2][0]))
181                     y_overlap = max(0, min(cords[i1][3], cords[i2][3]) - max(
                            cords[i1][1], cords[i2][1]))
182                     inter = x_overlap * y_overlap
183                     area1 = (cords[i1][2] - cords[i1][0]) * (cords[i1][3] -
                            cords[i1][1])
184                     area2 = (cords[i2][2] - cords[i2][0]) * (cords[i2][3] -
                            cords[i2][1])
185                     union = area1 + area2 - inter
186                     IoU = inter / union
187                     if IoU > tau:
188                         # считаем среднее по всем координатам между двух
                                рамок
189                         newcord = [(cords[i1][0] + cords[i2][0]) // 2, (cords
                                [i1][1] + cords[i2][1]) // 2,
190                                    (cords[i1][2] + cords[i2][2]) // 2, (cords
                                        [i1][3] + cords[i2][3]) // 2]
191                         newcords.append(newcord)
192
193                         newclasses.append(classes[i1])
194                         newprobs.append(probs[i1])
195
196                         # обнуляем класс, чтобы больше не крутить эту рамку
```

```
                        classes[i1] = 0
                        classes[i2] = 0
                        found = True

            # если ни с чем не обьединили, так и оставляем
            if found == False:
                newcords.append(cords[i1])
                newclasses.append(classes[i1])
                newprobs.append(probs[i1])

    return newcords, newclasses, newprobs


def detect_fire_in_image(path_for_one):
    image = cv2.imread(path_for_one)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    cords, classes, probs = detect_objects(image)
    for i in range(1):
        cords, classes, probs = prettify(cords, classes, probs, 0.8)
    result = visualize(image, cords, classes, probs, 0.5)

    buf = io.BytesIO()
    plt.figure()
    plt.imshow(result)
    plt.axis('off')  # Скрываем оси
    plt.savefig(buf, format='png')
    buf.seek(0)

    original_image = Image.open(buf)
    resized_image = original_image.resize(
        (int(original_image.width * 1.35), int(original_image.height * 1.15))
            )
    image_tk = ImageTk.PhotoImage(resized_image)

    image_field_ready.create_image(-192, -65, anchor=NW, image=image_tk, tags
        ='Старое')
    image_field_ready.image_tk = image_tk  # Сохраняем ссылку на изображение

    # Закрываем буфер
    buf.close()


def loadimage():
    global image
    global path_for_one
    path_for_one = filedialog.askopenfilename(filetypes=[("Изображения", "*.
        png;*.jpg;*.jpeg")])
    if path_for_one:
        pil_image = Image.open(path_for_one)
        pil_image = pil_image.resize((500, 500))
        image = ImageTk.PhotoImage(pil_image)
        image_field_raw.create_image(0, 0, anchor=NW, image=image)
```

```
248 def detect():
249     global path_for_one
250     detect_fire_in_image(path_for_one)
251 
252 
253 def check_and_install_packages():
254     required = {'PyQt5', 'sip'}
255     installed = {pkg.key for pkg in pkg_resources.working_set}
256     missing = required - installed
257     if missing:
258         python = sys.executable
259         subprocess.check_call([python, '-m', 'pip', 'install', *missing],
                stdout=subprocess.DEVNULL)
260 
261 
262 def run_labelimg():
263     showinfo("Обязательная информация",
264              "Для использования этого приложения необходимо иметь
                    установленные пакеты PyQt5 и sip")
265     check_and_install_packages()
266     labelimg_path = 'C:/ForestFireDiplomFinVer/labelimg/labelImg.py'
267     subprocess.run(['python', labelimg_path])
268 
269 
270 def create_tfrec_classifier():
271     showwarning("Предупреждение", "Классификатор настроен и не нуждается в
        добавлении новых данных")
272 
273 
274 def start_train_localizer():
275     loadmodel()
276     train()
277 
278 
279 def train_window_open():
280     window1 = Tk()
281     window1.geometry("720x910")
282     window1.resizable(False, False)
283     window1.title("Окно тренировки модели нейросети")
284     window1.iconbitmap(default="bplaicon.ico")
285     window1.configure(bg='silver')
286 
287     close_button = Button(window1, text="Закрыть окно", command=lambda:
        window1.destroy())
288     close_button.pack(anchor=NE, expand=0)
289 
290     processdataforonefile = Button(window1, font=40, text=f"Информация о
        картинке", command=dataprocess)
291     processdataforonefile.pack(anchor=NW)
292 
293     runlabelimg = Button(window1, font=40, text=f"Запустить LabelImg",
        command=run_labelimg)
294     runlabelimg.pack(anchor=NW)
295 
```

```
296     check_xml_list_but = Button(window1, font=40, text=f"Файлы для обучения",
            command=check_xml_list)
297     check_xml_list_but.pack(anchor=NW)
298
299     check_xml_list_but = Button(window1, font=40, text=f"Создать новую запись
            локализатора",
300                                 command=create_load_tfrec_for_localizer)
301     check_xml_list_but.pack(anchor=NW)
302
303     tfrec_classifier = Button(window1, font=40, text=f"Создать новую запись
            классификатора",
304                               command=create_tfrec_classifier)
305     tfrec_classifier.pack(anchor=NW)
306
307     classifier_test_but = Button(window1, font=40, text=f"Протестировать
            классификатор", command=test_classifier)
308     classifier_test_but.pack(anchor=NW)
309
310     true_classifier_test_but = Button(window1, font=40, text=f"Точный тест
            классификатора", command=imshow_and_pred)
311     true_classifier_test_but.pack(anchor=NW)
312
313     train_classifier = Button(window1, font=40, text=f"Тренировать модель
            классификатора", command=trainclass)
314     train_classifier.pack(anchor=NW)
315
316     save_classifier = Button(window1, font=40, text=f"Сохранить модель
            классификатора", command=saveclassifier)
317     save_classifier.pack(anchor=NW)
318
319     save_localizer = Button(window1, font=40, text=f"Сохранить модель
            локализатора", command=savemodel)
320     save_localizer.pack(anchor=NW)
321
322     load_localizer = Button(window1, font=40, text=f"Загрузить модель
            локализатора", command=loadmodel)
323     load_localizer.pack(anchor=NW)
324
325     test_localizer = Button(window1, font=40, text=f"Тест локализатора",
            command=testing)
326     test_localizer.pack(anchor=NW)
327
328     train_localizer = Button(window1, font=40, text=f"Тренировать модель
            локализатора", bg='Green', fg='black',
329                              command=start_train_localizer)
330     train_localizer.pack(anchor=NW)
331
332     text = tk.Text(window1)
333     text.pack(anchor='center')
334     sys.stdout = ConsoleRedirector(text)
335
336
337 def load_model_data():
338     global localizator
```

```
    model_path = filedialog.askopenfilename(filetypes=[("Модель keras", "*.keras")])
    if model_path:
        localizator = tf.keras.models.load_model(model_path)


load_image_but = Button(frame, font=40, text=f"Загрузить изображение", command=loadimage)
load_image_but.grid(column=0, row=0, padx=10, pady=20)

load_model_but = Button(frame, font=40, text=f"Загрузить свою модель нейросети", command=load_model_data)
load_model_but.grid(column=1, row=0, padx=10)

detect_but = Button(frame, font=40, text=f"Распознать", bg='Green', fg='black', command=detect)
detect_but.grid(column=2, row=0, padx=10)

train_window_but = Button(frame1, font=40, text=f"Открыть окно для обучения", command=train_window_open)
train_window_but.grid(column=0, row=0, padx=10, pady=20)

window.mainloop()
```

classifier.py

```
import tensorflow as tf
from IPython.display import clear_output
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.layers import Dense, Flatten, Input, Conv2D, GlobalAveragePooling2D, Dropout


dataset = tf.data.TFRecordDataset('classifier_dataset.tfrecord')

def parse_record(record):
    # нужно описать приходящий экземпляр
    # имена элементов как при записи
    feature_description = {
        'img': tf.io.FixedLenFeature([], tf.string),
        'name': tf.io.FixedLenFeature([], tf.string)
    }
    parsed_record = tf.io.parse_single_example(record, feature_description)
    img = tf.io.parse_tensor(parsed_record['img'], out_type=tf.float32)
    name = tf.io.parse_tensor(parsed_record['name'], out_type=tf.int32)
    return img, name


# пройдемся по записи и распакуем ее
dataset = dataset.map(parse_record)

# еще раз проверим

```

```
29
30 dataset = dataset.shuffle(50).cache().prefetch(buffer_size=tf.data.AUTOTUNE).
       batch(64).shuffle(50)
31
32 def test_classifier():
33     for i, n in dataset.take(1):
34         plt.figure(figsize=(10, 6))
35         i = i.numpy()
36         n = n.numpy()
37         for nn in range(32):
38             ax = plt.subplot(5, 10, 1 + nn)
39             plt.title(n[nn])
40             plt.imshow(i[nn])
41             plt.axis('off')
42         plt.show()
43
44
45 base_model = tf.keras.applications.MobileNetV2(weights='imagenet',
       include_top=False, input_shape=(128, 128, 3))
46 base_model.trainable = False
47
48 # Архитектура сети классификатора
49 inputs = Input((128,128,3))
50 x = base_model(inputs)
51 x = Flatten()(x)
52 x = Dropout(0.5)(x)
53 x = Dense(256, activation = 'relu')(x)
54 x = Dropout(0.2)(x)
55 x = Dense(3, activation = 'softmax')(x)
56
57 outputs = x
58
59 classifier = keras.Model(inputs, outputs)
60
61
62 # модель нейросети классификатора
63 class Model(tf.keras.Model):
64     def __init__(self, nn):
65         super(Model, self).__init__()
66         self.nn = nn
67         self.optimizer = tf.keras.optimizers.Adam(1e-3)
68
69     def get_loss(self, y, preds):
70         loss = tf.keras.losses.CategoricalCrossentropy()(tf.one_hot(y, 3),
               preds)
71         return loss
72
73     @tf.function
74     def training_step(self, x, y):
75         with tf.GradientTape() as tape:
76             preds = self.nn(x)
77             loss = self.get_loss(y, preds)
78
79         gradients = tape.gradient(loss, self.nn.trainable_variables)
```

```
80          self.optimizer.apply_gradients(zip(gradients, self.nn.
                trainable_variables))
81          return tf.reduce_mean(loss)
82
83
84  model = Model(classifier)
85
86
87  # Тензор
88  # for i, c in dataset.take(1):
89  #     print(tf.reduce_mean(model.training_step(i, c)))
90
91  # проверка тренировки как в training.py
92  def trainclass():
93      hist = np.array(np.empty([0]))
94      epochs = 10
95
96      for epoch in range(1, epochs + 1):
97          loss = 0
98          lc = 0
99          for step, (i, n) in enumerate(dataset):
100             loss += tf.reduce_mean(model.training_step(i, n))
101             lc += 1
102         clear_output(wait=True)
103         print(epoch)
104         hist = np.append(hist, loss / lc)
105         plt.plot(np.arange(0, len(hist)), hist)
106         plt.show()
107         if epoch != epochs:
108             plt.close()
109
110
111 def imshow_and_pred():
112     n = 5
113     plt.figure(figsize=(10, 6))
114     for images, labels in dataset.take(1):
115         preds = model.nn(images)
116         for i in range(n):
117             img = images[i]
118
119             pred = preds[i].numpy()
120             print(pred)
121             ax = plt.subplot(3, n, i + 1 + n)
122             plt.imshow(img, cmap='gist_gray')
123
124             ma = pred.max()
125             res = np.where(pred == ma)
126
127             plt.title(str(res[0][0]) + ' ' + str(round(pred[res[0][0]], 3)))
128             plt.axis('off')
129             ax.get_yaxis().set_visible(False)
130     plt.show()
131
132
```

```
133 def saveclassifier():
134     model.nn.save('my_classifier.keras')
```

training.py

```
1  import tensorflow as tf
2  from IPython.display import clear_output
3
4  gpus = tf.config.experimental.list_physical_devices('GPU')
5  for gpu in gpus:
6      tf.config.experimental.set_memory_growth(gpu, True)
7  from tensorflow import keras
8  from tensorflow.keras.layers import Dense, Flatten, Input, Conv2D,
       Conv2DTranspose, Concatenate, LeakyReLU, Dropout
9  import cv2
10 import numpy as np
11 import matplotlib.pyplot as plt
12
13 # Структура сети по детекции
14 inputs = Input((128, 128, 3))
15 x = Conv2D(32, 3, activation='relu', padding='same')(inputs)
16 x = Conv2D(64, 3, activation='relu', padding='same', strides=2)(x)
17 x = Conv2D(64, 3, activation='relu', padding='same')(x)
18 x = Conv2D(64, 3, activation='relu', padding='same', strides=2)(x)
19 x = Conv2D(128, 3, activation='relu', padding='same')(x)
20 x = Conv2D(128, 3, activation='relu', padding='same', strides=2)(x)
21 x = Conv2D(128, 3, activation='relu', padding='same')(x)
22 x = Conv2D(256, 3, activation='relu', padding='same', strides=2)(x)
23 x = Conv2D(256, 3, activation='relu', padding='same')(x)
24 x = Flatten()(x)
25 x = Dropout(0.2)(x)
26 x = Dense(256, activation='relu')(x)
27 x = Dense(30)(x)  # 3*10 = 30 у нейросети это просто выходы подряд
28
29 outputs = x
30
31 boxregressor = keras.Model(inputs, outputs)
32
33 # прочитаем запись
34 dataset = tf.data.TFRecordDataset('bounding_box_dataset.tfrecord')
35
36
37 def parse_record(record):
38     # имена элементов как при записи
39     feature_description = {
40         'img': tf.io.FixedLenFeature([], tf.string),
41         'cords': tf.io.FixedLenFeature([], tf.string)
42     }
43     parsed_record = tf.io.parse_single_example(record, feature_description)
44     img = tf.io.parse_tensor(parsed_record['img'], out_type=tf.float32)
45     cords = tf.io.parse_tensor(parsed_record['cords'], out_type=tf.float32)
46     return img, cords
47
48
49 # пройдемся по записи и распакуем ее
```

```
50 dataset = dataset.map(parse_record)
51
52 dataset = dataset.cache().prefetch(buffer_size=tf.data.AUTOTUNE).batch(32).
       shuffle(40)
53
54
55 # функция с вычислением IoU loss
56 def IoU_Loss(true, pred):
57     # (32, 5, 4)
58     t1 = true
59     t2 = pred
60
61     minx1, miny1, maxx1, maxy1 = tf.split(t1, 4, axis=2)
62
63     fminx, miny2, fmaxx = tf.split(t2, 3, axis=2)
64
65     minx2 = tf.minimum(fminx, fmaxx)
66     maxx2 = tf.maximum(fminx, fmaxx)
67
68     delta = maxx2 - minx2
69
70     maxy2 = miny2 + delta
71
72     intersection = 0.0
73
74     # найдем пересечение каждого из предсказанных с каждым из реальных
75     # сложим все вместе
76     for i1 in range(10):
77         for i2 in range(10):
78             x_overlap = tf.maximum(0.0, tf.minimum(maxx1[:, i1], maxx2[:, i2
                   ]) - tf.maximum(minx1[:, i1], minx2[:, i2]))
79             y_overlap = tf.maximum(0.0, tf.minimum(maxy1[:, i1], maxy2[:, i2
                   ]) - tf.maximum(miny1[:, i1], miny2[:, i2]))
80             intersection += x_overlap * y_overlap
81
82     # стремимся сделать площади всех элементов такими-же, как у реальных
           рамок
83     # просто среднеквадратичной ошибкой
84
85     beta1 = 0.0
86     for i1 in range(10):
87         for i2 in range(10):
88             x_overlap = tf.maximum(0.0, tf.minimum(maxx1[:, i1], maxx1[:, i2
                   ]) - tf.maximum(minx1[:, i1], minx1[:, i2]))
89             y_overlap = tf.maximum(0.0, tf.minimum(maxy1[:, i1], maxy1[:, i2
                   ]) - tf.maximum(miny1[:, i1], miny1[:, i2]))
90             if i1 == i2:
91                 beta1 += (x_overlap * y_overlap) ** 2
92             else:
93                 beta1 += x_overlap * y_overlap
94
95     beta2 = 0.0
96     for i1 in range(10):
97         for i2 in range(10):
```

```
98              x_overlap = tf.maximum(0.0, tf.minimum(maxx2[:, i1], maxx2[:, i2
                    ]) - tf.maximum(minx2[:, i1], minx2[:, i2]))
99              y_overlap = tf.maximum(0.0, tf.minimum(maxy2[:, i1], maxy2[:, i2
                    ]) - tf.maximum(miny2[:, i1], miny2[:, i2]))
100             if i1 == i2:
101                 beta2 += (x_overlap * y_overlap) ** 2
102             else:
103                 beta2 += x_overlap * y_overlap
104
105     loss = (beta1 - beta2) ** 2 - intersection
106
107     return loss
108
109
110 # Класс модели нейросети
111 class Model(tf.keras.Model):
112     def __init__(self, nn_box):
113         super(Model, self).__init__()
114         self.nn_box = nn_box
115
116         self.box_optimizer = tf.keras.optimizers.Adam(3e-4, clipnorm=1.0)
117
118     @tf.function
119     def training_step(self, x, true_boxes):
120         with tf.GradientTape() as tape_box:
121             pred = self.nn_box(x, training=True)
122             pred = tf.reshape(pred, [-1, 10, 3])
123
124             loss = IoU_Loss(true_boxes, pred)
125             #       print('test', tf.reduce_mean(IoU_Loss(true_boxes,
                        true_boxes) ))
126
127         # Backpropagation.
128         grads = tape_box.gradient(loss, self.nn_box.trainable_variables)
129         self.box_optimizer.apply_gradients(zip(grads, self.nn_box.
                trainable_variables))
130
131         return loss
132
133
134 model = Model(boxregressor)
135
136 # показывает тензор
137 # for i, c in dataset.take(1):
138 #     print(tf.reduce_mean(model.training_step(i, c)))
139
140
141 def savemodel():
142     # сохранить
143     model.nn_box.save('my_bb_model.keras')
144
145
146 def loadmodel():
147     # загрузить веса
```

```
148     model.nn_box.load_weights('my_bb_model.keras')
149
150
151 def testing():
152     for ii, cc in dataset.take(1):
153         # обрабатывем целый батч, используем только пять элементов
154         pred = model.nn_box(ii)
155         plt.figure(figsize=(10, 6))
156
157         for num in range(3):
158             i = ii[num]
159
160             pred = tf.reshape(pred, [-1, 10, 3])
161             c = pred[num]
162
163             ax = plt.subplot(1, 5, num + 1)
164             # переход в numpy для работы в opencv
165             i = i.numpy()
166             c = c.numpy()
167             c = (c + 1) / 2 * 128  # обратно из от -1...1 к 0...64
168             c = c.astype(np.int16)  # для opencv
169             for bb in c:
170                 bb0 = min(bb[0], bb[2])
171                 bb2 = max(bb[0], bb[2])
172                 i = cv2.rectangle(i, (bb0, bb[1]), (bb2, bb[1] + (bb2 - bb0))
                        , (0, 1, 0), 1)
173             plt.imshow(i)
174
175         plt.show()
176     # print(c)
177
178
179 def train():
180     hist = np.array(np.empty([0]))
181     epochs = 10                                  #<<<----- Кол-во эпох
182     ff = 0
183     for epoch in range(1, epochs + 1):
184         loss = 0
185         lc = 0
186         for step, (i, c) in enumerate(dataset):
187             loss += tf.reduce_mean(model.training_step(i, c))
188             lc += 1
189         clear_output(wait=True)
190         print(epoch)
191         hist = np.append(hist, loss / lc)
192
193         plt.plot(np.arange(0, len(hist)), hist)
194         plt.show()
195         testing()
196
197
198 #loadmodel()
199 #train()
```

Место для диска