МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Восточно-Сибирский государственный университет технологий и управления»

(ВСГУТУ)

Технологический колледж

Кафедра «Системы информатики»

Допущен к защите

«____» _____2022 г.

Зав. каф. «Системы информатики»
д.э.н., Михайлова С.С.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(ДИПЛОМНЫЙ ПРОЕКТ)
(Д.30.1.09.02.03.14.110.ПЗ)

на тему: «РАЗРАБОТКА СИСТЕМЫ ОБНАРУЖЕНИЯ ЛЕСНЫХ ПОЖАРОВ ПО СПУТНИКОВЫМ СНИМКАМ»

Исполнитель: обучающийся по специальности
«Программирование в компьютерных системах»
очной формы обучения группы К78/1

Павлова Виктория Алексеевна _____

Руководитель работы _____ Преподаватель Евдокимова И.С.

Рецензент /_____/ к.п.н., доцент каф. СИ ВСГУТУ Тулохонова И.С.

Нормоконтролер /_____/ Преподаватель Доржиева Э.Ц.

Улан-Удэ, 2022

ВОСТОЧНО-СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ
ТЕХНОЛОГИЧЕСКИЙ КОЛЛЕДЖ

Утверждаю:
_____
«____» _____2022 г.
Зав. каф. «Системы информатики»
д.э.н., доцент Михайлова С.С.

ЗАДАНИЕ

По подготовке выпускной квалификационной работы (ВКР)

Обучающемуся Павловой Виктории Алексеевне

1. Тема ВКР: Разработка системы обнаружения лесных пожаров по спутниковым снимкам
(утверждена приказом ВСГУТУ от «_29_» декабря 2021 № 4401у)

2. Срок сдачи обучающимся законченной ВКР «_6_» июня 2022г.

3. Исходные данные к ВКР: Case-средства ERWin, среда разработки Pycharm, Google, Collabotory

4. Перечень подлежащих разработке в ВКР вопросов или краткое содержание ВКР:
– анализ предметной областия;
– построение функциональной модели «Как надо»;
– разработка архитектуры модели нейронной сети;
– разработка модели нейронной сети;
– разработка алгоритма обучения нейронной сети;
– реализация программного приложения.

5. Перечень демонстрационного материалов;
– презентация Power Point;
– видеофильм в формате mp4.

6. Консультанты ВКР (с указанием относящихся к ним разделов и частей ВКР)

7. Дата выдачи задания «__» _____ 2022 г.

Руководитель _____/ Евдокимова И.С. /
(*подпись*) *(И.О.Фамилия)*

Обучающийся _____ / Павлова В.А. /
(*подпись*) *(И.О.Фамилия)*

АННОТАЦИЯ

Павлова В.А. Разработка системы обнаружения лесных пожаров по спутниковым снимкам. Выпускная квалификационная работа (дипломный проект). ТК ВСГУТУ, 2022. 59 страниц, 59 рис., 1 таблица, 10 источников, 1 Приложение

Данная внутренняя квалификационная работа посвящена разработке реализации системы обнаружения лесных пожаров по спутниковым снимкам. Разрабатываемая система предназначена для обнаружения лесных пожаров при помощи нейронной сети, которая будет анализировать снимки со спутника. В рамках работы будет выполнено подробное изучение предметной области, описана архитектура нейронной сети, разработана и обучена модель нейронной сети, а также разработано и протестировано приложение.

Пояснительная записка содержит введение, 3 раздела, заключение, список использованных источников и приложение.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6
1 Теоретический раздел 8
1.1 Описание предметной области 8
1.2 Обзор существующих решений 11
1.3 Актуальность проекта 15
1.4 Применение нейросетевого программирования для выявления возгораний на основе снимков со спутника 17
1.4.1 Снимки со спутников 21
1.4.2 Нейронные сети 24
1.4.3 Свёрточные нейронные сети 26
1.4.4 Функции активации 28
2 Проектный раздел 31
2.1 Модель «Как надо» 31
2.2 Алгоритм работы программного приложения 33
2.3 Алгоритм подготовки данных для обучения нейронной сети 34
2.4 Архитектура модели нейронной сети 39
2.5 Метод обучения нейронной сети 40
2.6 Алгоритм обучения нейронной сети 41
3 Программный раздел 44
3.1 Сравнение языков программирования 44
3.1.1 Python 44
3.1.2 C++ 44
3.1.3 JavaScript 44
3.1.4 Выбор средств разработки 45
3.2 Реализация нейронной сети 45

3.3 Описание интерфейса 46
3.3.1 Основное окно 46
3.3.2 Диалоговые окна 47
3.4 Описание подключаемых модулей и библиотек 49
3.4.1 TensorFlow 49
3.4.2 Keras 50
3.4.3 Tkinter 51
3.4.4 Os 51
3.4.5 PathLib 51
3.4.6 Numpy 52
3.5 Описание основных функций 53
3.5.1 Загрузка данных 53
3.5.2 Анализ 53
3.5.3 Отображение картинки 54
3.5.4 Распределение 55
3.6 Апробация нейросетевой модели 56
ЗАКЛЮЧЕНИЕ 57
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 58
ПРИЛОЖЕНИЕ А 60

ВВЕДЕНИЕ

Решение экологических проблем – одна из самых актуальных задач на данный момент, так как с каждым годом ресурсы земли начинают исчерпывать себя, а их восстановление занимает достаточно много времени. Лесные пожары довольно быстро охватывают огромную площадь и из-за этого страдают не только лесные массивы, которые сгорели, но и то, что находится рядом: реки, животные, почва и даже воздух. Восстановление пораженных мест может занимать несколько десятков лет в лучшем случае! А учитывая, что с каждым годом пожары становятся всё более масштабными, не говоря о том, что самих очагов возгорания становится больше, урон, нанесенный нашей планете, просто фатальный!

Выявление очагов возгорания сегодня можно решать разными способами, но наиболее эффективным и менее затратным может быть способ идентификации пожара на основе снимков из космоса.

Целью проекта является разработка механизма обнаружения лесных пожаров по спутниковым снимкам. Таким образом, для решения поставленной задачи необходимо:

1. провести анализ предметной области;
2. разработать архитектуру приложения;
3. разработать модель нейронной сети;
4. обучить и протестировать модель;
5. разработать приложение для возможности идентификации пожара, дыма на снимке со спутника.

Объект исследования – идентификация пожара.

Предмет исследования – вычисление вероятности пожара на спутниковом снимке.

Практическая значимость: использование нейронной сети в решении поставленной задачи позволяет автоматизировать процесс идентификации возгорания на изображении, снизить количество затрачиваемого времени и ресурсов на проверку больших объёмов данных.

Работа состоит из введения, трех глав, заключения и списка литературы.

В введении кратко описана актуальность, определён объект, предмет, цель и задачи. В первой главе рассматриваются предметная область, существующие решения, актуальность проекта и применение нейронных сетей для идентификации. Во второй главе проектируется модель «Как надо», алгоритм работы программы, алгоритм подготовки данных для обучения нейронной сети, архитектура модели нейронной сети, метод и алгоритм обучения нейронной сети. В третьей главе рассматриваются средства разработки приложений, разработка приложения, описание его интерфейса и функций. В заключении подводятся итоги исследования, формируются окончательные выводы по рассматриваемой теме.

1 Теоретический раздел

1.1 Описание предметной области

Лесной пожар – это чрезвычайная ситуация природного характера, неконтролируемое стихийное бедствие, которое сопровождается горением растительности на территории леса и огромным количеством дыма.

Лесные пожары опасны следующим:

1. быстрое и хаотичное распространение;
2. выгорание кислорода и загрязнение атмосферы углекислым газом;
3. колоссальный ущерб лесному массиву и, как следствие, уничтожение экосистем;
4. непосредственная угроза для населения и пожарных;

Исходя из выявленных угроз, можно выделить некоторые сферы жизни, на которых сказываются последствия пожаров, а именно: экологическая сфера, экономическая и социальная. Так как лес является неотъемлемой частью экологической сферы, то на неё приходится наибольший урон, а именно:

1. уничтожение огромного количества деревьев и растительности, что приводит к снижению качества леса и его биоразнообразия;
2. уничтожение различных видов животных, что приводит к потере уникальной экосистемы;
3. загрязнение воздуха и ухудшение его качества как из-за выбросов углекислого газа, так и из-за распространения пепла от сгоревшей растительности;
4. деградация почвы – потеря плодородности почвы; вероятность того, что почва вовсе станет бесплодной или будет подвержена эрозии; уничтожение важных микроорганизмов;

5. загрязнение водоёмов из-за распространения пепла и ухудшение качества воды.

С экономической стороны последствия от пожаров так же весьма «ощутимы». Чем больше гектаров охватывает пожар, тем больше выделяется средств на то, чтобы его потушить и на то, чтобы восстановить испорченное имущество и сам лесной массив.

Социальная сфера затрагивает людей и то, что с ними происходит во время буйства бедствия. Спасатели могут погибнуть при тушении пожара, а люди, если они живут недалеко от места возгорания, страдают от смога и могут потерять своё имущество, если оно сгорит.

Чем раньше обнаружат пожар, тем быстрее специалисты приступят к его устранению, что уменьшит сложность тушения пожара и количество затрачиваемых ресурсов. Именно поэтому используются все методы обнаружения пожаров: наземный, авиационный и аэрокосмический.

Наземное обнаружение проводится в районах, где имеется сеть дорог и устроены наблюдательные пункты, и включает в себя слежку за лесными массивами и патрулирование по специальным маршрутам. Сообщения о замеченном очаге дыма немедленно передаются в лесничество и, в соответствии с установленной формой, делается запись в журнале обнаружений. Так же стоит отметить, что данный метод может быть автоматизирован благодаря станциям с видеонаблюдением и тепловизионной съемкой, но дальность обнаружения пожара от этого особо не меняется.

Авиационное обнаружение заключается в обнаружение с воздуха и используют для этого легкомоторные самолёты, вертолёты или специальная техника, такие как беспилотные летательные аппараты. Авиапатрулирование проводят по утвержденным маршрутам и выполняют по заявкам от ответственных представителей авиабаз. Режимы авиапатрулирования могут быть изменены в зависимости от класса пожарной опасности.

Аэрокосмическое обнаружение действует на патрулируемой территории и в зонах неохраняемых лесов, посредством данных со спутника. Фотоснимки

позволяют отследить время для установления начала срока патрулирования и оценить погодные условия, которые могут повлиять на пожарную опасность. Данный способ помогает контролировать динамику развития крупных пожаров, учитывая площадь, пройденную огнём, и разрабатывать меры для предотвращения нанесения большего ущерба от пожара. Установленная сканерная аппаратура позволяет отследить как крупные, так и средние пожары, что бывает особенно полезно в зонах неохраняемых лесов, где каждая секунда на счету. Исходя из этих фактов, можно выделить несколько преимуществ по сравнению с другими способами обнаружения:

1. мониторинг больших территорий в труднодоступной для наземного и авиационного обнаружения территории;
2. оперативность получения данных;
3. доступность данных, так как снимки со спутника находятся в открытом доступе;
4. требует меньшего влияния человеческого фактора;
5. слабая зависимость от погодных условий.

Но данный способ обнаружения не лишен недостатков, а именно:

1. площадь возгорания должна быть значительной, то есть потушить очаг, если он находится в глубине лесного массива, быстро не получится;
2. в зависимости от того, какой тип съёмки используется на спутнике, несмотря на то, что влияние погодных условий малое, атмосфера может серьёзно задержать получение снимков из-за, например, облачности;
3. во избежание ошибок, данные необходимо проверять посредством других методов обнаружения, особенно это касается небольших и средних источников возгорания;
4. из-за периода обращения спутника, он может находится в невыгодном для съёмки положении;

1.2 Обзор существующих решений

Для специальных служб МЧС уже разработано приложение, которое позволяет сократить время передачи данных о возникновении пожара до специалистов. Для ликвидации возгораний в природной среде в кратчайшие сроки и минимизации риска перехода огня на населенные пункты, а также оперативного принятия управленческих решений служит приложение «Термические точки». Это приложение позволяет не только оперативно уведомлять о возникающем возгорании, но и пресекать правонарушения в области пожарной безопасности, привлекая виновных к административной ответственности. Вскоре после внедрения и распространения разработки оперативность реагирования на природные пожары повысилась в 3 раза. Система космического мониторинга круглосуточно обрабатывает спутниковые данные дистанционного зондирования Земли. В результате обработки 4 раза в сутки формируется слой с повышенной температурой поверхности. Через мобильное приложение информация о них доводится до представителей органов управления муниципальных образований или собственников, которые после проверки подтверждают, является ли термическая точка пожаром. Кроме того, вся информация дублируется на портале единых дежурно диспетчерских служб муниципальных районов. Интерфейс приложения представлен на рисунке 1.1

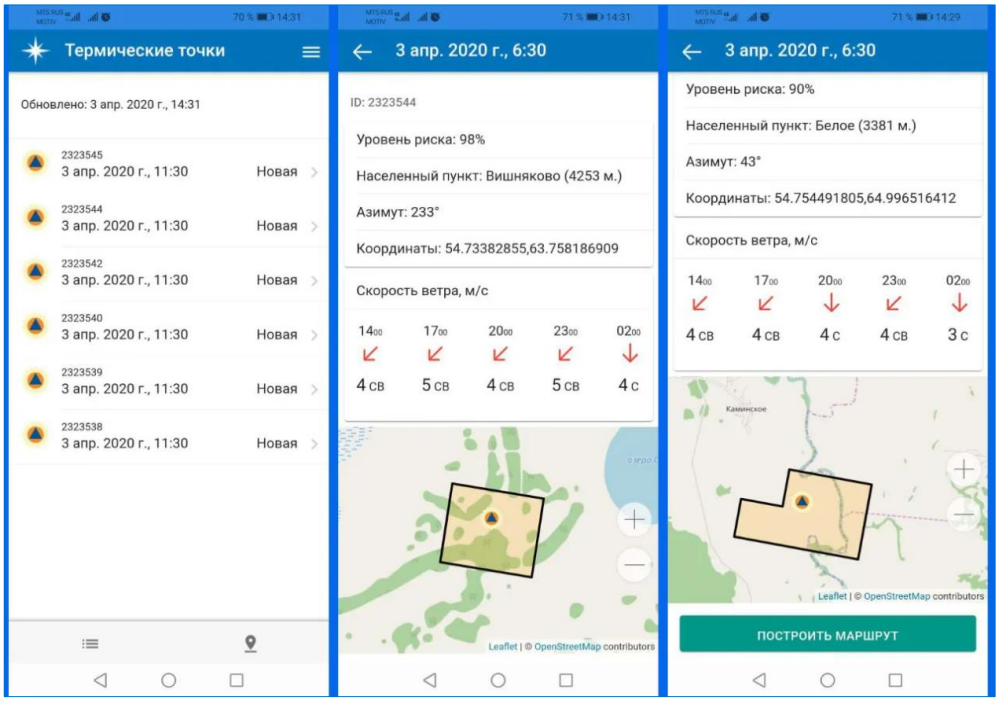

Рисунок 1.1 – Интерфейс приложения «Термические точки»

Одной из самых известных информационных систем, которые используют для обнаружения лесных пожаров, является «Лесной Дозор». Это программно-аппаратный комплекс, благодаря которому можно значительно снизить ущерб во время пожароопасного сезона. Камеры, установленные на телекоммуникационных объектах, работают в режиме реального времени и отвечают за обнаружение дыма. Система автоматизирована и сама определяет координаты, а после передает данные в центр контроля. Таким образом, пожар возможно предотвратить ещё на стадии возгорания.

На рисунке 1.2 представлен интерфейс программы.

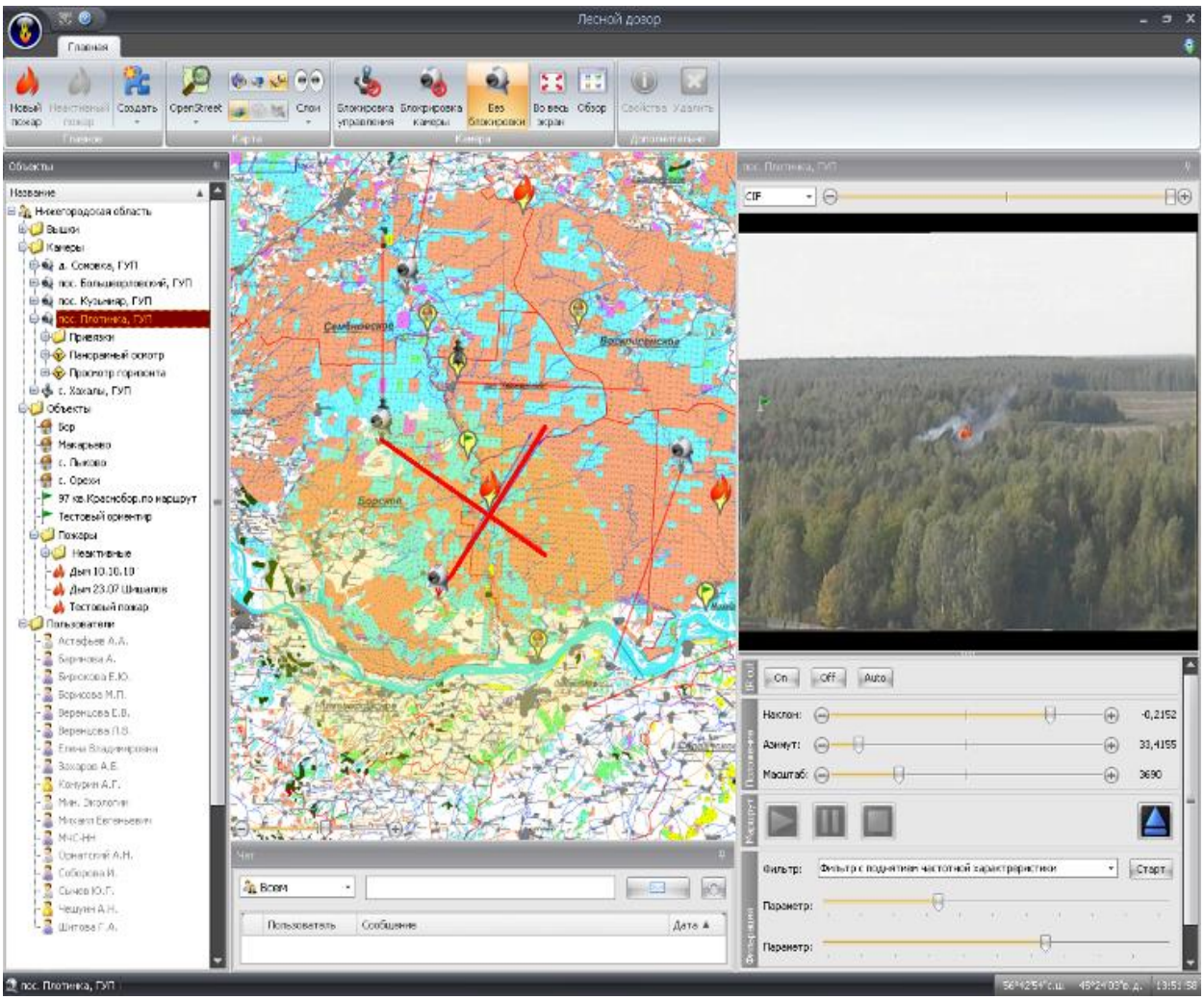

Рисунок 1.2 – Интерфейс программы «Лесной Дозор»

Аппаратная часть состоит из сети управляемых датчиков наблюдения (видеотехники, тепловизионных датчиков и специальных инфракрасных камер), что позволяет обнаружить пожар как на небольших территориях, так и на больших площадях. Эти датчики могут быть размещены на любом высотном сооружении с постоянным электроснабжением, но чаще их закрепляют на вышках провайдеров связи; вышках мобильной связи; телевизионных вышках; вышках освещения. На рисунке 1.3 показано, как работает функция наблюдения.

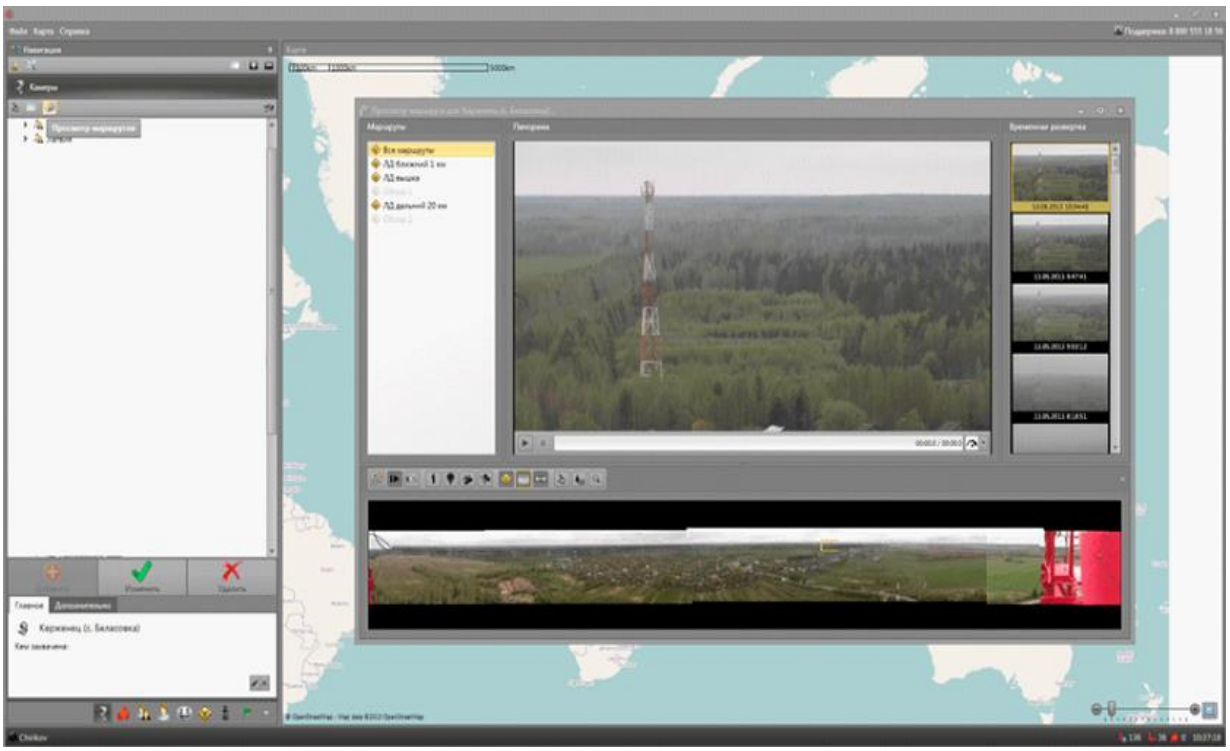

Рисунок 1.3 – функция наблюдения в программе «Лесной Дозор»

Программная часть – это узкоспециализированное программное обеспечение, отвечающее за контроль мониторинга леса в режиме реального времени и определение координат очагов возгораний. Она реализована на платформе .NET с использованием MS SQL Express и имеет систему распределенных серверов вместе с сервером для хранения головных баз данных. Блок раннего обнаружения пожаров, встроенный в камера контроллер, написан на языке C++. Таким образом система оборудована дружественным интерфейсом и располагает объёмным функционалом, а именно:

1. круглосуточная работа системы;
2. автоматическое определение пожароопасного объекта;
3. определение расстояния до очага возгорания и возможность проложить до него маршрут;
4. присвоение категории опасности замеченного пожара;
5. видеозапись пожароопасных объектов;
6. архивизация всех обнаруженных пожаров;
7. поддержка квартальных карт;
8. визуализация сил и средств тушения пожаров.

На рисунке 1.4 продемонстрирована работа функции для определения очага возгорания в системе координат.

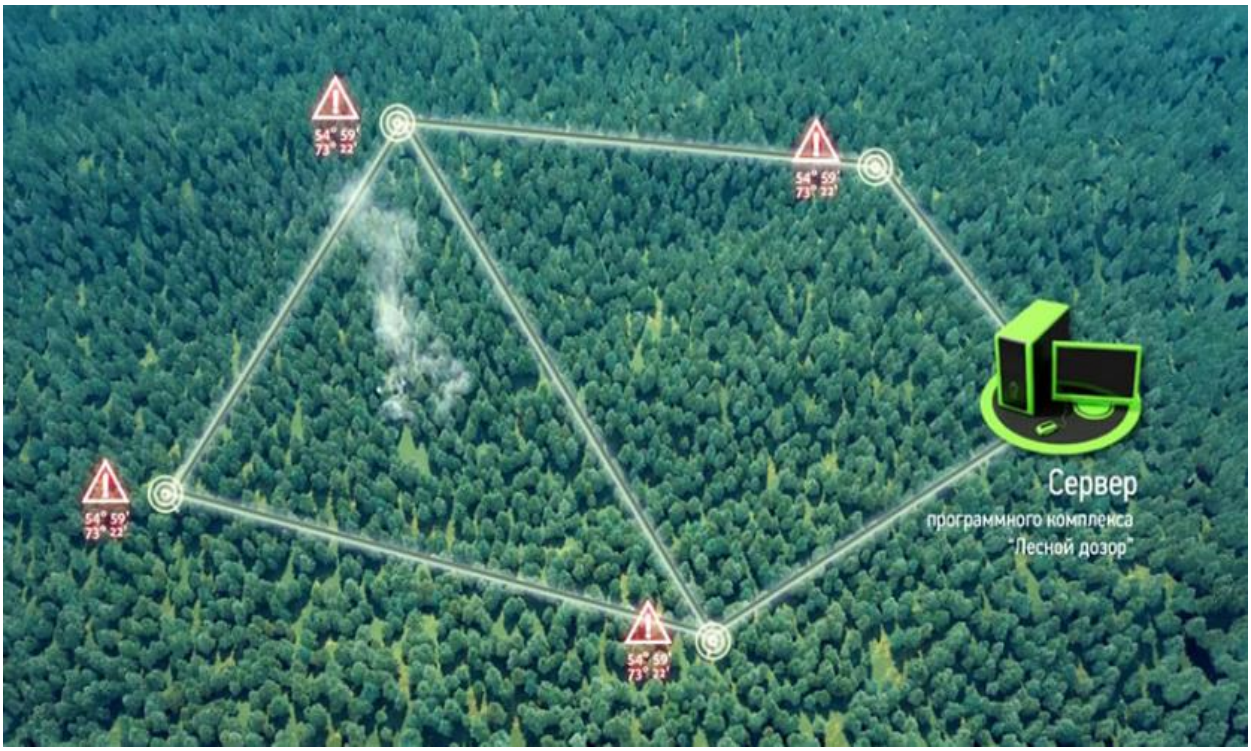

Рисунок 1.4 – определение пожара программной частью системы

Таким образом, система «Лесной Дозор» обладает рядом характеристик:

1. автоматическое обнаружение зарождающегося пожара;
2. учёт погодных условий;
3. радиус обзора из одной точки достигает 30 километров;
4. точные координаты с маленькой погрешностью;
5. возможность запрашивать данные со спутников;
6. возможность соотносить данные с других информационных систем;
7. получение результатов прямо на мобильные устройства;

1.3 Актуальность проекта

Согласно официальной статистике, собираемой МЧС России за 9 месяцев 2021 года было выявлено 314461 очагов возгораний. В результате чего погибло 5900 человек. На рисунке 1.5 показана диаграмма, на которой сравниваются данные за 2020 и 2021 год.

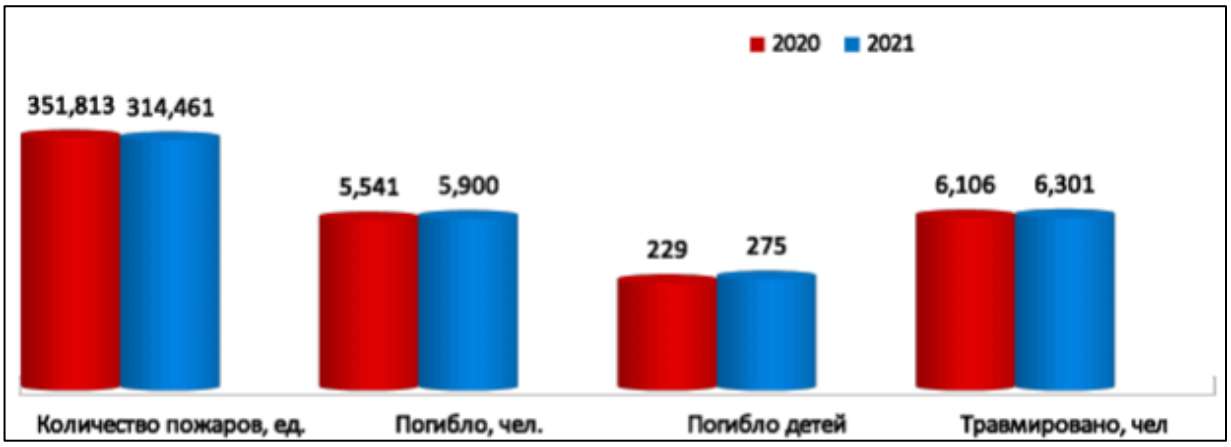

Рисунок 1.5 – диаграмма сравнения данных за 2020 и 2021 год

Можно заметить, что, несмотря на спад общего числа пожаров, во-первых, количество пожаров всё равно на высоком уровне, а во-вторых, во всех остальных областях потери наоборот увеличились. И это только в России

На рисунке 1.6 представлена диаграмма, разделенная по группам объектов, где возникали пожары.

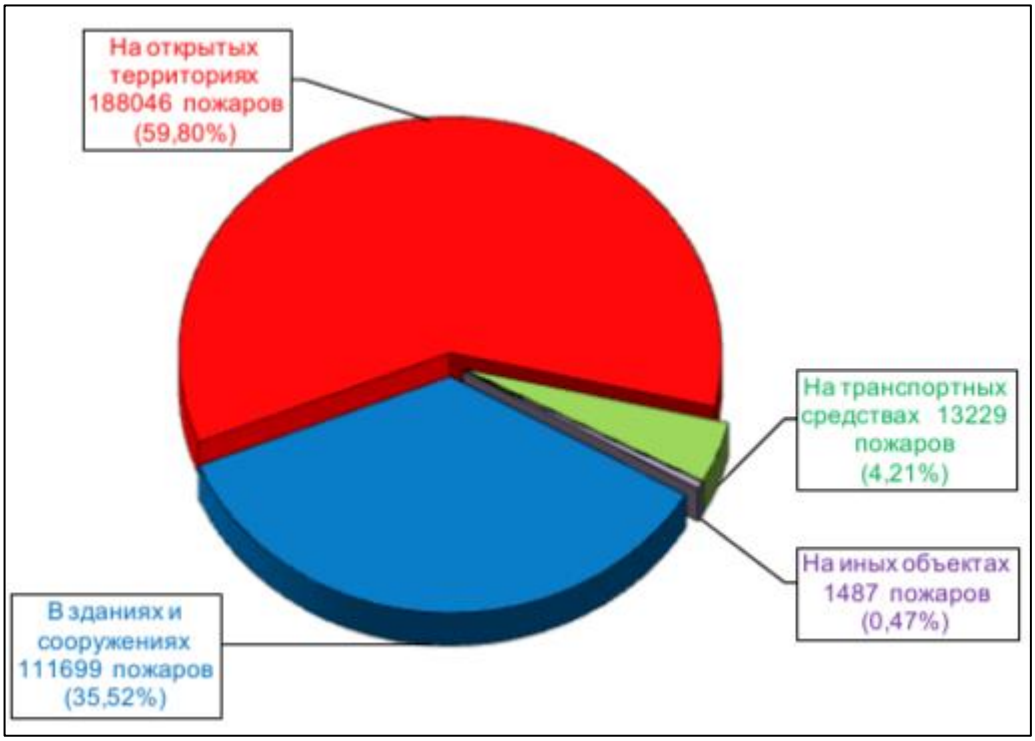

Рисунок 1.6 – диаграмма распределения пожаров по группам объектов.

Из этого можно сделать вывод, что лесные пожары занимают 60% от числа всех происшествий, то есть 188 тысяч очагов возгораний. Ущерб лесным массивам и окружающей среде от такого количества просто колоссальный.

Также большое значение имеет понимание того, из-за чего происходит возгорание – благодаря этому можно предполагать, где может возникнуть новый источник возгорания или же сделать так, чтобы такой ситуации не возникло. На рисунке 1.7 представлена диаграмма причин возникновения пожаров.

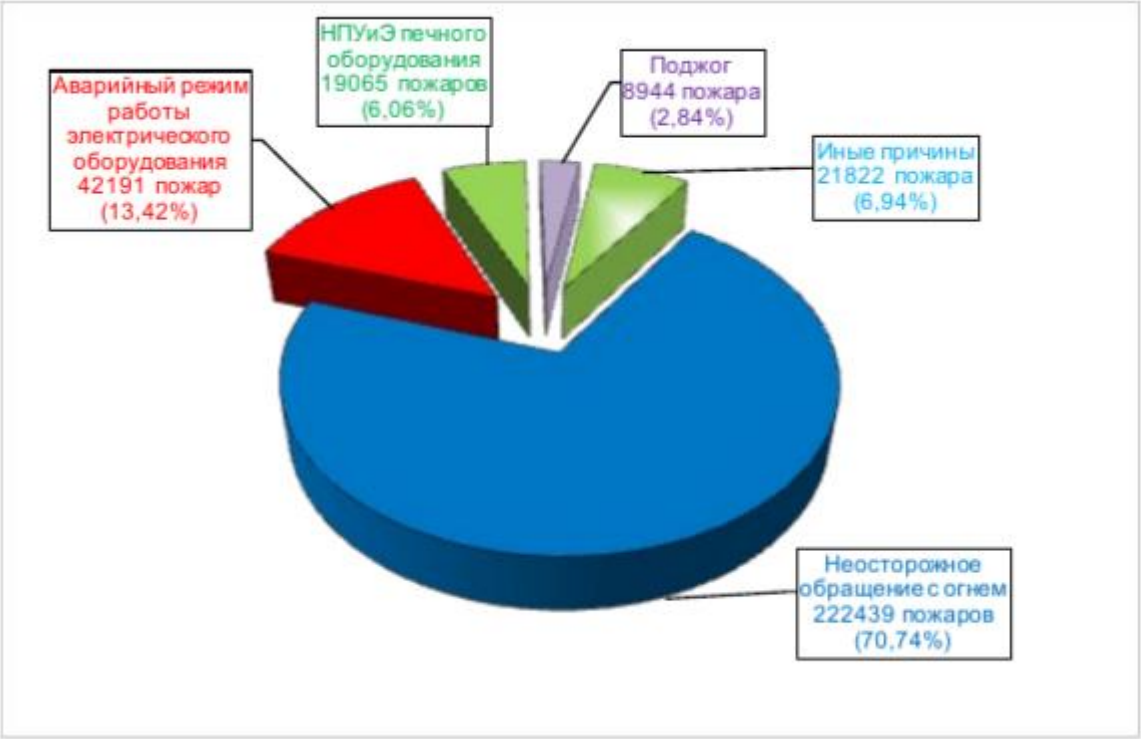

Рисунок 1.7 – диаграмма распределения пожаров по причинам

При правильном анализе причин можно определить возможное место возникновения пожаров.

1.4 Применение нейросетевого программирования для выявления возгораний на основе снимков со спутника

С развитием искусственного интеллекта появилось множество возможностей применить алгоритмы нейросетевого анализа для решения ряда различных задач. Одной из таких задач является определение возгорания на основе снимков со спутников. Это позволяет охватить огромную территорию для анализа и, при кооперации с источниками, при помощи которых можно получить качественные снимки из космоса, этот процесс можно полностью автоматизировать. Другой вопрос, каким образом можно идентифицировать пожар из космоса.

Одним из существующих подходов является сегментация изображения. Идея заключается в том, чтобы разделить изображение на отдельные части, которые соответствуют объектам, а затем классифицировать это изображение. Сегментация весьма сложный процесс, однако эта область весьма активно развивается. Есть несколько алгоритмов сегментации и большинство из них для классификации используют ключевые признаки, которые присутствуют на изображениях. Это может быть что угодно: цвет, форма, размер, текстура или всё в совокупности. На рисунке 1.8 и 1.9 показан пример сегментации изображения по ключевым признакам.

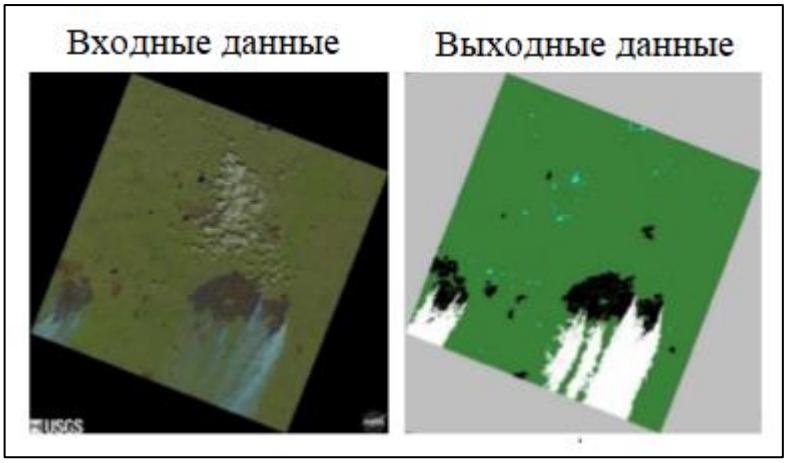

Рисунок 1.8 – первый пример сегментации изображения

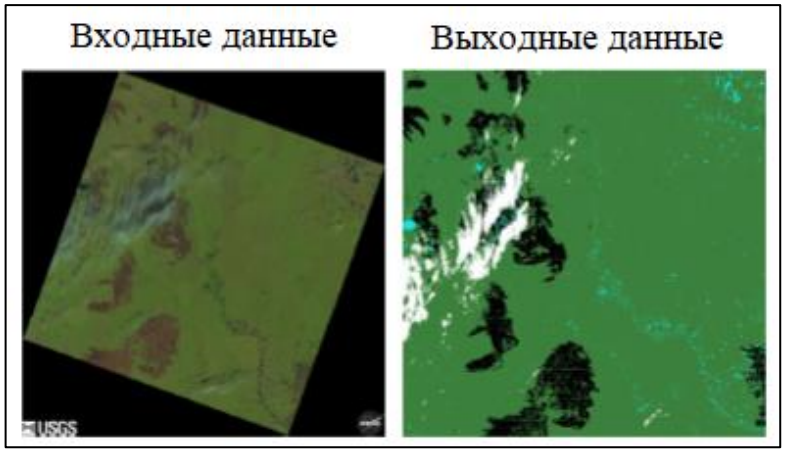

Рисунок 1.9 – второй пример сегментации изображения

Как можно заметить, нейронная сеть выделяет признаки, относящие к одному объекту одни цветом и то, что относится к другому объекту, соответственно,

другим цветом. Для этого примера зеленый цвет означает лес, синий – водоёмы, красный отвечает за огонь, белым показан дым, а чёрным – выгоревшая зона.

Некоторые алгоритмы используют особые точки – это те же самые ключевые признаки, но они сохраняют свои характеристики, несмотря на незначительную деформацию изображения: изменение масштаба или поворот изображения. Обычно это бывают достаточно уникальные признаки, такие как, например, угловые точки, места с резким изменением цвета, яркости. Самое сложное в этом подходе – это выбрать те точки, которые будут также присутствовать и на других изображениях и правильно описать эти особые точки, чтобы нейронная сеть могла соотнести эти точки. Для этого обычно используют следующие преобразования изображения: изменение яркости, изменение размера, изменение положения камеры, смещение и вращение. На рисунке 1.10 представлен один из методов – изменение яркости.

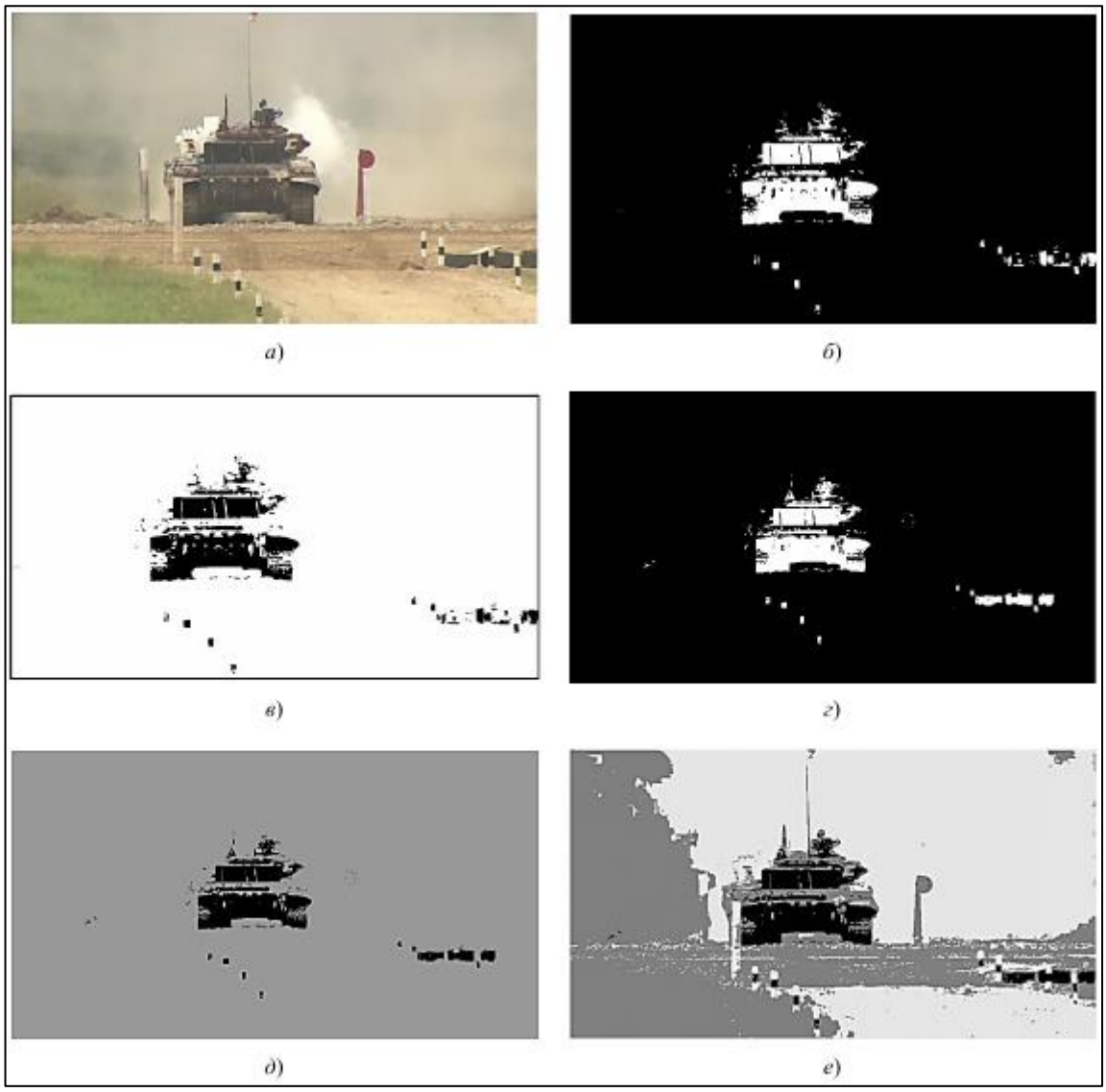

Рисунок 1.10 – поиск особых точек с помощью изменения яркости

На рисунке представлено 6 изображений. Они отличаются порогом яркости, которое использовали для преобразования изображения. Таким образом изображение «делится» на чёрное и белое, по которым нейронная сеть ищет особые точки. Если рассматривать конкретнее, то заметно, что первое изображение под буквой «а» – входное. На следующем изображении показан результат нижнего порога яркости – самые тёмные участки считаются признаком и при сегментации выделяются белым, а всё остальное считается «задним фоном». Как противопоставление на изображении под буквой «в» представлен результат верхнего порога – можно сказать, что произошла инверсия. Следом идёт «объединение» двух предыдущих способов – несмотря на то, что кажется, будто особых изменений не видно, если приглядеться, то можно заметить, что

сегментированное изображение более «плотное» и контур объекта выделен чётче. Под буквой «д» сегментация выполнена немного другим способом – вместо того, чтобы концентрироваться именно на объекте, нейронная сеть обращает внимание на фон, нивелируя все его детали, чтобы оставить только объект. Следующее применение сегментации является побочным от предыдущего – цель такого преобразования выделить все объекты с различной яркостью.

1.4.1 Снимки со спутников

Человечество не стоит на месте и стремится к открытиям во всех областях. Одной из самых интригующих областей является космос и, конечно же, наша планета, как часть солнечной системы. Таким образом можно следить за тем, что происходит с Землёй. С помощью спутников, летающих на орбите, можно собирать колоссальное количество данных. Но куда важнее найти достойное применение этим данным.

Существует множество проектов, которые собирают петабайты данных для статистики – на серверах хранятся снимки Земли за несколько десятилетий. На их основе можно проследить за изменениями, которые случились с поверхностью нашей планеты, например, какая часть ледников растаяла или как сильно пострадали леса от пожаров.

Может сложиться впечатление, что такие данные находятся в закрытом доступе, но это не так. Люди уже давно пользуются технологией просмотра снимков со спутника – интернет-карты, по которым можно построить маршрут между городами или узнать на какой автобус стоит сесть, чтобы доехать из одной точки города в другую. Однако эти снимки ориентированы на населенные пункты, что является незначительной частью, особенно по сравнению со всей остальной поверхностью планеты. Одним из сервисов, предоставляющих данные со всей Земли специально для исследований и анализа является Google Earth Engine. Этот сервис обладает многопетабайтным каталогом спутниковых изображений и наборов данных, обновляемый ежедневно, с помощью которого можно проследить за изменениями, произошедшими из-за нашего влияния за целых 37 лет! Но и это

не всё. Earth Engine предоставляет не только «простые» снимки, но и более узконаправленные, для которых требуется специальное оборудование. К примеру, в открытом доступе есть данные о климате: поток облаков в реальном времени, изображения с концентрацией метана, даже набор данных для мониторинга состава атмосферы, как показано на рисунках 1.11 – 1.13!

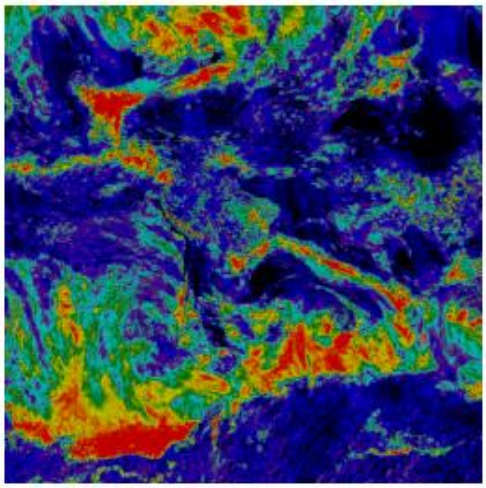

Рисунок 1.11 – поток облаков в режиме реального времени

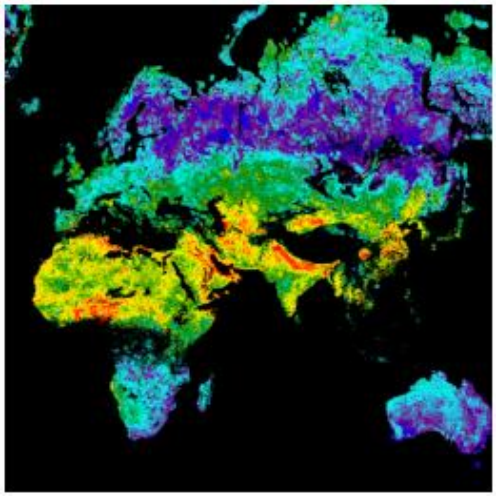

Рисунок 1.12 – концентрация метана на Земле

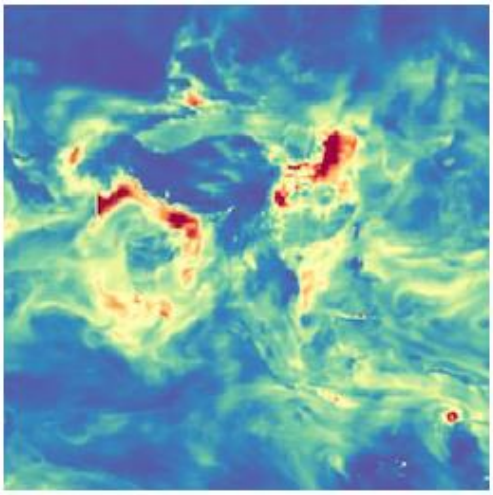

Рисунок 1.13 – набор данных для мониторинга состава атмосферы

Но большую часть данных составляют изображения дистанционного зондирования Земли, то есть изображения поверхности Земли такой, какая она есть. Большие объемы данных поставляются в свободный доступ примерно раз в две недели. Каждое изображение имеет разрешение 30 метров, как показано на рисунке 1.14.

Рисунок 1.14 – пример снимка со спутника

Также Google разработали Google Earth Engine. Это облачная платформа с доступом к высокопроизводительным вычислительным ресурсам, благодаря

которым обработка больших наборов данных значительно упрощается. Доступ к этой платформе осуществляется через интерфейс прикладного программирования и связанную с ним среду разработки, что позволяет быстро создавать алгоритмы и визуализировать их.

Другим представителем похожего сервиса является Planet Monitoring. По сравнению с последним источником база данных намного меньше – данные начали собирать только с 2009 года. Однако, несмотря на это, главная особенность данного сервиса в том, можно запросить самые свежие данные, которые будут почти в реальном времени – около 200 «спутников» вращаются вокруг Земли каждые 90 минут на протяжении всего дня и с каждым годом они обновляются. Также, для сравнения с прошлым ресурсом, Planet Monitoring представляет данные с расширением 3,7 метров, что позволяет проводить более точные исследования. Пример представлен на рисунке 1.15.

Рисунок 1.15 – пример изображения, поставляемого Planet Monitoring

1.4.2 Нейронные сети

Нейронная сеть представляет собой математическую модель и её воплощение, построенную по принципу организации и функционирования биологических нейронных сетей. Эта сеть состоит из нейронов, которые

предназначены для обработки поступившего сигнала и его вывода дальше по сети. Также нейроны объединяются в структуру – слой, созданный для определенных целей. Существуют три вида слоёв:

1. Входной слой – как следует из названия, с этого слоя всё начинается. В сеть подаются какие-то данные, входной слой их обрабатывает и передаёт следующему слою.
2. Скрытый слой – в зависимости от поставленной задачи, сети может потребоваться больше вычислительной мощи, которую, как раз и обеспечивает скрытый слой. Принимая данные из входного слоя и перемножая их на специальные весовые коэффициенты, скрытый слой получает новые значения и передаёт их дальше – к своему собрату, ещё одному скрытому слою, если это требуется, либо же к выходному слою.
3. Выходной слой – заключительный слой нейронной сети, который, можно сказать, подводит итог работы всех прошлых слоёв и передаёт окончательный результат сети обратно в систему.

На рисунке 1.16 представлена упрощенная схема нейронной сети с входным, скрытым и выходным слоем.

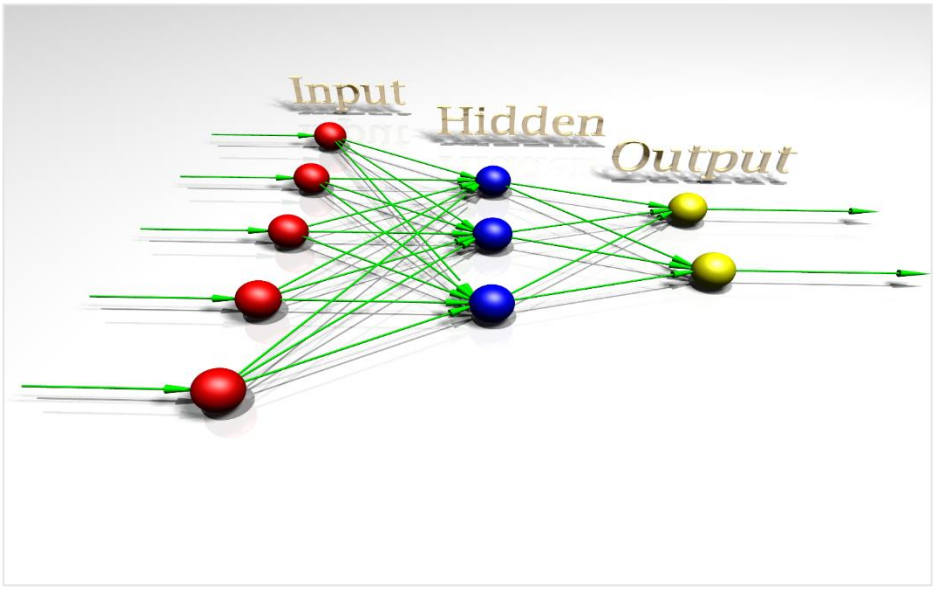

Рисунок 1.16 – слои нейронов

В описании скрытого слоя был упомянут весовой коэффициент – это условная сила связи между нейронами разных слоёв, но в математическом виде это

просто какое-то значение. Изначально оно устанавливается случайным образом и только после первого обучения «веса» изменяют для более точной настройки нейронной сети.

1.4.3 Свёрточные нейронные сети

Свёрточная нейронная сеть – более узконаправленная нейронная сеть. За счёт своей топологии эта сеть показывает наилучшие результаты в области работы с изображениями. Огромным отличием от стандартной нейронной сети является концепция «общих весов» – каждый нейрон в слое обладает фиксированным весом. Это помогает при обработке изображений и даёт устойчивость к некоторым «искажениям» изображений, что повышает точность распознавания на 10-15% по сравнению с обычными нейронными сетями.

Принцип работы свёрточной нейронной сети заключается в следующем: каждое изображение разбивается на 3 канала: красный, синий, зеленый. Пример изображения, представленного в разных каналах, показано на рисунке 1.17.

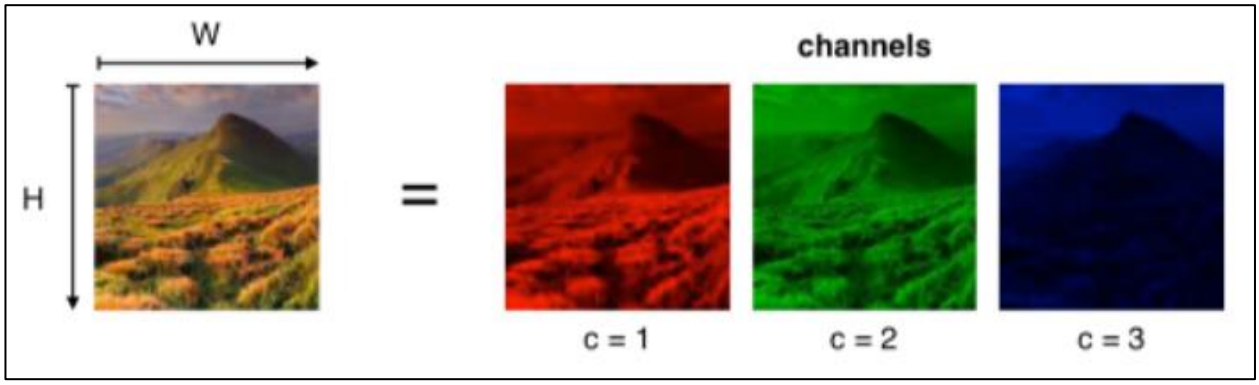

Рисунок 1.17 – изображение, разбитое по разным каналам

Затем входной слой вычисляет значение интенсивности канала в каждом пикселе для формирования начальной карты признаков, как представлено на рисунке 1.18.

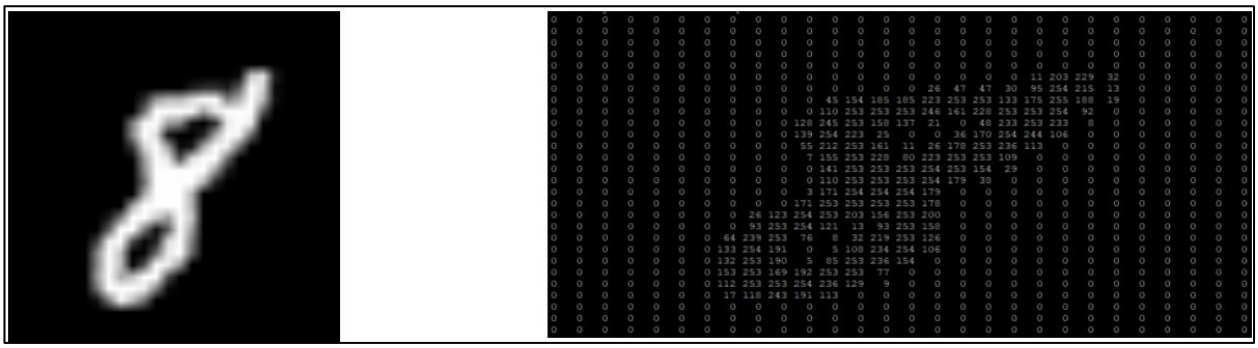

Рисунок 1.18 – интенсивность каналов в пикселе

После этого результат идёт дальше, в свёрточный слой. Этот слой отвечает за выделение признаков на основе входных данных и формировании новой карты признаков с помощью фильтра – тех самых «общих весов». Фильтр проходит по всей области входной карты и находит определенные признаки объектов. Например, если сеть обучали на распознавании лиц, то один из фильтров выделит признаком линию глаза, рта, брови или носа. Размер фильтра обычно берут в пределах от 3x3 до 7x7, чтобы суметь выделить признаки и не перегрузить сеть. Также размер фильтра выбирается таким, чтобы размер карт сверточного слоя был четным, это позволяет не терять информацию при уменьшении размерности в подвыборочном слое. На рисунке 1.19 продемонстрирована работа слоя свёртки: по входной карте признаков проходит фильтр (ядро) размерностью 3x3 и получается новая карта признаков.

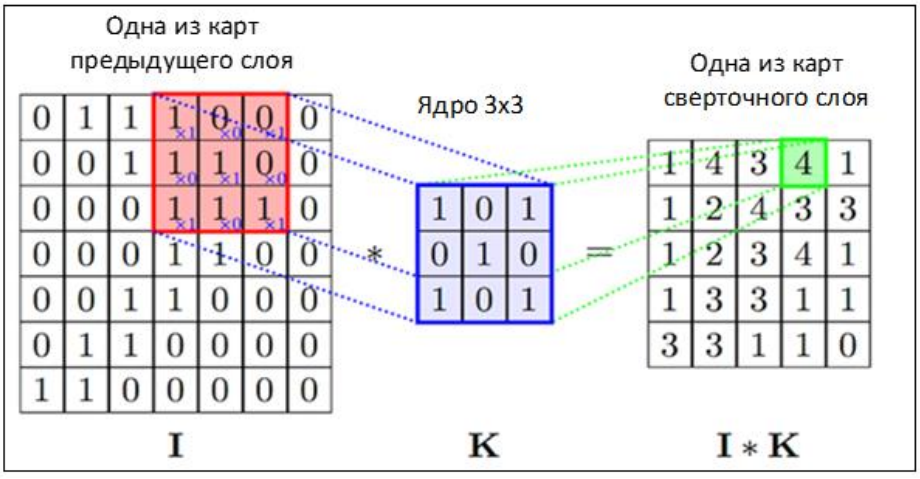

Рисунок 1.19 демонстрация работы слоя свёртки

Обычно после слоя свёртки идёт подвыборочный слой, также называемый слоем пуллинга. Его цель состоит в уменьшении входной карты признаков. Если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного и это помогает избежать переобучения модели – так снижается вероятность того, что сеть ошибочно выделит ненужный признак. Чаще всего каждая карта имеет фильтр размером 2x2, что позволяет уменьшить предыдущие карты сверточного слоя в 2 раза. Операция подвыборки по максимальному значению показана на рисунке 1.20.

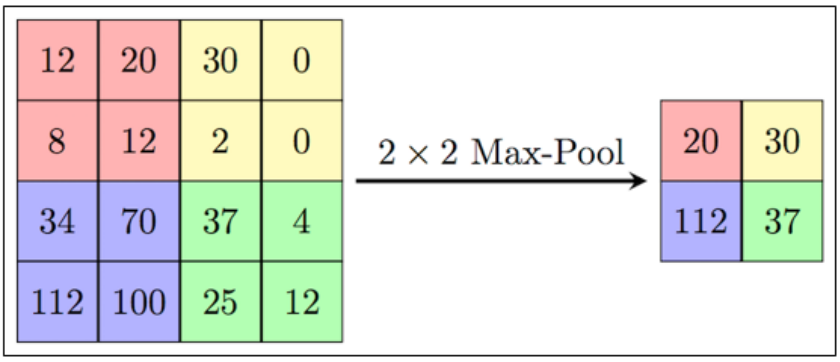

Рисунок 1.20 демонстрация работы слоя подвыборки

Если изображение большое и признаки не так очевидны, то для лучшего результата слои свёртки и подвыборки повторяются, но в конце концов завершаются его полносвязным выходным слоем. Этот тип слоя –обычный многослойный персептрон, отвечающий за классификацию собранных признаков.

1.4.4 Функции активации

Немаловажную роль в нейронной сети играет функция активации – математическое выражение, которое определяет ответ нейрона на поступившие в него сигналы. В зависимости от этой функции разнится диапазон принимаемых значений, что влияет на точность и скорость обучения нейронной сети. Наиболее часто используют три функции.

Сигмойд – самая распространенная функция активации. Уровень активации нейрона колеблется: от отсутствия результата, представленного значением 0, до

полного «попадания», то есть 1, как показано на рисунке 1.21. Подходит для задач «чёткой» классификации.

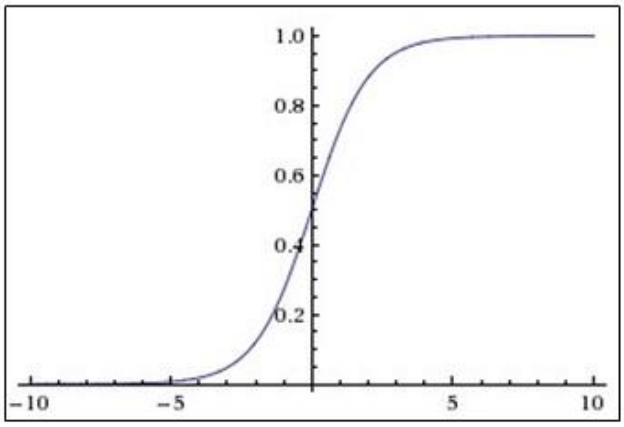

Рисунок 1.21 – график принимаемых значений функции Сигмойд

Гиперболический тангенс является скорректированной сигмоидной функцией, как можно заметить по рисунку 1.22.

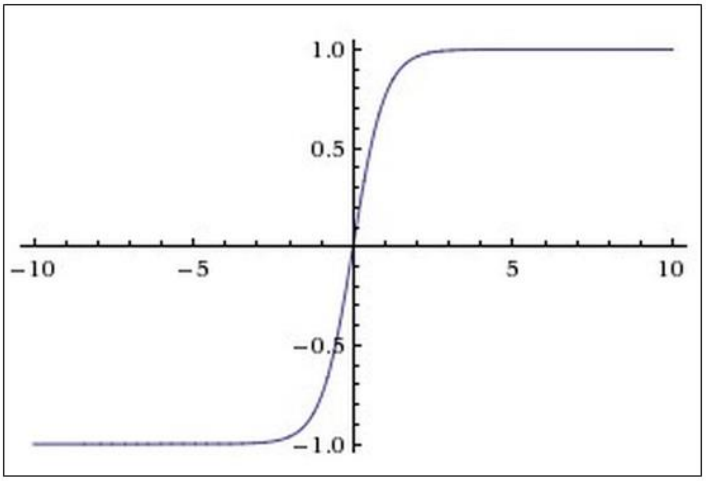

Рисунок 1.22 график принимаемых значений функции Tanh

ReLu простая функция, поэтому она менее требовательна к вычислительным ресурсам и более быстрая, что весьма важно в нагроможденных и глубоких нейронных сетях. Применение ReLU существенно повышает скорость сходимости стохастического градиентного спуска по сравнению с сигмоидой и гиперболическим тангенсом. Диапазон принимаемых значений представлен на рисунке 1.23.

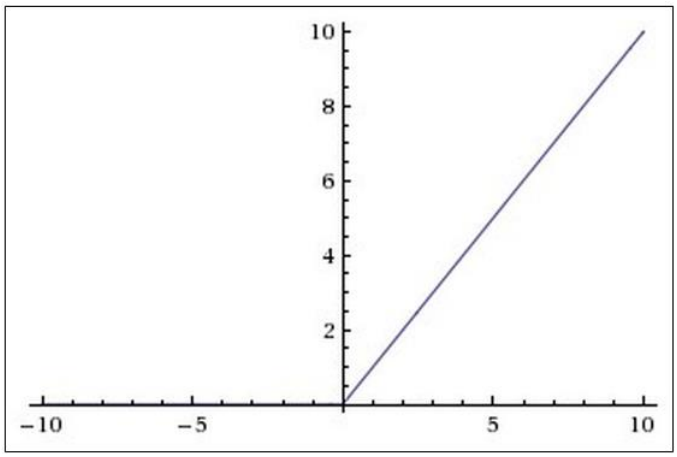

Рисунок 1.23 график принимаемых значений функции RELU

На основании данных первого раздела был составлен подробный план разработки программы дипломного проекта.

2 Проектный раздел

2.1 Модель «Как надо»

На основе плана разработки, была спроектирована концептуальная модель «Как надо». Она отражает работу программы с разработанной и обученной нейронной сетью. Для составления модели была использована методология функционального моделирования IDEF0, предназначенная для формализации и описания бизнес-процессов.

На рисунках 2.1 – 2.4 представлена модель IDEF0.

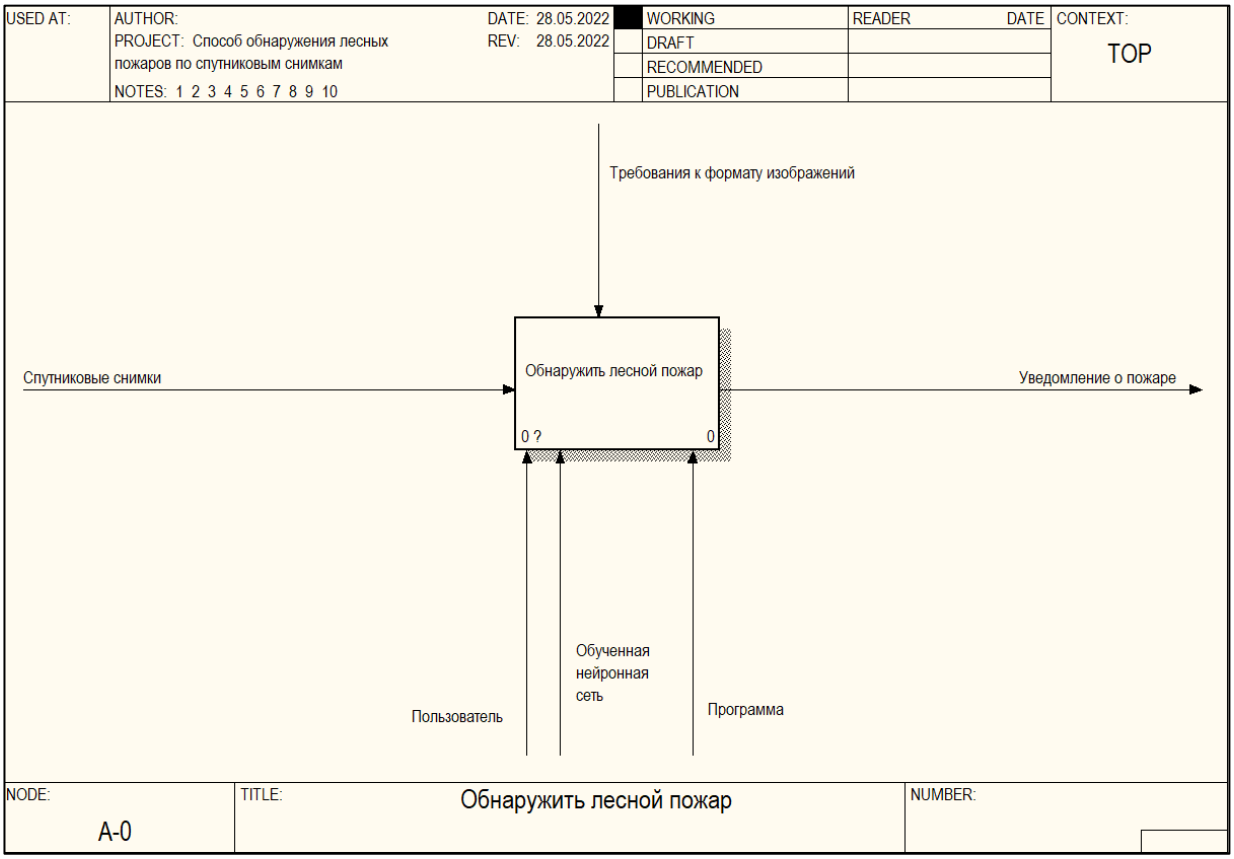

Рисунок 2.1 Основной блок модели «Как надо»

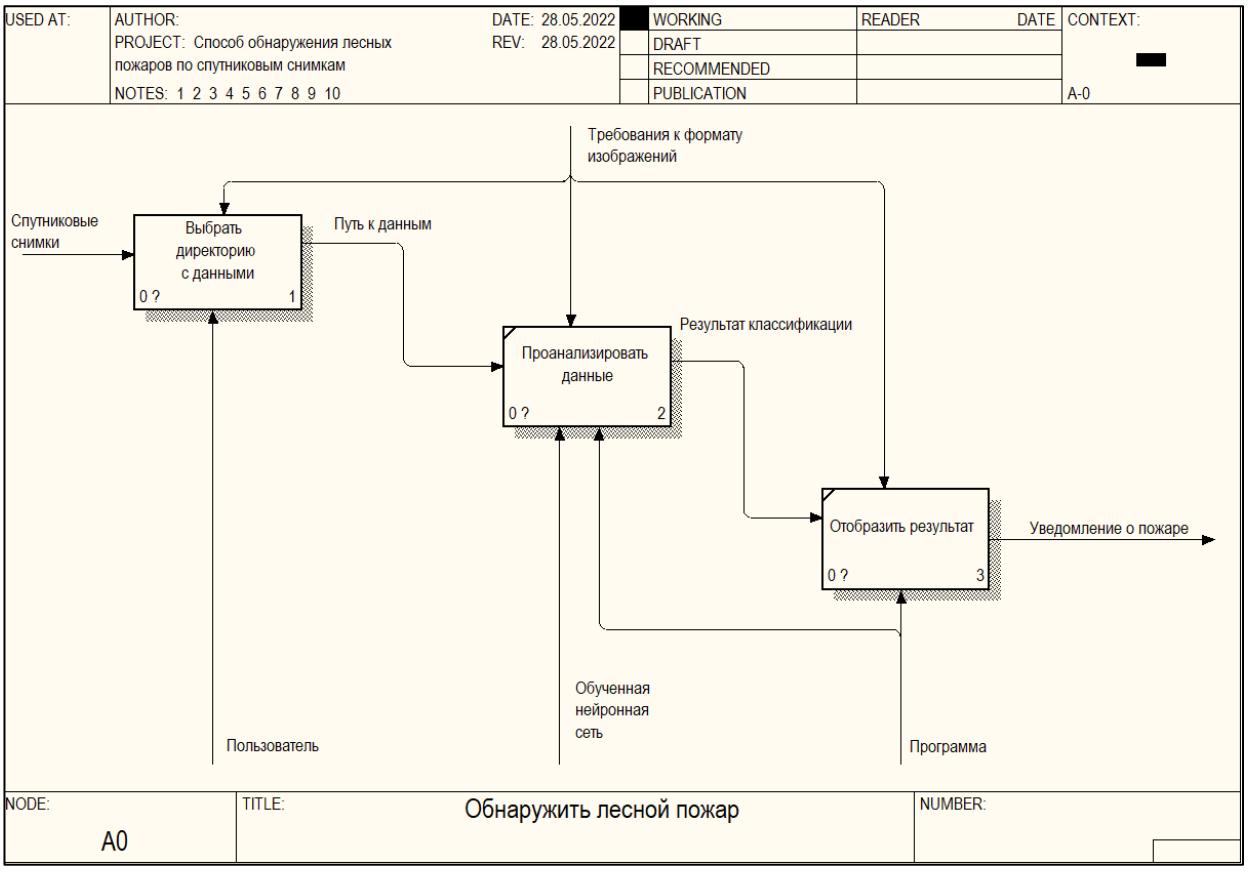

Рисунок 2.2 декомпозиция основного блока модели

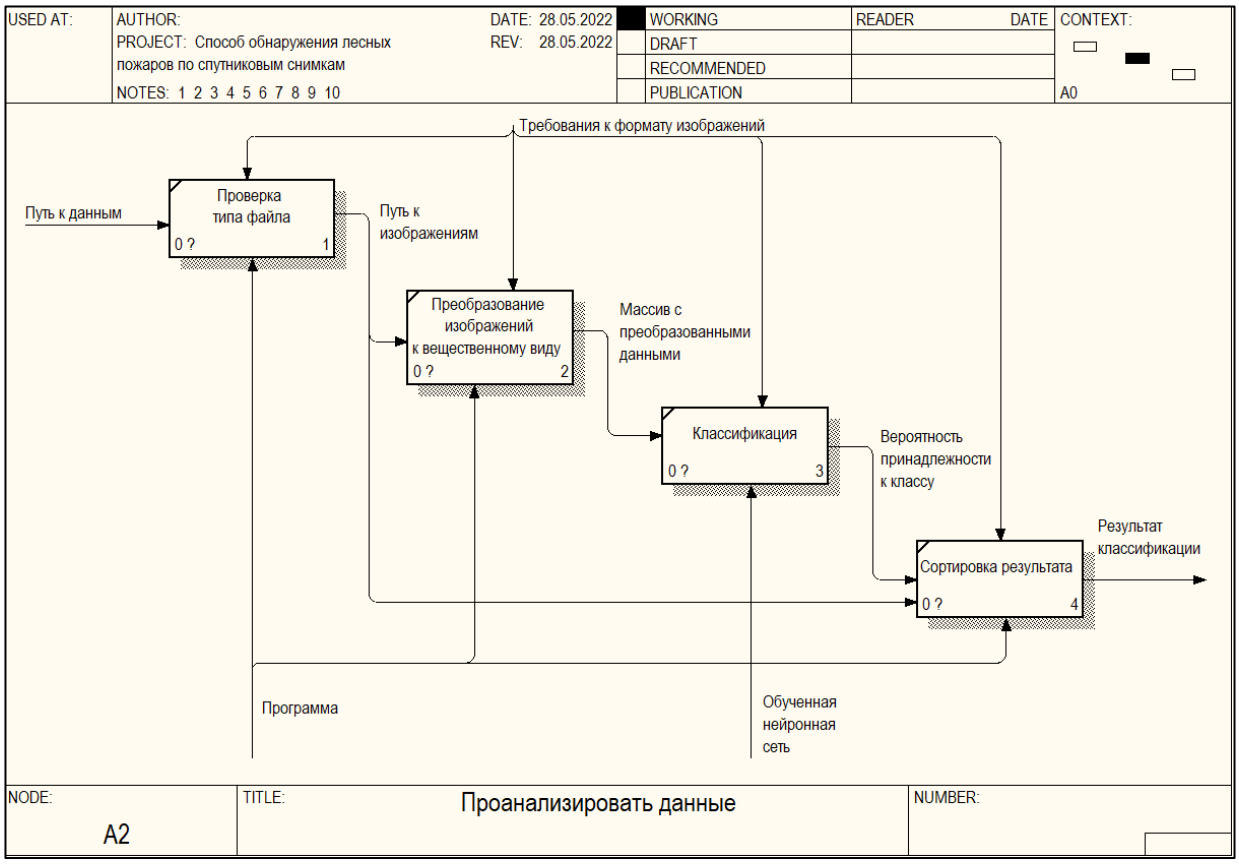

Рисунок 2.3 декомпозиция блока «Проанализировать данные»

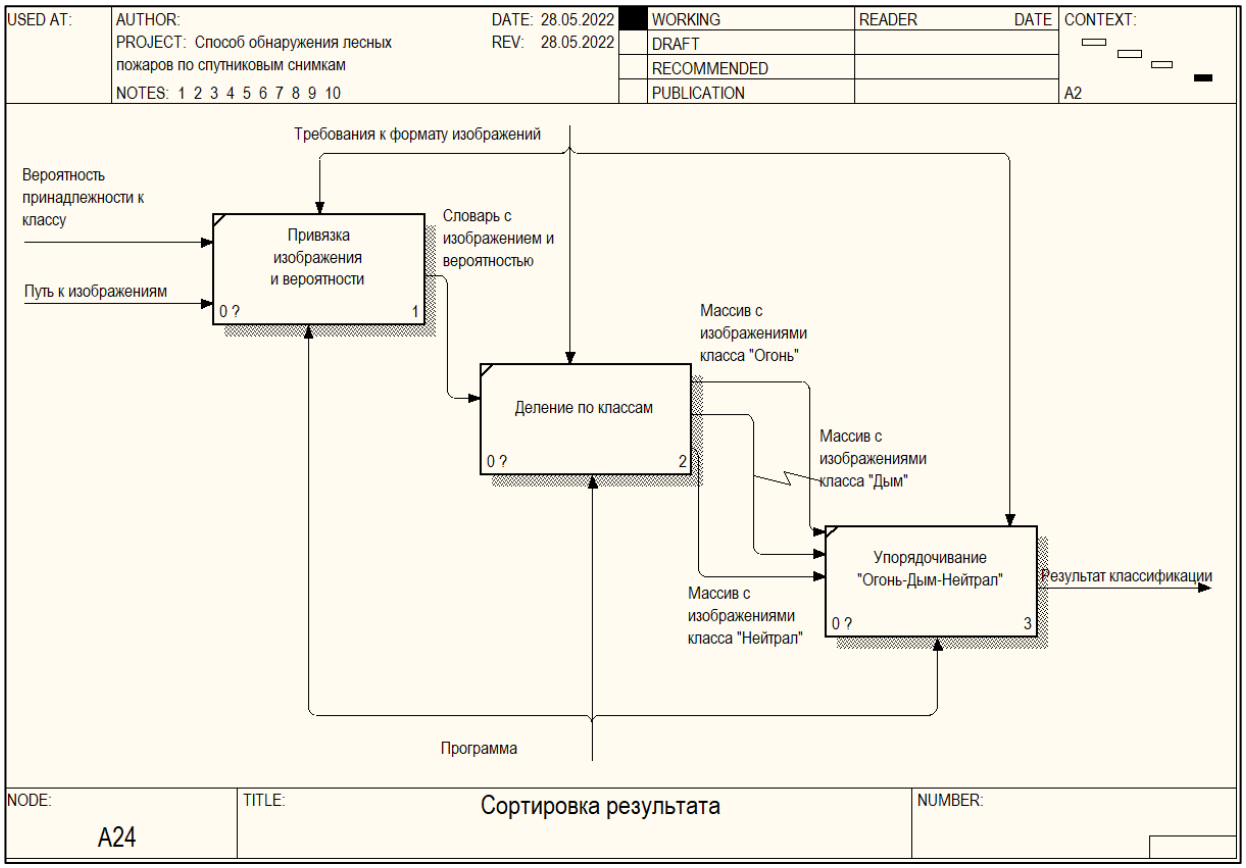

Рисунок 2.4 декомпозиция блока «Сортировка результата»

На основе декомпозиции был разработан алгоритм программного приложения.

2.2 Алгоритм работы программного приложения

Программное приложение стоит из следующих частей:

1. хранилище;
2. модуль подготовки данных;
3. обученная нейронная сеть.

Хранилищем может быть что угодно: база данных, облачное хранилище или же жёсткий диск компьютера пользователя, главное, чтобы там содержались данные, пригодные для обработки.

Модуль подготовки данных обрабатывает предоставленные изображения из хранилища.

Затем модуль передаёт эти изображения в нейронную сеть, где происходит преобразование изображения к виду входной карты признаков и эта карта признаков проходит через все слои нейронной сети, выделяя признаки.

И, наконец, результат классификации передаётся в приложение.

Схема описанного алгоритма показана на рисунке 2.5.

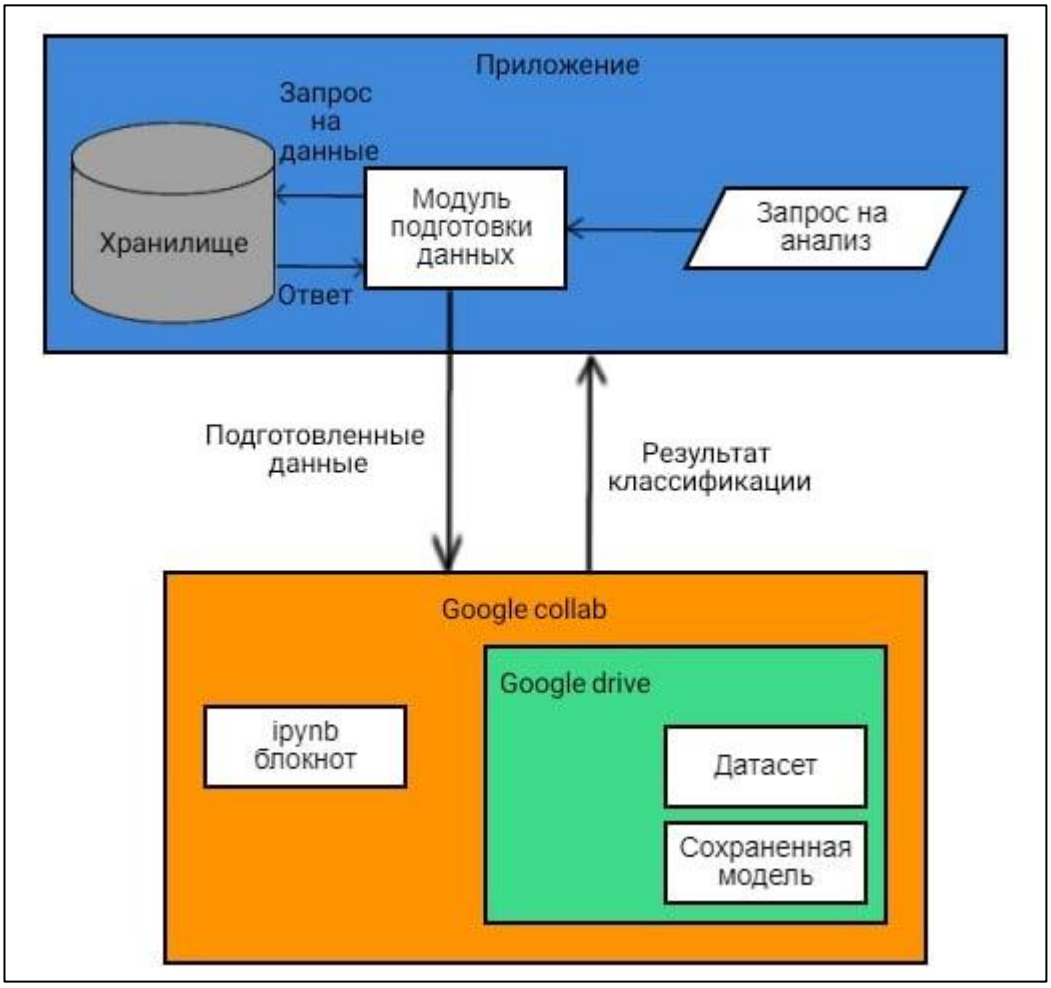

Рисунок 2.5 схема взаимодействий частей программы

2.3 Алгоритм подготовки данных для обучения нейронной сети

Предполагается, что существует три класса, которые может идентифицировать проектируемая нейронная сеть.

Класс «Fire» – в качестве признаков выступает интенсивный цвет, резко выделяющийся на фоне. К сожалению, содержит признаки других классов, из-за чего могут возникать ошибки в идентификации, особенно если этих признаков будет больше. Чаще всего этот класс дополняет и дополним классом «Smoke».

Класс «Smoke» – также, как и предыдущий класс, выделяется на фоне, но цвет менее «яркий». В некоторых случаях нейронная сеть может идентифицировать его как другой, последний класс «Neutral».

Класс «Neutral» – можно сказать, что этот класс существует только для того, чтобы обучить нейронную сеть методом обратной ошибки. Если говорить грубо, то основным признаком этого класса будет отсутствие признаков других классов.

Алгоритм подготовки начинается с загрузки данных – для данного проекта это архив, который после этого распаковывается и в котором изображения разделены на три класса («Fire», «Neutral», «Smoke»), по соответствующим директориям. Данные обязательно должны быть разделены по классам, чтобы правильно сформировать датасет. На рисунке 2.6 показана структура директорий.

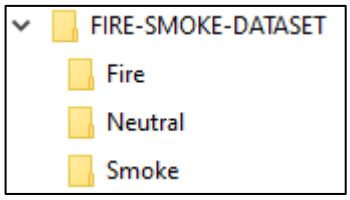

Рисунок 2.6 – структура архива с изображениями

Следующий этап – это проверка формата изображений. Для использования функции преобразования изображений в датасет нужно, чтобы входные данные соответствовали формату, который поддерживается TensorFlow, иначе датасет не сформируется. Это также автоматически отбросит файлы, которые могли по ошибке попасть в архив, и изображения, формат которых написан ошибочно. На рисунке 2.7 представлен результат работы такой проверки.

```
/content/FIRE-SMOKE-DATASET/Smoke/xqvZdn2U1TbcTY9WLFMq8Q7d9dYH9VodoOaTdZiy25HQ.jpg is not an image
/content/FIRE-SMOKE-DATASET/Fire/raging-pine-tree-fire-across-260nw-114369937.jpg is not an image
/content/FIRE-SMOKE-DATASET/Fire/загруженное.jpg is not an image
/content/FIRE-SMOKE-DATASET/Neutral/les_hvojnyj_vid_sverhu_149423_300x255.jpg is not an image
/content/FIRE-SMOKE-DATASET/Neutral/image_22.jpg is a tiff, not accepted by TensorFlow
/content/FIRE-SMOKE-DATASET/Neutral/image_77.jpg is a tiff, not accepted by TensorFlow
/content/FIRE-SMOKE-DATASET/Neutral/image_863.jpg is a tiff, not accepted by TensorFlow
/content/FIRE-SMOKE-DATASET/Neutral/image_72.jpg is a tiff, not accepted by TensorFlow
```

Рисунок 2.7 – проверка формата изображений

После этого нужно привести все изображения к единому размеру. Это обусловлено тем, что для преобразования изображений в датасет указывается

фиксированный размер. Если размер будет больше указанного, то нейросеть может не «захватить» участок, где будут находится ключевые признаки и, соответственно, не сможет обучится правильно. В проектируемой нейронной сети фиксированный размер составит 256 пикселей. Это наиболее оптимальный размер – для обучения нейронной сети желательно, чтобы размер был как можно меньше, но с учётом специфики задачи уменьшение изображения ещё сильнее снизит точность классификации из-за того, что признаки «слепятся» в одну кучу. На рисунках 2.8 – 2.13 представлены примеры сжатия изображений в формате «до» и «после».

Рисунок 2.8 – изображение класса «Smoke» до сжатия

Рисунок 2.9 – изображение класса «Smoke» после изменения

Рисунок 2.10 – изображение класса «Fire» до изменения размера

Рисунок 2.11 – изображение класса «Fire» после сжатия

Рисунок 2.12 – изображение класса «Neutral» до масштабирования

Рисунок 2.13 – изображение класса «Neutral» после изменения размера

Следующим шагом идёт преобразование изображений в датасет. В данном случае использовалась специальная функция, встроенная в Keras. Благодаря ей изображения преобразуются в тензоры, содержащие пакеты изображений, а также выделяются классы, по которым нейронная сеть будет классифицировать изображения. Для обучения выделяются два типа данных – «тренировочные», те, на которых нейронная сеть будет обучаться и «проверочные», те, на которых будут проверять работоспособность и точность сети. На рисунке 2.14 представлен результат преобразования изображений в датасет.

```
Found 3031 files belonging to 3 classes.
Using 2425 files for training.
Found 3031 files belonging to 3 classes.
Using 606 files for validation.
```

Рисунок 2.14 – преобразование изображений в датасет с выделением классов

Завершающим пунктом в подготовке данных является нормализация. Так как при обработке rgb изображений нейронная сеть собирает карту признаков используя значения в диапазоне от 0 до 255. Примеры входной карты признаков показаны в приложении А. Однако, это весьма большой диапазон, поэтому принято приводить входную карту признаков к диапазону от 0 до 1.

На рисунке 2.15 представлен алгоритм подготовки данных

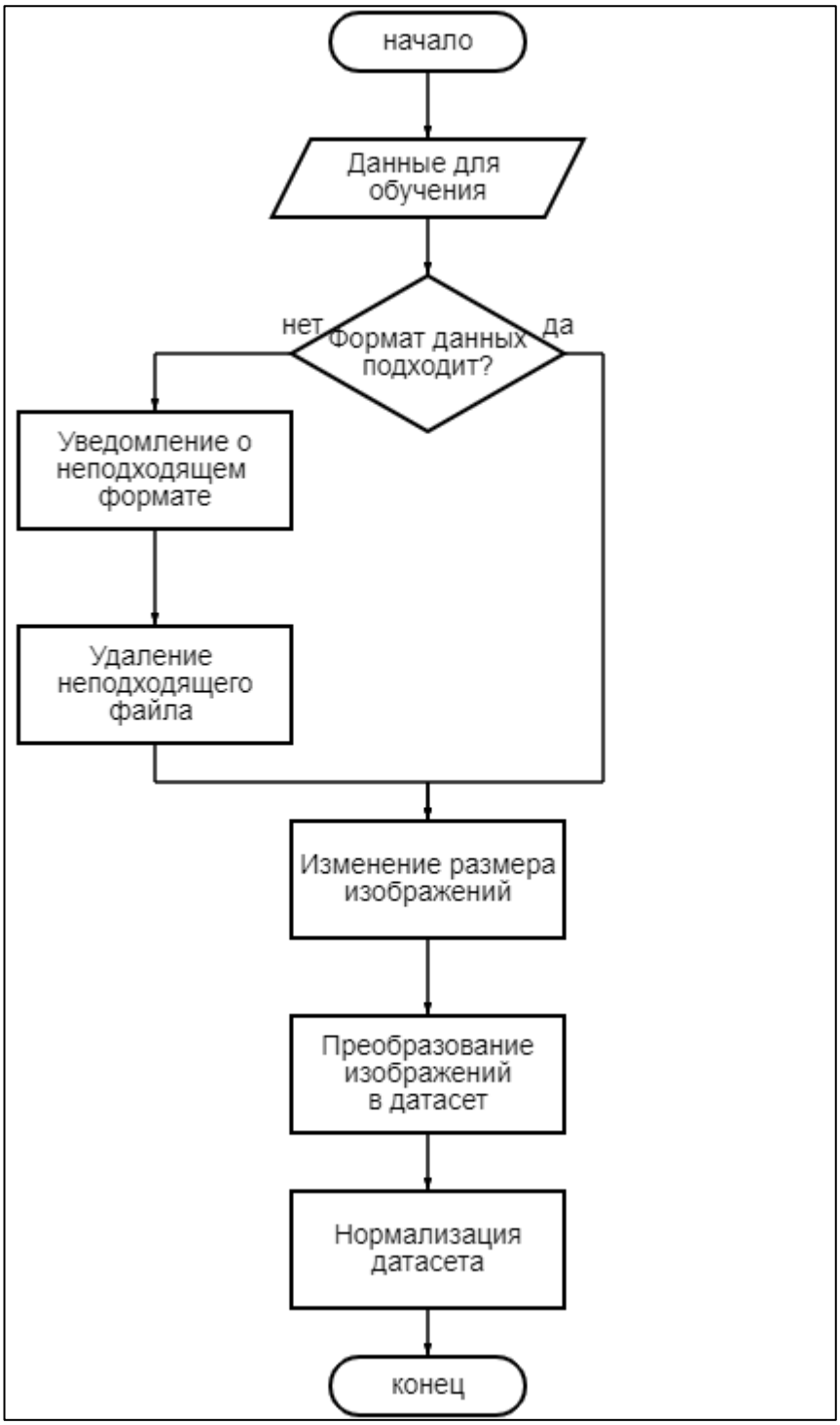

Рисунок 2.15 – алгоритм подготовки данных

2.4 Архитектура модели нейронной сети

Данная модель нейронной сети использует искусственное увеличение данных. Самый первый слой предназначен для нормализации начальной карты признаков. После этого идут три пары слоёв «свёртка-пуллинг». Затем слой «выброса» данных для того, чтобы уменьшить вероятность переобучения.

Следующий слой преобразовывает карты признаков в «вектора». И, наконец, полносвязный слой, который классифицирует изображение на основе полученных «векторов». Полная архитектура представлена на рисунке 2.16.

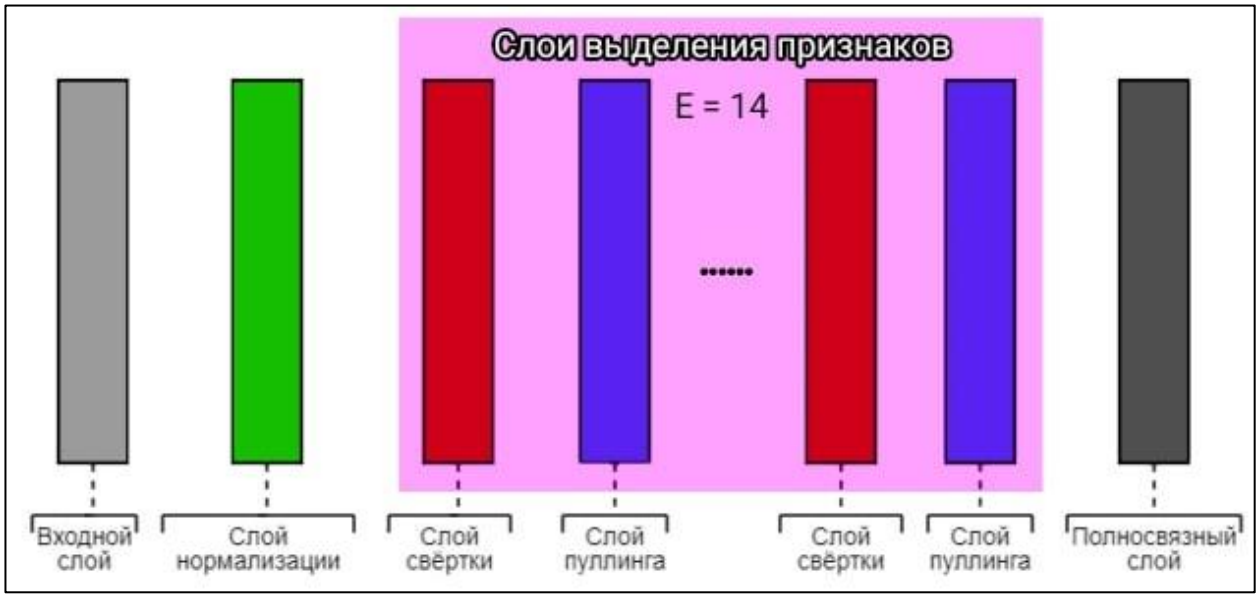

Рисунок 2.16 архитектура модели нейронной сети

2.5 Метод обучения нейронной сети

Выбранный метод относится к подразделу обучения с учителем, и подразумевает под собой наличие отсортированного набора данных (Dataset) где каждое изображение соответствует одному из трёх классов.

Метод заключается в следующем: в нейронную сеть подают данные, сеть выделяет признаки и только на их основе классифицирует изображение. Затем нейронная сеть вычисляет ошибки на каждом слое, опираясь на искомый результат из датасета. И, в конце, применяет все полученные знания. Важно иметь достаточно большой, и точный дата сет.

Недостаток данного метода в том, что сложно подобрать хороший и большой набор данных.

2.6 Алгоритм обучения нейронной сети

Для обучения нейронной сети нужны подготовленные данные – это первый шаг. От этих данных зависит точность, с которой нейронная сеть будет классифицировать изображения.

Следующий этап помогает в том случае, если данных для обучения не так много и они могут повторяться, что, обычно, приводит к переобучению модели и зацикливании нейронной сети на побочных признаках. Искусственное увеличение данных использует существующие примеры из обучающих данных для создания дополнительных данных путем случайного преобразования. На рисунках 2.17 – 2.19 показано, как на основе одного изображения и с помощью различных наклонов и поворотов были созданы новые изображения.

Рисунок 2.17 – пример искусственного увеличения данных для класса «Smoke»

Рисунок 2.18 – увеличение количества изображений класса «Fire»

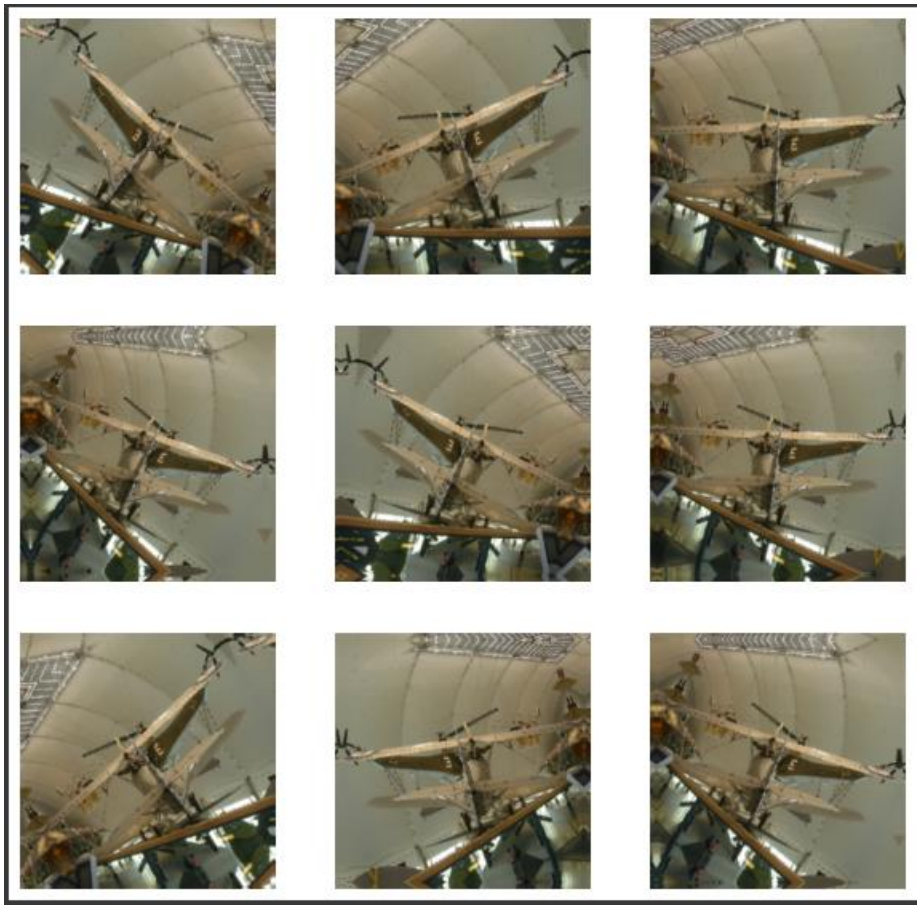

Рисунок 2.19 – искусственное увеличение данных класса «Neutral»

Следующий шаг включает в себя создание модели на основе спроектированной архитектуры.

Обучение модели – заключительный и самый продолжительный этап. Во время этого этапа данные проходят через каждый слой модели и обучается в соответствии с выбранным методом обучения.

Краткий алгоритм обучения представлен на рисунке 2.20.

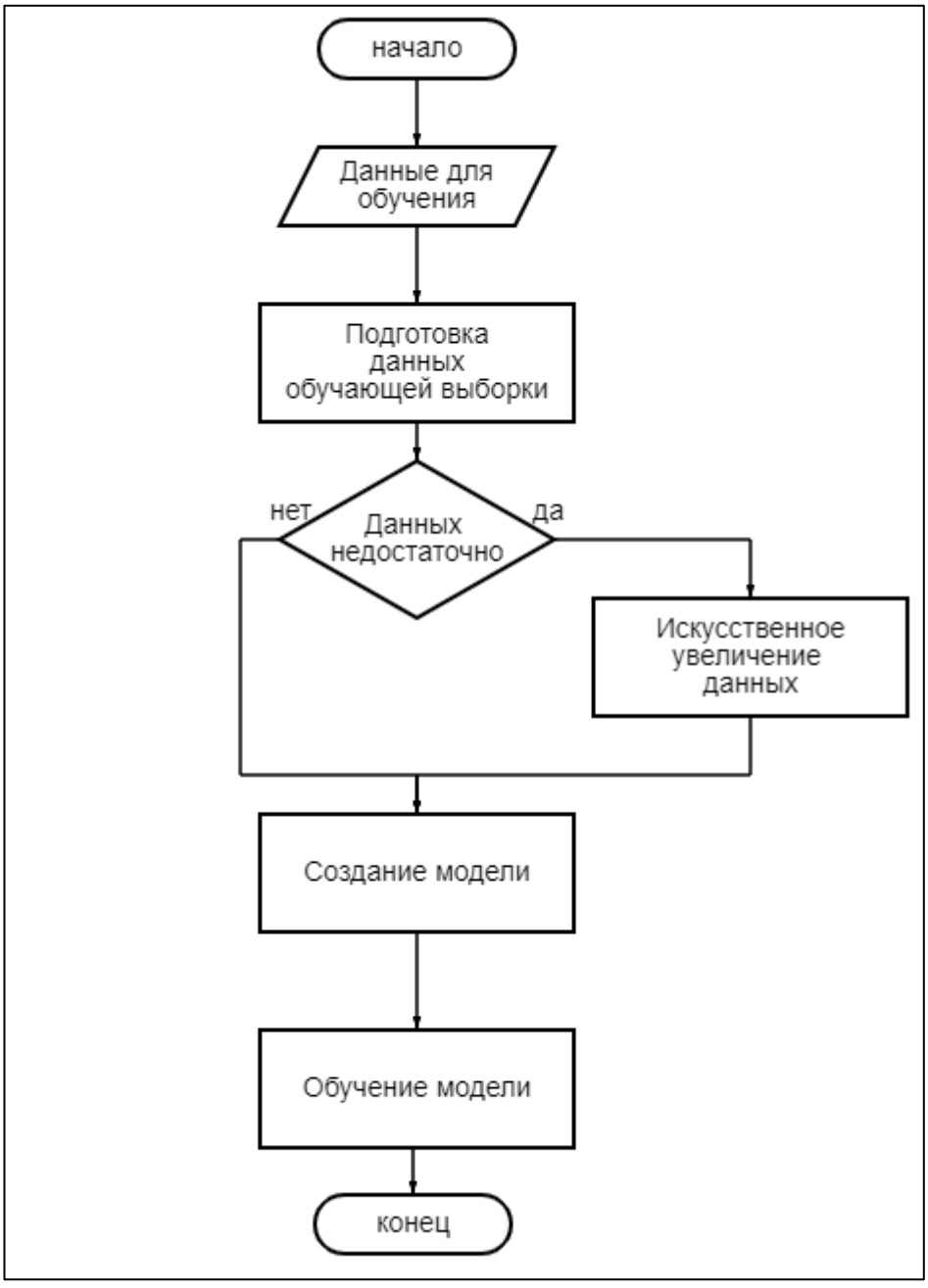

Рисунок 2.20 –алгоритм обучения нейронной сети

3 Программный раздел

3.1 Сравнение языков программирования

3.1.1 Python

Python – широко применяемый в сфере нейронных сетей язык программирования из-за его обширных средств для работы с нейронными сетями, простоты в использовании и гибкости. Синтаксис Python позволяет легко реализовать различные алгоритмы, что также позволяет сократить время разработки по сравнению с другими доступными языками программирования. Другие функции включают возможность тестирования алгоритмов без необходимости их реализации. Как язык высокого уровня Python поддерживает объектно-ориентированные, функциональные и процедурно-ориентированные стили программирования.

3.1.2 C++

C++ один из самых известных и распространенных языков программирования, который был разработан для баланса производительности, эффективности и гибкости, что делает его идеальным выбором для многих проектов, которым необходима скорость. За счёт этого, по сравнению с другими языками программирования, C++ имеет более быстрое выполнение и более низкую задержку, что делает его полезным для поиска решений сложных проблем, которые затрачивают много вычислительных ресурсов. Он также позволяет широко использовать алгоритмы и является эффективным средством написания статистических методов ИИ, таких как нейронные сети.

3.1.3 JavaScript

Java, чрезвычайно популярный язык программирования, также может рассматриваться как хороший выбор для программирования нейронных сетей, поскольку он обеспечивает алгоритмы поиска и нейронные сети. Этот язык отличается тем, что предлагает графическое представление, отладку и масштабируемость. Его портативность делает его предпочтительным для реализации различных приложений на основе наличия различных встроенных типов.

3.1.4 Выбор средств разработки

Для реализации данного проекта был выбран язык программирования Python за счёт его гибкости и наличия средств для работы с нейронными сетями. В качестве среды разработки использовался Google Collaboratory – для работы с нейронной сетью и ускорению обучения благодаря специальным серверам, а также PyCharm – уже для оформления самого приложения.

3.2 Реализация нейронной сети

После проектирования различных частей нейронной сети идёт разработка модели и её обучение. На рисунке 3.1 и 3.2 представлен процесс обучения нейронной сети и график точности и потерь, которую достигла модель во время обучения.

```
Epoch 1/18
76/76 [==============================] - 117s 1s/step - loss: 0.9564 - accuracy: 0.5196 - val_loss: 0.7474 - val_accuracy: 0.6650
Epoch 2/18
76/76 [==============================] - 107s 1s/step - loss: 0.7832 - accuracy: 0.6565 - val_loss: 0.7112 - val_accuracy: 0.7013
Epoch 3/18
76/76 [==============================] - 108s 1s/step - loss: 0.6812 - accuracy: 0.7204 - val_loss: 0.6349 - val_accuracy: 0.7426
Epoch 4/18
76/76 [==============================] - 109s 1s/step - loss: 0.6210 - accuracy: 0.7373 - val_loss: 0.7115 - val_accuracy: 0.7343
Epoch 5/18
76/76 [==============================] - 108s 1s/step - loss: 0.5764 - accuracy: 0.7691 - val_loss: 0.6275 - val_accuracy: 0.7624
Epoch 6/18
76/76 [==============================] - 108s 1s/step - loss: 0.5842 - accuracy: 0.7579 - val_loss: 0.5103 - val_accuracy: 0.8069
Epoch 7/18
76/76 [==============================] - 107s 1s/step - loss: 0.5357 - accuracy: 0.7810 - val_loss: 0.5081 - val_accuracy: 0.8036
Epoch 8/18
76/76 [==============================] - 108s 1s/step - loss: 0.5236 - accuracy: 0.7798 - val_loss: 0.6160 - val_accuracy: 0.7426
Epoch 9/18
76/76 [==============================] - 108s 1s/step - loss: 0.5128 - accuracy: 0.7852 - val_loss: 0.5723 - val_accuracy: 0.7657
Epoch 10/18
76/76 [==============================] - 108s 1s/step - loss: 0.4856 - accuracy: 0.7975 - val_loss: 0.4971 - val_accuracy: 0.8053
Epoch 11/18
76/76 [==============================] - 107s 1s/step - loss: 0.4936 - accuracy: 0.8000 - val_loss: 0.5073 - val_accuracy: 0.7970
Epoch 12/18
76/76 [==============================] - 107s 1s/step - loss: 0.4609 - accuracy: 0.8058 - val_loss: 0.4950 - val_accuracy: 0.7970
Epoch 13/18
76/76 [==============================] - 107s 1s/step - loss: 0.4700 - accuracy: 0.8087 - val_loss: 0.4616 - val_accuracy: 0.8185
Epoch 14/18
76/76 [==============================] - 107s 1s/step - loss: 0.4380 - accuracy: 0.8115 - val_loss: 0.4689 - val_accuracy: 0.8152
Epoch 15/18
76/76 [==============================] - 108s 1s/step - loss: 0.4264 - accuracy: 0.8194 - val_loss: 0.4338 - val_accuracy: 0.8366
Epoch 16/18
76/76 [==============================] - 108s 1s/step - loss: 0.4382 - accuracy: 0.8132 - val_loss: 0.4713 - val_accuracy: 0.8152
Epoch 17/18
76/76 [==============================] - 107s 1s/step - loss: 0.4237 - accuracy: 0.8268 - val_loss: 0.4580 - val_accuracy: 0.8003
Epoch 18/18
76/76 [==============================] - 108s 1s/step - loss: 0.4330 - accuracy: 0.8210 - val_loss: 0.6839 - val_accuracy: 0.7442
```

Рисунок 3.1 – процесс обучения модели

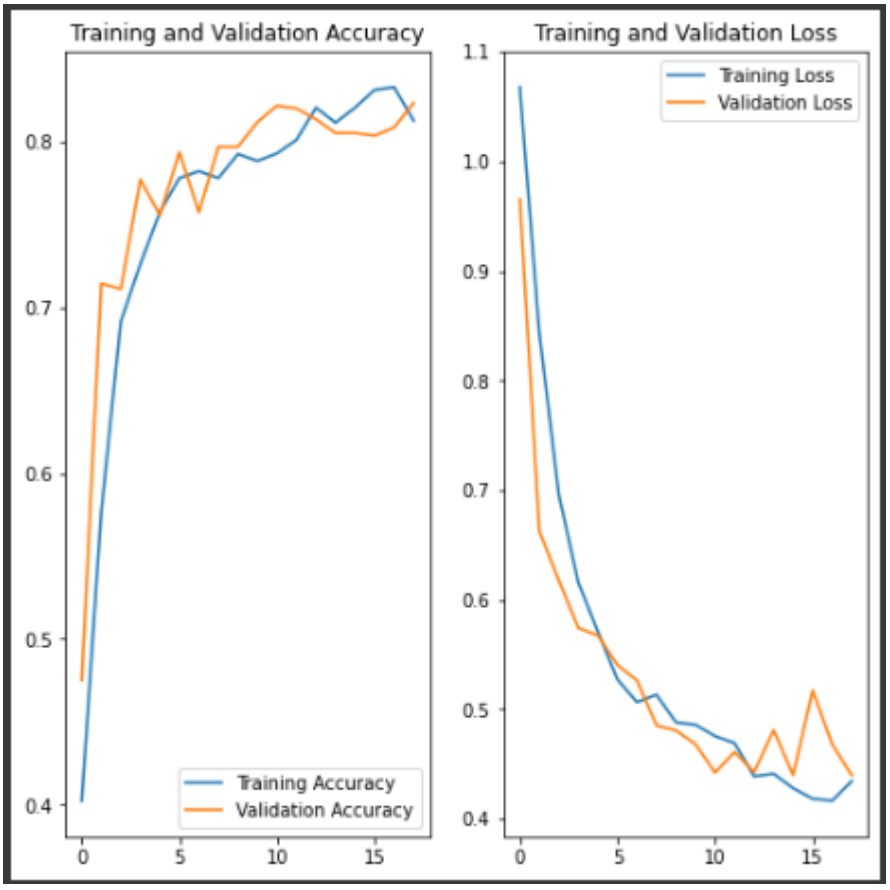

Рисунок 3.2 – график точности и потерь модели

Как можно заметить, обучение длилось 18 эпох, точность постепенно возрастала, но после достижения показателя в 80% особого прогресса не заметно – сказывается малочисленность данных для обучения.

3.3 Описание интерфейса

3.3.1 Основное окно

Приложение состоит из одного основного окна, поэтому весь функционал приложения представлен сразу же, как показано на рисунке 3.3.

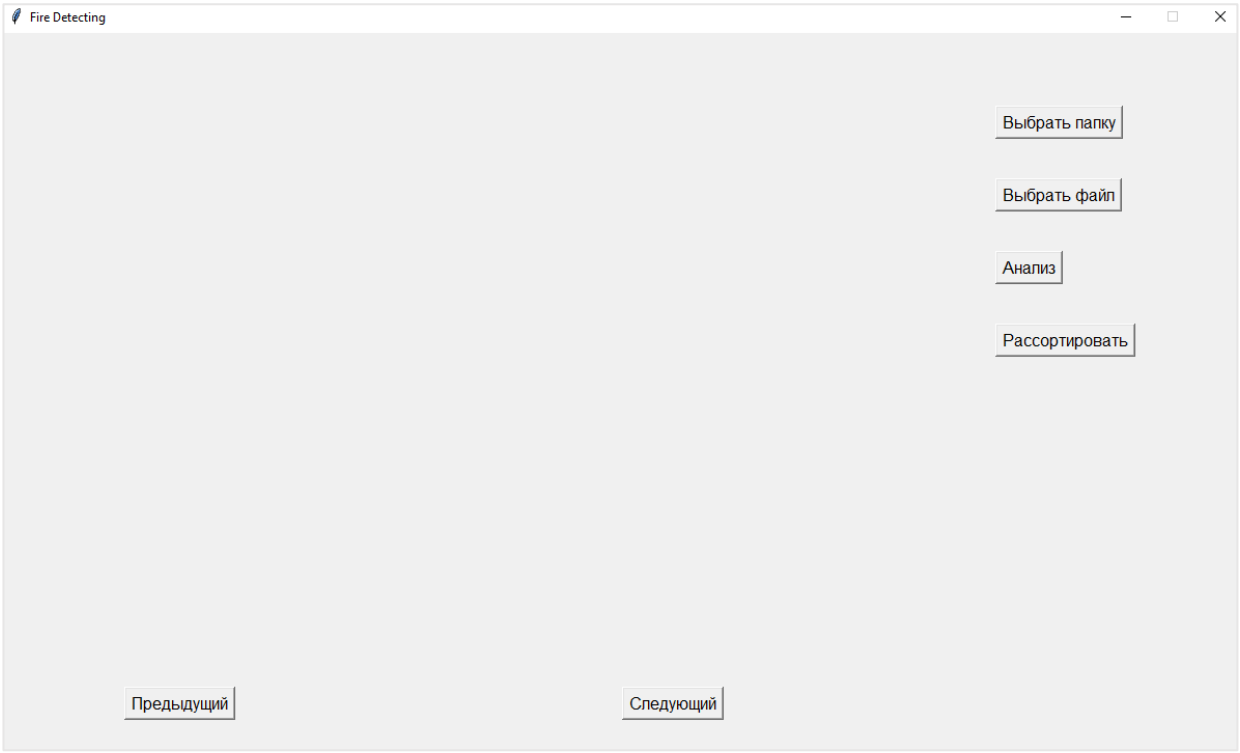

Рисунок 3.3 – основное окно приложения

3.3.2 Диалоговые окна

Для удобства будущего пользователя в приложении задействованы диалоговые окна. В текущем проекте они делятся на две категории: окна-запросы и информационные окна. Окна-запросы появляются, когда пользователь должен выбрать директорию или один конкретный файл, как показано на рисунке 3.4 и 3.5 соответственно.

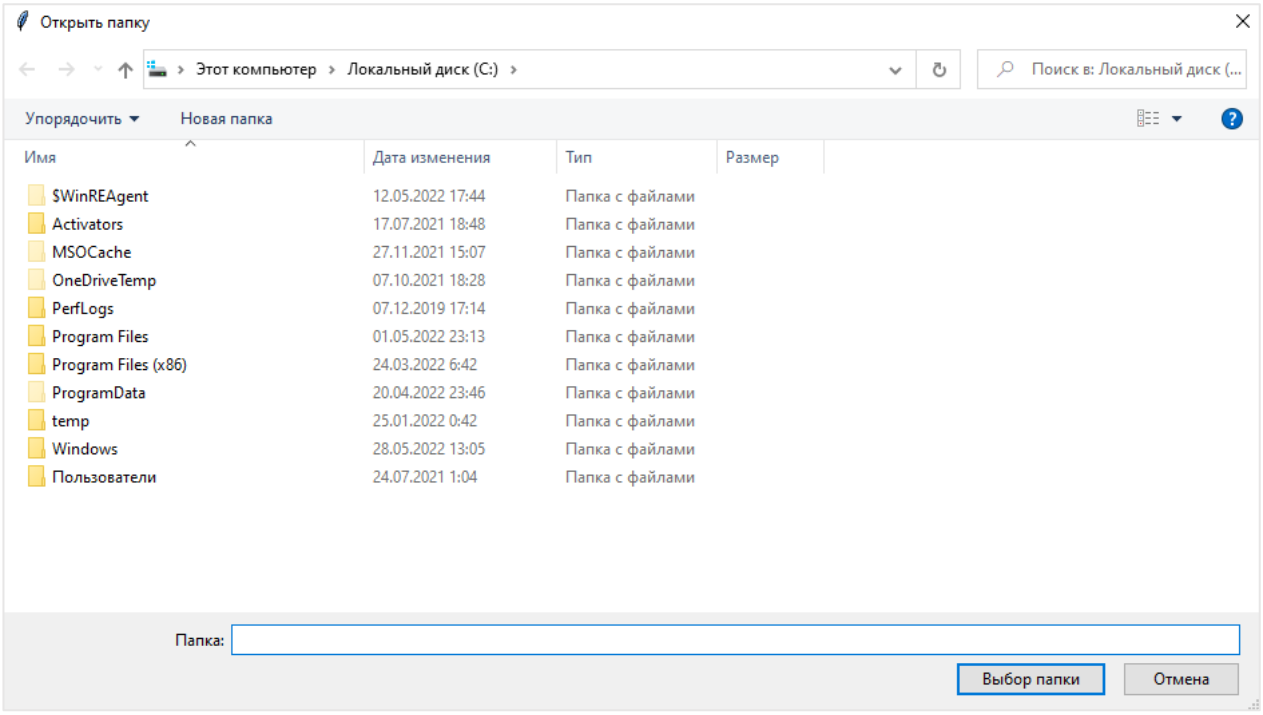

Рисунок 3.4 – выбор директории пользователем

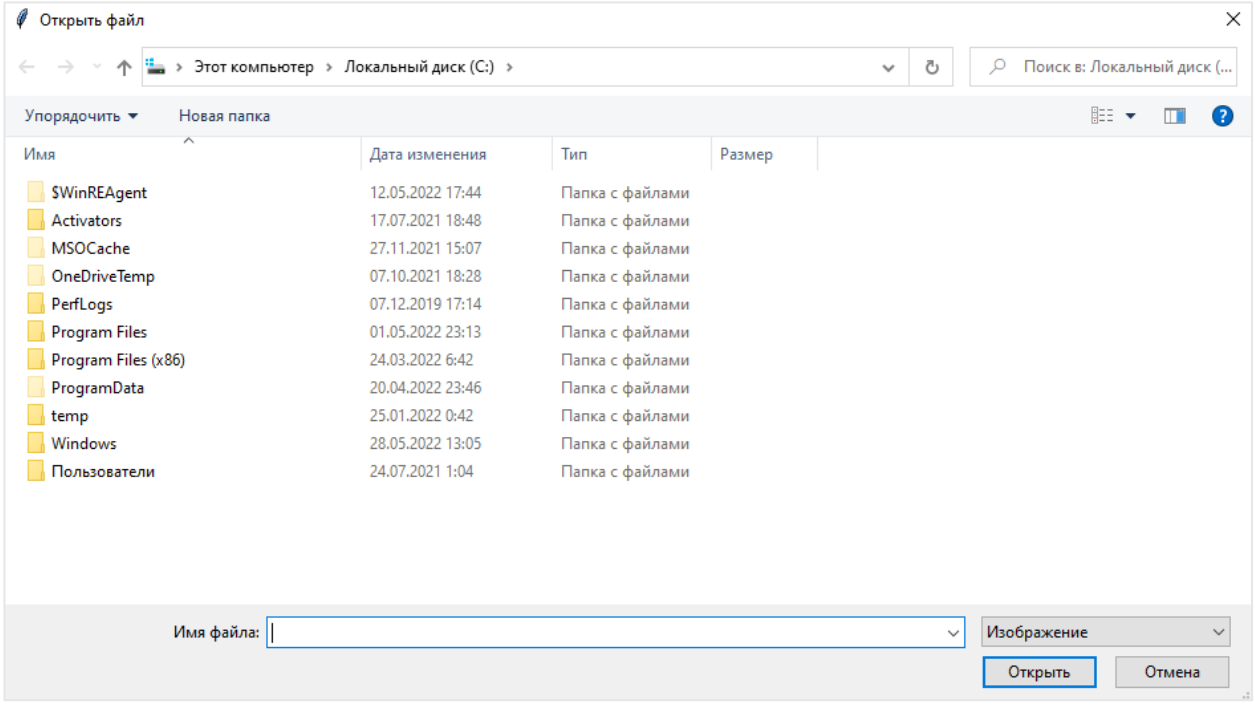

Рисунок 3.5 – выбор файла пользователем

Информационные окна предупреждают пользователя:

1. о важной информации, как показано на рисунке 3.6;

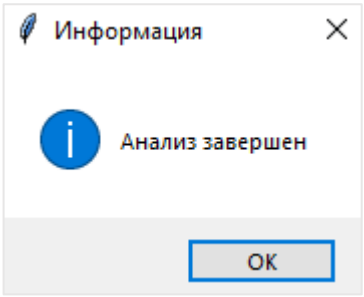

Рисунок 3.6 – всплывающее информационное окно

2. о спорной ситуации, как представлено на рисунке 3.7;

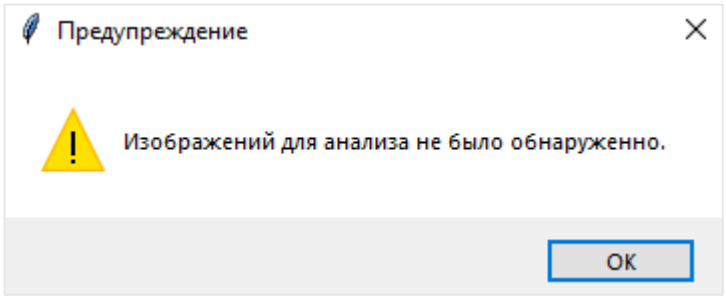

Рисунок 3.7 – всплывающее окно с предупреждением

3. об ошибке, как видно на рисунке 3.8;

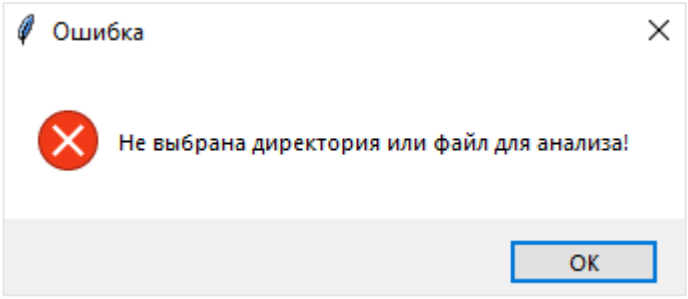

Рисунок 3.8 – всплывающее окно с ошибкой

3.4 Описание подключаемых модулей и библиотек

3.4.1 TensorFlow

TensorFlow – это весьма известная комплексная платформа с открытым исходным кодом для машинного обучения. Она имеет мощную и гибкую экосистему различных инструментов и библиотек, что позволяет исследователям продвигать новейшие достижения в области машинного обучения, а разработчикам легко разрабатывать приложения с использованием машинного обучения.

Преимущества TensorFlow – это простое построение модели, надежное производство машинного обучения в любом месте и мощные инструменты для исследований.

3.4.2 Keras

Keras – это одна из библиотек глубокого обучения от TensorFlow, представляющая из себя высокоуровневый API, написанный на Python, с расчетом на быстрое обучение. Способность переходить от проектирования к разработке с наименьшими временными затратами является ключом к проведению успешных исследований. Базовая структура данных библиотеки Keras — это модель, описывающая способ организации слоев. Простейшим типом модели является модель Sequential, представляющая собой линейный стек слоев.

Основные преимущества Keras:

1. позволяет легко и быстро создавать прототипы;
2. имеет гибкую систему, которая поддерживает как свёрточные и рекуррентные сети, так и их комбинации;
3. поддерживает работу как на процессоре (CPU), так и на графическом процессоре (GPU).

Основные идеи Keras.

1. Удобство для пользователя. Keras использует передовые методы снижения когнитивной нагрузки: он предлагает согласованный и простой API, минимизирует количество действий пользователя, необходимых для решения распространенных задач, предоставляет четкую и действенную обратную связь в случае возникновения ошибок.
2. Модульность. Под моделью понимается последовательность или граф автономных, полностью сконфигурированных модулей, в которые входят нейронные слои, функции ошибки, оптимизаторы, схемы инициализации, функции активации и схемы регуляризации.

3. Расширяемость. Keras предоставляет разработчику возможность собрать свою нейронную сеть с помощью описанных в предыдущем пункте модулей, как конструктор. Именно эта возможность делает Keras максимально привлекательным средством для проведения передовых исследований.
4. Работа в Python. Все модели написаны на Python, благодаря чему код компактен, легко читается и отлаживается, а также легко расширяется.

3.4.3 Tkinter

Это графическая библиотека, которая позволяет создавать программы с простейшим оконным интерфейсом, поддерживаемый большинством операционных систем. Tkinter, несмотря на простоту, добавляет изрядное количество собственной логики.

3.4.4 Os

Модуль os предоставляет множество функций для работы с операционной системой. Довольно часто его использую для простейшей обработки файлов, но важно знать, что модуль os используется не только для этого. Он включает массу методов и инструментов для других, более сложных и узкоспециализированных операций: обработки переменных среды, управления системными процессами и аргументами командной строки, включая изменение атрибутов файлов, которые есть только в Linux.

3.4.5 PathLib

Модуль Pathlib в Python упрощает работу с файлами и папками за счёт удобного для чтения и простого способа создания путей, а именно: вместо строкового представления пути создаётся объект типа Path. Модуль Pathlib также предлагает набор классов и удобных методов, что могут использоваться для популярных операций с путями, выгодно отличая этот модуль от простого объектно-ориентированного способа.

3.4.6 Numpy

NumPy это открытый модуль для Python, известный за свои общие математические и числовые операции в виде пре-скомпилированных, быстрых функций. Они объединяются в высокоуровневые пакеты и обеспечивают функционал, который можно сравнить с функционалом MatLab. Главной особенностью Numpy является объект array. Массивы схожи со списками, исключая тот факт, что элементы массива должны иметь одинаковый тип данных. С массивами можно проводить числовые операции с большим объемом информации в разы быстрее и, главное, намного эффективнее чем со списками.

Плюсы Numpy:

1. Мощные массивы. Быстрые и универсальные функции векторизации, индексации и вещания NumPy задают стандарт обработки массивов на сегодняшний день.
2. Вычислительные возможности. Библиотека предлагает комплексные математические функции, генераторы случайных чисел, процедуры линейной алгебры, преобразования Фурье и многое другое.
3. Совместимость. NumPy поддерживает широкий спектр аппаратных и вычислительных платформ и хорошо работает с распределенными, графическими процессорами и библиотеками
4. Производительность. Ядро NumPy написано на языке Си и является хорошо оптимизированным, что позволяет наслаждаться гибкостью Python со скоростью компиляции кода, которым так знаменит язык Си.
5. Простота в использовании. Синтаксис высокого уровня NumPy делает его доступным и продуктивным для программистов с любым уровнем фона или опыта.
6. С открытым исходным кодом. Распространяемый под либеральной лицензией BSD, NumPy разрабатывается и поддерживается на GitHub динамичным, отзывчивым и разнообразным.

3.5 Описание основных функций

3.5.1 Загрузка данных

Данные загружаются только после того, как пользователь указал директорию, в которой могут находится файлы и нажал на кнопку «Анализ». После этого вызывается функция, которая выбирает только изображения из указанной директории и передаёт путь к ним для анализа. Код для этой функции показан на рисунках 3.9.

```
image_extensions = [".png", ".jpg", "bmp", "gif", "jpeg"]
c = len(list(pathlib.Path(self.directory).rglob("*")))
progress_bar = ttk.Progressbar(self, orient="horizontal", mode="determinate", maximum=c, value=0)
progress_bar.place(relx=0.46, rely=0.48)
self.count = 0
for filepath in pathlib.Path(self.directory).rglob("*"):
    if filepath.suffix.lower() in image_extensions:
        self.load_image_to_array(filepath)
        progress_bar['value'] += 1
        self.update()
progress_bar.destroy()
```

Рисунок 3.9 – функция, отбирающая только изображения

3.5.2 Анализ

После того, как были отобраны изображения, с помощью специального метода они преобразуются в входную карту признаков. Затем обученная модель предсказывает, с какой вероятностью на изображении показан один из трёх классов. Полученный результат заносится в специальные словари, обозначающие класс, чтобы привязать изображение и вероятность. Это можно увидеть, просмотрев код на рисунке 3.10.

```
def load_image_to_array(self, filepath):
    img = tf.keras.utils.load_img(
        filepath, target_size=(256, 256))
    img_array = tf.keras.utils.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0)
    predictions = model.predict(img_array)
    self.score.append(tf.nn.softmax(predictions[0]))
    class_name = self.class_names[np.argmax(self.score[-1])]
    if self.count != -1:
        if class_name == "Огонь":
            self.fire[str(self.count)] = filepath, 100 * np.max(self.score[-1]), "Огонь"
        if class_name == "Дым":
            self.smoke[str(self.count)] = filepath, 100 * np.max(self.score[-1]), "Дым"
        if class_name == "Нейтрал":
            self.neutral[str(self.count)] = filepath, 100 * np.max(self.score[-1]), "Нейтрал"
        self.count += 1
    else:
        self.array_images[0] = filepath, 100 * np.max(self.score[-1]), class_name
```

Рисунок 3.10 – функция загрузки выбранных изображений

3.5.3 Отображение картинки

После того, как все изображения были проанализированы, пользователь может просмотреть результат. При нажатии на навигационные кнопки «Следующий» и «Предыдущий» на холсте отображается изображение и привязанная к нему вероятность. Так как изображения могут быть разного размера, их тут же приводит к одному фиксированному размеру. На рисунке 3.11 показан код для этой функции

```
def show_image(self):
    print(self.score[0])
    self.canvas.delete("all")
    ite = list(self.array_images.keys())
    l = self.array_images[ite[self.count_image]]
    text = "Вероятность, что это {} составляет {:.2f} процентов.".format(l[2], l[1])
    fixed_width = 512
    self.image = Image.open(l[0])
    width_percent = (fixed_width / float(self.image.size[0]))
    height_size = int((float(self.image.size[0]) * float(width_percent)))
    new_image = self.image.resize((fixed_width, height_size))
    self.photo = ImageTk.PhotoImage(new_image)
    self.c_image = self.canvas.create_image(300, 250, image=self.photo)
    self.text = self.canvas.create_text(300, 530, text=text, font=20)
    self.canvas.place(relx=0.05, rely=0.05)
```

Рисунок 3.11 – функция отображения изображений

3.5.4 Распределение

Цель этой функции заключается в том, чтобы распределить изначальные изображения по директориям с названиями классов. Эта функция создана на случай, если требуется хранить проанализированные данные, например, для отчётности. Пользователь выбирает директорию, куда хочет поместить эти данные, затем в выбранной директории создаются каталоги, отображающие класс. Во избежание того, что директории с таким названием уже существуют, название генерируется следующим образом: название класса, сегодняшняя дата и время в миллисекундах.

Код для функции создания директорий представлен на рисунке 3.12.

```
def dialog_save_image(self):
    directory = fd.askdirectory(title="Выберите директорию для того, чтобы сохранить данные", initialdir="/")
    if directory != "":
        t = str(datetime.datetime.now().time().microsecond)
        p = []
        for i in range(3):
            s = str(directory) + "/" + self.class_names[i] + "_" + str(datetime.date.today()).replace("-", "_") \
                + "_" + t
            os.mkdir(s)
            os.chmod(s, 0o754)
            p.append(s)
        self.save_image(p)
```

Рисунок 3.12 – генерация директорий для распределения проанализированных изображений

После вызывается другая функция, которая переносит файлы в нужную директорию. На рисунке 3.13 представлена программная реализация этой функции.

```
def save_image(self, path):
    for items in self.fire.items():
        p = str(items[1][0])
        p = list(p.split('\\'))
        os.replace(items[1][0], path[0] + '/' + p[3])
    for items in self.smoke.items():
        p = str(items[1][0])
        p = list(p.split('\\'))
        os.replace(items[1][0], path[2] + '/' + p[3])
    for items in self.neutral.items():
        p = str(items[1][0])
        p = list(p.split('\\'))
        os.replace(items[1][0], path[1] + '/' + p[3])
```

Рисунок 3.13 – код функции распределения изображений

3.6 Апробация нейросетевой модели

Апробация нейросетевой модели осуществлялась на серии снимков. Результаты апробации приведены в таблице 3.1.

Таблица 3.1 – Результаты вычислительных экспериментов

Класс	Количество снимков	Корректное распознавание	Ошибки
“Smoke”	50	92%	8%
“Fire”	50	76%	24%
“Neutral”	50	78%	22%

Таким образом, можно сделать вывод, что предложенная нейросетевая модель может корректно распознавать огонь и дым. Ошибочные результаты в основном связаны с тем, что на снимках присутствуют оба класса признаков. Для повышения качества распознавания требуется продолжить исследования в части совершенствования архитектуры нейронной сети и увеличения признакового пространства для обучения.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе было спроектировано и реализовано приложение, позволяющее идентифицировать пожар используя снимки со спутника. Все цели проекта достигнуты и все задачи выполнены.

Но к сожалению, как приложение, так и разработанная нейронная сеть имеют свои недостатки. Во-первых, достигнуть высокой точности распознавания не получилось. В первую очередь это из-за малого количества данных для обучения и довольно скудной подборки – найти подходящие изображения очень сложно, особенно в большом количестве. Во-вторых, процесс не полностью автоматизирован. Пользователю самому нужно подготавливать данные, запускать анализ и получать координаты с местом возгорания. В-третьих, пользовательский интерфейс мало функционален.

Однако, если использовать данную систему не со снимками со спутника, а со снимками со специальных дронов, то результат будет более точным из-за более детальных изображений.

Предложенная нейросетевая модель может корректно распознавать огонь и дым. Ошибочные результаты в основном связаны с тем, что на снимках присутствуют оба класса признаков. Для повышения качества распознавания требуется продолжить исследования в части совершенствования архитектуры нейронной сети и увеличения признакового пространства для обучения.

В перспективе планируется продолжить исследования для минимизации ошибочных предположений системы о наличии пожара или дыма. Для этого необходимо исследовать архитектуру нейронной сети и продумать состав и объемы, подаваемых на вход нейронной сети датасетов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Багаев, И.И. Анализ понятий нейронная сеть и свёрточная нейронная сеть, обучение свёрточной нейросети при помощи модуля TensorFlow / Багаев И.И. – Текст : непосредственный // Математическое и программное обеспечение систем в промышленной и социальной сферах; учредитель Магнитогорский государственный технический университет им. Г.И. Носова. – 2020. – N 1. – С. 15-22.
2. Анализ обстановки с пожарами и их последствиями на территории Российской Федерации за 9 месяцев 2021 года. – Текст : электронный // МЧС России : [сайт] – 2022. – URL: https://39.mchs.gov.ru/uploads/resource/2021-11-01/11-statisticheskie-dannye_1635768651911545997.docx (дата обращения: 27.01.2022).
3. Тархов, Д.А. Нейросетевые модели и алгоритмы: справочник / Д.А Тархов – Москва: Радиотехника, 2014. – 352 с.
4. Ибрагимов Р.М. Влияние функции активации нейронных сетей на скорость обучения на примере нейронной сети с обратным распространением ошибки : [Материалы 21-й Всероссийской молодежной научной школы-семинара; Ульяновский гос. тех. ун-т. – Ульяновск, 04-06 декабря 2018] / Ибрагимов Р.М. – Текст : непосредственный // Актуальные проблемы физической и функциональной электроники. – С. 125-126.

5. Галушкин, А.И. Нейронные сети. Основы теории / Галушкин А.И ; редактор Рысев Ю.Н. – Москва – 2012. – 496 с. – ISBN 978-5-9912-0082-0. – Текст : электронный. – URL : https://www.rulit.me/data/programs/resources/pdf/Galushkin_Neyronnye-seti-osnovy-teorii_RuLit_Me_603891.pdf (дата обращения: 13.02.2022).
6. Джулли, А. Библиотека Keras – инструмент глубокого обучения / А. Джулли, С. Пал ; [перевод с английского А.А. Слинкин]. – Москва : ДМК Пресс, 2018. – 294 с. – ISBN 978-5-97060-573-8. – Текст : электронный. – URL : https://www.rulit.me/data/programs/resources/pdf/Dzhulli_Biblioteka-Keras-instrument-glubokogo-obucheniya_RuLit_Me_603503.pdf (дата обращения: 16.03.2022).
7. Луцив В.Р. Свёрточные искусственные нейронные сети глубокого обучения : Оптический журнал / Санкт-Петербургский государственный университет аэрокосмического приборостроения – 2015. – N 8. – С 11-23. – Текст : непосредственный.
8. Годунов, А.И. Сегментация изображений и распознавание объектов на основе технологии свёрточных нейронных сетей / Годунов А.И., Баланян С.Т., Егоров П.С. – Текст : электронный // Компьютерные и информационные науки. – 2021. – N 3. – С. 62-72. – URL: https://cyberleninka.ru/article/n/segmentatsiya-izobrazheniy-i-raspoznavanie-obektov-na-osnove-tehnologii-svertochnyh-neyronnyh-setey/viewer (дата обращения: 28.04.2022).
9. Русскоязычная документация Keras. – Текст : электронный // Keras : [сайт] – 2022. – URL: https://ru-keras.com/guide-functional-api/#P4_6 (дата обращения: 14.04.2022).
10. Спутниковый мониторинг в реальном времени. – Текст : электронный // Planet : [сайт] – 2022. – URL: http://planet.com/products/monitoring (дата обращения: 17.03.2022).

ПРИЛОЖЕНИЕ А

```
[[[ 18.  35.  29.]
  [ 38.  55.  49.]
  [ 33.  50.  44.]
  ...
  [195. 205. 214.]
  [193. 203. 212.]
  [189. 199. 208.]]

 [[ 23.  40.  34.]
  [ 25.  42.  36.]
  [ 34.  51.  45.]
  ...
  [196. 206. 215.]
  [194. 204. 213.]
  [191. 201. 210.]]

 [[ 37.  54.  48.]
  [ 17.  34.  28.]
  [ 37.  54.  48.]
  ...
  [197. 207. 216.]
  [195. 205. 214.]
  [193. 203. 212.]]

 ...

 [[ 54.  67.  60.]
  [ 25.  38.  29.]
  [ 38.  51.  42.]
  ...
  [ 37.  47.  56.]
  [ 36.  46.  55.]
  [ 39.  49.  58.]]

 [[ 30.  43.  36.]
  [ 28.  41.  34.]
  [ 43.  56.  49.]
  ...
  [ 39.  49.  58.]
  [ 39.  49.  58.]
  [ 43.  53.  62.]]

 [[ 58.  71.  64.]
  [ 21.  34.  27.]
  [ 26.  39.  32.]
  ...
  [ 38.  48.  57.]
  [ 39.  49.  58.]
  [ 44.  54.  63.]]]
```

Рисунок 1 – карта признаков для изображения класса «Fire»

```
[[[ 87.  82.  84.]
  [ 88.  82.  83.]
  [ 88.  82.  82.]
  ...
  [147. 123.  99.]
  [147. 123.  99.]
  [144. 120.  96.]]

 [[ 86.  82.  83.]
  [ 87.  81.  83.]
  [ 87.  81.  81.]
  ...
  [148. 124. 100.]
  [148. 124. 100.]
  [145. 121.  97.]]

 [[ 85.  81.  83.]
  [ 85.  81.  83.]
  [ 87.  81.  81.]
  ...
  [149. 125. 101.]
  [149. 125. 101.]
  [146. 121.  98.]]

 ...

 [[  6.  17.  35.]
  [  6.  17.  36.]
  [  4.  15.  34.]
  ...
  [ 45.  62.  71.]
  [ 45.  63.  72.]
  [ 34.  52.  61.]]

 [[  5.  16.  34.]
  [  7.  18.  36.]
  [  5.  16.  35.]
  ...
  [ 36.  54.  63.]
  [ 42.  60.  69.]
  [ 37.  55.  64.]]

 [[  4.  15.  32.]
  [  7.  18.  35.]
  [  8.  19.  37.]
  ...
  [ 34.  49.  53.]
  [ 43.  59.  63.]
  [ 37.  52.  57.]]]
```

Рисунок 2 – карта признаков изображения класса «Smoke»

```
[[[ 77.  94. 120.]
  [ 79.  96. 122.]
  [ 80.  97. 123.]
  ...
  [129. 147. 171.]
  [128. 145. 171.]
  [126. 143. 169.]]

 [[ 77.  94. 120.]
  [ 79.  96. 122.]
  [ 80.  97. 123.]
  ...
  [130. 148. 172.]
  [129. 146. 172.]
  [127. 144. 170.]]

 [[ 77.  94. 120.]
  [ 79.  96. 122.]
  [ 80.  97. 123.]
  ...
  [132. 150. 174.]
  [131. 148. 174.]
  [129. 146. 172.]]

 ...

 [[  5.   0.   4.]
  [  3.   1.   4.]
  [  3.   1.   4.]
  ...
  [  1.   1.   0.]
  [  1.   1.   0.]
  [  1.   1.   0.]]

 [[  5.   0.   4.]
  [  3.   1.   4.]
  [  3.   1.   4.]
  ...
  [  1.   1.   0.]
  [  1.   1.   0.]
  [  1.   1.   0.]]

 [[  4.   0.   3.]
  [  2.   0.   3.]
  [  3.   1.   4.]
  ...
  [  1.   1.   0.]
  [  2.   2.   0.]
  [  3.   3.   1.]]]
```

Рисунок 3 – карта признаков изображения класса «Neutral»