

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки  
09.03.04 Программная инженерия

Направленность (профиль) Промышленная разработка программного обеспечения

Студента Беликова Павла Геннадьевича                   шифр 180818

Институт приборостроения, автоматизации и информационных технологий

Тема выпускной квалификационной работы

«Разработка программного обеспечения для получения сведений об автомобилях  
на основе их изображений с использованием методов машинного обучения»

Студент

П.Г. Беликов

Руководитель

О.В. Конюхова

Нормоконтроль

О.В. Захарова

Зав. кафедрой  
информационных систем  
и цифровых технологий

В.Н. Волков

Орёл 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

Институт приборостроения, автоматизации и информационных технологий  
Кафедра информационных систем и цифровых технологий  
Направление 09.03.04 Программная инженерия  
Направленность (профиль) Промышленная разработка программного обеспечения

УТВЕРЖДАЮ:  
Зав. кафедрой

Б.Н. Волков  
«12» ноябрь 2022 г.

## ЗАДАНИЕ

## на выполнение выпускной квалификационной работы

студента Беликова Павла Геннадьевича      шифр 180818

- # 1 Тема ВКР «Разработка программного обеспечения для получения сведений об автомобилях на основе их изображений с использованием методов машинного обучения»

Утверждена приказом по университету от «28» октября 2021г. № 2-2993

2 Срок сдачи студентом законченной работы «14» июня 2022г.

- 3 Исходные данные к работе
  - Теоретический материал; информация о предметной области
  - 4 Содержание пояснительной записки (перечень подлежащих разработке вопросов)
    - Анализ предметной области
    - Моделирование системы распознавания марок и моделей транспортных средств
    - Проектирование программного обеспечения
    - Реализация программного обеспечения

5 Перечень демонстрационного материала  
Презентация, отображающая основные этапы и результаты выполнения ВКР

Дата выдачи задания                    « 12 » мая 2022 г.

Руководитель

  
(подпись)

О.В. Конюхова

Задание принял к исполнению

  
(подпись)

П.Г. Беликов

### КАЛЕНДАРНЫЙ ПЛАН

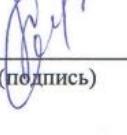
Наименование этапов работы	Сроки выполнения этапов работы	Примечание
Анализ предметной области, постановка задачи, формирование основных требований	12.05.2022 – 18.05.2022	
Моделирование системы	18.05.2022 – 23.05.2022	
Проектирование обучающего набора данных	23.05.2022 – 26.05.2022	
Проектирование программного обеспечения	26.05.2022 – 02.06.2022	
Реализация программного обеспечения	02.06.2022 – 08.06.2022	
Оформление выпускной квалификационной работы	08.06.2022 – 14.06.2022	

Студент

  
(подпись)

П.Г. Беликов

Руководитель

  
(подпись)

О.В. Конюхова

## **АННОТАЦИЯ**

ВКР 96 с., 65 рис., 3 табл., 16 источников, 3 прил.

**НЕЙРОННАЯ СЕТЬ, ГЕНЕРАЛЬНАЯ СОВОКУПНОСТЬ,  
КЛАСТЕРИЗАЦИЯ, АУГМЕНТАЦИЯ, КЛАССИФИКАЦИЯ ОБРАЗОВ.**

Выпускная квалификационная работа посвящена проектированию и реализации программного обеспечения для получения сведений об автомобилях методами машинного обучения.

В первой главе была проанализирована предметная область, определены основные задачи, сформированы основные требования к разработке. Также было проанализировано аналогичное программное обеспечение и выбраны методы разработки.

Во второй главе были смоделированы основные процессы разрабатываемой системы, а также ее архитектура.

В третьей главе были спроектированы основные алгоритмы работы программного обеспечения, алгоритмы сбора обучающего набора данных, база данных.

В четвертой главе были реализованы спроектированное программное обеспечение и обучающий набор данных. Также были рассчитаны метрики эффективности нейронной сети.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	7
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ .....	9
1.1 Распознавание транспортных средства сегодня .....	9
1.1.1 Распознавание с помощью человека .....	9
1.1.2 Распознавание и идентификация номера транспортного средства с использованием нейронных сетей.....	9
1.1.3 Распознавание моделей транспортных средств .....	10
1.2 Обзор аналогичных решений.....	11
1.3 Задача распознавания образов .....	13
1.4 Архитектуры нейронных сетей .....	16
1.4.1 Виды архитектур нейронных сетей .....	16
1.4.2 Сверточные нейронные сети.....	18
1.5 Методы обучения нейронных сетей .....	20
1.5.1 Обучение с учителем.....	20
1.5.2 Обучение без учителя.....	20
1.5.3 Обучение с подкреплением .....	21
1.6 Обученные нейронные сети.....	22
1.7 Числовое представление изображения.....	24
1.8 Функция потерь нейронной сети .....	25
1.9 Общая структура системы .....	27
1.10 Функциональные и нефункциональные требования .....	27
1.11 Выбор методов разработки.....	28
2 МОДЕЛИРОВАНИЕ СИСТЕМЫ РАСПОЗНАВАНИЯ МАРОК И МОДЕЛЕЙ ТРАНСПОРТНЫХ СРЕДСТВ.....	29
2.1 Моделирование взаимодействия системы со внешней средой.....	29
2.2 Моделирование взаимодействия элементов внутри системы .....	30
2.3 Моделирование процессов системы.....	33

2.4 Концептуальная модель БД .....	35
2.5 Логическая модель БД .....	37
2.6 Построение концептуальной модели ПО .....	37
2.7 План подготовки набора данных для обучения.....	40
<b>3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ .....</b>	<b>42</b>
3.1 Проектирование обучающего набора данных .....	42
3.1.1 Сбор первоначального набора данных.....	42
3.1.2 Разметка изображений .....	45
3.1.3 Увеличение набора данных с помощью аугментации изображений.....	47
3.2 Проектирование базы данных .....	48
3.2.1 Выявление перечня ограничений целостности и выбор способа реализации его контроля.....	48
3.2.2 Физическая модель БД.....	49
3.3 Проектирование пользовательского интерфейса .....	50
3.3.1 Проектирование жизненных циклов пользовательского интерфейса .....	50
3.3.2 Карта диалоговых окон пользовательского интерфейса .....	51
3.3.3 Проектирование алгоритмов работы ПО .....	51
3.4 Проектирование модели нейронной сети .....	61
<b>4 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ .....</b>	<b>63</b>
4.1 Реализация модели нейронной сети .....	63
4.1.1 Описание работы аугментации изображений .....	63
4.1.2 Особенности реализации нейронной сети .....	64
4.1.2.1 График распространения ошибки.....	64
4.1.2.2 Кластеризация модели .....	65
4.1.2.3 Метрика полноты .....	66
4.1.2.4 Конвертация модели нейронной сети .....	66
4.2 Реализация программного обеспечения.....	66
4.2.1 Особенности программной реализации .....	66
4.2.2 Описание пользовательского интерфейса.....	69
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>78</b>

СПИСОК ЛИТЕРАТУРЫ .....	79
Приложение 1 – Примеры аугментаций изображения .....	81
Приложение 2 – Фрагмент программного кода обучения модели нейронной сети.....	85
Приложение 3 – Фрагмент программного кода главного окна ПО .....	88
УДОСТОВЕРЯЮЩИЙ ЛИСТ № 180818 К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ.....	94
ИНФОРМАЦИОННО-ПОИСКОВАЯ ХАРАКТЕРИСТИКА ДОКУМЕНТА НА ЭЛЕКТРОННОМ НОСИТЕЛЕ.....	95

## ВВЕДЕНИЕ

В наше время, когда технологии развиваются с немыслимой скоростью, появляется потребность в системах, которые будут выполнять не четко запрограммированную последовательность действий, а самостоятельно анализировать входящую информацию, искать в ней закономерности, а также прогнозировать ее и выполнять операции с наилучшим исходом.

Для этих целей больше всего подходит структура, которая пришла из биологии – нейронная сеть. Это последовательность нейронов, соединенных между собой синапсами.

Человеческий мозг и цифровой компьютер выполняют совершенно разные задачи и имеют различные свойства. В типичном мозгу человека имеется в 1000 раз больше нейронов, чем логических элементов в процессоре типичного компьютера высокого класса. Различие в отношении количества элементов является незначительным по сравнению с различием в скорости переключения и степени распараллеливания. Микросхемы компьютера способны выполнить отдельную команду меньше чем за наносекунду, тогда как нейроны действуют в миллионы раз медленнее. Но мозг сторицей восполняет этот свой недостаток, поскольку все его нейроны действуют одновременно, тогда как большинство современных компьютеров имеет только один процессор (но с несколькими ядрами) или небольшое количество процессоров. Таким образом, даже несмотря на то, что компьютер обладает преимуществом более чем в миллион раз в физической скорости переключения, оказывается, что мозг по сравнению с ним выполняет все свои действия примерно в 100 000 раз быстрее.

На данный момент нейронные сети получили большое распространение и используются во многих сферах человеческой деятельности. Будь то медицина, автомобилестроение, компьютерная безопасность или повседневная жизнь.

Нейронные сети позволяют распознать и классифицировать какой-либо объект с точностью выше, чем у человека. Такое решение актуально в задачах распознавания и классификации в таких областях, где количество объектов очень

велико. В частности, такой областью является распознавание марок и моделей транспортных средств. Решая эту задачу, можно не только классифицировать транспортные средства, но получать на основе этих данных какую-либо информацию о самих моделях, что в свою очередь позволит охватить больший спектр проблем, таких как проблема ориентации в пространстве на основе 2D изображения (на основе данных распознанного транспортного средства можно узнать расстояние, которое он проехал и вычислить скорость — такое решение может быть полезно в правоохранительных органах).

Целью данной работы является повышение эффективности работы нейронной сети с помощью снятия ограничения на максимальное число распознаваемых классов.

Нейронная сеть должна научиться по переданному изображению определять принадлежность транспортного средства к той или иной модели.

В рамках ВКР были поставлены следующие задачи:

1. Описание предметной области нейронной сети.
2. Обзор и выбор архитектуры и способа обучения нейронной сети.
3. Обзор существующих аналогов и выбор наиболее подходящего.
4. Проектирование программ для сбора и разметки данных для датасета.
5. Проектирование и разработка базы данных.
6. Проектирование и разработка пользовательского интерфейса.

## 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

### 1.1 Распознавание транспортных средства сегодня

В настоящее время существует множество различных способов распознавания и идентификации транспортных средств. Рассмотрим самые распространенные из них.

#### 1.1.1 Распознавание с помощью человека

Тут все предельно понятно: специализированный человек смотрит на предоставленное ему изображение, находит и идентифицирует транспортное средство. Такой метод требует больших затрат времени и сил со стороны человека и считается устаревшим.

#### 1.1.2 Распознавание и идентификация номера транспортного средства с использованием нейронных сетей

Суть метода заключается в том, что нейронная сеть самостоятельно находит транспортное средство на изображении, после чего находит его номер и распознает символы на нем.

Способ работоспособен, но для идентификации транспортного средства потребуется дальнейшее постоянное обращение к базе данных номеров, доступ к которой получить очень сложно.

Помимо этой проблемы, номер может быть загрязнен, прикрыт или вовсе отсутствовать. Также нередки случаи, когда государственный номер был зарегистрирован на иное транспортное средство.

Все это затрудняет решение задачи идентификации транспортных средств. Стоит отметить, что при распознавании номеров человек все еще наблюдает и контролирует результаты работы системы.

### 1.1.3 Распознавание моделей транспортных средств

Способ, при котором вмешательство человека в процесс распознавания и идентификации минимально.

Распознавание и идентификация происходит в два этапа: сначала первая нейронная сеть находит транспортное средство на изображении и вырезает его, после чего передает вырезанный прямоугольник на идентификацию другой нейронной сети, которая выдает свой прогноз – определяет модель транспортного средства.

Подобная система применяется в нескольких регионах России. Ntechlab (технологический партнер Ростеха) разработал систему распознавания лиц, которая уже применяется в ряде регионов РФ, теперь компания объединила ее с функцией распознавания силуэтов людей и машин. Компания представляет платформу мультиобъектной видеоаналитики, которая в режиме реального времени отображает видеопотоки, распознавая силуэты прохожих и автомобилей, на конференции "Цифровая индустрия промышленной России" (ЦИПР) [1].

Однако узнать, как именно работает эта система, равно как и получить к ней доступ, для нас невозможно.

Другой аналог подобной системы находится в открытом доступе. Она создана на основе модели нейронной сети ResNet-152 в Стэнфордском университете. Нейронная сеть для идентификации автомобилей обучена на большом количестве автомобилей и прекрасно справляется со своей задачей. Так в чем же проблема?

Автомобильная промышленность не стоит на месте каждый год выпускается сотни новых моделей транспортных средств. И на сегодняшний день подобная модель не сможет распознать и половины автомобилей, встречаемых нами на улице ежедневно.

Разрабатываемая нейронная сеть позволит распознавать все современные транспортные средства с высокой точностью, а также, за счет использования новых методов разработки нейронных сетей, не потеряет своей актуальности по прошествии лет.

Подводя итог, можно сказать, что нужна нейронная сеть, которая будет обладать высокой точностью распознавания, не требовать вмешательства человека в свою работу и не потеряет актуальность из-за увеличения числа моделей транспортных средств на рынке.

## 1.2 Обзор аналогичных решений

На сегодняшний день существует большое количество ПО для распознавания транспортных средств. Для сравнения были выбраны наиболее популярные решения: Авто.ру, Intlab Auto MMR и Blippar. Сравнение аналогов представлено в таблице 1.

В качестве основных критериев сравнения были выбраны:

- отсутствие ограничений на количество распознаваемых марок и моделей транспортных средств;
- двустороннее распознавание (с обеих сторон транспортного средства).

Таблица 1 – Обзор аналогов ПО

	Отсутствие ограничений на количество распознаваемых транспортных средств.	Двустороннее распознавание.
Авто.ру	-	+
Intlab Auto MMR	-	+
Blippar	-	-
Наше ПО.	+	+-

Авто.ру выполняет распознавание транспортного средства основываясь на его кузове и эмблеме марки. Из-за этого при загрузке одного транспортного средства с разных ракурсов сервис может выдать разные результаты. Всего Авто.ру поддерживает около 100 марок и 1000 моделей транспортных средств. На рисунке 1 представлен интерфейс сервиса.

Intlab Auto MMR выполняет распознавание транспортного средства основываясь на его кузове. Данное ПО поддерживает 6 различных категорий,

более 100 марок и 1300 моделей транспортных средств. На рисунке 2 представлен пример работы встраиваемого ПО Intlab Auto MMR.

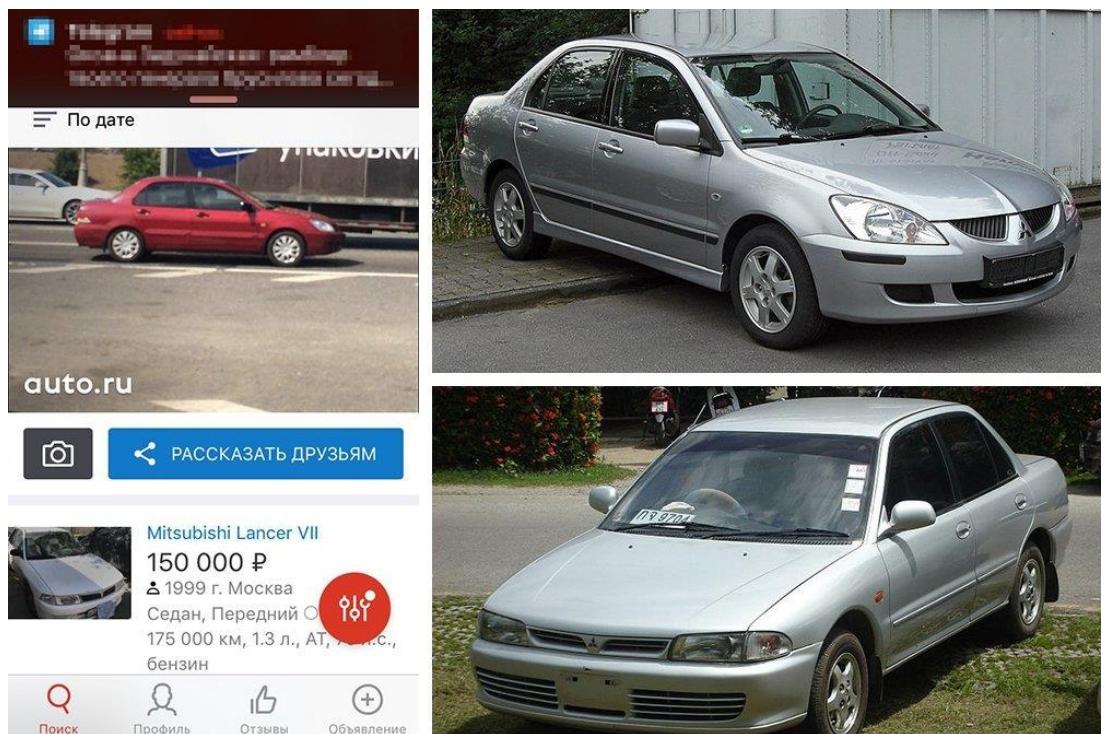


Рисунок 1 – Пример работы сервиса Авто.ру

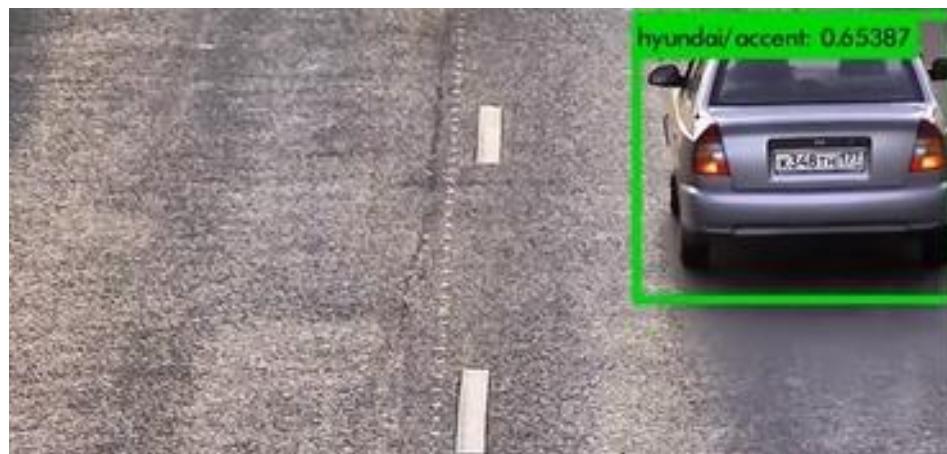


Рисунок 2 – Пример работы Intlab Auto MMR

Blippar выполняет распознавание транспортного средства основываясь на его кузове. В данном приложении представлено около 1000 моделей иностранных автомобилей. На рисунке 3 представлен пример работы приложения.

Все рассмотренное ПО имеет ограничение на количество распознаваемых марок и моделей. Это означает, что если в это ПО нужно будет добавить новую модель, то необходимо будет повторить процесс сбора данных для нового

транспортного средства, а затем обучить используемую нейронную сеть заново на всех имеющихся данных – даже тех, которые она уже была обучена.

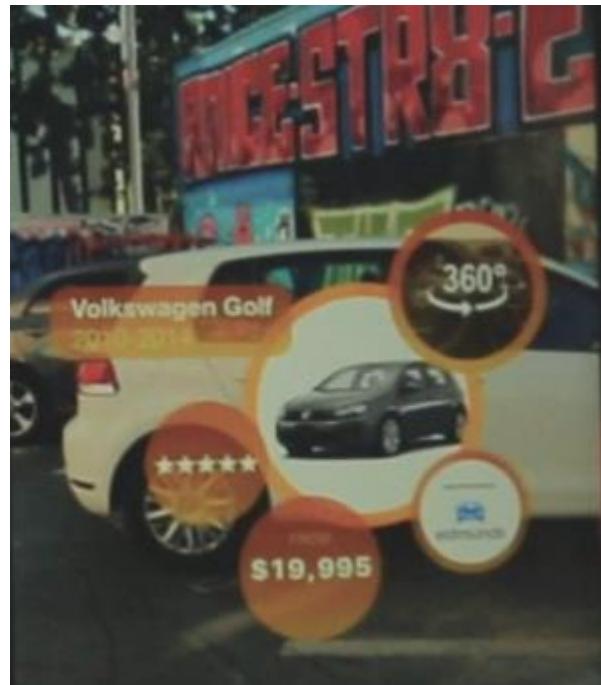


Рисунок 3 – Пример работы приложения Blipper

Однако Авто.ру и Intlab Auto MMR могут без проблем распознать транспортное средство с любой его стороны. Нам для такого необходимо будет добавить соответствующий вид в базу данных.

### 1.3 Задача распознавания образов

Распознавание человеком конкретных образов можно рассматривать как психофизиологическую задачу, связанную с процессом взаимодействия индивида с определенным физическим раздражителем. Фактически, распознавание человеком образов основывается на схожести однотипных объектов.

Очевидное решение задачи распознавания образов заключается в применении к отдельным предъявленным образам ряда простых тестов для выявления признаков каждого класса. Совокупность всех этих тестов должна распознавать все допустимые образы из разных классов.

Если следовать такому интуитивному подходу, то построение автоматической системы распознавания образов может показаться достаточно

простой задачей. Однако, не существует общей теории, которая бы позволяла определить, какие из всего множества возможных тестов следует применить к предъявленным образам. И подобный подход чрезмерно зависит от опыта и технической интуиции разработчика и поэтому часто не дает удовлетворительного решения задач распознавания образов, встречающихся в практической деятельности [2].

Функциональная схема систем распознавания представлена на рисунке 4.

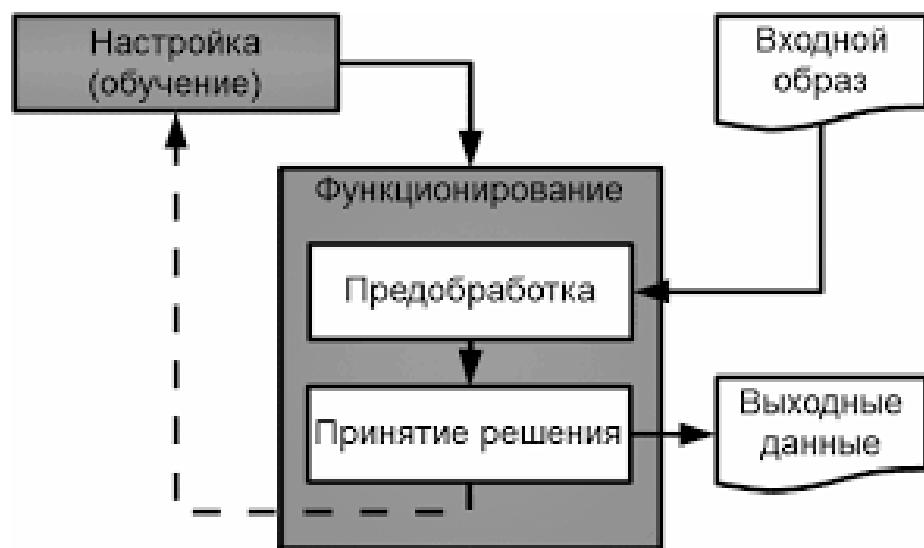


Рисунок 4 – Функциональная схема систем распознавания

Входные данные, которые подлежат распознаванию, подаются на вход системы и подвергаются предобработке для их дальнейшего преобразования в необходимый для следующего этапа вид или для выделения из них необходимых отличительных черт. В последующем на этапе принятия решения над обработанным массивом данных производится ряд вычислений и на основе их результатов формируется ответ, содержащий ожидаемые от системы сведения о входных данных. Содержание входных и выходных данных зависит от задачи, которую решает система. Например, если входом служит изображение транспортного средства, то в качестве выходных данных система может выдать его марку, модель или иную характеристику.

Кроме описанных этапов функционирования, системы распознавания предусматривают свою настройку на множество возможных входных данных; этот

этап называют этапом обучения системы. Целью обучения системы является формирование в ее памяти набора сведений, необходимых для распознавания предполагаемого класса входных данных.

На этапе предобработки решается задача формирования формализованного описания объектов распознавания, подходящего для использования алгоритмами распознавания. Как правило, исходные данные о наблюдаемых объектах, поданных в форме, неподходящей для распознавания. Это могут быть растровые изображения, аудиофайлы, статистические данные (наборы чисел), видеозаписи и прочее. Некоторые алгоритмы распознавания требуют более высокоуровневого представления. Это приводит к необходимости произвести одно или более преобразований исходных данных.

Этап принятия решения является наиболее существенным в процессе работы системы распознавания с точки зрения ее характеристики в целом. Т.е. задача, которая решается на данном этапе, во многом определяет назначение системы.

Основными классами задач, решаемых на этапе принятия решения являются распознавание и классификация.

Распознавание — распределение предъявляемых объектов по определенным классам посредством применения известных правил классификации. Это одна из наиболее характерных задач для систем распознавания. Перед тем, как система сможет выполнять данную функцию, предполагается ее обучение на множестве разнообразных примеров — обучающей выборке объектов распознавания.

Классификация — это разбиение множества объектов на непересекающиеся классы по их формализованным описаниям. Данная задача решается в случаях, когда от системы не требуется отнести входные образы к каким-либо определенным классам, а требуется лишь способность различать их каким-либо способом по определенным признакам [3].

Исходя из вышесказанного можно сделать вывод, что при разработке нашего ПО необходимо решить задачу классификации.

## 1.4 Архитектуры нейронных сетей

### 1.4.1 Виды архитектур нейронных сетей

Сегодня нейронные сети имеют множество различных архитектур, среди которых самыми распространенными являются следующие:

1. Нейронные сети прямого распространения.
2. Рекуррентные нейронные сети.
3. Сверточные нейронные сети.

Сеть прямого распространения сигнала - нейронная сеть без обратных связей.

В этой сети распространение сигнала односторонне. От входного слоя сигнал обрабатывается слой за слоем в направлении выхода. Через известное число шагов на выходном слое появляется ответ сети.

Сети прямого распространения являются хорошо изученными и относительно простыми в реализации. Их недостатком является необходимость большого числа нейронов для выполнения сложных задач. Схема нейронной сети прямого распространения представлена на рисунке 5.

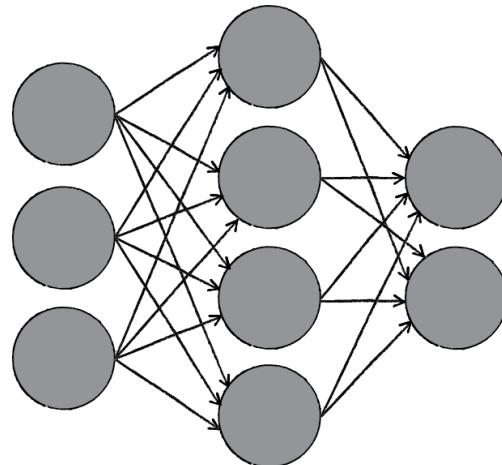


Рисунок 5 – Схема нейронной сети прямого распространения

Рекуррентные сети — это глубокие сети, в которых присутствуют обратные связи. Это значит, что присутствует хотя бы один слой, сигналы с которого поступают на него же, либо на один из предыдущих слоев. Нейроны участвуют в обработке информации многократно, что позволяет использовать динамические свойства сети. Такие сети позволяют сократить число нейронов. На основе рекуррентных сетей разработаны различные модели ассоциативной памяти. Также

данный вид нейронных сетей используется для задач распознавания речи. Схема рекуррентной нейронной сети представлена на рисунке 6.

Сверточные сети получили свое название из-за наличия операции свертки, в которой каждый фрагмент изображения умножается на матрицу свертки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения. Они используются в основном для обработки изображений, иногда для аудио и других видов входных данных. Типичным способом их применения является классификация изображений [4]. Схема сверточной нейронной сети представлена на рисунке 7.

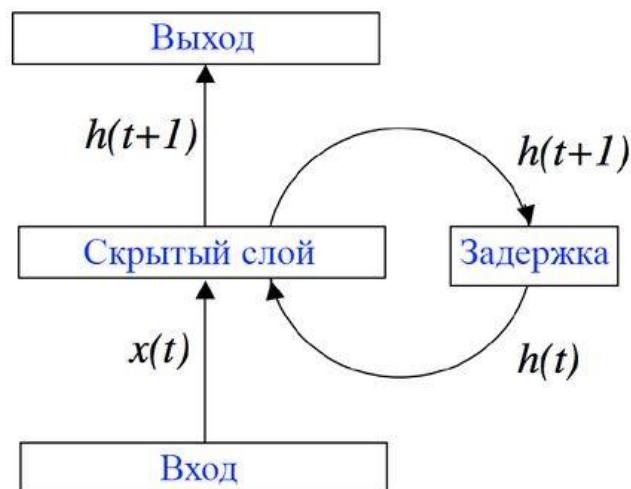


Рисунок 6 – Схема рекуррентной нейронной сети

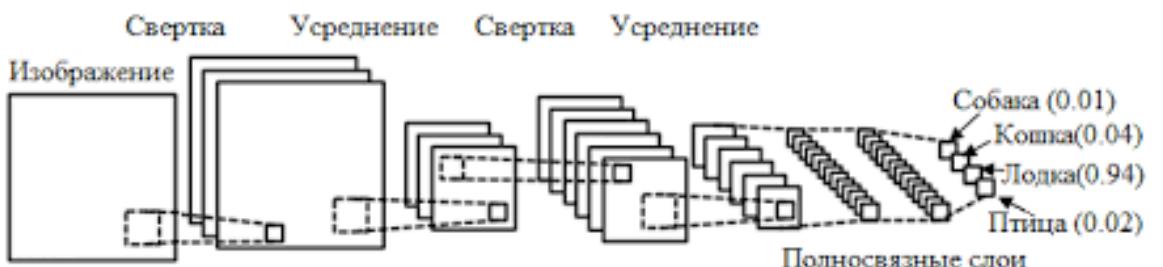


Рисунок 7 – Схема сверточной нейронной сети

Исходя из вышесказанного можно сделать вывод, что для задачи классификации изображений будет лучше использовать сверточные нейронные сети. Рассмотрим принцип их работы более подробно.

### 1.4.2 Сверточные нейронные сети

Сверточная нейронная сеть состоит из следующих типов слоев:

1. Входной слой.
2. Сверточные слои.
3. Субдискретизирующие слои.
4. “Обычные” слои (полносвязные).
5. Выходной слой.

Входной слой или входные данные представляет собой изображение, которое нейросети предстоит проанализировать. Входной слой учитывает двумерную топологию изображений и состоит из нескольких карт (матриц). Карта может быть одна в случае, если изображение представлено в оттенках серого, иначе их три, где каждая карта соответствует изображению с конкретным каналом (красным, синим и зеленым) [5].

В сверточном слое нейронной сети в операции свертки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале по входному изображению), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются общие веса матрица весов, которую также называют набором весов или ядром свертки. Она построена таким образом, что графически кодирует какой-либо один признак, например, наличие наклонной линии под определенным углом. Тогда следующий слой, получившийся в результате операции свертки такой матрицей весов, показывает наличие данной наклонной линии в обрабатываемом слое и ее координаты, формируя так называемую карту признаков. Естественно, в сверточной нейронной сети набор весов не один, а целая гамма, кодирующая всевозможные линии и дуги под разными углами. При этом такие ядра свертки не закладываются исследователем заранее, а формируются самостоятельно путем обучения сети классическим методом распространения ошибки. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многомерной (много независимых карт признаков на одном слое).

Схема операции свертки представлена на рисунке 8.

Операция субдискретизации выполняет уменьшение размерности сформированных карт признаков. В данной архитектуре сети считается, что информация о факте наличия искомого признака важнее точного знания его координат, поэтому из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон карты признаков уменьшенной размерности. Также иногда применяют операцию нахождения среднего между соседними нейронами. За счет данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения. Схема операции субдискретизации представлена на рисунке 9.

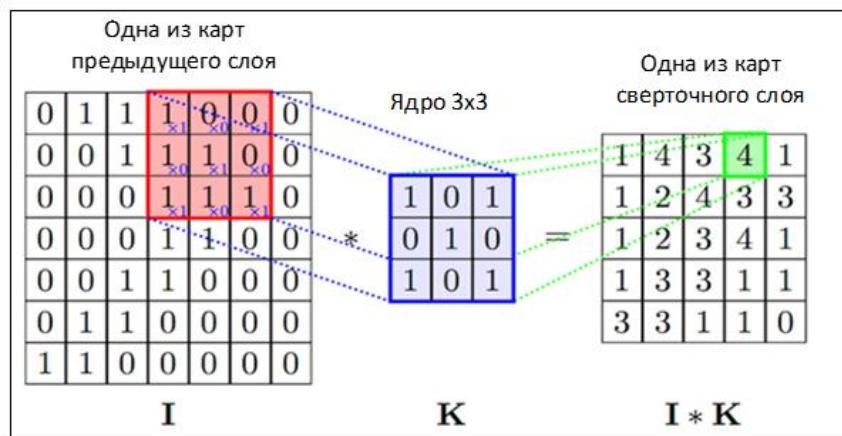


Рисунок 8 – Схема операции свертки

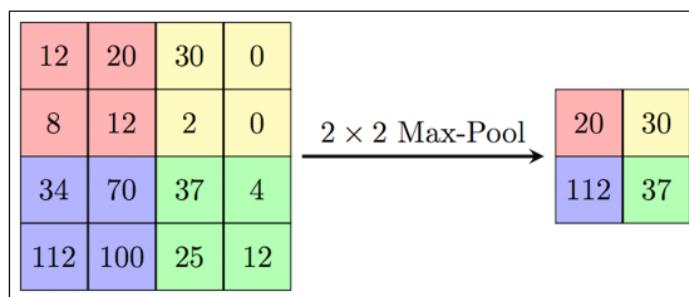


Рисунок 9 – Схема операции субдискретизации

Последний из типов слоев это слой обычного многослойного персептрона. Цель слоя – классификация, моделирует сложную нелинейную функцию, оптимизируя которую, улучшается качество распознавания.

Выходной слой связан со всеми нейронами предыдущего слоя. Количество нейронов на нем соответствует количеству распознаваемых классов.

## 1.5 Методы обучения нейронных сетей

Существует несколько способов обучения нейронной сети:

1. Обучение с учителем.
2. Обучение без учителя.
3. Обучение с подкреплением.

### 1.5.1 Обучение с учителем

Обучение с учителем предполагает наличие полного набора размеченных данных для тренировки модели на всех этапах ее построения.

Наличие полностью размеченного набора данных означает, что каждому примеру в обучающем наборе соответствует ответ, который алгоритм и должен получить. Таким образом, размеченный набор данных из фотографий цветов обучит нейронную сеть, где изображены розы, ромашки или нарциссы. Когда сеть получит новое фото, она сравнит его с примерами из обучающего набора данных, чтобы предсказать ответ.

В основном обучение с учителем применяется при решении задач классификации.

В задачах классификации алгоритм предсказывает дискретные значения, соответствующие номерам классов, к которым принадлежат объекты. В обучающем наборе данных с фотографиями транспортных средств каждое изображение будет иметь соответствующую метку — «автомобиль», «автобус» или «велосипед». Качество алгоритма оценивается тем, насколько точно он может правильно классифицировать новые фото различными транспортными средствами.

### 1.5.2 Обучение без учителя

Идеально размеченные и чистые данные достать нелегко. Поэтому иногда перед алгоритмом стоит задача найти заранее неизвестные ответы. В таких задачах и применяется данный тип обучения.

В обучении без учителя у модели есть набор данных, и нет явных указаний, что с ним делать. Нейронная сеть пытается самостоятельно найти корреляции в данных, извлекая полезные признаки и анализируя их.

В обучении без учителя сложно вычислить точность алгоритма, так как в данных отсутствуют правильные ответы или метки. Но размеченные данные часто ненадежные или их слишком дорого получить. В таких случаях, предоставляя модели свободу действий для поиска зависимостей, можно получить хорошие результаты [6].

### **1.5.3 Обучение с подкреплением**

Обучение с подкреплением действует по принципу, в котором агенты ИИ пытаются найти оптимальный способ достижения цели или улучшения производительности для конкретной среды.

При принятии решения агент изучает обратную связь, новые тактики и решения способные привести к большему выигрышу. Этот подход использует долгосрочную стратегию — так же как в шахматах: следующий наилучший ход может не помочь выиграть в конечном счете. Поэтому агент пытается максимизировать суммарную награду.

Это итеративный процесс. Чем больше уровней с обратной связью, тем лучше становится стратегия агента. Такой подход особенно полезен для обучения роботов, которые управляют автономными транспортными средствами или инвентарем на складе [7].

Так же, как и ученики в школе, каждый алгоритм учится по-разному. Но благодаря разнообразию доступных методов, вопрос в том, чтобы выбрать подходящий и научить вашу нейронную сеть разбираться в среде.

Исходя из вышесказанного можно сделать вывод, что обучение нейронной сети с учителем решает установленную нами задачу. Поэтому этот тип обучения и будет использоваться.

## 1.6 Обученные нейронные сети

С увеличением размера обучающей выборки растет потребность в ресурсах машины, на которой будет производиться обучение. Обучение нейронной сети с нуля может занимать от одного дня до месяцев непрерывной работы компьютера. В таком случае на помощь приходят уже обученные на крупном наборе данных нейронные сети.

В настоящее время существует множество обученных нейронных сетей в свободном доступе. Преимущество использования подобных решений очевидно – вы можете сэкономить время не только на написании кода для стартовой архитектуры сети или подборе слоев, но и на обучении.

Однако, чтобы нейронная сеть в дальнейшем решала конкретную, необходимую нам задачу, ее потребуется модифицировать и переобучить на новом наборе данных с обновленным классификатором. В связи с этим исходная архитектура нейронной сети и решаемая ей изначальная задача должны быть максимально приближены к нашей.

В качестве кандидатов были рассмотрены пять наиболее распространенных и ресурсоемких моделей сверточных нейронных сетей: Dlib, FaceNet, OpenFace, VggFace, DeepFace.

Сравнение будет проводиться путем сравнения точности результата работы нейронных сетей на различных наборах данных в комбинации с различными детекторами лиц. Размер моделей (в Мб) не будет рассматриваться при сравнении, т.к. все они удовлетворяют поставленным требованиям размерности. Сравнение точности моделей нейронных сетей представлено в таблице 2.

Для проведения сравнения использовались наборы данных Labeled Faces in the Wild и Masked. LFW представляет собой огромный набор лиц людей всех этничностей и возрастов, а Masked – это набор данных людей, на которых надеты медицинские маски (т.е. часть лица скрыта). Для набора данных LFW использовались различные детекторы лиц, такие как MTCNN, Dlib и PWC, а также два готовых набора уже размеченных лиц.

Среди представленных моделей самую низкую точность работы имеет OpenFace. Помимо этого, у данной нейронной сети отсутствует поддержка GPU, а значит, что запустить ее можно только на CPU, что не подходит в соответствии с поставленными требованиями.

Таблица 2 – Сравнение точности моделей нейронных сетей

	LFW1	LFW2 (mtcnn)	LFW3 (dlib)	LFW4	LFW5 (pwc)	Masked
Dlib	99.38%	95.18%	97.90%	87.53%	99.38%	-
FaceNet	99.63%	97.17%	98.22%	99.17%	99.63%	67.48%
OpenFace	93.80%	93.29%	93.29%	93.29%	92.92%	63.18%
VggFace	97.78%	-	-	-	98.78%	68.17%
DeepFace (FB)	97.35%	-	-	-	98.37%	63.78%

Следом идет нейронная сеть, разработанная Facebook – DeepFace. Сама модель достаточно старая (2015 год), а точность работы около человеческая: другие нейронные сети уже давно превзошли этот показатель.

Модель VggFace имеет высокую точность работы, однако уступает Dlib и FaceNet. Нейронная сеть работает на основе нейронной сети VGG и имеет на выходе вектор размерности 2048D.

Модель FaceNet имеет самую высокую точность работы с различными средствами выделения лиц. Данная нейронная сеть поддерживает кластеризацию по умолчанию и имеет на выходе вектор размерностью 128D.

Dlib, как и FaceNet, имеет очень высокую точность работы, однако ее рекомендуется использовать со встроенными в модуль системами выделения лиц: с использованием аналогов точность распознавания значительно падает. Этот фактор является решающим – для системы распознавания транспортных средств невозможно использовать систему выделения лиц.

Подводя итог, можно сказать, что нейронная сеть FaceNet является лучшим выбором из рассмотренных.

## 1.7 Числовое представление изображения

Измерения, используемые для классификации образов, называются признаками. Признак – это некоторое количественное измерение объекта произвольной природы. Совокупность признаков, относящихся к одному образу, называется вектором признаков. Вектора признаков принимают значения в пространстве признаков. В рамках задачи распознавания считается, что каждому образу ставится в соответствие единственное значение вектора признаков и наоборот: каждому значению вектора признаков соответствует единственный образ.

Вектор признаков представляет собой такой числовой вектор, который был получен из сущности изображения. Вектором признаков размерности  $k$  называют список из  $k$  чисел, в котором порядок чисел строго определен. Например, трехмерным вектором можно считать  $(2.7, 1.1, 5.55)$ , а  $(1., 0., 2., 0.2, 0., 0., 10.1, 11.1)$  – восьмимерным числовым вектором.

Таким образом, получаемый в результате работы нейронной сети вектор – это набор чисел, полученный с помощью многократного выполнения операций свертки и усреднения.

Нейронная сеть формирует эти числа в результате выделения признаков (которые она научилась выделять) из изображения.

Приведу простой пример в контексте разрабатываемой системы. Представим, что у одного автомобиля фары окружной формы, а у другого прямоугольной. В таком случае, в большом векторе представления, нейронная сеть выделит форму фар как признак, и векторы, полученные в результате работы сети, будут отличаться на определенном участке. Округлые фары будут представлены последовательностью чисел  $(2, 0, 1)$ , а прямоугольные –  $(1, 1, 1)$ .

Конечно, в реальности нейронная сеть сама определит, какими числами она будет описывать признаки, а сам вектор будет в разы больше, чем представлено выше.

## 1.8 Функция потерь нейронной сети

Функция потерь находится в центре нейронной сети. Она используется для расчета ошибки между реальными и полученными ответами. Наша глобальная цель — минимизировать эту ошибку. Таким образом, функция потерь эффективно приближает обучение нейронной сети к этой цели. Для решения нашей задачи была выбрана потеря триплетов (triplet loss).

Потеря триплетов — это функция потерь для алгоритмов машинного обучения, при которой эталонный ввод (якорь), сравнивается с совпадающим вводом (положительный ввод) и несовпадающим вводом (отрицательный ввод). В процессе обучения расстояние от якоря до положительного ввода сводится к минимуму, а расстояние между якорем и отрицательным вводом увеличивается.

За счет данной функции ошибки модель выставляется таким образом, что расстояние между похожими вводами будет меньше, чем между разными.

Функцию потерь триплетов можно описать с помощью Евклидова расстояния:

$$L(A, P, N) = \max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha), \quad (1)$$

где A — эталонный ввод (якорь);

P — положительный ввод;

N — отрицательный ввод.

На рисунке 10 на примере человеческих лиц представлен процесс формирования триплета для нейронной сети.

Исходя из принципа формирования триплетов для обучения, становится ясно, что для корректной работы выбранной функции потерь необходимо быть уверенным, что при обучении на каждой эпохе будут изображения как минимум из двух разных классов (один класс для якоря и положительного ввода, второй класс — для негативного ввода) — если в одну эпоху хоть на одну итерацию попадут изображения только из одного класса, то нейронная сеть не сможет найти негативный ввод для формирования триплета и, следовательно, все обучение будет провалено.

Чтобы избежать возникновения подобной ситуации для генерации выборок данных на каждой итерации эпохи будет использована специальная библиотека tensorflow – kerasgen. Генератор выборок, встроенный в данную библиотеку позволяет гарантировать возможность создания триплетов на каждой итерации каждой из эпох обучения.

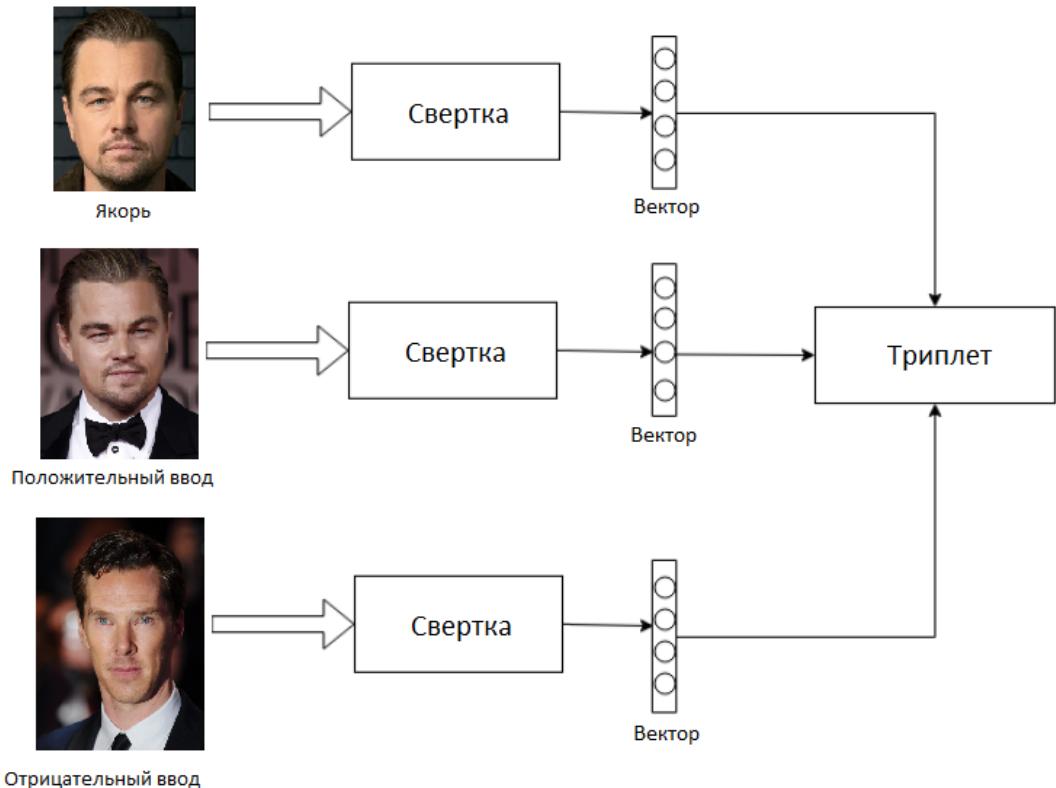


Рисунок 10 – Формирование tripleta

На рисунке 11 представлен процесс обучения с использованием функции потерь триплетов.

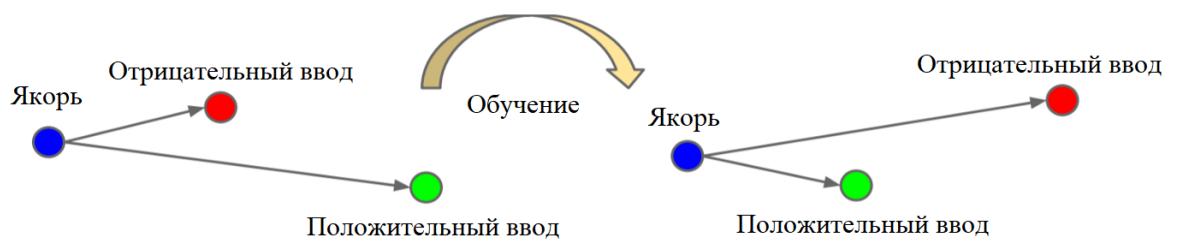


Рисунок 11 – Обучение с использованием функции потерь triplets

## 1.9 Общая структура системы

Разрабатываемая система может быть представлена в виде совокупности структур, изображенных на рисунке 12.



Рисунок 12 – Общая структура системы

## 1.10 Функциональные и нефункциональные требования

Функциональные требования:

- модель должна уметь работать с изображениями (прием и свертка изображения);
- в результате работы модель должна выдавать вектор (числовое представление изображения, на основе которого выполняется классификация транспортного средства);
- система предоставляет возможность получения данных о транспортном средстве на основе его изображения;
- система должна позволять добавлять, изменять и удалять данные из базы данных;
- система должна позволять подключить другую базу данных.

Нефункциональные требования:

- модель должна уметь работать с изображениями типа jpg, jpeg, png;
- в результате работы модель должна выдавать вектор представления размерностью 128D (такой формат данных показывает лучшее соотношение точности и скорости работы нейронной сети);
- исполняемый файл модели не должен превышать размерности 200 Мб;

- модель должна быть совместима с компактными компьютерами серии NVIDIA Jetson (именно Jetson в конечном итоге будет использоваться для выполнения нейронной сети);
- модель должна запускаться на GPU ускорителе (GPU в десятки раз быстрее CPU; медленные нейронные сети, выполняемые на CPU, используются только в узкоспециализированных задачах, где не важна скорость работы модели – наша задача к таким не относится);
- время распознавания одного изображения не должно превышать 0.9 секунды (данная цифра исходит из числа событий, которые нужно обработать за 1 час; в пиковой ситуации с камеры приходит 1800 траекторий (проезд транспортного средства мимо камеры) – выходит на обработку одной траектории должно быть выделено не более 2 секунд; нагрузка на выполнение другого ПО составляет приблизительно 1.1 секунды, а значит на выполнение нашей нейронной сети может быть отведено максимум  $2 - 1.1 = 0.9$  секунд).

## **1.11 Выбор методов разработки**

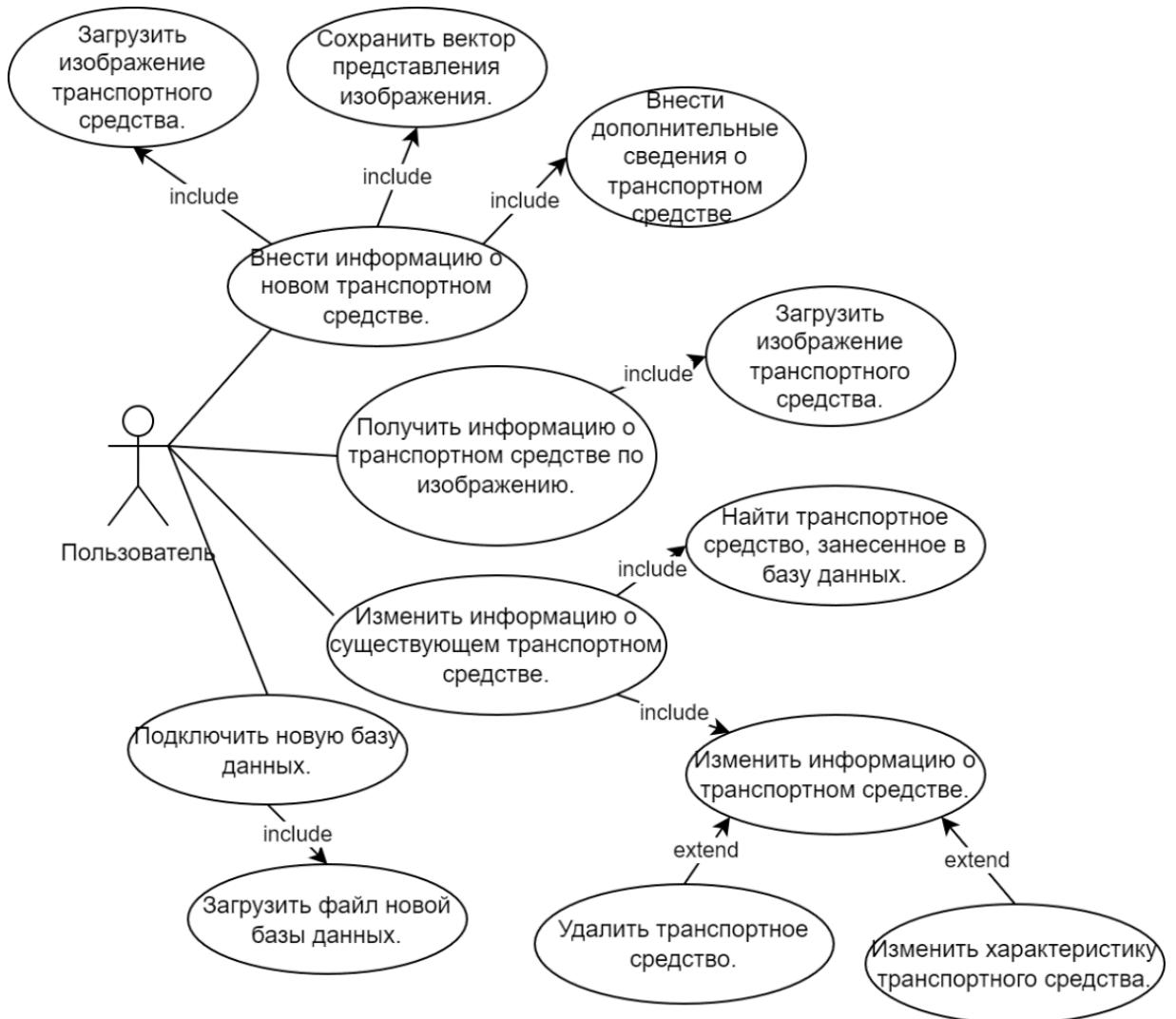
Для разработки модели нейронной сети был выбран язык программирования Python – он достаточно прост в понимании, имеет отличную производительность при обработке данных, а также именно под Python написано огромное количество фреймворков для машинного обучения, которые упрощают процесс написания кода и время на разработку. С помощью данного языка будет происходить сбор данных для обучения модели, обучение модели и ее взаимодействие с базой данных. Для разработки БД был выбран язык запросов SQL. В качестве СУБД будет использоваться SQLite – благодаря архитектуре движка данную СУБД возможно использовать на как на различных ОС, так и на разных встраиваемых системах.

Обучение модели будет проводиться с помощью библиотеки языка Python - Keras. Данная библиотека предоставляет набор готовых методов для большинства алгоритмов, используемых при обучении нейронных сетей.

## 2 МОДЕЛИРОВАНИЕ СИСТЕМЫ РАСПОЗНАВАНИЯ МАРОК И МОДЕЛЕЙ ТРАНСПОРТНЫХ СРЕДСТВ

### 2.1 Моделирование взаимодействия системы со внешней средой

На основе функциональных требований была разработана диаграмма вариантов использований, представленная на рисунке 13.



Согласно диаграмме, система позволяет пользователю использовать три функции: добавить новый объект в БД, выполнить распознавание транспортного средства по его изображению или изменить информацию о существующем в базе данных транспортном средстве.

Добавление нового объекта в БД происходит путем загрузки изображения с транспортным средством в систему. Данное изображение получает нейронная сеть, после чего выполняет преобразование изображения в его числовое представление (вектор). После этого пользователь должен ввести необходимые данные о транспортном средстве на изображении. Выполнив все действия, БД сохраняет числовое представление изображение и остальные данные, введенные пользователем, в одну новую строку.

Чтобы провести распознавание и получить информацию об транспортном средстве, пользователю необходимо просто загрузить соответствующее изображение в систему. Нейронная сеть также преобразует изображение в его числовое представление, а затем будет найден самый похожий вариант из существующих в БД.

Чтобы изменить информацию о существующем транспортном средстве, пользователю необходимо указать искомое название модели транспортного средства. Если подобная модель существует в базе данных, пользователь сможет изменить ее параметры или вовсе удалить модель из базы данных.

Чтобы загрузить новую базу данных, пользователь должен выбрать файл с расширением .db для загрузки.

## **2.2 Моделирование взаимодействия элементов внутри системы**

Для пояснения вариантов использования для каждого варианта была разработана диаграмма последовательности.

Диаграмма последовательности отображает последовательность действий в рамках взаимодействия системы с пользователем и последовательность действий объектов системы и внешних систем в рамках выполнения прецедента, а также жизненный цикл объектов.

Диаграмма последовательности для добавления нового объекта представлена на рисунке 14.

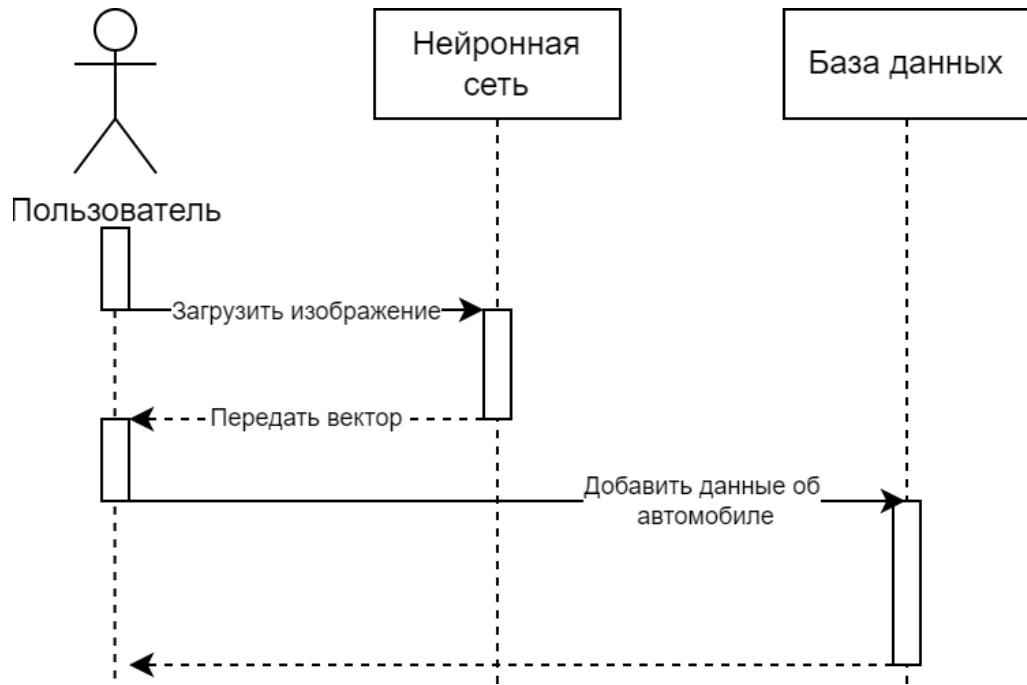


Рисунок 14 – Взаимодействие элементов системы при добавлении транспортного средства

На первом шаге пользователь загружает изображение, которое получает нейронная сеть и переводит его в числовое представление. После этого управление возвращается пользователю, который должен ввести данные о новом транспортном средстве, изображение которого он загрузил ранее. Затем все введенные данные вместе с числовым представлением изображения заносятся в базу данных.

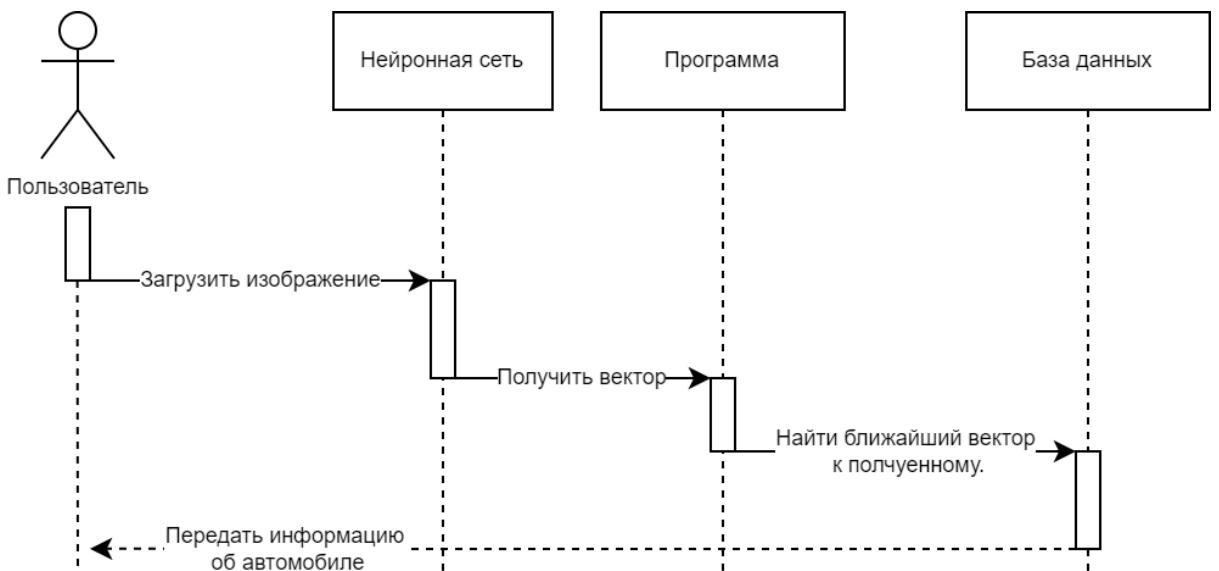


Рисунок 15 – Взаимодействие элементов системы при распознавании транспортного средства по его изображению

На рисунке 15 представлена диаграмма последовательности для распознавания транспортного средства по его изображению.

На первом шаге пользователь загружает изображения транспортного средства, информацию о котором он хочет получить. Это изображение получает нейронная сеть, которая переводит изображение в числовое представление. Затем этот вектор передается программе, которая делает запрос к БД, стараясь найти наиболее близкий по расстоянию к полученному вектор [8]. После нахождения такого, база данных передает информацию об объекте с найденным вектором.

На рисунке 16 представлена диаграмма последовательности для изменения информации о существующем транспортном средстве.

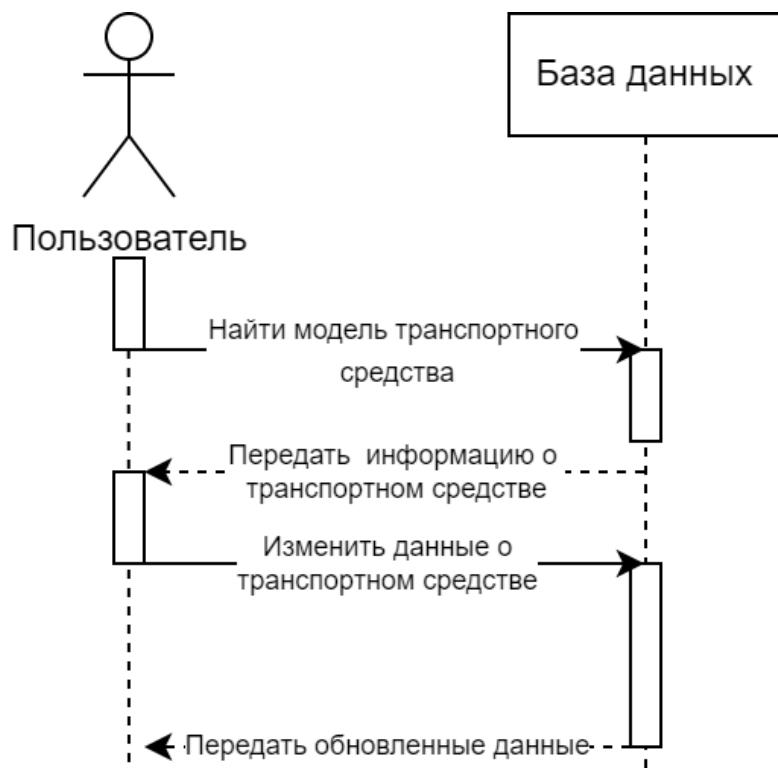


Рисунок 16 - Взаимодействие элементов системы при изменении данных о транспортном средстве

На первом шаге пользователь вводит название модели искомого транспортного средства. Затем база данных находит похожую модель и возвращает информацию о ней пользователю. Пользователь изменяет данные и передает обновленные данные обратно в БД. В конце база данных выводит пользователю обновленные данные.

## 2.3 Моделирование процессов системы

Для пояснения вариантов использования для каждого варианта была разработана диаграмма деятельности.

Диаграмма деятельности (активностей) отражает динамические аспекты поведения системы. По существу, эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

На рисунке 17 представлена диаграмма деятельности для добавления нового объекта.

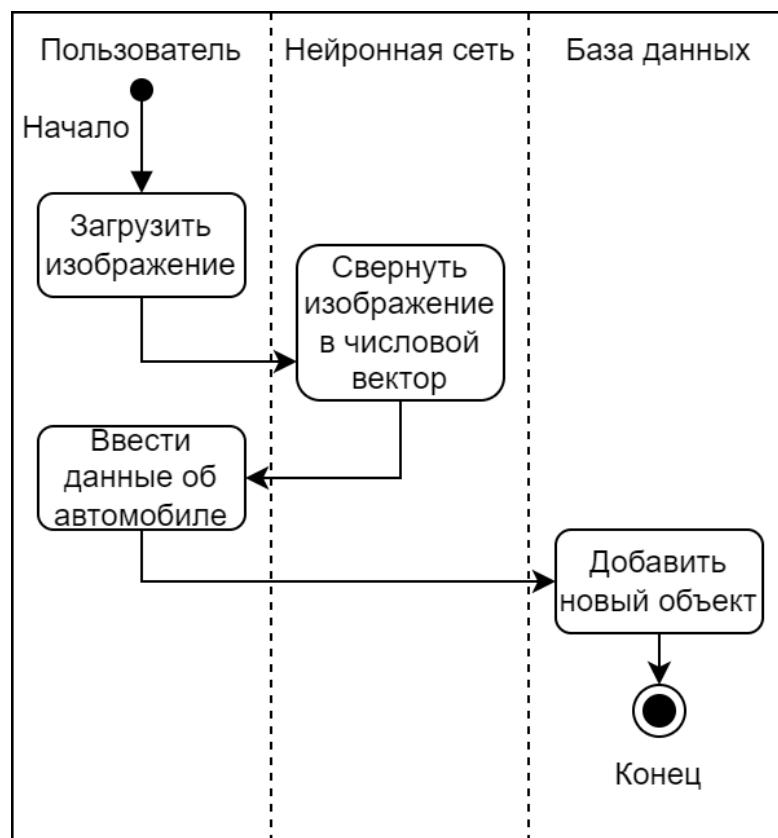


Рисунок 17 – Описание процесса добавления нового транспортного средства

В самом начале пользователь должен загрузить изображение транспортного средства, который он хочет добавить. Затем нейронная сеть выполнит преобразование переданного изображения в числовой вектор. После этого пользователь должен будет ввести необходимые данные. И в конце база данных, на основе вектора и данных, введенных пользователем, сформирует и сохранит в себе новый объект.

На рисунке 18 представлена диаграмма деятельности для распознавания транспортного средства по его изображению.

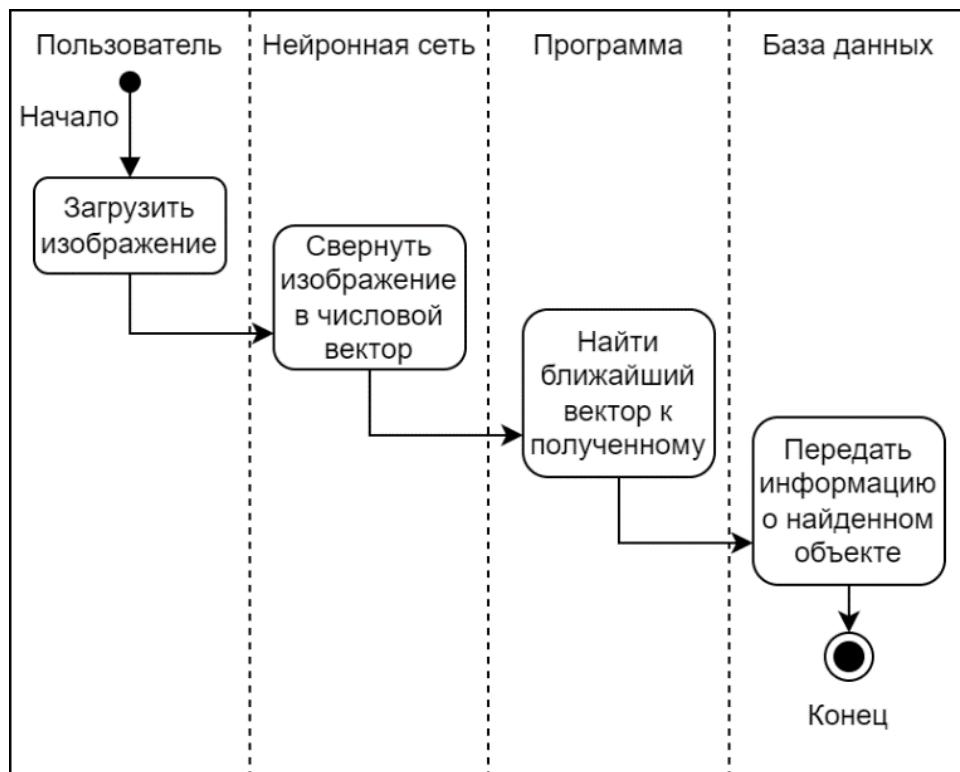


Рисунок 18 – Описание процесса распознавания транспортного средства

В самом начале пользователь должен загрузить изображения транспортного средства, информацию о котором он хочет получить. Затем нейронная сеть выполнит преобразование переданного изображения в числовой вектор. После этого программа сформирует и сделает запрос к БД, чтобы найти в ней ближайший по расстоянию вектор к полученному. В конце база данных передаст информацию об найденном объекте.

На рисунке 19 представлена диаграмма деятельности для изменения информации о существующем транспортном средстве.

В самом начале пользователь должен ввести название искомой модели. Затем база данных найдет запрошенную модели и передаст информацию о ней. После этого пользователь сможет изменить данные о транспортном средстве. После внесения изменений база данных сохраняет все изменения параметров модели и передает пользователю обновленную информацию о транспортном средстве.

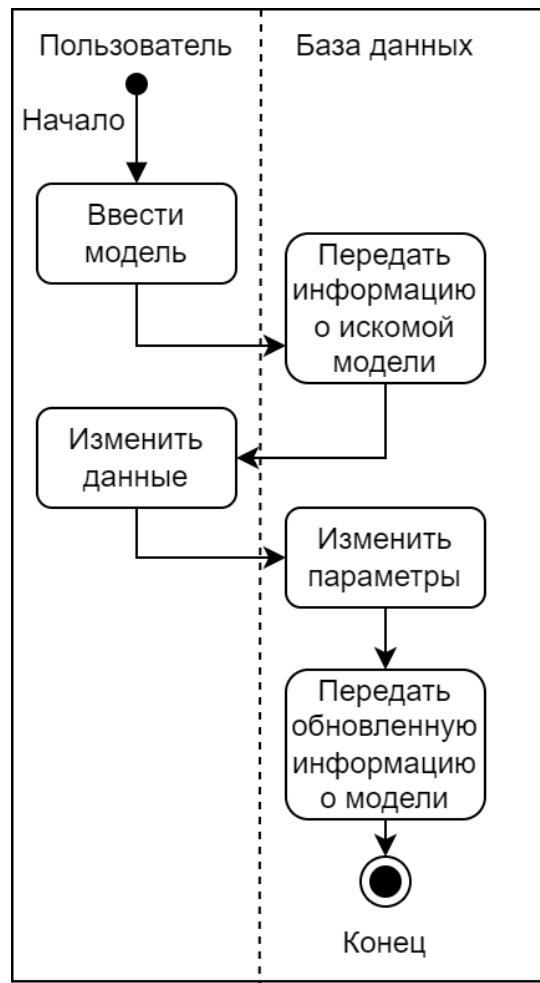


Рисунок 19 – Описание процесса изменения информации о транспортном  
средстве

#### 2.4 Концептуальная модель БД

На концептуальной модели в визуально удобном виде прописываются связи между объектами и их характеристиками.

Для единообразия проектирования концептуальных моделей введены следующие понятия:

- сущность – фактическая вещь или объект;
- атрибут – характеристика объекта;
- отношение – показывает связь между объектами.

Основным объектом проектируемой базы данных является Транспортное средство.

Сущность объект характеризуется следующими атрибутами:

- модель;
- высота кузова;
- длина кузова;
- ширина кузова;
- высота кузова;
- колесная база (расстояние между центрами колес);
- клиренс;
- ширина передней колеи;
- ширина задней колеи;
- вектор представления.

На основании этого была построена концептуальная модель базы данных, представленная на рисунке 20.



Рисунок 20 – Концептуальная модель базы данных

## 2.5 Логическая модель БД

Логическая модель находится на более низком уровне предметной области, нежели концептуальная. Она представляет собой модель базы данных, которая не привязана к определенной СУБД. В ней выделяют основные объекты БД и определяют связи между этими объектами.

Логическая модель ограничивается решениями, принятыми при разработке концептуальной модели. Таким образом устанавливаются пределы, в рамках которых мы можем принимать решения при создании логической модели.

Приняв за основу концептуальную схему, была построена логическая модель. В нее были включены сущности: “ID транспортного средства”, “Модель”, “Длина кузова”, “Ширина кузова”, “Высота кузова”, “Колесная база”, “Клиренс”, “Ширина передней колеи”, “Ширина задней колеи”, “Вектор представления”.

База данных состоит из одной таблицы и ее последующее усложнение не целесообразно.

Логическая модель базы данных представлена на рисунке 21.

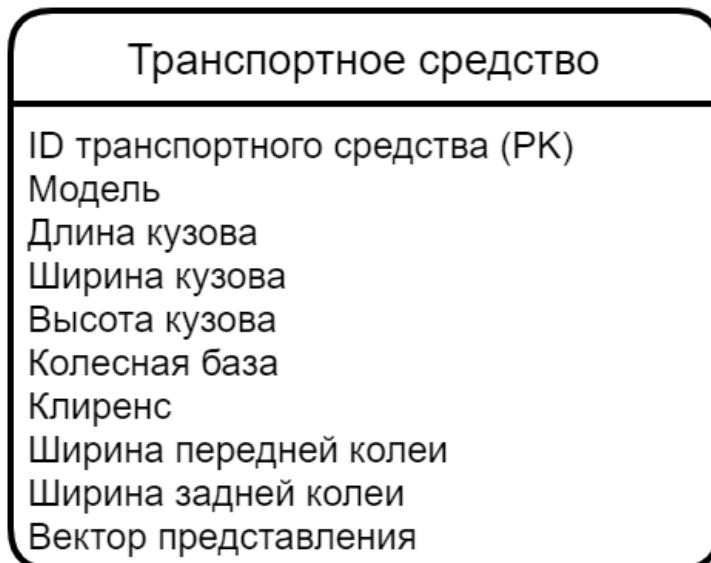


Рисунок 21 – Логическая модель БД

## 2.6 Построение концептуальной модели ПО

Концептуальная модель отражает семантику предметной области в виде совокупности сущностей, их характеристик и связей. Концептуальное

моделирование предметной области удобно выполнять посредством диаграммы классов.

На рисунке 22 представлена концептуальная модель ПО в виде диаграммы классов.

Класс “Классификатор” является средством взаимодействия ПО с нейронной сетью. Он обладает закрытыми полями:

- “Путь до модели” – содержит путь до исполняемой модели нейронной сети в файловой системе;
- “Минимальная дистанция близости векторов” – числовое значение, которое разделяет известные и неизвестные вектора по близости;
- “Размеры входного изображения” – параметр для преобразования размеров изображения для подачи в нейронную сеть.

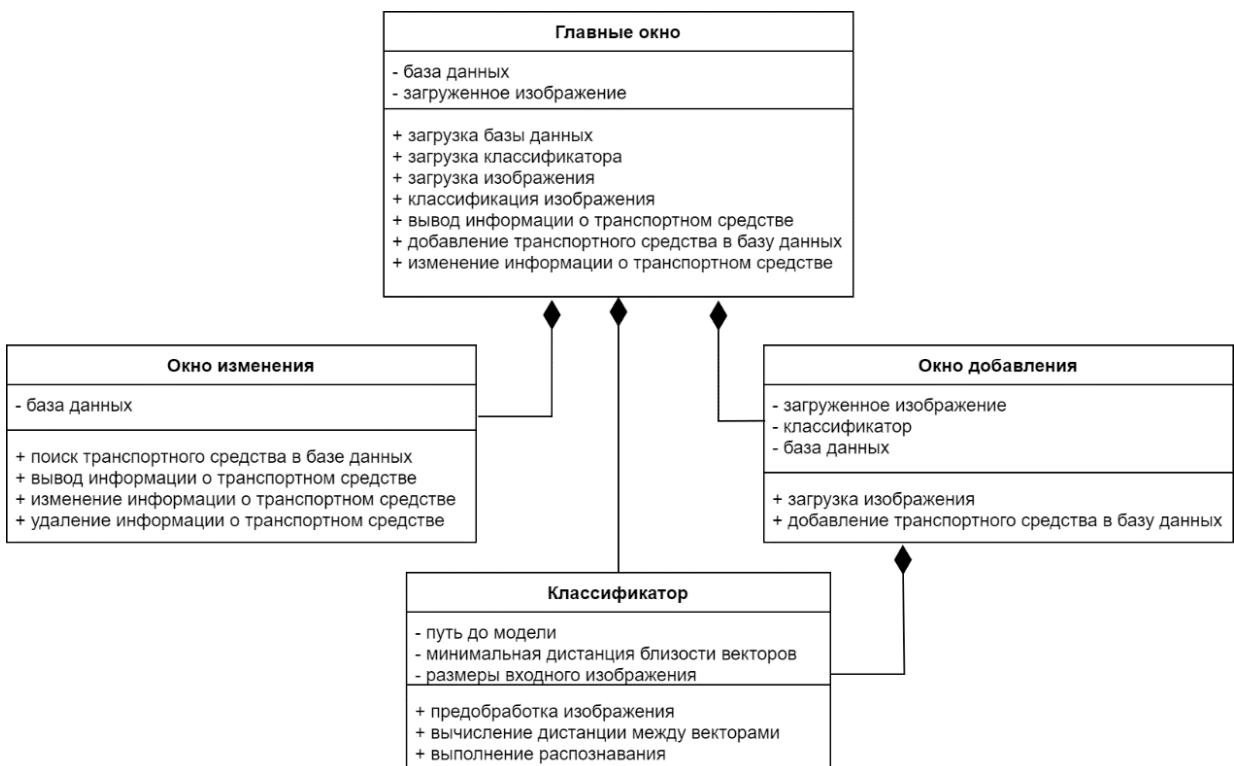


Рисунок 22 – Концептуальная модель ПО в виде диаграммы классов

Класс “Классификатор” обладает следующими методами:

- “Предобработка изображения” – производит обработку изображения для подачи в нейронную сеть;

- “Вычисление дистанции между векторами” – определяет значение близости двух векторов;

- “Выполнение распознавания” – выполняет классификацию изображения и передает результирующий вектор.

Класс “Окно добавления” отвечает за окно интерфейса, которое позволяет пользователю добавить новое транспортное средство в базу данных. Класс обладает следующими закрытыми полями:

- “Загруженное изображение” – содержит путь до изображения, которое выбрал пользователь;

- “Классификатор” – экземпляр класса “Классификатор”;
- “База данных” – установленное соединение с базой данных.

Класс “Окно добавления” обладает следующими методами:

- “Загрузка изображения” – загружает путь до добавляемого изображения из файловой системы;

- “Добавление транспортного средства в базу данных” – добавляет новую запись в базу данных.

Класс “Окно изменения” отвечает за окно интерфейса, которое позволяет пользователю изменить информацию о существующем в базе данных транспортном средстве. Класс обладает одним закрытым полем “База данных” – которое содержит установленное соединение с базой данных.

Класс “Окно изменения” обладает следующими методами:

- “Поиск транспортного средства в базе данных” – находит в базе данных запись о транспортном средстве;

- “Вывод информации о транспортном средстве” – выводит информацию о транспортном средстве из базы данных;

- “Изменение информации о транспортном средстве” – изменяет информацию о транспортном средстве в базе данных;

- “Удаление информации о транспортном средстве” – удаляет запись о транспортном средстве из базы данных целиком.

Класс “Главное окно” отвечает за главное окно интерфейса, которое позволяет пользователю выполнить классификацию изображения или перейти к окнам добавления или изменения. Он обладает закрытыми полями:

- “База данных” – установленное соединение с базой данных;
- “Загруженное изображение” – содержит путь до изображения, которое выбрал пользователь.

Класс “главное окно” обладает следующими методами:

- “Загрузка базы данных” – устанавливает соединение с базой данных;
- “Загрузка классификатора” – инициализирует классификатор;
- “Загрузка изображения” – загружает путь до добавляемого изображения из файловой системы;
- “Классификация изображения” – переводит изображение в вектор с помощью нейронной сети и выполняет поиск ближайшего вектора к полученному в базе данных;
- “Вывод информации о транспортном средстве” – выводит информацию из найденной записи на экран;
- “Добавление транспортного средства в базу данных” – инициализирует и вызывает объект класса “Окно добавления”;
- “Изменение информации о транспортном средстве” – инициализирует и вызывает объект класса “Окно изменения”.

## **2.7 План подготовки набора данных для обучения**

Самым важным и объемным этапом работы при реализации системы является процесс подготовки набора данных для обучения нейронной сети.

Набор данных должен быть подготовлен и размечен таким образом, чтобы нейронная сеть смогла научиться выделять признаки транспортных средств.

Чтобы подготовить необходимый набор данных, нужно выполнить следующие задачи: получить изображения различных транспортных средств в требуемом количестве, определить их марку и модель, распределить по классам и

наконец разбить на обучающую и валидационную выборки данных для обучения модели.

Чтобы получить изображения различных транспортных средств необходимо спроектировать и реализовать программу, которая будет из переданного на вход видео вырезать и сохранять соответствующие изображения. Формат входных данных – видео – был выбран ввиду большей эффективности этого способа по времени (распознавать автомобили по отдельным фотографиям требует очень много времени).

Чтобы определить марку и модель транспортного средства необходимо спроектировать и реализовать программу, которая по номеру помошью стороннего сервиса будет определять марку и модель транспортного средства, которому принадлежит номер.

Для выполнения последних двух этапов подготовки набора данных необходимо спроектировать и реализовать программы, которые будут манипулировать собранными файлами в файловой системе согласно требованиям разметки набора данных.

## 3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### 3.1 Проектирование обучающего набора данных

Процесс подготовки набора данных представляет собой важный этап, в результате которого получается обработанный набор очищенных данных, пригодный для обработки алгоритмами машинного обучения.

Первичный набор исходных данных принято называть генеральной совокупностью. Выборка – это конечное подмножество элементов генеральной совокупности, изучив которое можно понять поведение исходного множества.

При этом для каждого этапа обучения необходим свой набор данных:

- обучающая выборка;
- тестовая выборка;
- проверочная (валидационная) выборка.

В настоящее время в открытом доступе представлено большое множество информации, которую можно легко использовать для создания наборов данных. Однако данный процесс усложняется при наличии каких-либо дополнительных требований, предъявляемых к данным или при недостаточном их количестве.

В нашем случае, таким требованием является особый способ выделения транспортного средства на изображении.

Нейронная сеть выделяет признаки из изображения полностью, а значит на нем должно быть минимальное число внешних (не относящихся к транспортному средству) элементов.

#### 3.1.1 Сбор первоначального набора данных

Как уже было сказано ранее, форматом входных данных для сбора изображений будет видео.

Программа должна будет делить видео на кадры, и каждый из них обрабатывать последовательно.

В процессе обработки кадра будет использована специально обученная нейронная сеть, способная найти координаты транспортного средства на

изображении и распознать его номер. Данная нейронная сеть допускает погрешности, так что для каждого набора координат было решено сделать рамку размером в 10 пикселей. После получения координат необходимо вырезать соответствующее изображение с транспортным средством и перевести его в черно-белый формат.

Чтобы получить изображение более высокого качества, программа должна будет рассматривать изображения в ближайшей к наблюдателю зоне – нижней половине кадра. Помимо этого, чтобы не нагружать набор данных дубликатами одного и того же транспортного средства, было решено определить и добавить в набор данных только самый удачное изображение транспортного средства, полученное из всего видео. Критерием для отбора такого изображения стала уверенность нейронной сети в предсказании номера транспортного средства.

Для работы с видео была выбрана библиотека OpenCV.

OpenCV — это библиотека с открытым кодом, поддерживающая множество платформ, включая Windows, MacOs и Linux. Также эта библиотека существует и для многих языков программирования. Но наиболее часто она используется для написания приложений машинного обучения на языке Python, особенно в сфере компьютерного зрения.

Помимо кроссплатформенности и поддержки многих языков программирования, которые позволяют использовать приложения на различных системах, библиотека OpenCV весьма эффективна (по сравнению с другими похожими библиотеками) с точки зрения вычислений, так как почти все функции и операторы в ней векторизированы [9].

На рисунке 23 представлен алгоритм работы программы.

При получении изображений выполняется следующий алгоритм:

1. Проверить, было ли загружено видео. Если да, то переходим к шагу 2, если нет, то переходим к шагу 11.
2. Получаем следующий кадр из видео.
3. Переводим полученный кадр в черно-белый формат.
4. Получить координаты транспортного средства.

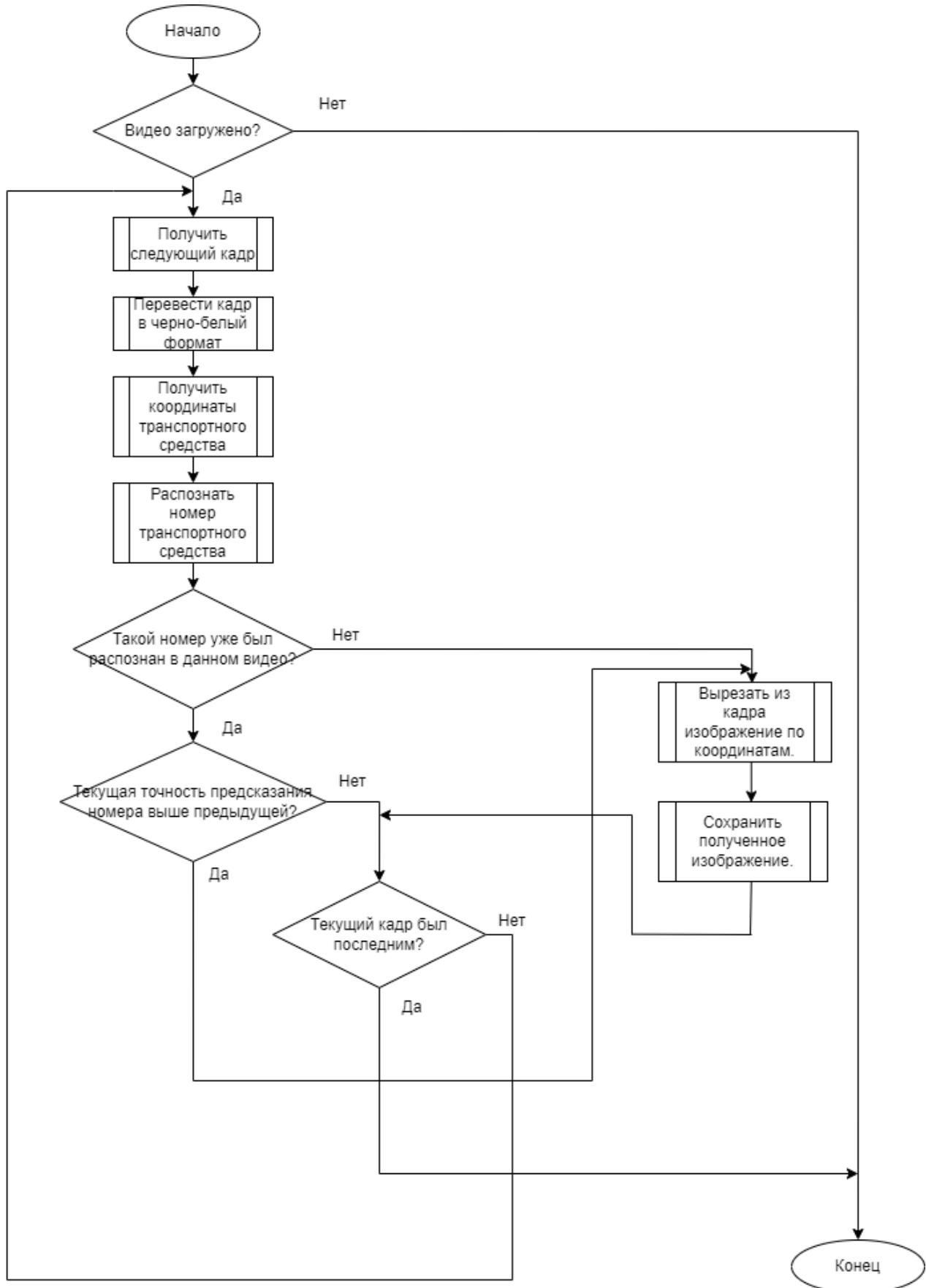


Рисунок 23 – Алгоритм получения изображений транспортных средств из видео

5. Распознать номер транспортного средства.
6. Если такой номер уже был распознан в данном видео, то переходим к шагу 7. Если такого номера еще не было, то переходим к шагу 8.
7. Если точность распознавания текущего номера выше предыдущей точности распознавания номера, то переходим к шагу 8, иначе переходим к шагу 10.
8. Вырезать изображение из полученного кадра используя координаты транспортного средства.
9. Сохранить полученное изображение использовав распознанный номер.
10. Проверить, был ли текущий кадр последним в видео. Если еще остались карды, то перейти к шагу 2. Если кадр был последним, перейти к шагу 11.
11. Завершить выполнение.

В результате работы программы должно быть получено какое-то число изображений, названных в соответствии с номером изображенного на них транспортного средства.

### **3.1.2 Разметка изображений**

Выбранным способом обучения модели стало обучение с учителем. Это значит, что для каждого изображения нужно точно знать, какое транспортное средство на нем изображено. Ручная разметка займет большое количество времени (к тому же для этого нужно обладать знаниями о моделях транспортных средств). Поэтому было решено воспользоваться помощью стороннего сервиса SpectrumData. Данный сервис специализируется на Big data и Data mining.

Требуемой функцией стала определение марки и модели транспортного средства по его государственному номеру.

Однако делать запрос для каждого автомобиля в выборке отдельно не целесообразно, поэтому необходимо автоматизировать передачу и извлечение информации через систему запросов.

На рисунке 24 представлен алгоритм получения информации о транспортном средстве.

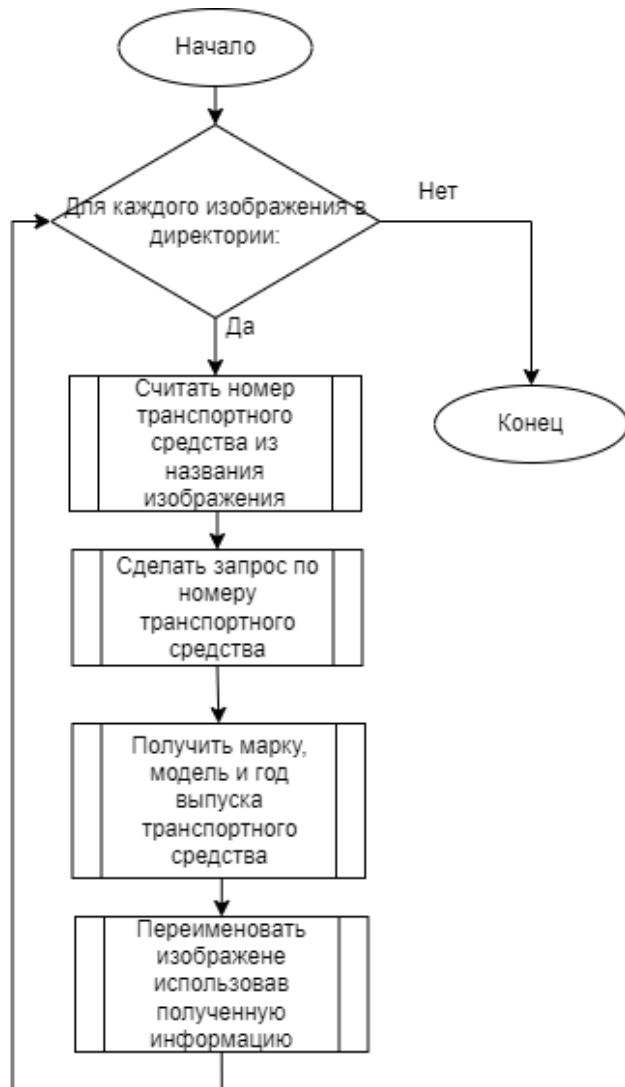


Рисунок 24 – Алгоритм получения информации о транспортном средстве

При получении информации о транспортном средстве используется следующий алгоритм:

1. Для каждого изображения, находящегося в директории, перейти на шаг 2.
2. Если изображений не осталось, перейти на шаг 6.
2. Считать номер транспортного средства из названия изображения.
3. Используя считанный номер сформировать и сделать запрос к SpectrumData.
4. Из полученного в ответ отчета извлечь марку, модель и год выпуска транспортного средства.
5. Переименовать исходное изображение в соответствии с полученной информацией. Перейти к шагу 1.

6. Завершить работу.

В результате работы программы должны получиться изображения, названные в соответствии с маркой, моделью и годом выпуска изображенного на них автомобиля.

### **3.1.3 Увеличение набора данных с помощью аугментации изображений**

Аугментация данных — это важный этап обучения моделей машинного обучения. Под аугментацией данных понимается увеличение выборки данных для обучения через модификацию существующих данных. Использование методов аугментации данных показало себя хорошо на задаче классификации изображений. Несмотря на это, исследований влияния аугментации на точность моделей распознавания не было. Учитывая ресурсы на разметку изображений, для задачи распознавания объектов аугментация может быть более полезна [10].

В качестве возможных аугментационных преобразований были выбраны:

- поворот изображения в пределах 20 градусов;
- шум;
- контраст;
- погодные эффекты (дождь, снег, туман);
- обрезка (используется для случаев, когда транспортное средство по каким-либо причинам попало в кадр не полностью);
- отражение;
- размытие при движении.

В одном изображении могут быть использованы несколько эффектов одновременно (не относится к разновидностям погодных эффектов).

С помощью аугментационных преобразований можно увеличить сравнительно небольшой набор данных до минимальных размеров, необходимых для обучения.

Программа для аугментации должна принимать на вход изображение, а на выходе получать  $n$  аугментированных изображений оригинала.

С помощью аугментации будет получена генеральная совокупность набора данных для обучения.

Для выполнения аугментации была выбрана библиотека `imgaug` языка `python`. В ней встроено множество различных вариантов преобразований изображения, которые могут быть скомбинированы между собой.

После сбора всех данных генеральная совокупность будет разделена на выборки данных в соотношении:

- обучающая 0.75;
- валидационная 0.15;
- тестовая 0.1.

На каждом этапе разработки модели будут использованы сторонние ресурсы. В таблице 3 показано, какой ресурс на каком этапе будет использован.

Таблица 3 – Сторонние ресурсы

Сбор данных.	1. Библиотека <code>python-opencv</code> . 2. Нейронная сеть для нахождения транспортного средства и его номера на изображении. 3. Нейронная сеть для распознавания символов.
Разметка данных.	<code>SpectrumData</code> .
Аугментация данных.	Библиотека <code>imgaug</code> .
Обучение модели.	1. Библиотека <code>tensorflow.keras</code> . 2. Библиотека <code>tensorflow_addons</code> . 3. Библиотека <code>kerasgen</code> .

### 3.2 Проектирование базы данных

#### 3.2.1 Выявление перечня ограничений целостности и выбор способа реализации его контроля

Для корректной работы базы данных необходимо определить ограничения, следуя которым она будет функционировать, а также определить способ реализации их контроля.

Атрибут “ID транспортного средства”, являющийся первичным ключом, заполняется автоматически.

Атрибуты “Длина кузова”, “Ширина кузова”, “Высота кузова”, “Колесная база”, “Клиренс”, “Ширина передней колеи” и “Ширина задней колеи” могут быть числом больше нуля, нулем и пустым (“NULL”) значением.

Атрибут “Модель” должен быть уникальным и не может быть пустым.

Данное ограничение будет реализовано ограничением символов, доступных для ввода в соответствующие поля.

### **3.2.2 Физическая модель БД**

Физическая модель базы данных находится на более низком уровне, чем логическая модель. Она содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей и т.п. Физическая модель строится на основе логической с учетом ограничений, накладываемых возможностями выбранной СУБД.

На основании логической модели определим следующую таблицу: Отношение “Транспортное средство” – таблица “vehicle”.

Для таблицы определим следующие атрибуты:

- атрибут “ID транспортного средства” – атрибут “id”;
- атрибут “Модель” – атрибут “model”;
- атрибут “Длина кузова” – атрибут “vehicle\_length”;
- атрибут “Ширина кузова” – атрибут “vehicle\_width”;
- атрибут “Высота кузова” – атрибут “vehicle\_height”;
- атрибут “Колесная база” – атрибут “wheel\_distance”;
- атрибут “Клиренс” – атрибут “clearance”;
- атрибут “Ширина передней колеи” – атрибут “front\_track\_width”;
- атрибут “Ширина задней колеи” – атрибут “back\_track\_width”;
- атрибут “Вектор представления” – атрибут “embedding”.

В базе данных применяются два стандартных типа данных:

1. «INTEGER» - обычновенный целочисленный тип. Используется для «ID» ключей (первичных и наследуемых), а также обозначения цены или количества.

2. «TEXT» - значение представляет собой текстовую строку, хранящуюся с использованием кодировки базы данных (UTF-8, UTF-16BE или UTF-16LE).

Атрибуты “model” и “embedding” получили характеристику NOT NULL, что делает их обязательными для заполнения. Атрибут “model” также получил характеристику UNIQUE, что устанавливает ограничение на то, что все значения в столбце должны быть различны.

На рисунке 25 представлена физическая модель БД.

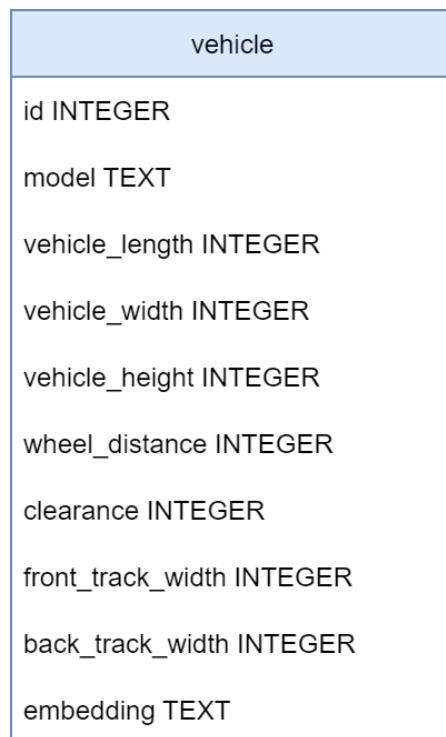


Рисунок 25 – Физическая модель БД

### **3.3 Проектирование пользовательского интерфейса**

#### **3.3.1 Проектирование жизненных циклов пользовательского интерфейса**

На сегодняшний день при проектировании сложной Системы принято делить ее на части, каждую из которых затем рассматривать отдельно. Таким образом, при

объектной декомпозиции Система разбивается на объекты или компоненты, которые взаимодействуют друг с другом, обмениваясь сообщениями. Сообщения описывают или представляют собой некоторые события. Получение объектом сообщения активизирует его и побуждает выполнять предписанные его программным кодом действия.

При данном подходе Система становится событийно управляемой, поэтому разработчикам зачастую важно знать, как должен реагировать тот или иной объект на определенные события. Инициаторами событий могут быть как объекты самой Системы, так и ее внешнее окружение.

Описать поведение отдельно взятого объекта помогает диаграмма состояний. Также зачастую диаграмма состояний используется аналитиками для описания последовательности переходов объекта из одного состояния в другое.

Диаграмма состояний покажет нам все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния [11].

На рисунке 26 представлена диаграмма состояний пользовательского интерфейса разрабатываемого ПО.

### **3.3.2 Карта диалоговых окон пользовательского интерфейса**

Диалоговое окно в графическом пользовательском интерфейсе — специальный элемент интерфейса, окно, предназначенное для вывода информации и (или) получения ответа от пользователя. Получило свое название потому, что осуществляет двустороннее взаимодействие компьютер-пользователь («диалог»): сообщая пользователю что-то и ожидая от него ответа.

На рисунке 27 представлена карта диалоговых окон.

### **3.3.3 Проектирование алгоритмов работы ПО**

Для работы программы необходимо спроектировать алгоритмы работы следующих кнопок:

- кнопка ”Пуск” – главное окно;

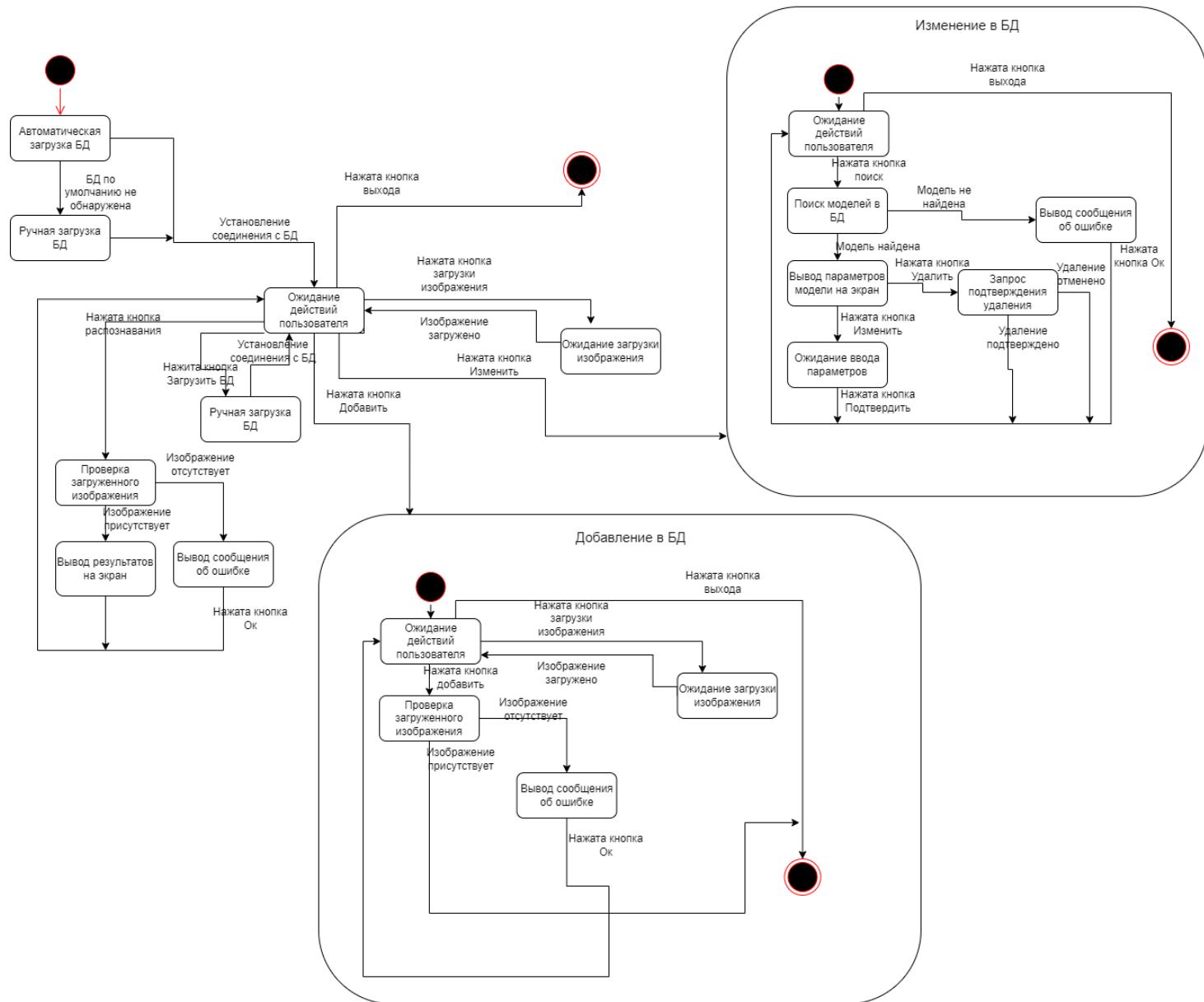


Рисунок 26 – Диаграмма состояний пользовательского интерфейса

- кнопка "Поиск" – окно изменения;
- кнопка "Изменить" – окно изменения;
- кнопка "Удалить" – окно изменения;
- кнопка "Подтвердить" – окно изменения;
- кнопка "Добавить" – окно добавления.

На рисунке 28 представлен алгоритм получения информации о транспортном средстве, который выполняется при нажатии кнопки "Пуск".

При нажатии на кнопку "Пуск" используется следующий алгоритм:

1. Если изображение загружено, то перейти на шаг 3, иначе на шаг 2.
2. Вывести сообщение об ошибке, перейти на шаг 15.
4. Инициализировать нейронную сеть.
5. Получить вектор представления изображения.
6. Получить все вектора из БД.
7. Для каждого извлеченного вектора: перейти на шаг 7. Если все извлеченные векторы были обработаны, перейти на шаг 11.
8. Вычислить Евклидово расстояние между вектором признаков изображения и вектором из БД.
9. Если полученное расстояние меньше минимального, то перейти на шаг 9, иначе на шаг 6.
10. Сделать минимальное значение расстояния равным текущему.
11. Заменить предсказываемый класс на класс, соответствующий текущему извлеченному вектору. Перейти на шаг 6.
12. Если предсказываемый класс был получен, то перейти на шаг 13, иначе перейти на шаг 12.
13. Вывести сообщение об отсутствии похожих транспортных средств в БД. Перейти на шаг 15.
14. Получить информацию о найденном транспортном средстве.

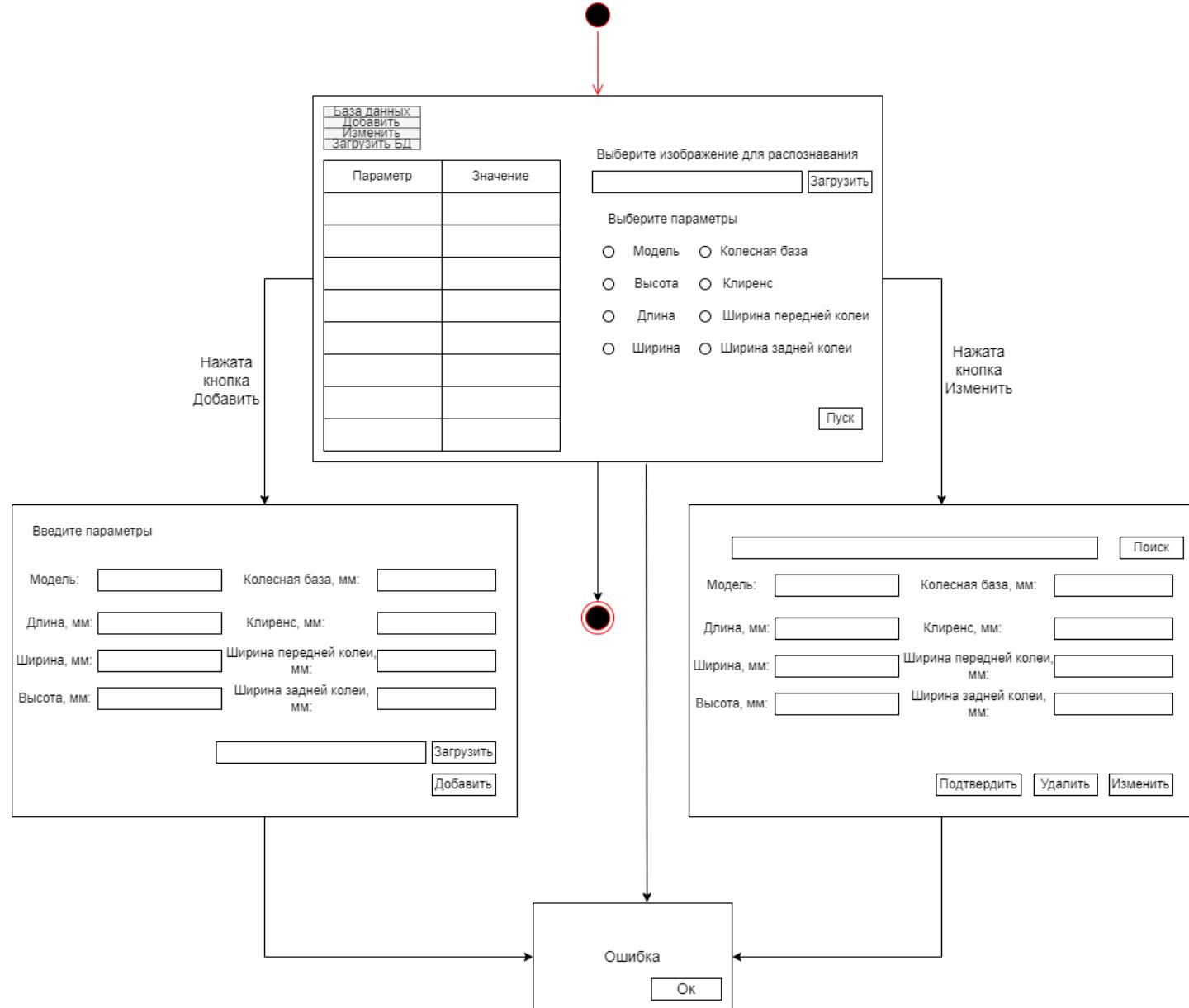


Рисунок 27 - Карта диалоговых окон

15. Вывести информацию на экран.
16. Завершить выполнение.

При нажатии на кнопку “Поиск” исполняется следующий алгоритм:

1. Получить текст из строки поиска.
2. Если изображение загружено, то перейти на шаг 4, иначе на шаг 3.
3. Вывести сообщение об ошибке. Перейти на шаг 8.
4. Выполнить поиск по БД.
5. Если модель транспортного средства была найдена, то перейти на шаг 7, иначе на шаг 6.
6. Вывести сообщение об отсутствии совпадений. Перейти на шаг 8.
7. Вывести информацию о транспортном средстве на экран.
8. Завершить выполнение.

На рисунке 29 представлен алгоритм поиска транспортного средства в БД, который выполняется при нажатии на кнопку “Поиск”.

При нажатии на кнопку “Изменить” исполняется следующий алгоритм:

1. Если поле с названием модели пустое, то перейти на шаг 2, иначе на шаг 3.
2. Вывести сообщение об ошибке. Перейти на шаг 8.
3. Разблокировать все поля для ввода информации.
4. Заблокировать кнопку “Изменить”.
5. Заблокировать строку поиска.
6. Заблокировать кнопку “Удалить”.
7. Разблокировать кнопку “Подтвердить”.
8. Завершить выполнение.

На рисунке 30 представлен алгоритм изменения информации о транспортном средстве, который выполняется при нажатии на кнопку “Изменить”.

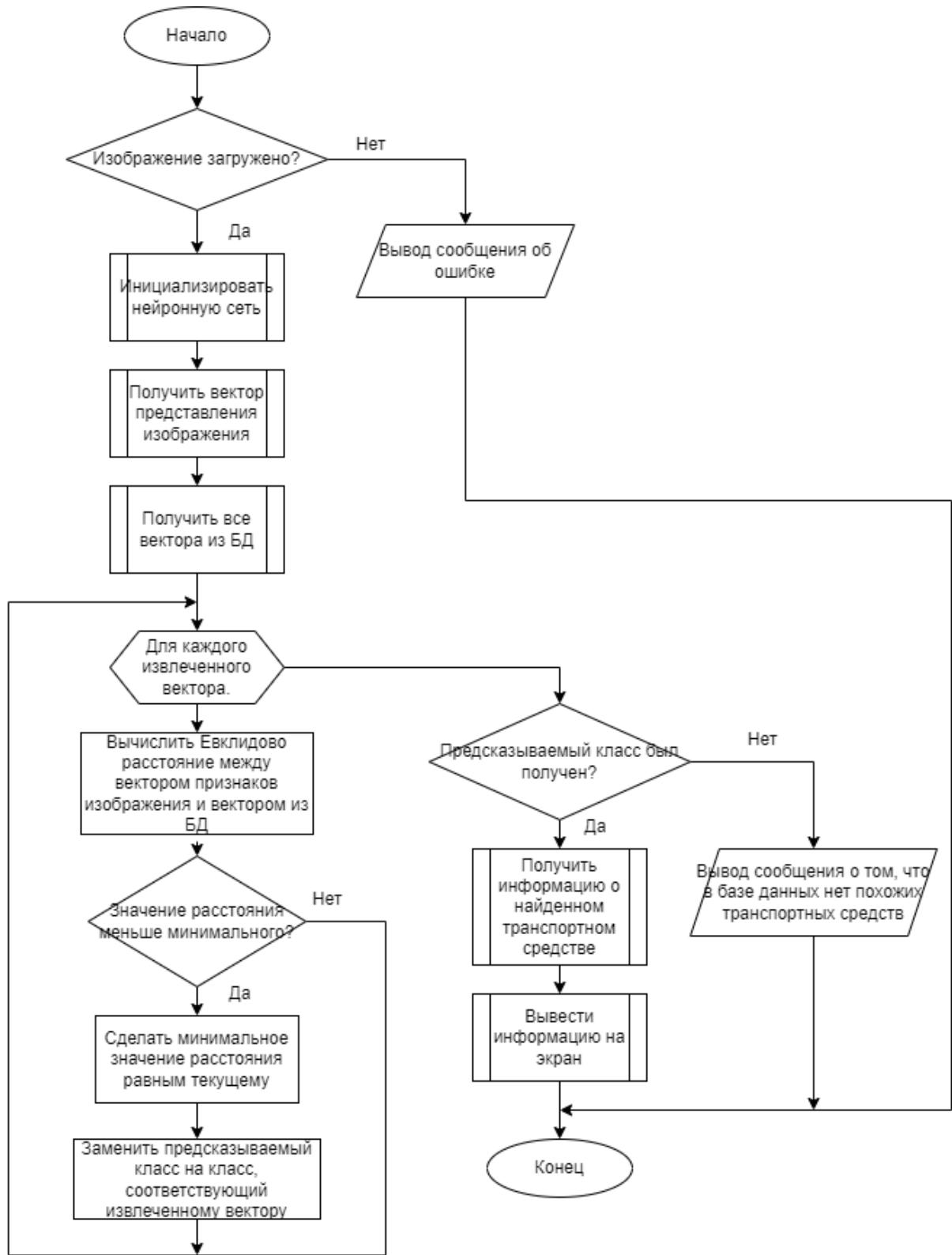


Рисунок 28 – Алгоритм получения информации о транспортном средстве

При нажатии на кнопку “Подтвердить” выполняется следующий алгоритм:

1. Считать информацию из полей ввода.
2. Обновить информацию в базе данных в соответствии со считанной.

3. Подтвердить изменения в БД.
4. Заблокировать все поля ввода информации.
5. Разблокировать кнопку “Изменить”.
6. Разблокировать строку поиска.
7. Разблокировать кнопку “Удалить”.
8. Заблокировать кнопку “Подтвердить”.
9. Завершить выполнение.

На рисунке 31 представлен алгоритм сохранения изменений информации в БД, который выполняется при нажатии на кнопку “Подтвердить”.

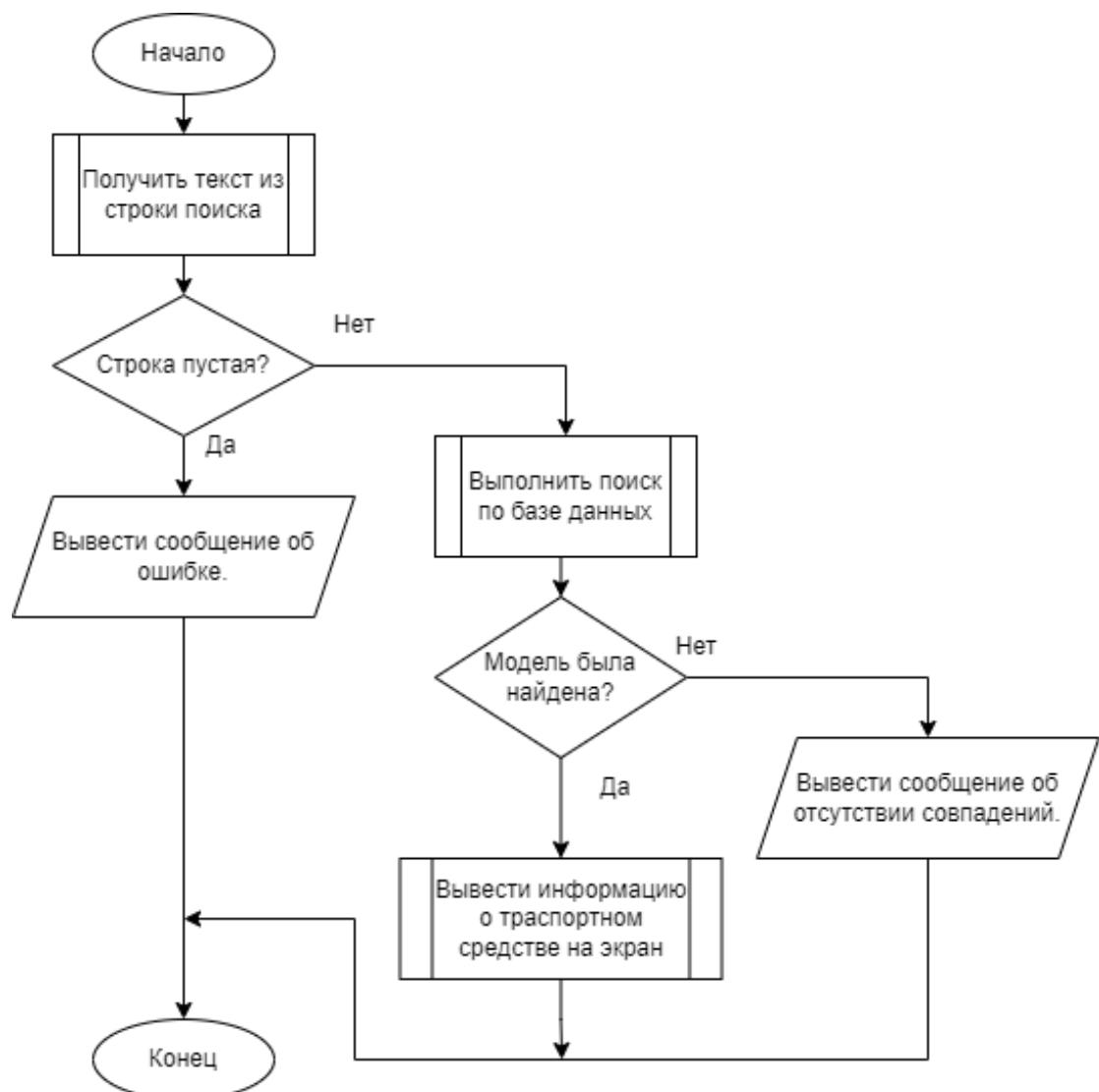


Рисунок 29 – Алгоритм поиска транспортного средства в БД

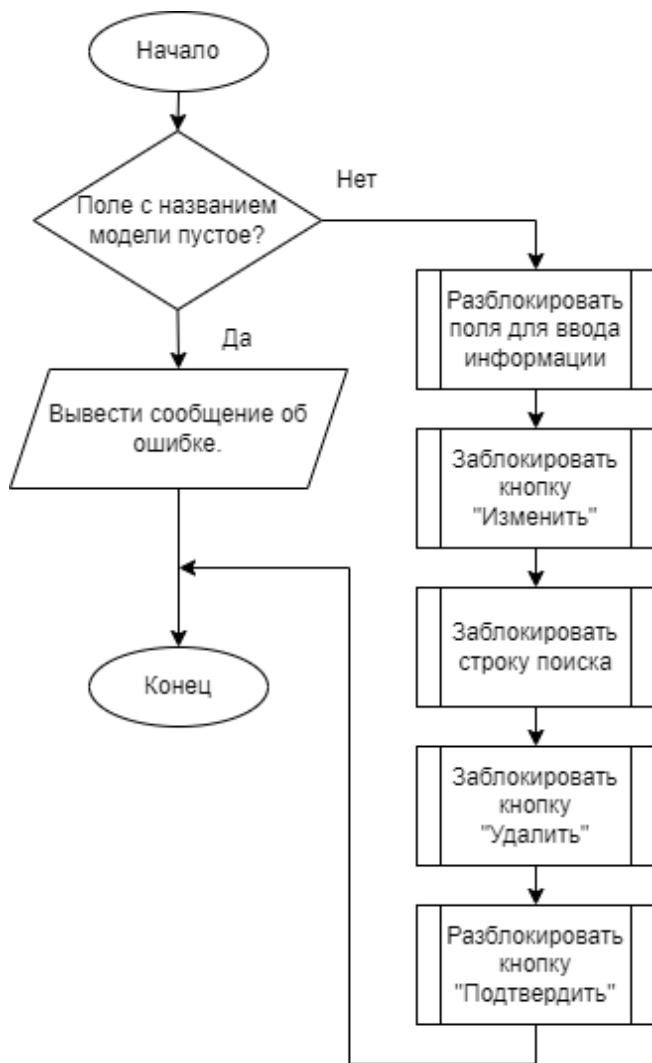


Рисунок 30 – Алгоритм изменения информации о транспортном средстве

При нажатии на кнопку “Удалить” выполняется следующий алгоритм:

1. Если поле с названием модели пустое, то перейти на шаг 2, иначе на шаг 3.
2. Вывести сообщение об ошибке. Перейти на шаг 10.
3. Найти в БД модель с введенным названием.
4. Если запись с введенным названием была найдена, то перейти на шаг 6, иначе на шаг 5.
5. Вывести сообщение об ошибке. Перейти на шаг 10.
6. Запросить подтверждение удаления записи у пользователя.
7. Если удаление было подтверждено, перейти на шаг 8, иначе на шаг 10.
8. Удалить запись из базы данных.

9. Подтвердить изменения в базе данных.
10. Завершить выполнение.

На рисунке 32 представлен алгоритм удаления записи о транспортном средстве из БД, который выполняется при нажатии на кнопку “Удалить”.

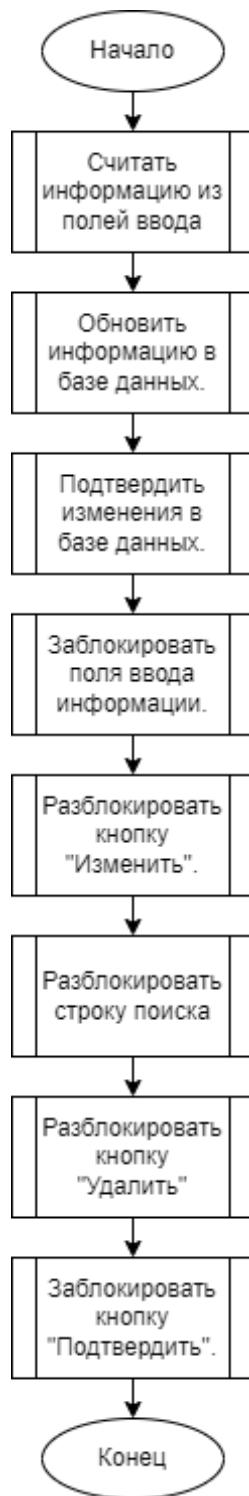


Рисунок 31 – Алгоритм сохранения изменений информации в БД

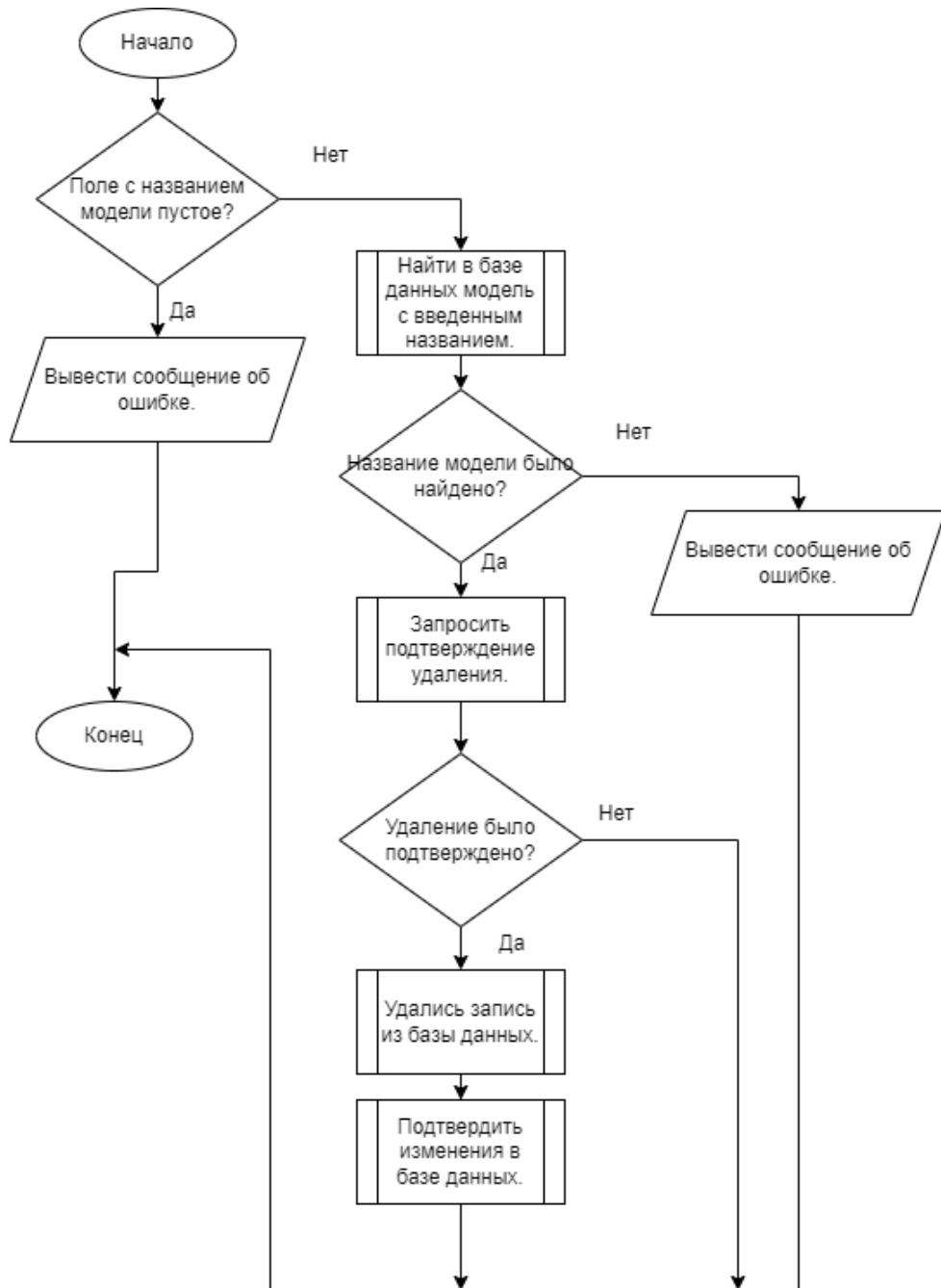


Рисунок 32 – Алгоритм удаления записи о транспортном средстве из БД

При нажатии на кнопку “Добавить” выполняется следующий алгоритм:

1. Если изображение загружено, то перейти на шаг 3, иначе на шаг 2.
2. Вывести сообщение об ошибке. Перейти на шаг 7.
3. Считать информацию из полей ввода.
4. Получить вектор представления изображения.
5. Создать новую запись в БД.
6. Подтвердить изменения в БД.

7. Завершить выполнение.

На рисунке 33 представлен алгоритм добавления записи о транспортном средстве в БД, который выполняется при нажатии кнопки “Добавить”.

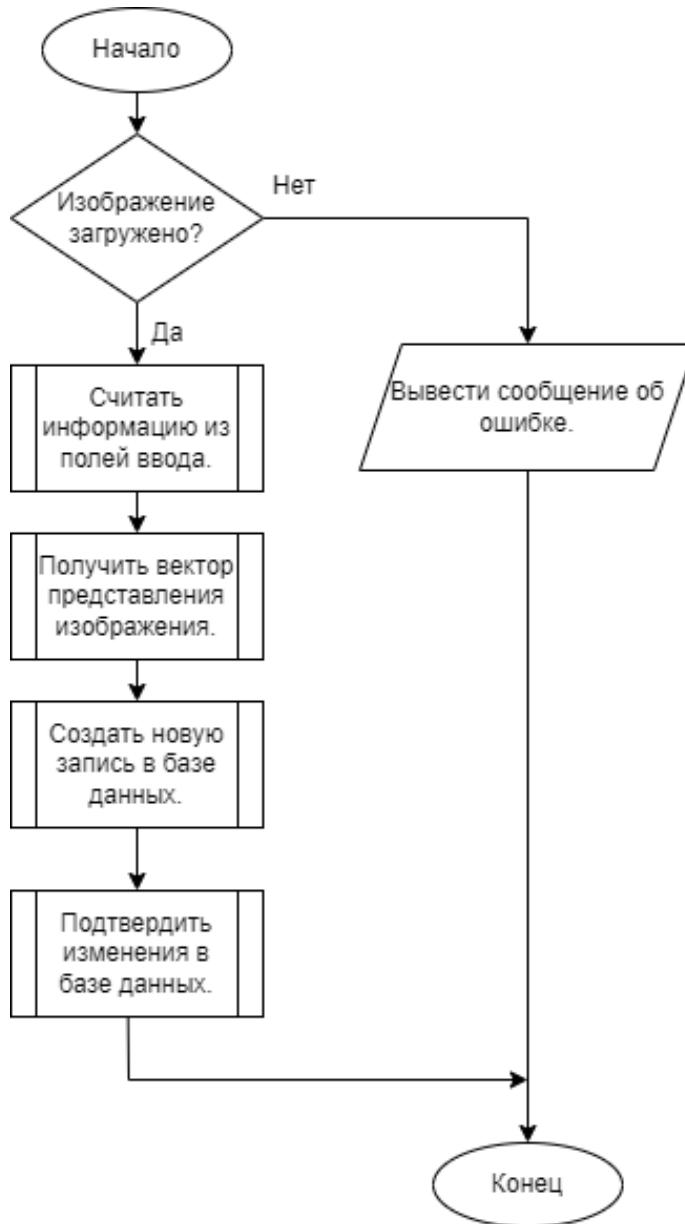


Рисунок 33 – Алгоритм добавления записи о транспортном средстве в БД

### 3.4 Проектирование модели нейронной сети

Выбранная для переобучения модель нейронной сети – Facenet – отлично справляется с задачей выделения признаков у человеческих лиц. Однако у человеческого лица и транспортного средства есть масса различий.

Facenet использует в качестве выходного слоя батч-нормализацию (batch normalization). Данный слой решает проблему, препятствующую эффективному обучению нейронных сетей: по мере распространения сигнала по сети, даже если мы нормализовали его на входе, пройдя через внутренние слои, он может сильно исказиться как по математическому ожиданию, так и по дисперсии, что чревато серьезными несоответствиями между градиентами на различных уровнях. Подобных слоев в модели Facenet очень много, однако нас интересует только конечная нормализация.

Даже с подобным решением, большой проблемой нейронных сетей является переобучение – когда модель в погоне за минимизацией затрат на обучение теряет способность к обобщению (особенно эта проблема заметна на небольших наборах данных, когда модель может просто запомнить признаки и стать совершенно неэффективной на новом наборе данных).

Самым простым способом держать переобучение под контролем является метод dropout, который, как и батч-нормализация, не один раз используется в Facenet. Но есть и другие способы нормализации, и самый популярный из них – l2 нормализация.

Такая нормализация использует более прямой подход, нежели классический dropout – l2 накладывает штрафы на веса с наибольшим значением, минимизируя их L2 норму с использованием параметра регуляризации.

Добавление слоя с подобной нормализацией поможет обеспечить более качественное функционирование модели, особенно с небольшим изначальным набором данных.

Так как Facenet обладает большой и сложной архитектурой модели, было принято решение добавить новый слой в конец, сразу после батч-нормализации.

## 4 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### 4.1 Реализация модели нейронной сети

В результате сборки набора данных общий объем генеральной совокупности составил 660 уникальных изображений транспортных средств, принадлежащих к 132 различным классам.

Листинг программного кода, использовавшегося при реализации модели нейронной сети представлен в Приложении 2.

#### 4.1.1 Описание работы аугментации изображений

Аугментация изображений позволяет значительно увеличить размер собранного набора данных. Однако для каждого типа преобразований существуют правила возникновения.

Так, например, на одном изображении может возникнуть только один из погодных эффектов: дождь, снег или туман.

Фрагмент реализации данного правила на языке python представлен ниже.

OneOf[

```
Rain(drop_size=(0.10, 0.15)),
Snowflakes(flake_size=(0.7, 0.75), speed=(0.001, 0.03)),
Fog(0)
])
```

Точно также не могут вместе быть наложены эффекты шума, размытия при движении и контраста.

Отражение, обрезка и поворот возникают независимо как друг от друга, так и от других эффектов и могут комбинироваться с ними.

На рисунке 34 представлен пример наложения погодного эффекта снега. Примеры для всех остальных эффектов представлены в Приложении 1.

После проведения аугментаций общий объем генеральной совокупности составил около 52 тыс. изображений.



Рисунок 34 – Погодный эффект снега

#### 4.1.2 Особенности реализации нейронной сети

##### 4.1.2.1 График распространения ошибки

В процессе обучения происходит минимизация ошибки. Чем меньше значение ошибки, тем лучше модель обучена.

На рисунке 35 представлены графики ошибки на обучающем (оранжевый) и на валидационном (синий) наборах данных.

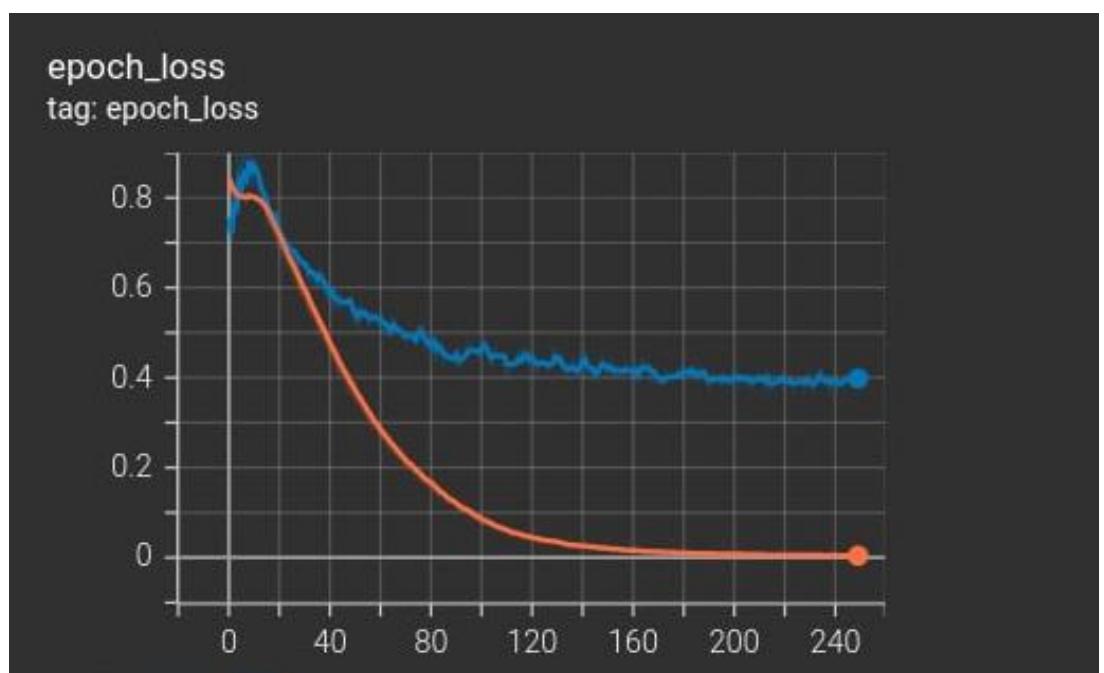


Рисунок 35 – График распространения ошибки

#### 4.1.2.2 Кластеризация модели

Помимо минимизации ошибки модель должна научиться формировать кластеры. Кластеризация – это автоматическое разбиение элементов некоторого множества на группы в зависимости от их схожести [12]. Элементами множества может быть что угодно, например, данные или вектора характеристик. Сами же группы принято называть кластерами [13].

Чтобы проверить способность модели формировать кластеры векторов, все изображения из генеральной совокупности данных были представлены в виде векторов, а после отрисованы в 2D пространстве. Каждый отдельный класс изображений был обозначен своим цветом [14].

На рисунке 36 представлено кластерное пространство векторов генеральной совокупности для 18 классов. Каждый класс, отмеченный определенным цветом, относится к определенному пространству на рисунке. Лишь некоторые отдельные вектора находятся не в своем кластере [15].

Исходя из Рисунка 36 можно сделать вывод, что модель может формировать кластеры данных.

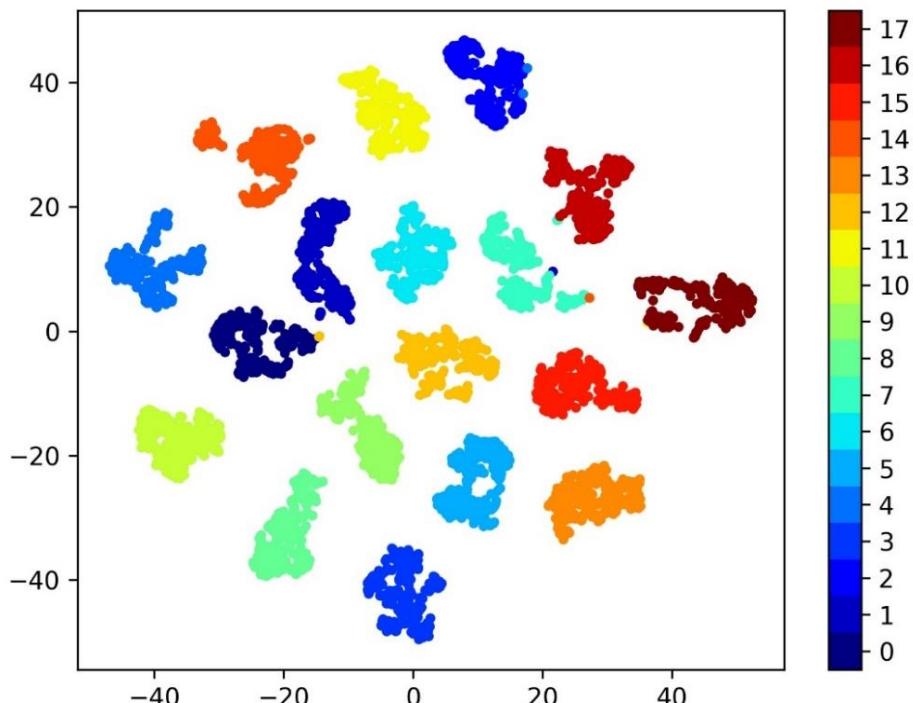


Рисунок 36 – Кластерное пространство векторов генеральной совокупности для 18 классов

#### **4.1.2.3 Метрика полноты**

Полнота (recall) – пропорция всех верно-положительных предсказанных объектов к общему количеству действительно положительных. То есть, полнота показывает, сколько образцов из всех положительных примеров были классифицированы правильно. Чем выше значение полноты, тем меньше положительных примеров пропущено в классификации [16].

Для обучающей выборки данная метрика составила 94.78%, а для тестовой – 80%.

#### **4.1.2.4 Конвертация модели нейронной сети**

Получившаяся в результате обучения модель весит 295 Мб, а скорость ее обработки одного изображения составляет 0.25 с.

Чтобы уменьшить вес модели, а также немного сократить время обработки изображения, формат модели был изменен с HDF5 на tflite. Подобное преобразование сжимает модель, замораживает ее веса. Модель становится закрытой и в нее невозможно более вносить изменения. Помимо этого, значительно уменьшается время, затрачиваемое на первичную инициализацию модели.

После конвертации вес модели составил 91 Мб, а скорость обработки одного изображения сократилась до 0.21 с.

### **4.2 Реализация программного обеспечения**

#### **4.2.1 Особенности программной реализации**

Для создания графического интерфейса была использована библиотека языка python – PyQt5. Она является одним из наиболее используемых модулей для создания GUI приложений в python.

Для разделения ответственности и различных целей были созданы четыре класса:

- класс формы главного окна;
- класс формы окна добавления;
- класс формы окна изменения;

- класс исполнения классификации с помощью нейронной сети.

Класс `Ui_MainWindow` создает главное окно программы, а также обрабатывает взаимодействия с его элементами.

Данный класс имеет множество методов, которые вызываются при взаимодействии с элементами окна.

Класс `Ui_MainWindow` содержит следующие поля:

- `image_path` – путь до загруженного в форму изображения;
- `db` – путь до загружаемой базы данных;
- `sqlite_connection` – установленное соединение с базой данных;
- `cursor` – инструмент выполнения запросов к базе данных.

Класс `Ui_MainWindow` содержит следующие методы:

- `load_db` – загружает базу данных и обработчик запросов;
- `close_db_connectoin` – закрывает соединение с базой данных;
- `load_new_db` – запрашивает загрузку базы данных у пользователя;
- `setupUi` – создает главное окно и размещает основные элементы на нем;
- `retranslateUi` – создает надпись на каждом объекте на русском языке;
- `load_tflite` – инициализирует нейронную сеть, создает объект класса

`Classifier`:

- `statusbar_text` – выводит сообщение в поле статуса формы;
- `set_table` – выводит информацию из базы данных на экран;
- `browse_button_click` – нажатие на кнопку загрузки;
- `prep` – предобработка изображения для загрузки в нейронную сеть;
- `recognize_button_click` – нажатие на кнопку распознавания;
- `menu_load_db` – загрузка новой базы данных;
- `db_add_vehicle` – вызывает окно добавления;
- `db_search_vehicle` – вызывает окно изменения.

Класс `Ui_db_add_window` создает окно добавления нового транспортного средства в базу данных, а также обрабатывает взаимодействия с его элементами.

Данный класс имеет несколько методов, которые вызываются при взаимодействии с элементами окна.

Класс `Ui_db_add_window` содержит следующие поля:

- `classifier` – объект класса `Classifier`;
- `image_path` – путь до загруженного изображения;
- `sqlite_connection` – установленное соединение с базой данных;
- `cursor` – инструмент выполнения запросов к базе данных;

Класс `Ui_db_add_window` содержит следующие методы:

- `setupUi` – создает главное окно и размещает основные элементы на нем;
- `retranslateUi` - создает надпись на каждом объекте на русском языке;
- `browse_button_click` – нажатие на кнопку загрузки;
- `add_button_click` – нажатие на кнопку добавления.

Класс `Ui_search_window` создает окно добавления нового транспортного средства в базу данных, а также обрабатывает взаимодействия с его элементами.

Данный класс имеет несколько методов, которые вызываются при взаимодействии с элементами окна.

Класс `Ui_search_window` содержит следующие поля:

- `sqlite_connection` – установленное соединение с базой данных;
- `cursor` – инструмент выполнения запросов к базе данных.

Класс `Ui_search_window` содержит следующие методы:

- `setupUi` – создает главное окно и размещает основные элементы на нем;
- `retranslateUi` - создает надпись на каждом объекте на русском языке;
- `change_edits_condition` – изменяет состояния полей ввода информации;
- `search` – нажатие на кнопку поиска;
- `clear` – очищает поля ввода информации;
- `db_change` – нажатие на кнопку изменить;
- `db_commit` – нажатие на кнопку подтвердить;
- `db_delete` – нажатие на кнопку удалить.

Класс `Classifier` загружает и подготавливает нейронную сеть для исполнения.

Класс `Classifier` содержит следующие поля:

- `model_path` – путь до модели нейронной сети;
- `add_array` – список ближайших векторов к искомому;

- required\_size – входная размерность изображения;
- min\_distance – минимальная дистанция близости векторов.

Класс Classifier содержит следующие методы:

- get\_embedding – производит обработку изображения для подачи в нейронную сеть;
- get\_distance – определяет значение близости двух векторов;
- get\_embedding\_vector – выполняет распознавание изображения и передает результирующий вектор.

Пример программного кода класса главного окна представлен в Приложении 3.

#### **4.2.2 Описание пользовательского интерфейса**

После запуска программы перед пользователем открывается окно, представленное на рисунке 37. Вместе со стартом программа инициализирует соединение с базой данных.

Для чтобы программа выполнила классификацию, пользователь должен загрузить желаемое изображение (нажатием кнопки “Загрузить” открывается файловая система, откуда можно выбрать необходимое изображение), выбрать параметры, которые будут выведены на экран и нажать кнопку “Пуск”.

На рисунке 38 представлен результат нажатия кнопки “Пуск”.

Если для выбранного изображения программа не найдет совпадений в БД, то будет выведено предупреждение об отсутствии искомого транспортного средства. Предупреждение об отсутствии транспортного средства представлено на рисунке 39.

Если кнопка “Пуск” была нажата до того, как пользователь загрузил изображение, то программа выдаст ошибку. Ошибка отсутствия загруженного изображения представлена на рисунке 40.

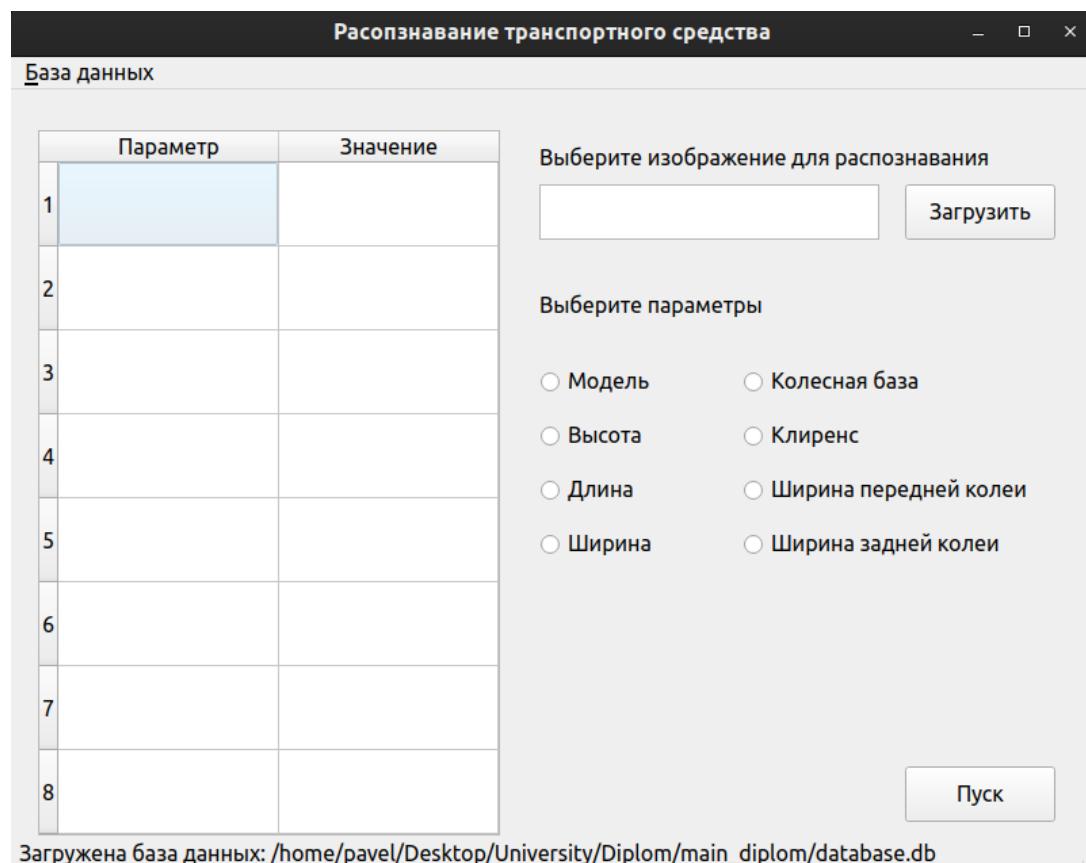


Рисунок 37 – Главное окно

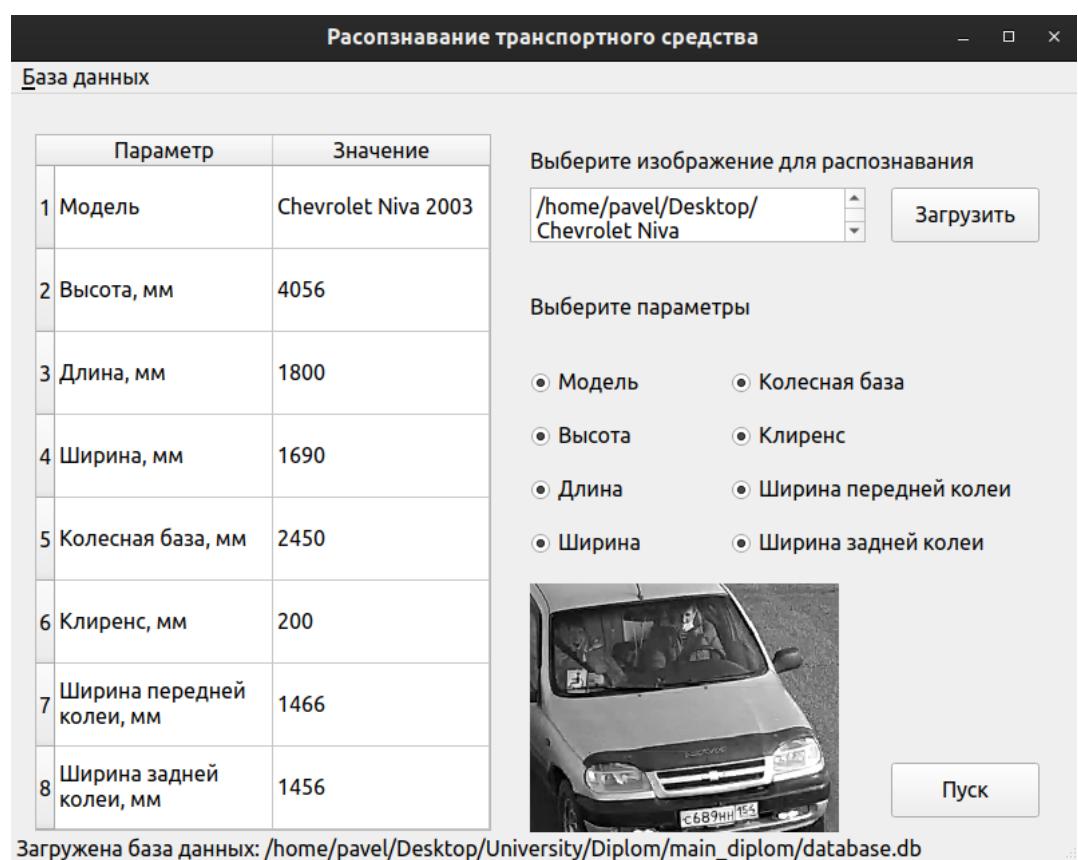


Рисунок 38 – Нажатие кнопки “Пуск”

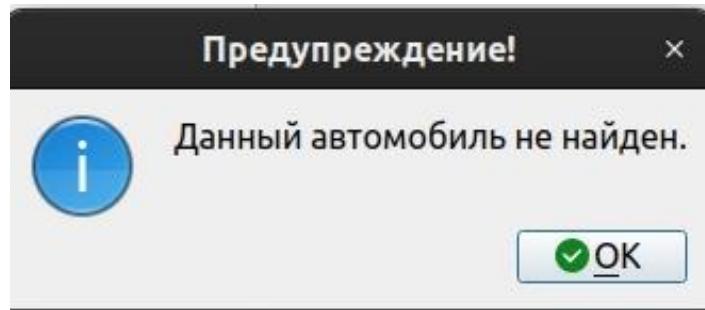


Рисунок 39 – Предупреждение об отсутствии искомого транспортного средства

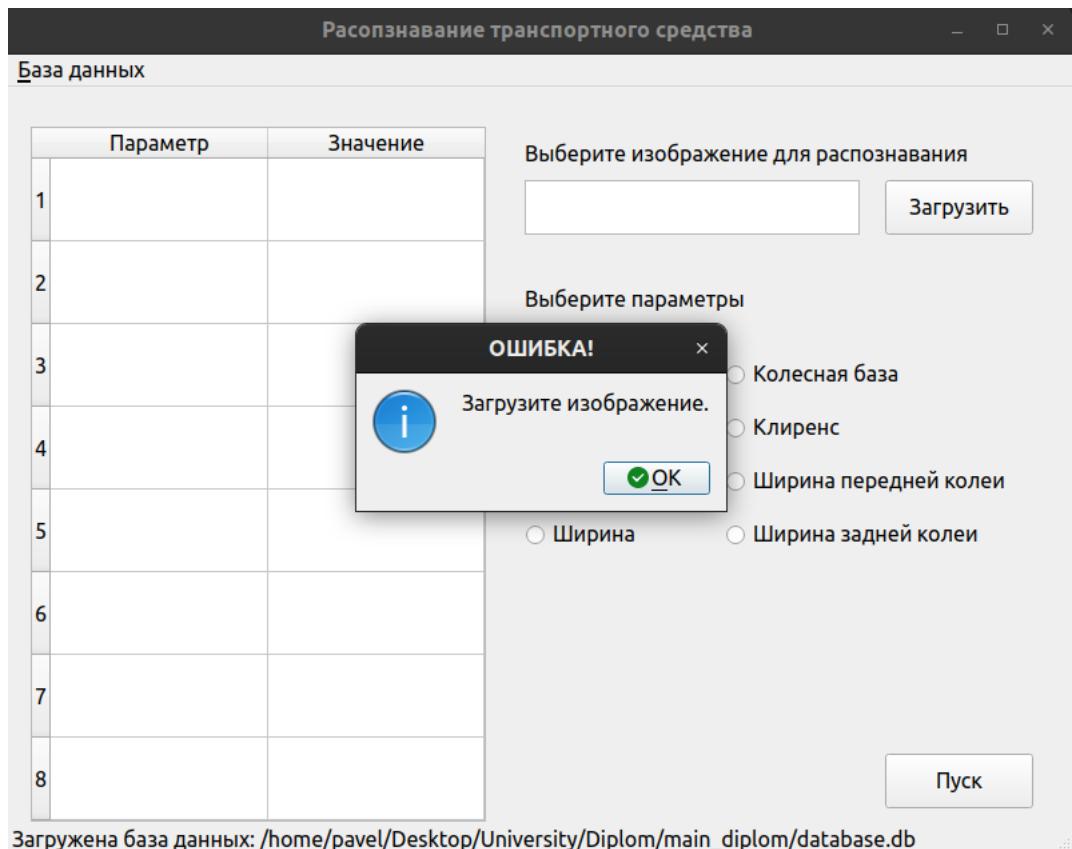


Рисунок 40 – Ошибка отсутствия загруженного изображения

Функции взаимодействия с базой данных находятся в выпадающем меню “База данных”. Всего в нем представлено 3 варианта действий: добавление, изменение и загрузка новой БД. На рисунке 41 представлено выпадающее меню.

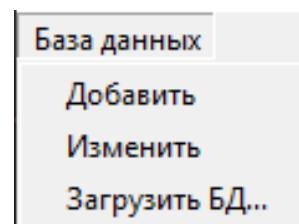


Рисунок 41 – Выпадающее меню

При нажатии на кнопку “Добавить” открывается окно добавления. Окно добавления представлено на рисунке 42.

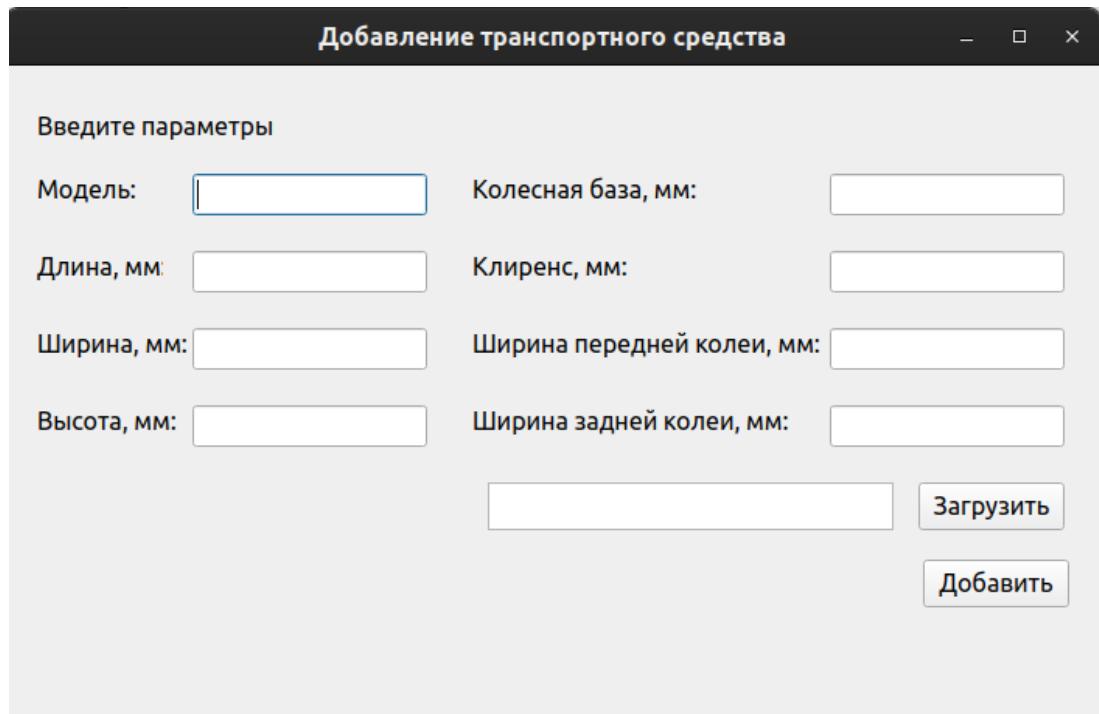


Рисунок 42 – Окно добавления

Чтобы добавить транспортное средство в базу данных пользователю необходимо заполнить поля информации о модели, после чего загрузить изображение модели (нажатием кнопки “Загрузить” открывается файловая система, откуда можно выбрать необходимое изображение). Далее, после нажатия кнопки “Добавить”, будет создана и сохранена новая запись в базе данных. После выполнения кнопки окно добавления сразу будет закрыто.

Если кнопка “Добавить” была нажата до того, как пользователь загрузил изображение, то программа выдаст ошибку. Ошибка отсутствия загруженного изображения представлена на рисунке 43.

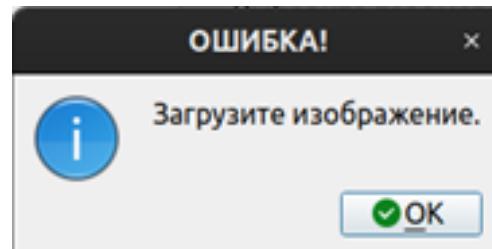


Рисунок 43 – Ошибка отсутствия изображения

При нажатии на кнопку “Загрузить БД...” откроется проводник, из которого пользователь сможет загрузить новый файл с расширением .db. После загрузки новой базы данных будет выведено сообщение, подтверждающее загрузку, а также загруженная БД станет базой данных по умолчанию (будет загружаться при старте программы). На рисунке 44 представлено сообщение успешной загрузки новой БД.

Загружена база данных: /home/pavel/Desktop/database.db

Рисунок 44 – Успешная загрузка БД

При нажатии на кнопку “Изменить” будет открыто окно изменения. Окно изменения представлено на рисунке 45.

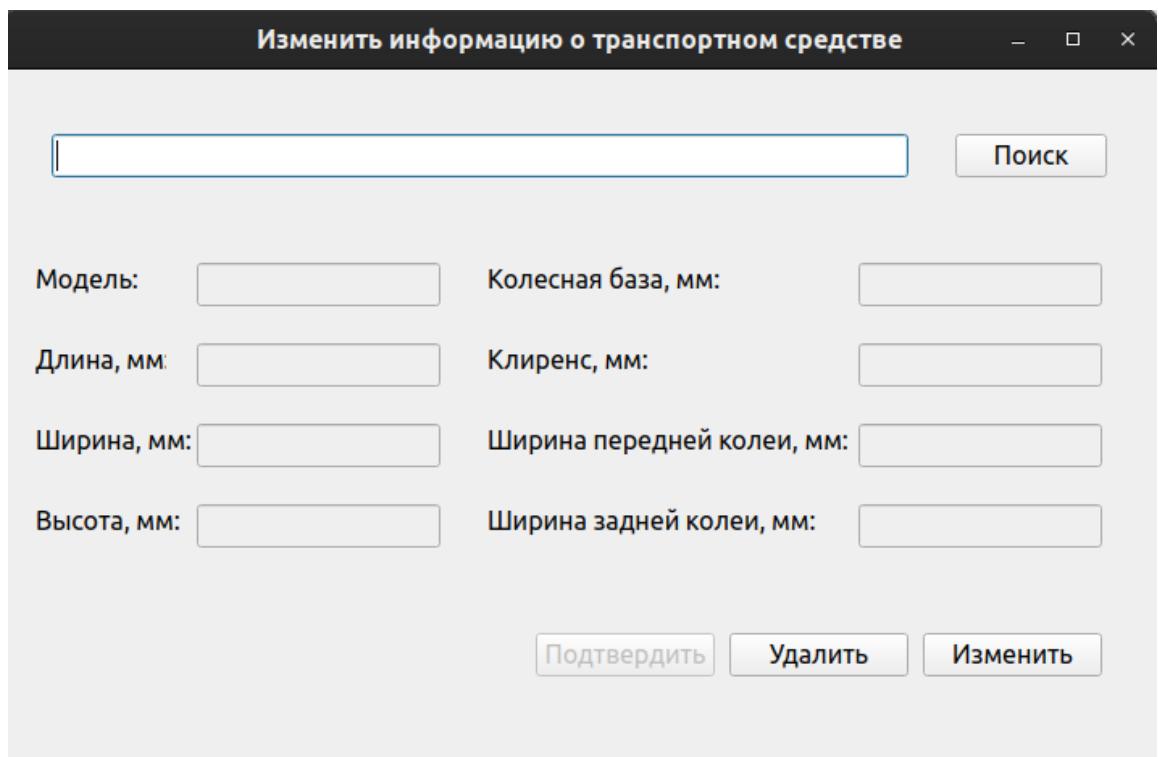


Рисунок 45 – Окно изменения

При открытии окна все строки ввода, за исключением строки поиска, заблокированы, и пользователь не может изменять их содержимое.

Чтобы изменить информацию о транспортном средстве, пользователь должен сперва найти необходимую модель в базе данных.

При нажатии на кнопку “Поиск” программа находит запрашиваемую пользователем модель транспортного средства и выводит информацию о нем в заблокированные поля. Пример работы кнопки “Поиск” представлен на рисунке 46.

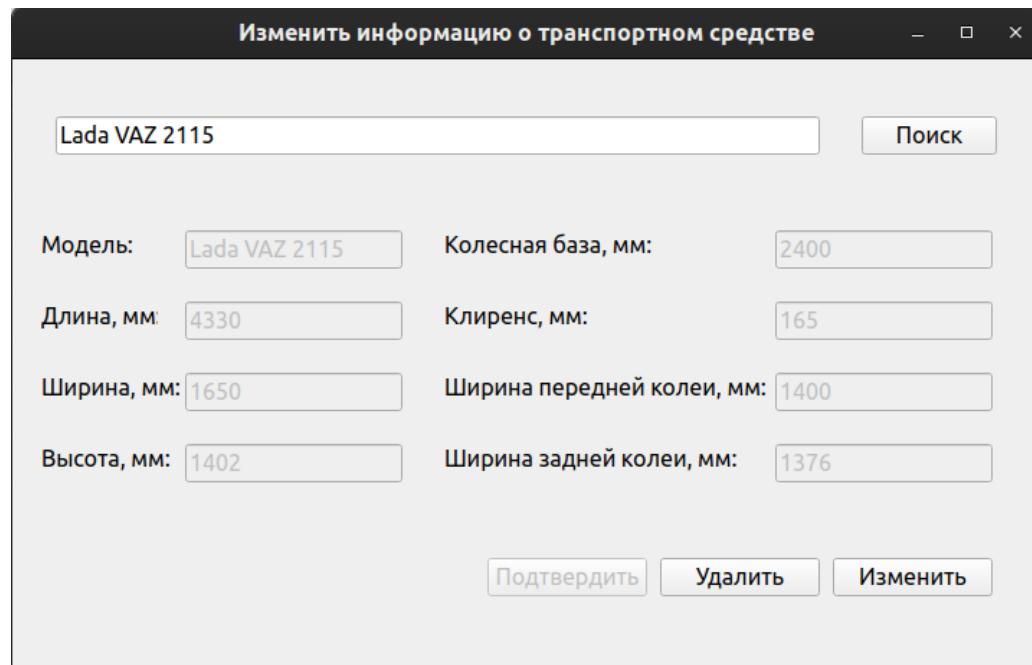


Рисунок 46 – Нажатие кнопки “Поиск”

Если вводимая модель отсутствует в базе данных, то пользователю будет выведено предупреждение об отсутствии совпадений. Данное предупреждение представлено на рисунке 47.

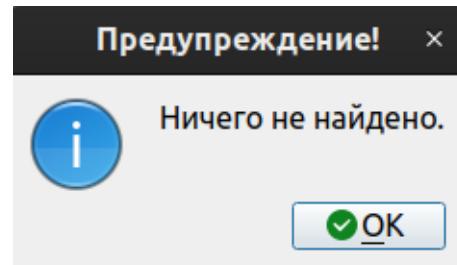


Рисунок 47 – Предупреждение об отсутствии совпадений

Если пользователь нажмет кнопку “Поиск” при пустой строке поиска, то программа выведет ошибку, представленную на рисунке 48.

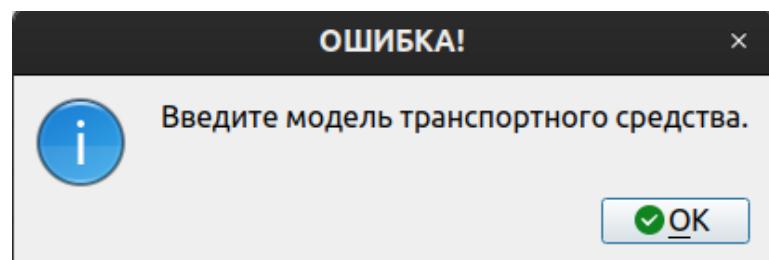


Рисунок 48 – Ошибка пустой строки поиска

При нажатии на кнопку “Изменить” строка поиска будут заблокированы строка поиска, кнопки “Изменить” и “Удалить”, а поля ввода информации и кнопка “Подтвердить” будут разблокированы. Пользователь сможет редактировать значения данных полей. Результат нажатия кнопки “изменить” представлен на рисунке 49.

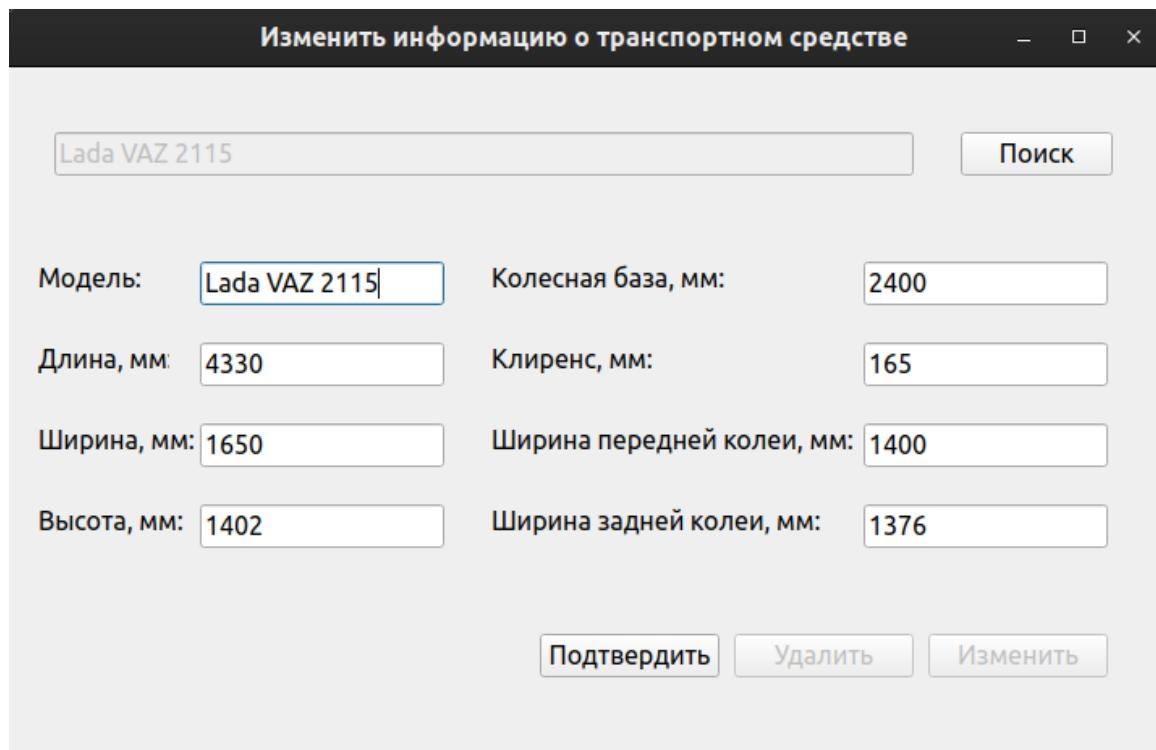


Рисунок 49 – Нажатие кнопки “Изменить”

Если кнопка “Изменить” будет нажата, а модель транспортного средства не найдена, то программа выведет предупреждение, представленное на рисунке 50.

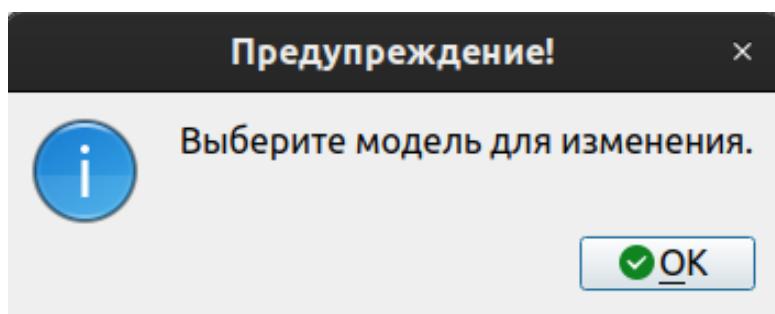


Рисунок 50 – Предупреждение отсутствия модели для изменения

При нажатии на кнопку “Подтвердить” все изменения в полях ввода информации будут внесены в базу данных. В поля ввода информации будет записана актуальная информация о транспортном средстве, а сами поля и кнопка

“подтвердить” будут заблокированы. Стока поиска, кнопки “Изменить” и “Удалить” будут разблокированы. На рисунке 51 представлен результат работы кнопки “Подтвердить”.

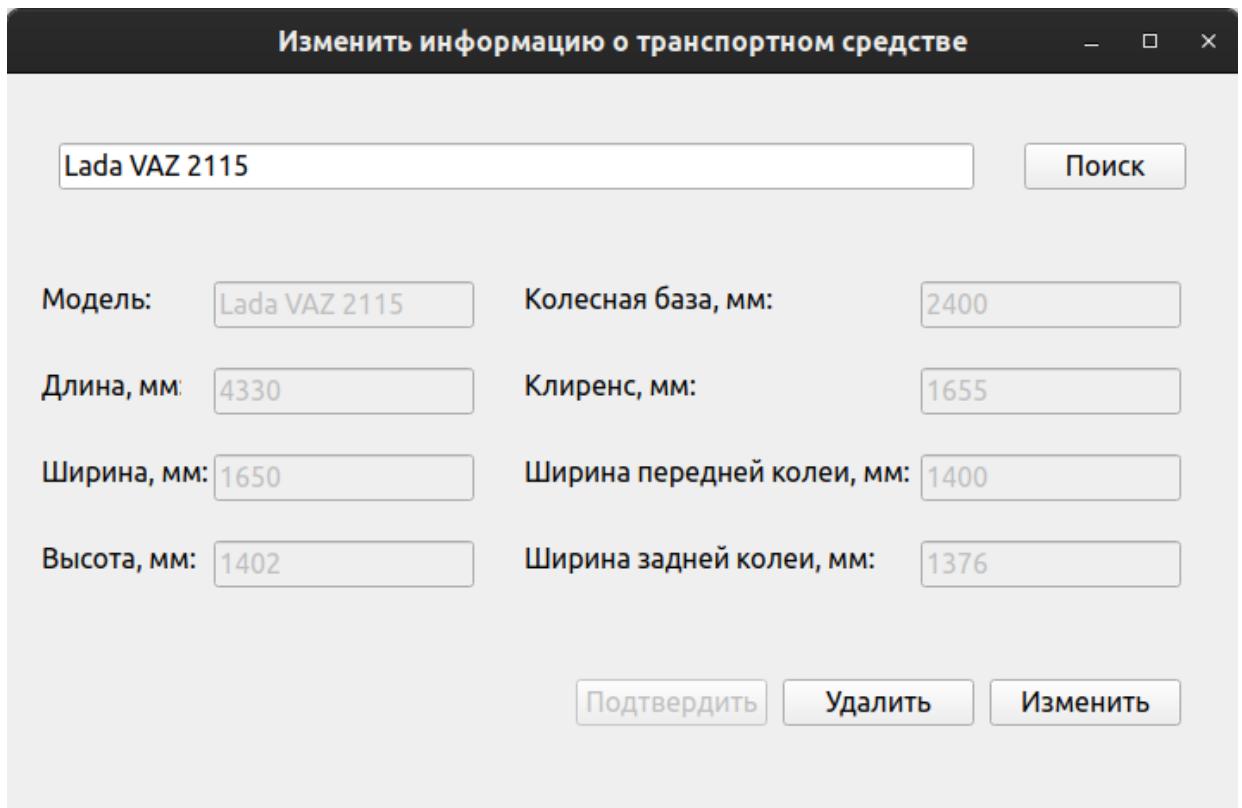


Рисунок 51 – Нажатие кнопки “Подтвердить”

При нажатии на кнопку “Удалить” программа запросит подтверждение удаления модели. Запрос подтверждения представлен на рисунке 52. Если пользователь подтвердит удаление, то произойдет удаление найденной модели, а все поля будут очищены. Если пользователь отклонит удаление, то программа вернется к состоянию до нажатия кнопки “Удалить”. Результат работы кнопки “Удалить” с подтверждением удаления представлен на рисунке 53.

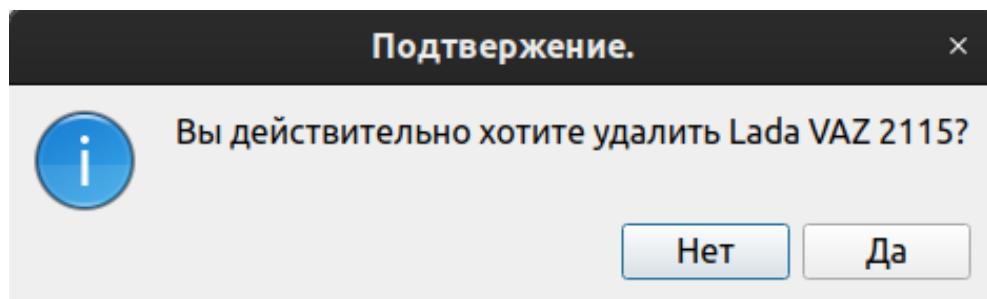


Рисунок 52 – Запрос подтверждения удаления

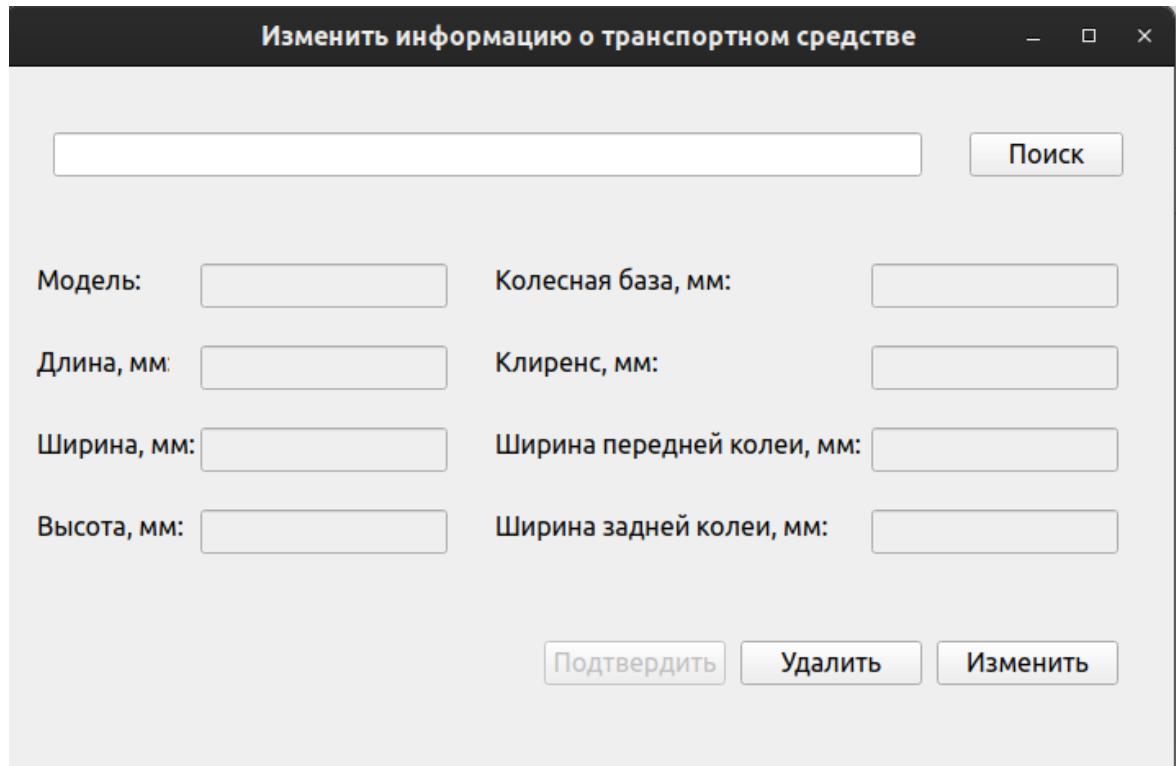


Рисунок 53 – Нажатие кнопки “Удалить” с подтверждением удаления

Если кнопка “Удалить” будет нажата, а модель транспортного средства не найдена, то программа выведет предупреждение, представленное на рисунке 54.

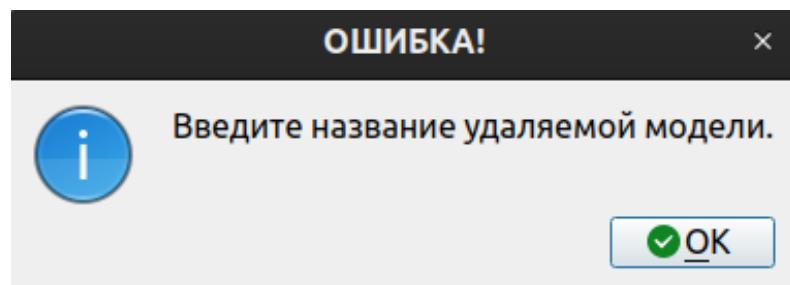


Рисунок 54 – Предупреждение отсутствия совпадений модели в БД

Для завершения работы любого окна необходимо нажать на крестик в правом верхнем углу окна, после чего программа закроет соединение с базой данных и завершит работу.

## ЗАКЛЮЧЕНИЕ

В результате выполнения ВКР мы выбрали метод решения поставленной задачи, смоделировали подходящие структуры данных, разработали алгоритмы сборки набора данных и работы программы, реализовали программу на языке высокого уровня Python с использованием объектно-ориентированного программирования, описали пользовательский интерфейс.

В результате нейронная сеть смогла научиться различать транспортные средства лишь единожды увидев их изображение. Таким образом, мы достигли поставленной задачи, сняв ограничение нейронной сети на максимальное количество распознаваемых классов.

Для полученной модели нейронной сети существует несколько вариантов дальнейшего развития.

Самым очевидным способом улучшения качества модели является увеличение количества уникальных изображений в наборе данных. С увеличением разнообразия исходных данных улучшится способность модели выявлять разнообразные признаки автомобилей. Для сравнения, в открытом наборе данных Cars196 представлено 196 классов и 16 тыс. уникальных изображений автомобилей.

Более сложным вариантом является переработка генератора изображений таким образом, чтобы максимизировать пользу подаваемых на обучение изображений: например, брать самый непохожий на якорь положительный ввод и загружать его вместе с самым похожим на якорь отрицательным вводом.

Еще одним способом улучшения модели является переработка или замена функции потерь. В последнее время появляется все больше экспериментальных методов, которые пытаются максимизировать качество обучения нейронных сетей. Однако, как правило, подобные методы требуют большого количества данных для эффективной работы, что возвращает нас к первому варианту развития.

## СПИСОК ЛИТЕРАТУРЫ

1. Тасс. Власти начали применять в России систему распознавания лиц и силуэтов людей и машин [Электронный ресурс]. – Режим доступа: <https://tass.ru/obschestvo/11736281>. – Дата доступа: 21.05.2022.
2. Ту, Дж. Принципы распознавания образов / Дж. Ту, Р. Гонсалес. – М.: Мир, 1978. – 416 с.
3. Распознавание образов [Электронный ресурс]. – Режим доступа: [http://it-claim.ru/Persons/Zelencov/Lecture\\_presentation.pdf](http://it-claim.ru/Persons/Zelencov/Lecture_presentation.pdf). – Дата доступа: 12.05.2022.
4. Москалев, Н.С. Виды архитектур нейронных сетей / Н.С. Москалев. – Молодой ученый, 2016. – 34 с.
5. Habr. Сверточная нейронная сеть [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/348000/>. – Дата доступа: 13.05.2022.
6. Neurohive. Обучение нейронной сети [Электронный ресурс]. – Режим доступа: <https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem/>. – Дата доступа: 15.05.2022.
7. Буркатов, А.В. Машинное обучение без лишних слов / А.В. Буркатов. – СПб.: Питер, 2019. – 192 с.
8. ResearchGate. Distance metric learning for large margin nearest neighbor classification [Электронный ресурс]. – Режим доступа: [https://www.researchgate.net/publication/221618254\\_Distance\\_Metric\\_Learning\\_for\\_Large\\_Margin\\_Nearest\\_Neighbor\\_Classification](https://www.researchgate.net/publication/221618254_Distance_Metric_Learning_for_Large_Margin_Nearest_Neighbor_Classification). – Дата доступа: 17.05.2022.
9. Pythonist. Введение в OpenCv библиотеки компьютерного зрения [Электронный ресурс]. – Режим доступа: <https://pythonist.ru/vvedenie-v-opencv-biblioteku-kompyuternogo-zreniya-na-python/>. – Дата доступа: 25.05.2022.
10. Neurohive. Как делать аугментации для распознавания объектов [Электронный ресурс]. – Режим доступа: <https://neurohive.io/ru/novosti/kak-delat-augmentaciju-dannyh-dlya-raspoznavaniya-obektov/>. – Дата доступа: 26.05.2022.

11. IT-Ghost. Теория и практика UML. Диаграмма состояний [Электронный ресурс]. – Режим доступа: [http://it-gost.ru/articles/view\\_articles/97](http://it-gost.ru/articles/view_articles/97). – Дата доступа: 01.06.2022.
12. Гитис, Л.Х. Кластерный анализ в задачах классификации, оптимизации и прогнозирования / Л.Х. Гитис. – М.: МГГУ, 2001. – 104 с.
13. Кластеризация данных [Электронный ресурс]. – Режим доступа: <https://logic.pdmi.ras.ru/~yura/internet/02ia-seminar-note.pdf>. – Дата доступа: 03.06.2022.
14. Алгоритмы кластеризации и многомерного шкалирования [Электронный ресурс]. – Режим доступа: <http://www.ccas.ru/voron/download/Clustering.pdf>. – Дата доступа: 04.06.2022.
15. Концептуальная модель кластеризации данных [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/kontseptualnaya-model-klasterizatsii-dannyh.pdf>. – Дата доступа: 04.06.2022.
16. Метрики оценки классификаторов [Электронный ресурс]. – Режим доступа: <https://core.ac.uk/download/pdf/196226627.pdf>. – Дата доступа: 07.06.2022.

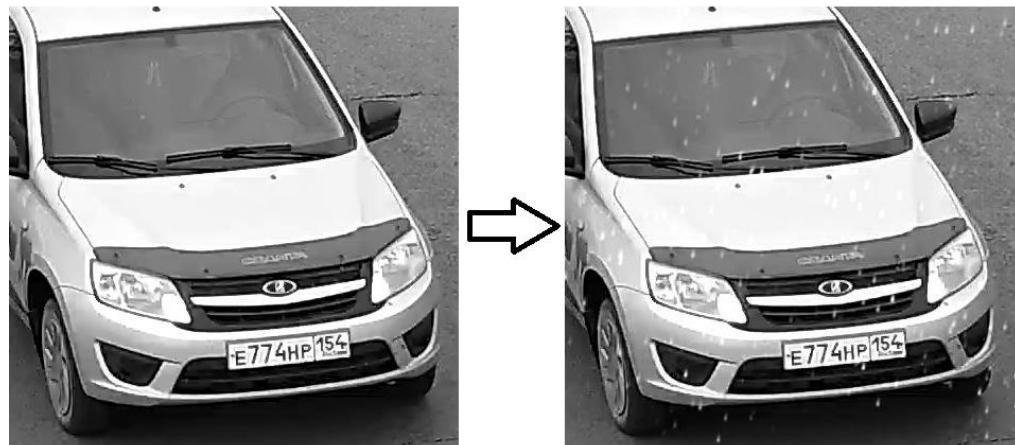
**Примеры аугментаций изображения**

Рисунок 1.1 – Погодный эффект снега

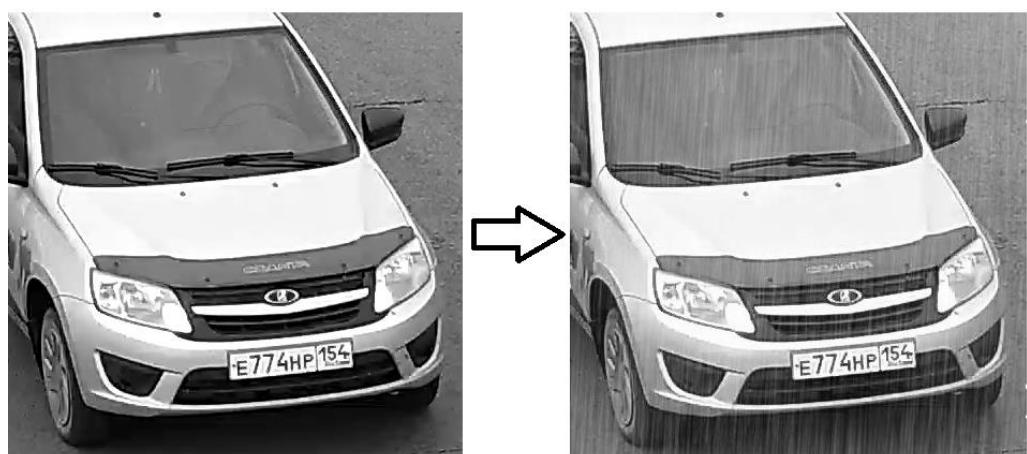


Рисунок 1.2 – Погодный эффект дождя



Рисунок 1.3 – Погодный эффект тумана



Рисунок 1.4 – Эффект размытия при движении

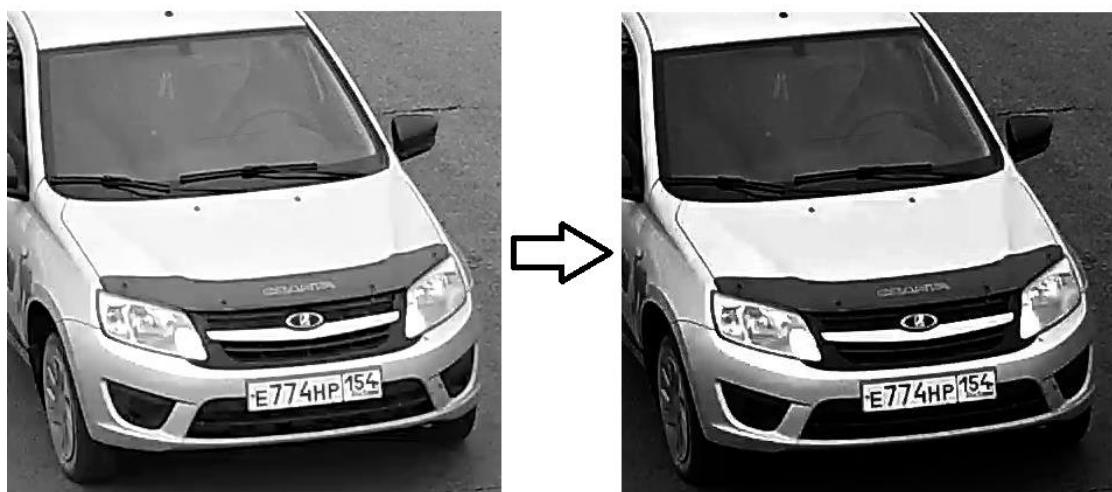


Рисунок 1.5 – Эффект контраста

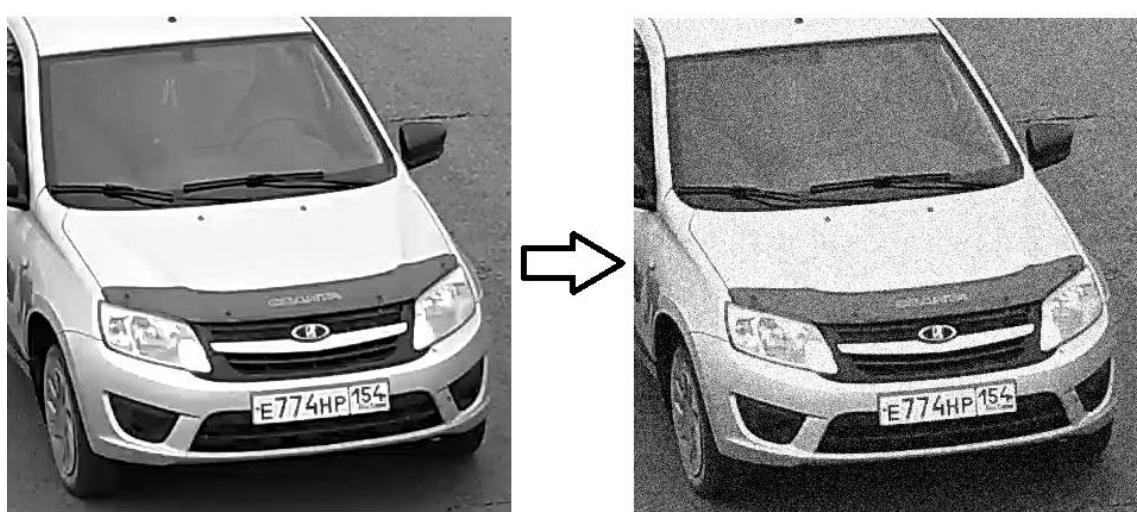


Рисунок 1.6 – Эффект шума



Рисунок 1.7 – Обрезка изображения



Рисунок 1.8 – Отражение изображения



Рисунок 1.9 – Поворот изображения



Рисунок 1.10 – Комбинация эффекта размытия при движении и поворота изображения



Рисунок 1.11 – Комбинация погодного эффекта дождя и отражения изображения

### Фрагмент программного кода обучения модели нейронной сети

```

'''Обучение нейронной сети.''''

import argparse
import os
import datetime
import tensorflow as tf
import tensorflow_addons as tfa

from tensorflow.keras.models import load_model
from tensorflow_addons.losses.metric_learning import pairwise_distance
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from tensorflow.keras.callbacks import ReduceLROnPlateau
from dataset.balanced_image_dataset import balanced_image_dataset_from_directory


def get_arguments():
    ap = argparse.ArgumentParser()
    ap.add_argument(
        "-mp",
        "--model_path",
        required=True,
        help="Path to .h5 keras model."
    )
    ap.add_argument(
        "--train_data",
        required=True,
        help="Path to training dataset."
    )
    ap.add_argument(
        "-vd",
        "--val_data",
        required=True,
        help="Path to val dataset."
    )
    ap.add_argument(
        "--epochs",
        type=int,
        required=True,
        help="Training epochs amount."
    )
    ap.add_argument(
        "--batch_size",
        type=int,
        default=128,
        help="Batch size."
    )
    ap.add_argument(
        "--model_name",
        type=str,
        default="model_new",
        help="Название эксперимента."
    )
    return vars(ap.parse_args())

```

```

def check(path):
    i = 0
    for folder in os.listdir(path):
        n = os.path.join(path, folder)
        i += len(os.listdir(n))
    return i

def triplet_accuracy(y_true, y_pred):
    batch_size = tf.cast(tf.size(y_true), dtype=tf.dtypes.float32)
    # Build pairwise distance matrix
    pdist_matrix = pairwise_distance(y_pred, squared=False) # was pairwise_distance
    # Build pairwise binary adjacency matrix.
    adjacency = tf.cast(tf.math.equal(y_true, tf.transpose(y_true)),
    dtype=tf.dtypes.float32)
    # Invert so we can select negatives only.
    adjacency_not = 1 - adjacency
    # Convert to decision with thresholding at 0.5
    predicted = tf.cast(tf.math.less_equal(pdist_matrix, 0.5), dtype=tf.dtypes.float32)
    # Calculate true positives and true negatives
    tp = tf.reduce_sum(
        tf.cast(
            tf.math.multiply(predicted, adjacency),
            dtype=tf.dtypes.float32
        )
    )
    tn = tf.reduce_sum(
        tf.cast(
            tf.math.multiply(1-predicted, adjacency_not),
            dtype=tf.dtypes.float32
        )
    )
    # Calculate percentage
    return (tp + tn) / (batch_size * batch_size)

def main():
    args = get_arguments()
    seed = 3
    img_width, img_height = 160, 160 # целевой размер изображения для обучения
    save_model_path = './models/' # путь к месту сохранения обученной модели
    # Train batch (128) = num_classes_per_batch_train * num_images_per_class_train (должно
    # быть кратно 2)
    num_classes_per_batch_train = 8
    num_images_per_class_train = 16
    # Val batch (64) = num_classes_per_batch_val * num_images_per_class_val (должно быть
    # кратно 2)
    num_classes_per_batch_val = 8
    num_images_per_class_val = 8
    nb_train_samples = check(args['train_data'])
    nb_val_samples = check(args['val_data'])
    model = load_model(args['model_path'], compile=False)
    exp_time = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    exp_name = "_".join(
        [args['model_name'], exp_time]
    )
    log_dir = os.path.join("./logs/", exp_time)
    # model = clone_model(model)
    # opt = SGD(lr=0.001, momentum=0.001, nesterov=True)

```

```

# opt = Adam(learning_rate=1e-5)
opt = RMSprop(learning_rate=0.001, centered=True)

model.compile(
    loss=tfa.losses.TripletSemiHardLoss(),
    optimizer=opt,
    metrics=[triplet_accuracy]
)

train_generator = balanced_image_dataset_from_directory(
    args['train_data'],
    num_classes_per_batch=num_classes_per_batch_train,
    num_images_per_class=num_images_per_class_train,
    image_size=(img_width, img_height),
    seed=seed,
    augment=False,
    safe_triplet=True
)
val_generator = balanced_image_dataset_from_directory(
    args['val_data'],
    num_classes_per_batch=num_classes_per_batch_val,
    num_images_per_class=num_images_per_class_val,
    image_size=(img_width, img_height),
    seed=seed,
    augment=False,
    safe_triplet=True
)

# datagen = ImageDataGenerator(rescale=1. / 255)
# train_generator = datagen.flow_from_directory(args['train_data'],
#                                               target_size=(img_width, img_height),
#                                               batch_size=args['batch_size'],
#                                               class_mode='sparse')
# val_generator = datagen.flow_from_directory(args['val_data'],
#                                              target_size=(img_width, img_height),
#                                              batch_size=args['batch_size'],
#                                              class_mode='sparse')
model_checkpoint = tf.keras.callbacks.ModelCheckpoint(
    filepath=os.path.join(save_model_path, f'{exp_name}_best.h5'),
    monitor='val_loss',
    mode='min',
    save_best_only=True
)
tensorboard_callback = tf.keras.callbacks.TensorBoard(
    log_dir=log_dir, histogram_freq=1
)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=25, verbose=1)
model.fit(
    train_generator,
    epochs=int(args['epochs']),
    callbacks=[tensorboard_callback, reduce_lr, model_checkpoint],
    validation_data=val_generator,
)

model.save(os.path.join(save_model_path, f'{exp_name}.h5'))

if __name__ == '__main__':
    main()

```

### Фрагмент программного кода главного окна ПО

```

# -*- coding: utf-8 -*-

import os
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog, QButtonGroup, QHeaderView, QWidget, QMessageBox
from PyQt5.QtGui import QIcon, QPixmap
from os.path import expanduser
from PyQt5.QtCore import Qt
import sqlite3
import numpy as np
from config_actions import *
from add_window import Ui_db_add_window
from search_window import Ui_search_window


class Ui_MainWindow(object):
    def __init__(self):
        self.image_path = None
        self.db = config_read()
        self.sqlite_connection = None
        self.cursor = None
        #self.add_array = None

    def load_db(self, db):
        '''Заружает БД и обработчик запросов.'''
        try:
            if not os.path.exists(db):
                print("База данных не обнаружена. Требуется ручная загрузка.")
                db = self.load_new_db()
                #return
            self.sqlite_connection = sqlite3.connect(db)
            self.cursor = self.sqlite_connection.cursor()
            print("База данных создана и успешно подключена к SQLite")
            sqlite_select_query = "select sqlite_version();"
            self.cursor.execute(sqlite_select_query)
            record = self.cursor.fetchall()
            print("Версия базы данных SQLite: ", record)
            #cursor.close()
            self.statusbar_text("Загружена база данных: " + db)
        except sqlite3.Error as error:
            print("Ошибка при подключении к sqlite", error)

    def close_db_connection(self):
        '''Закрытие соединения с БД.'''
        if self.cursor:
            self.cursor.close()
            self.sqlite_connection.close()
            print("Соединение с SQLite закрыто")
        return 0

    def load_new_db(self, message='БД не найдена, требуется загрузка новой...'):
        home = expanduser("~")

```

```

fname, _ = QFileDialog.getOpenFileName(None, message, home, "DB (*.db)")
config_write(fname)
print("Загружена новая база данных: ", fname)
return fname

def setupUi(self, MainWindow):
    self.main_window = MainWindow
    MainWindow.setObjectName("MainWindow")
    MainWindow.resize(800, 600)
    self.centralwidget = QtWidgets.QWidget(MainWindow)
    self.centralwidget.setObjectName("centralwidget")
    self.results_table = QtWidgets.QTableWidget(self.centralwidget)
    self.results_table.setGeometry(QtCore.QRect(20, 30, 341, 521))
    self.results_table.setObjectName("results_table")
    self.results_table.setColumnCount(2)
    self.results_table.setRowCount(8)

    #self.results_table.insertRow(rowPosition)
    #self.results_table.setItem(0 , 0, QtWidgets.QTableWidgetItem("Параметр"))
    #self.results_table.setItem(0 , 1, QtWidgets.QTableWidgetItem("Значение"))
    row_labels = ["Параметр", "Значение"]
    self.results_table.setHorizontalHeaderLabels(row_labels)

self.results_table.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeToContents)
self.results_table.horizontalHeader().setMinimumSectionSize(0)

    self.browse_button = QtWidgets.QPushButton(self.centralwidget)
    self.browse_button.setGeometry(QtCore.QRect(660, 70, 111, 41))
    self.browse_button.setObjectName("browse_button")
    self.browse_button.clicked.connect(self.browse_button_click) # обработка нажатия
    self.path_edit = QtWidgets.QTextEdit(self.centralwidget)
    self.path_edit.setGeometry(QtCore.QRect(390, 70, 251, 41))
    self.path_edit.setReadOnly(True)
    self.path_edit.setObjectName("path_edit")
    self.recognize_button = QtWidgets.QPushButton(self.centralwidget)
    self.recognize_button.setGeometry(QtCore.QRect(660, 500, 111, 41))
    self.recognize_button.setObjectName("recognize_button")
    self.recognize_button.clicked.connect(self.recognize_button_click)
    self.path_label = QtWidgets.QLabel(self.centralwidget)
    self.path_label.setGeometry(QtCore.QRect(390, 40, 341, 20))
    self.path_label.setAutoFillBackground(False)
    self.path_label.setScaledContents(False)
    self.path_label.setObjectName("path_label")

    self.image_label = QtWidgets.QLabel(self.centralwidget)
    self.image_label.setGeometry(QtCore.QRect(390, 365, 231, 231))
    self.image_label.setAutoFillBackground(False)
    self.image_label.setScaledContents(False)
    self.image_label.setObjectName("image_label")

    self.parameters_label = QtWidgets.QLabel(self.centralwidget)
    self.parameters_label.setGeometry(QtCore.QRect(390, 150, 181, 19))
    self.parameters_label.setObjectName("parameters_label")

    self.group1 = QButtonGroup()
    self.model_radiobutton = QtWidgets.QRadioButton(self.centralwidget)
    self.model_radiobutton.setGeometry(QtCore.QRect(390, 200, 151, 31))
    self.model_radiobutton.setObjectName("model_radiobutton")
    self.group1.addButton(self.model_radiobutton)

    self.group2 = QButtonGroup()

```

```

self.height_radiobutton = QtWidgets.QRadioButton(self.centralwidget)
self.height_radiobutton.setGeometry(QtCore.QRect(390, 240, 151, 31))
self.height_radiobutton.setObjectName("height_radiobutton")
self.group2.addButton(self.height_radiobutton)

self.group3 = QButtonGroup()
self.length_radiobutton = QtWidgets.QRadioButton(self.centralwidget)
self.length_radiobutton.setGeometry(QtCore.QRect(390, 280, 151, 31))
self.length_radiobutton.setObjectName("length_radiobutton")
self.group3.addButton(self.length_radiobutton)

self.group4 = QButtonGroup()
self.width_radiobutton = QtWidgets.QRadioButton(self.centralwidget)
self.width_radiobutton.setGeometry(QtCore.QRect(390, 320, 151, 31))
self.width_radiobutton.setObjectName("width_radiobutton")
self.group4.addButton(self.width_radiobutton)

self.group5 = QButtonGroup()
self.wheel_distance_radiobutton = QtWidgets.QRadioButton(self.centralwidget)
self.wheel_distance_radiobutton.setGeometry(QtCore.QRect(540, 200, 151, 31))
self.wheel_distance_radiobutton.setObjectName("wheel_distance_radiobutton")
self.group5.addButton(self.wheel_distance_radiobutton)

self.group6 = QButtonGroup()
self.clearance_radiobutton = QtWidgets.QRadioButton(self.centralwidget)
self.clearance_radiobutton.setGeometry(QtCore.QRect(540, 240, 151, 31))
self.clearance_radiobutton.setObjectName("clearance_radiobutton")
self.group6.addButton(self.clearance_radiobutton)

self.group7 = QButtonGroup()
self.front_track_width_radiobutton = QtWidgets.QRadioButton(self.centralwidget)
self.front_track_width_radiobutton.setGeometry(QtCore.QRect(540, 280, 221, 31))
self.front_track_width_radiobutton.setObjectName("front_track_width_radiobutton")
self.group7.addButton(self.front_track_width_radiobutton)

self.group8 = QButtonGroup()
self.back_track_width_radiobutton = QtWidgets.QRadioButton(self.centralwidget)
self.back_track_width_radiobutton.setGeometry(QtCore.QRect(540, 320, 211, 31))
self.back_track_width_radiobutton.setObjectName("back_track_width_radiobutton")
self.group8.addButton(self.back_track_width_radiobutton)

MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 24))
self.menubar.setObjectName("menubar")
# ПОДМЕНЮ для пререхода на другие окна.
self.databaseMenu = self.menubar.addMenu('&База данных')

add_databaseMenu = QtWidgets.QAction('&Добавить', self.centralwidget)
add_databaseMenu.triggered.connect(self.db_add_vehicle)
self.databaseMenu.addAction(add_databaseMenu)

add_databaseMenu = QtWidgets.QAction('&Изменить', self.centralwidget)
add_databaseMenu.triggered.connect(self.db_search_vehicle)
self.databaseMenu.addAction(add_databaseMenu)

load_databaseMenu = QtWidgets.QAction('&Загрузить БД...', self.centralwidget)
load_databaseMenu.triggered.connect(self.menu_load_db)
self.databaseMenu.addAction(load_databaseMenu)

MainWindow.setMenuBar(self.menubar)

```

```

self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)
self.load_db(self.db)
self.load_tflite()

def load_tflite(self):
    '''Загружает tflite модель нейронной сети.'''
    from vtc.classifier import Classifier
    self.classifier = Classifier("./vtc/model_exp2.tflite", 3.4)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Распознавание транспортного
средства"))
    self.browse_button.setText(_translate("MainWindow", "Загрузить"))
    self.recognize_button.setText(_translate("MainWindow", "Пуск"))
    self.path_label.setText(_translate("MainWindow", "Выберите изображение для
распознавания"))
    self.parameters_label.setText(_translate("MainWindow", "Выберите параметры"))
    self.model_radiobutton.setText(_translate("MainWindow", "Модель"))
    self.height_radiobutton.setText(_translate("MainWindow", "Высота"))
    self.length_radiobutton.setText(_translate("MainWindow", "Длина"))
    self.width_radiobutton.setText(_translate("MainWindow", "Ширина"))
    self.wheel_distance_radiobutton.setText(_translate("MainWindow", "Колесная база"))
    self.clearance_radiobutton.setText(_translate("MainWindow", "Клиренс"))
    self.front_track_width_radiobutton.setText(_translate("MainWindow", "Ширина
передней колеи"))
    self.back_track_width_radiobutton.setText(_translate("MainWindow", "Ширина задней
колеи"))

def statusbar_text(self, text):
    '''Выводит сообщение в statusbar.'''
    self.statusbar.showMessage(text)

def set_table(self, model, height, length, width, wheel_distance, clearance, ftw, btw):
    i = 0
    if self.model_radiobutton.isChecked():
        self.results_table.setItem(i, 0, QtWidgets.QTableWidgetItem("Модель"))
        self.results_table.setItem(i, 1, QtWidgets.QTableWidgetItem(model))
        i += 1
    if self.height_radiobutton.isChecked():
        self.results_table.setItem(i, 0, QtWidgets.QTableWidgetItem("Высота, мм"))
        self.results_table.setItem(i, 1, QtWidgets.QTableWidgetItem(str(height)))
        i += 1
    if self.length_radiobutton.isChecked():
        self.results_table.setItem(i, 0, QtWidgets.QTableWidgetItem("Длина, мм"))
        self.results_table.setItem(i, 1, QtWidgets.QTableWidgetItem(str(length)))
        i += 1
    if self.width_radiobutton.isChecked():
        self.results_table.setItem(i, 0, QtWidgets.QTableWidgetItem("Ширина, мм"))
        self.results_table.setItem(i, 1, QtWidgets.QTableWidgetItem(str(width)))
        i += 1
    if self.wheel_distance_radiobutton.isChecked():
        self.results_table.setItem(i, 0, QtWidgets.QTableWidgetItem("Колесная база,
мм"))
        self.results_table.setItem(i, 1, QtWidgets.QTableWidgetItem(str(wheel_distance)))
        i += 1,

```

```

        i += 1
    if self.clearance_radiobutton.isChecked():
        self.results_table.setItem(i , 0, QtWidgets.QTableWidgetItem("Клиренс, мм"))
        self.results_table.setItem(i , 1, QtWidgets.QTableWidgetItem(str(clearance)))
        i += 1
    if self.front_track_width_radiobutton.isChecked():
        self.results_table.setItem(i , 0, QtWidgets.QTableWidgetItem("Ширина передней
колеи, мм"))
        self.results_table.setItem(i , 1, QtWidgets.QTableWidgetItem(str(ftw)))
        i += 1
    if self.back_track_width_radiobutton.isChecked():
        self.results_table.setItem(i , 0, QtWidgets.QTableWidgetItem("Ширина задней
колеи, мм"))
        self.results_table.setItem(i , 1, QtWidgets.QTableWidgetItem(str(btw)))
        i += 1

def browse_button_click(self):
    '''Нажатие на кнопку загрузки изображения, выводит и передает путь до
изображения.'''
    home = expanduser("~/")
    fname, _ = QFileDialog.getOpenFileName(None, 'Open file', home, "Image (*.png *.jpg
*jpeg)")
    self.path_edit.setPlainText(fname)
    pixmap = QPixmap(fname)
    pixmap_resized = pixmap.scaled(231, 231, QtCore.Qt.KeepAspectRatio)
    self.image_label.setPixmap(pixmap_resized)
    self.image_path = fname

def prep(self, l):
    l = l[1:]
    l[-1] = l[-1][:-2]
    if l[-1] == '':
        l = l[:-1]
    for num in l:
        num = float(num)
    l = np.asarray(l).astype(np.float32)
    return l

def recognize_button_click(self):
    if not self.image_path:
        msgBox = QMessageBox()
        msgBox.setIcon(QMessageBox.Information)
        msgBox.setText("Загрузите изображение.")
        msgBox.setWindowTitle("ОШИБКА!")
        msgBox.setStandardButtons(QMessageBox.Ok)
        returnValue = msgBox.exec()
        if returnValue == QMessageBox.Ok:
            print('OK clicked')
            return
    embedding = self.classifier.get_embedding_vector(self.image_path)
    sql = '''SELECT model, embedding FROM main.vehicle'''
    self.cursor.execute(sql)
    rows = self.cursor.fetchall()
    min = 3.4 # 100
    vehicle_class = None
    for row in rows:
        new_distance      =      self.classifier.get_distance(self.prep(row[1].split())),
embedding)
        if new_distance < min:
            min = new_distance

```

```

        vehicle_class = row[0]
print(vehicle_class) # предсказанный класс
if vehicle_class != None:
    sql_get_info = '''SELECT      *      FROM      main.vehicle      WHERE      model      =
 "{0}""".format(vehicle_class)
    self.cursor.execute(sql_get_info)
    info = self.cursor.fetchall()
    print(info[0])
    self.set_table(info[0][1],  info[0][2],  info[0][3],  info[0][4],  info[0][5],
info[0][6],  info[0][7],  info[0][8])
else:
    msgBox = QMessageBox()
    msgBox.setIcon(QMessageBox.Information)
    msgBox.setText("Данный автомобиль не найден.")
    msgBox.setWindowTitle("Предупреждение!")
    msgBox.setStandardButtons(QMessageBox.Ok)
    returnValue = msgBox.exec()
    if returnValue == QMessageBox.Ok:
        print('OK clicked')
        return
    return

def menu_load_db(self):
    self.load_db(self.load_new_db("Загрузите базу данных..."))

def db_add_vehicle(self):#, model, height, length, width, wheel_distance, clearance,
front_track_width, back_track_width, embedding):
    '''Добавляет строку в таблицу vehicle в БД.'''
    self.add_window = QtWidgets.QMainWindow()
    self.ui = Ui_db_add_window(self.classifier, self.sqlite_connection, self.cursor) # parameters
    self.ui.setupUi(self.add_window, self)
    self.add_window.show()

def db_search_vehicle(self):
    '''Выполняет поиск автомобиля по названию модели.'''
    self.search_window = QtWidgets.QMainWindow()
    self.ui = Ui_search_window(self.sqlite_connection, self.cursor) # parameters
    self.ui.setupUi(self.search_window, self)
    self.search_window.show()

```

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

УДОСТОВЕРЯЮЩИЙ ЛИСТ № 180818  
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ  
на демонстрационный материал, представленный в электронном виде

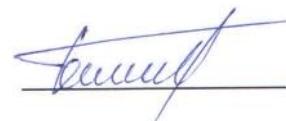
Студента Беликова Павла Геннадьевича        шифр 180818  
Институт приборостроения, автоматизации и информационных технологий  
Кафедра информационных систем и цифровых технологий  
Направление 09.03.04 Программная инженерия  
Направленность (профиль) Промышленная разработка программного обеспечения

Наименование документа: Демонстрационные плакаты к выпускной  
квалификационной работе

Документ разработал:

Студент

Беликов П.Г.



Документ согласован:

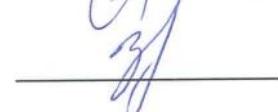
Руководитель

Конюхова О.В.



Нормоконтроль

Захарова О.В.



Документ утвержден:

Зав. кафедрой

Волков В.Н.



Орел 2022

**ИНФОРМАЦИОННО-ПОИСКОВАЯ ХАРАКТЕРИСТИКА  
ДОКУМЕНТА НА ЭЛЕКТРОННОМ НОСИТЕЛЕ**

Наименование		Характеристики документа на электронном носителе
группы атрибутов	атрибута	
1. Описание документа	Обозначение документа (идентификатор(ы) файла(ов))	\Плакаты\Презентация.pptx
	Наименование документа	Демонстрационные плакаты к выпускной квалификационной работе
	Класс документа	ЕСКД
	Вид документа	Оригинал документа на электронном носителе
	Аннотация	Демонстрационный материал, отображающий основные этапы выполнения выпускной квалификационной работы
2. Даты и время	Использование документа	Операционная система Windows 10, Microsoft PowerPoint 2016
	Дата и время копирования документа	11.06.2022
	Дата создания документа	06.06.2022
3. Создатели	Дата утверждения документа	11.06.2022
	Автор	Беликов П.Г.
	Изготовитель	Беликов П.Г.
4. Внешние ссылки	Ссылки на другие документы	Удостоверяющий лист № 180818
5. Защита	Санкционирование	ОГУ имени И.С. Тургенева
	Классификация защиты	По законодательству РФ
6. Характеристики содержания	Объем информации документа	6280729 Б

7. Структура документа(ов)	Наименование плаката (слайда) №1	Титульный лист
	Наименование плаката (слайда) №2	Цель и задачи работы
	Наименование плаката (слайда) №3	Сравнение аналогов
	Наименование плаката (слайда) №4	Требования к системе и сравнение точности моделей нейронных сетей
	Наименование плаката (слайда) №5	Общая структура системы и ее функционал
	Наименование плаката (слайда) №6	Концептуальная модель ПО в виде диаграммы классов
	Наименование плаката (слайда) №7	Состояния и переходы системы во время работы
	Наименование плаката (слайда) №8	Карта диалоговых окон
	Наименование плаката (слайда) №9	Основные алгоритмы работы ПО
	Наименование плаката (слайда) №10	Аугментация изображений
	Наименование плаката (слайда) №11	Результат обучения модели
	Наименование плаката (слайда) №12	Пример работы программы
	Наименование плаката (слайда) №13	Демонстрация работы программного обеспечения



## СПРАВКА

о результатах проверки текстового документа  
на наличие заимствований

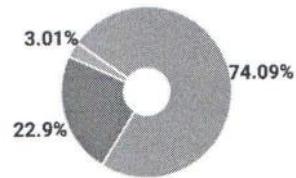
Орловский государственный университет  
имени И.С. Тургенева

ПРОВЕРКА ВЫПОЛНЕНА В СИСТЕМЕ АНТИПЛАГИАТ.ВУЗ

Автор работы: Беликов Павел Геннадиевич  
Самоцитирование  
рассчитано для: Беликов Павел Геннадиевич  
Название работы: VKR\_090304\_Belikov\_P\_G\_2022  
Тип работы: Выпускная квалификационная работа  
Подразделение: ИПАИТ, кафедра ИСиЦТ

### РЕЗУЛЬТАТЫ

ЗАИМСТВОВАНИЯ	<div style="width: 22.9%;"></div>	22.9%
ОРИГИНАЛЬНОСТЬ	<div style="width: 74.09%;"></div>	74.09%
ЦИТИРОВАНИЯ	<div style="width: 3.01%;"></div>	3.01%
САМОЦИТИРОВАНИЯ	<div style="width: 0%;"></div>	0%



ДАТА ПОСЛЕДНЕЙ ПРОВЕРКИ: 14.06.2022

Модули поиска: ИПС Адилет; Библиография; Сводная коллекция ЭБС; Интернет Плюс; Сводная коллекция РГБ; Цитирование; Переводные заимствования (RuEn); Переводные заимствования по eLIBRARY.RU (EnRu); Переводные заимствования по Интернету (EnRu); Переводные заимствования издательства Wiley (RuEn); eLIBRARY.RU; СПС ГАРАНТ; Медицина; Диссертации НББ; Перефразирования по eLIBRARY.RU; Перефразирования по Интернету; Патенты СССР, РФ, СНГ; СМИ России и СНГ; Шаблонные фразы; Модуль поиска "ФГБОУ ВО ОГУ им. И.С.Тургенева"; Кольцо вузов; Издательство Wiley; Переводные заимствования

Работу проверил: Ужаринский Антон Юрьевич

ФИО проверяющего

Дата подписи: 14.06.2022

Подпись проверяющего



Чтобы убедиться  
в подлинности справки, используйте QR-код,  
который содержит ссылку на отчет.

Ответ на вопрос, является ли обнаруженное заимствование  
корректным, система оставляет на усмотрение проверяющего.  
Предоставленная информация не подлежит использованию  
в коммерческих целях.