

Минобрнауки России
Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ПО ПРОГРАММЕ БАКАЛАВРИАТА

09.03.04 Программная инженерия

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

«Разработка программно-информационных систем»

Интеллектуальная система распознавания объектов по цветовым

характеристикам на основе нечетких нейронных сетей

(название темы)

Дипломный проект

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР

Б.В. Тягунов

(подпись, дата)

(инициалы, фамилия)

Группа ПО-026

Руководитель ВКР

Р.А. Томакова

(подпись, дата)

(инициалы, фамилия)

Нормоконтроль

А. А. Чаплыгин

(подпись, дата)

(инициалы, фамилия)

ВКР допущена к защите:

Заведующий кафедрой

А. В. Малышев

(подпись, дата)

(инициалы, фамилия)

Курск 2024 г.

Минобрнауки России
Юго-Западный государственный университет

Кафедра программной инженерии

УТВЕРЖДАЮ:
Заведующий кафедрой

(подпись, инициалы, фамилия)

«_____» 20____ г.

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ
РАБОТУ ПО ПРОГРАММЕ БАКАЛАВРИАТА**

Студента Тягунова В.В., шифр 20-06-0149, группа ПО-026

1. Тема «Интеллектуальная система распознавания объектов по цветовым характеристикам на основе нечетких нейронных сетей» утверждена приказом ректора ЮЗГУ от «04» апреля 2024 г. № 1616-с.

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

3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

- 1) проанализировать ИТ-инфраструктуру предприятия;
- 2) разработать концептуальную модель системы управления ИТ-инфраструктурой предприятия на основе подхода к управлению и организации ИТ-услуг ITSM;
- 3) спроектировать программную систему управления ИТ-инфраструктурой предприятия;
- 4) сконструировать и протестировать программную систему управления ИТ-инфраструктурой предприятия.

3.2. Входные данные и требуемые результаты для программы:

- 1) Входными данными для программной системы являются: данные справочников комплектующих, конфигураций, ПО, критериев качества SLA,

ИТ-услуг, департаментов компании; технические данные ИТ-ресурсов; данные входящих заявок на ИТ-ресурсы; данные запросов поставщикам на комплектующие.

2) Выходными данными для программной системы являются: сформированные заявки на обслуживание ИТ-ресурсов; сформированные запросы на закупку комплектующих; сведения о выполненных работах по заявкам; статусы заявок; выходные отчеты (инфографика) – по качеству услуг, по состоянию ИТ-ресурсов, по деятельности ИТ-отдела, по стоимости обслуживания ИТ-ресурсов, воронка заявок.

4. Содержание работы (по разделам):

4.1. Введение

4.1. Анализ предметной области

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

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

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

4.5. Заключение

4.6. Список использованных источников

5. Перечень графического материала:

Лист 1. Сведения о ВКРБ

Лист 2. Цель и задачи разработки

Лист 3. Диаграммы компонентов

Лист 4. Архитектура нейронной сети

Лист 5. Функция принадлежности

Лист 6. База данных

Лист 7. Интерфейс приложения

Лист 8. Демонстрация работы программы

Лист 9. Заключение

Руководитель ВКР

(подпись, дата)

Р.А. Томакова

(инициалы, фамилия)

Задание принял к исполнению

(подпись, дата)

В.В. Тягунов

(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 90 страницам. Работа содержит 37 иллюстраций, 1 таблицу, 13 библиографических источников и 9 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: компьютерное зрение, язык Python, средства MATLAB, приложение, нечёткая нейронная сеть, система, информатизация, автоматизация, информационные технологии, классы, ANFIS, база данных, компонент, нечёткая логика, модуль, сущность, метод, пользователь, нейросеть, стационарное приложение.

Объектом разработки является стационарное приложение, обеспечивающее возможность распознавания объектов по цветовым характеристикам на основе нечётких нейронных сетей.

Целью выпускной квалификационной работы является проведение глубокого анализа и исследования передовых технологий в области компьютерного зрения и искусственного интеллекта. Работа фокусируется на применении нечётких нейронных сетей, что позволяет значительно повысить точность и эффективность процесса распознавания объектов.

В ходе разработки приложения был выполнен ряд ключевых шагов: определение и структурирование основных сущностей через создание информационных модулей, применение классов и методов для управления данными в соответствии с предметной областью и для обеспечения надёжности функционирования приложения. Была спроектирована архитектура нечёткой нейронной сети и базы данных, а также разработан пользовательский интерфейс, облегчающий взаимодействие с приложением.

Для создания веб-сайта были использованы возможности языка программирования Python в сочетании с инструментарием библиотек MATLAB, что позволило реализовать необходимые функциональные возможности.

ABSTRACT

The amount of work is equal to 90 pes. The work contains 37 illustrations, 1 tables, 13 bibliographic sources and 9 sets of graphic material. Number of applications – 2. Graphic material is presented in Appendix A. Source code fragments are presented in Appendix B.

List of keywords: computer vision, Python language, MATLAB tools, application, fuzzy neural network, system, informatization, automation, information technology, classes, ANFIS, database, component, fuzzy logic, module, entity, method, user, neural network, stationary application.

The object of development is a stationary application that provides the ability to recognize objects by color characteristics based on fuzzy neural networks.

The purpose of the final qualifying work is to conduct in-depth analysis and research of advanced technologies in the field of computer vision and artificial intelligence. The work focuses on the use of fuzzy neural networks, which can significantly improve the accuracy and efficiency of the object recognition process.

During the development of the application, a number of key steps were completed: defining and structuring the main entities through the creation of information modules, using classes and methods to manage data in accordance with the subject area and to ensure the reliability of the application. The architecture of the fuzzy neural network and database was designed, and a user interface was developed to facilitate interaction with the application.

To create the website, the capabilities of the Python programming language were used in combination with the tools of the MATLAB libraries, which made it possible to implement the necessary functionality.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	11
1 Анализ предметной области	13
1.1 Понятие искусственной нейронной сети	13
1.1.1 Распознавание образов и классификация	14
1.1.2 Образы и классификация	14
1.1.3 Кластеризация и новые классы	17
1.1.4 Архитектура нейронных сетей	17
1.1.5 Прогнозирование	18
1.1.6 Аппроксимация	18
1.1.7 Сжатие данных и ассоциативная память	19
1.2 Понятие нечёткой логики	19
1.2.1 Символическая нечёткая логика	19
1.2.2 Синтез функций непрерывной логики заданных таблично	20
1.2.3 Теория приближённых вычислений	21
1.2.4 Нечёткая логика и нейронные сети	21
1.2.5 Байесовская вероятность	22
1.3 Понятие адаптивной системы нейро-нечеткого вывода	22
1.3.1 Слой фазификации	24
2 Техническое задание	25
2.1 Основание для разработки	25
2.2 Цель и назначение разработки	25
2.3 Требования пользователя к интерфейсу приложения	25
2.4 Моделирование вариантов использования	26
2.5 Требования к оформлению документации	27
3 Технический проект	28
3.1 Общая характеристика организации решения задачи	28
3.2 Обоснование выбора технологии проектирования	28
3.2.1 Python и его библиотеки	28
3.2.2 Архитектура нечёткой нейронной сети	29

3.2.3 Описание базы данных	30
3.3 Диаграмма компонентов	31
3.3.1 Структура компонентов	31
3.3.2 Взаимодействие компонентов	32
3.4 Содержание информационных блоков. Основные сущности	32
3.4.1 Структура сущности графический интерфейс	33
3.4.2 Структура сущности предварительная обработка	34
3.4.3 Структура сущности нейронная сеть	34
3.4.4 Структура сущности база данных	34
3.4.5 Структура сущности анализ данных	34
4 Рабочий проект	35
4.1 Классы, используемые при разработке сайта	35
4.2 Модульное тестирование разработанного приложения	42
4.3 Системное тестирование разработанного приложения	46
ЗАКЛЮЧЕНИЕ	70
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	71
ПРИЛОЖЕНИЕ А Представление графического материала	74
ПРИЛОЖЕНИЕ Б Фрагменты исходного кода программы На отдельных листах (CD-RW в прикрепленном конверте)	84
Сведения о ВКРБ (Графический материал / Сведения о ВКРБ.png)	Лист 1
Цель и задачи разработки (Графический материал / Цель и задачи разработки.png)	Лист 2
Диаграммы компонентов (Графический материал / Диаграммы компонентов.png)	Лист 3
Архитектура нейронной сети (Графический материал / Архитектура нейронной сети.png)	Лист 4
Функция принадлежности (Графический материал / Функция принадлежности.png)	Лист 5
База данных (Графический материал / База данных.png)	Лист 6
Интерфейс приложения (Графический материал / Интерфейс приложения.png)	Лист 7

Демонстрация работы программы (Графический материал / Демонстрация работы программы.png) Лист 8

Заключение (Графический материал / Заключение.png) Лист 9

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ИНС - Искусственная нейронная система

SOM(Self-Organizing Maps) - Самоорганизующиеся карты

ANFIS(Adaptive Neuro Fuzzy Inference System) - Адаптивная система
нейро-нечёткого вывода

ВВЕДЕНИЕ

В эпоху цифровизации и автоматизации процессов, особое значение приобретает развитие технологий компьютерного зрения. Одним из перспективных направлений является создание систем, способных распознавать и классифицировать объекты на изображениях с высокой точностью и скоростью. Это становится возможным благодаря применению адаптивных нечётких нейронных сетей, которые могут анализировать цветовые характеристики объектов и адаптироваться к различным условиям освещения и цветовым профилям. Такие системы находят широкое применение в разнообразных областях, включая медицинскую диагностику, системы безопасности и многие другие.

Современные достижения в области искусственного интеллекта и машинного обучения открывают новые горизонты в разработке программного обеспечения, способного выполнять задачи, ранее считавшиеся недостижимыми для автоматизированных систем. Разработка стационарного приложения с адаптивной нечёткой нейронной сетью для распознавания объектов на изображениях по цветовым характеристикам является одним из таких прорывных проектов. Стационарные системы обладают рядом преимуществ, включая повышенную вычислительную мощность и расширенные возможности для интеграции с другими технологическими решениями, что делает их идеальными для сложных задач компьютерного зрения.

Цель настоящей работы – разработка приложения на базе адаптивной нечёткой нейронной сети для распознавания и классификации объектов на изображениях по цветовым характеристикам. Приложение предназначено для улучшения процессов анализа изображений, обеспечивая высокую точность и скорость обработки данных. Особое внимание уделяется разработке алгоритмов, способных адаптироваться к изменяющимся условиям и разнообразию цветовых профилей объектов, что делает его применимым в различных областях, от медицинской диагностики до систем безопасности. Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ предметной области;
- разработать концептуальную модель приложения;
- спроектировать приложение;
- реализовать приложение.

Структура и объем работы. Отчет состоит из введения, 4 разделов основной части, заключения, списка использованных источников, 2 приложений. Текст выпускной квалификационной работы равен 90 страницам.

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

В первом разделе на стадии описания технической характеристики предметной области приводится сбор информации о используемой архитектуре.

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

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

В четвертом разделе приводится список классов и их методов, использованных при разработке сайта, производится тестирование разработанного сайта.

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

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

1 Анализ предметной области

1.1 Понятие искусственной нейронной сети

Нейронная сеть, также именуемая искусственной нервной сетью (ИНС), это математическая модель, которая послужила программным или аппаратным воплощением принципов функционирования биологических нейронных сетей - сетей нервных клеток живых организмов. Этот термин возник в результате изучения процессов, происходящих в мозге, и попыток эмулировать их. Первые усилия по созданию нейронных сетей были предприняты У. Маккалоком и У. Питтсом. С развитием алгоритмов обучения этих моделей стали широко применяться в практических задачах, таких как предсказание, распознавание образов, управление и прочие.

Искусственная нейронная сеть (ИНС) - это архитектура, в которой простые вычислительные элементы, называемые искусственными нейронами, взаимодействуют между собой для выполнения сложных задач. Каждый нейрон обрабатывает входные сигналы и передает выходные сигналы другим нейронам в сети. Хотя отдельные нейроны могут быть простыми, их коллективное взаимодействие в большой сети позволяет выполнять сложные вычисления и задачи.

Искусственная нейронная сеть (ИНС) представляет собой комплекс систематически связанных и взаимодействующих между собой простых вычислительных элементов, называемых искусственными нейронами. В контексте машинного обучения, они функционируют как специализированный вид методов для распознавания образов и дискриминантного анализа. Математически они представляют собой сложную многопараметрическую задачу нелинейной оптимизации. В кибернетике они используются для адаптивного управления и в робототехнике. С точки зрения развития вычислительной техники и программирования, они являются мощным инструментом для решения проблемы эффективного параллелизма.

Важно отметить, что нейронные сети не programmedны в традиционном понимании этого термина; они обучаются. Процесс обучения заклю-

чается в настройке параметров связей между нейронами на основе представленных данных. Этот процесс позволяет нейронным сетям обнаруживать сложные закономерности в данных, выполнять обобщение и возвращать верные результаты даже на основе данных, которые не были представлены в процессе обучения или были представлены с искажениями.

1.1.1 Распознавание образов и классификация

Когда мы говорим о различных ”образах“ мы имеем в виду разнообразные объекты, такие как текстовые символы, изображения, звуковые образцы и так далее. В процессе обучения нейронной сети предъявляются эти различные образы, каждому из которых присваивается определенный класс. Образец представляется в виде вектора значений признаков, и сеть учится определять, к какому классу относится каждый образец. Важно, чтобы набор признаков однозначно определял класс образца. Если признаков недостаточно, сеть может неправильно классифицировать образец, связывая его с несколькими классами.

После завершения обучения количество нейронов в выходном слое сети обычно соответствует количеству классов. Каждый нейрон в выходном слое представляет определенный класс, и сеть выдаёт ответ о принадлежности образца к тому или иному классу. Если сеть уверена в классификации, на одном из выходов появится признак принадлежности к классу, а на других выходах этот признак отсутствует. Однако, если сеть не уверена, может возникнуть ситуация, когда на нескольких выходах присутствует признак принадлежности к классу, что указывает на неопределенность сети в своём ответе.

1.1.2 Образы и классификация

При обучении нейронной сети различные типы данных, такие как текст, изображения и звук, представлены в виде образов и привязаны к определенным классам. Сеть изучает эти образы и их признаки, чтобы точно классифицировать их. Важно, чтобы эти признаки явно указывали на класс об-

разца. По завершении обучения количество нейронов в выходном слое сети соответствует количеству классов. Каждый нейрон представляет определенный класс, и сеть выдает ответ о принадлежности образца к определенному классу. В случае неопределенности сеть может указать на несколько возможных классов.

Классификация по формату входной информации:

1. Аналоговые нейронные сети: работают с информацией в форме действительных чисел;
2. Двоичные нейронные сети: оперируют с информацией, представленной в двоичном виде;
3. Образные нейронные сети: оперируют с информацией, представленной в виде образов, таких как знаки, иероглифы или символы.

Классификация по типу обучения:

1. Обучение с учителем: Нейронная сеть использует известные пары входных данных и соответствующих им выходных значений для обучения. В этом случае выходное пространство решений известно заранее;
2. Обучение без учителя: Нейронная сеть формирует выходное пространство решений только на основе входных данных, без предоставления соответствующих выходных значений. Такие сети называют самоорганизующимися, так как они обнаруживают структуры или паттерны в данных без явного учителя;
3. Обучение с подкреплением: В этом типе обучения система взаимодействует со средой, принимая последовательность действий и получая за них награды или наказания. Цель состоит в том, чтобы оптимизировать стратегию действий так, чтобы максимизировать суммарную награду в долгосрочной перспективе.

Существует два типа классификации синапсов по характеру настройки:

1. Сети с фиксированными связями: Весовые коэффициенты нейронной сети выбираются заранее и остаются неизменными на протяжении работы сети. Это означает, что они не подвергаются изменениям в процессе обучения и задаются исходными условиями задачи;

2. Сети с динамическими связями: В этих сетях весовые коэффициенты синапсов настраиваются в процессе обучения. Это позволяет сети адаптироваться к новой информации и улучшать свою производительность в зависимости от задачи или окружающей среды;

3. Обучение с подкреплением: В этом типе обучения система взаимодействует со средой, принимая последовательность действий и получая за них награды или наказания. Цель состоит в том, чтобы оптимизировать стратегию действий так, чтобы максимизировать суммарную награду в долгосрочной перспективе.

Классификация по характеру связей в нейронных сетях

1. Нейронные сети прямого распространения:

Все связи направлены строго от входных нейронов к выходным.

Примеры: перцептрон Розенблатта, многослойный перцептрон, сети Ворда;

2. Рекуррентные нейронные сети:

Сигнал с выходных нейронов или нейронов скрытого слоя частично передаётся обратно на входы нейронов входного слоя (обратная связь).

Рекуррентная сеть Хопфилда решает задачи компрессии данных и построения ассоциативной памяти.

Частный случай: двунаправленные сети;

3. Радиально-базисные функции (RBF):

Используются нейронные сети с единственным скрытым слоем.

Нелинейная активационная функция только у нейронов скрытого слоя.

Синаптические веса связей входного и скрытого слоёв равны единице;

4. Самоорганизующие карты (Self-Organizing Maps, SOM), представляют собой модель нейронных сетей, которая используется для визуализации и кластеризации данных. Это метод проецирования данных из многомерного пространства в пространство с более низкой размерностью, обычно двумерное. SOM также применяются в моделировании, прогнозировании и других задачах. Они являются разновидностью сетей Кохонена.

В сети Кохонена сигнал поступает на все нейроны одновременно, а выходной сигнал формируется по принципу "победитель забирает всё". В процессе обучения веса синапсов настраиваются таким образом, чтобы узлы сети описывали кластерную структуру данных. Удобно представлять SOM как двумерную сетку узлов в многомерном пространстве.

Начальное вложение сетки в пространство данных выбирается произвольно, а затем узлы сети перемещаются на каждом этапе обучения в направлении данных. Алгоритм обучения состоит из двух этапов: грубой настройки, где узлы двигаются коллективно для грубого отображения структуры данных, и тонкой настройки, где настраиваются индивидуальные положения узлов.

Этот процесс повторяется определённое число эпох, причем количество шагов может изменяться в зависимости от задачи.

1.1.3 Кластеризация и новые классы

Кластеризация подразумевает разделение входных сигналов на классы, неизвестные заранее по числу и признакам. После обучения сеть может определить, к какому классу относится входной сигнал, либо указать на его новизну. Такие сети могут обнаруживать новые, ранее неизвестные классы сигналов. Соответствие между выделенными сетью классами и классами в предметной области устанавливается человеком.

1.1.4 Архитектура нейронных сетей

Нейронные сети Кохонена имеют ограниченный размер, разделяясь на гиперслои и ядра. Идеальное количество параллельных слоев ограничено до 112, где каждый слой содержит от 500 до 2000 микроколонок. Эти микроколонки обеспечивают кодирование и вывод результатов. Регулирование числа нейронов и слоев осуществляется с помощью суперкомпьютеров, делая нейронные сети пластичными и адаптивными.

1.1.5 Прогнозирование

Способность нейронной сети к прогнозированию происходит из ее способности к обобщению и выявлению скрытых зависимостей между входными и выходными данными. После обучения сеть может предсказать будущее значение последовательности, основываясь на предыдущих значениях и/или текущих факторах. Прогнозирование возможно лишь в случае, если предыдущие изменения в некоторой степени влияют на будущие. Например, прогнозирование цен акций на основе предыдущих недельных котировок может быть успешным (но не обязательно), в то время как прогнозирование результатов лотереи на основе 50-летних данных практически бесполезно.

1.1.6 Аппроксимация

Способность нейронной сети к прогнозированию напрямую вытекает из ее способности обобщать и выявлять скрытые зависимости между входными и выходными данными. После обучения сеть может предсказывать будущее значение последовательности, опираясь на предыдущие значения и/или текущие факторы. Прогнозирование возможно только в том случае, если предыдущие изменения в некоторой степени определяют будущие. Например, прогнозирование цен акций на основе предыдущих недельных котировок может быть успешным (но не обязательно), в то время как прогнозирование результатов лотереи на основе 50-летних данных практически бесполезно.

Нейронные сети способны аппроксимировать непрерывные функции. Обобщённая аппроксимационная теорема показывает, что с помощью линейных операций и каскадного соединения можно получить устройство, способное вычислить любую непрерывную функцию с некоторой наперёд заданной точностью. Это означает, что нейроны могут иметь различные нелинейные характеристики, от сигмоидальной до волновых пакетов или вейвлетов, синусов или многочленов. Выбор нелинейной функции может влиять на сложность сети, но при правильном выборе структуры нейронная сеть остаётся

универсальным аппроксиматором и может достаточно точно аппроксимировать функционирование любого непрерывного автомата.

1.1.7 Сжатие данных и ассоциативная память

Нейросети обладают способностью выявлять взаимосвязи между различными параметрами, что позволяет более компактно представлять данные большой размерности в случае их тесной взаимосвязи. Процесс обратного восстановления исходного набора данных из части информации называется (авто)ассоциативной памятью. Ассоциативная память также способна восстанавливать исходный сигнал или образ из зашумленных или повреждённых входных данных. Решение задачи гетероассоциативной памяти позволяет реализовать память, адресуемую по содержимому.

1.2 Понятие нечёткой логики

Нечёткая логика представляет собой раздел математики, который расширяет традиционную логику и теорию множеств, используя концепцию нечетких множеств. Она была впервые предложена Лотфи Заде в 1965 году. В отличие от классической логики, где элементы либо принадлежат множеству (имеют значение 1), либо не принадлежат (имеют значение 0), нечеткие множества могут иметь значения на интервале $[0, 1]$, отражая степень принадлежности элемента к множеству. На основе этой концепции разрабатываются различные логические операции и определяются лингвистические переменные, значениями которых являются нечеткие множества.

Область применения нечеткой логики включает исследование рассуждений в условиях нечеткости, размытости, аналогичных рассуждениям в обычной логике, а также их применение в вычислительных системах.

1.2.1 Символическая нечёткая логика

Нечёткая логика, также известная как символическая нечёткая логика, базируется на концепции t -нормы. После выбора определённой t -нормы появляется возможность определить основные операции над пропозициональ-

ными переменными: конъюнкцию, дизъюнкцию, импликацию, отрицание и другие.

Теорема о дистрибутивности, свойственная классической логике, выполняется лишь при использовании t-нормы Гёделя. Импликация обычно определяется операцией, называемой residuum, которая, в свою очередь, зависит от выбранной t-нормы.

Эти базовые операции приводят к формальному определению базовой нечёткой логики, имеющей сходства с классической булевой логикой (исчислением высказываний).

Существуют три основные базовые нечёткие логики: логика Лукасевича, логика Гёделя и вероятностная логика. Интересно, что объединение любых двух из этих логик приводит к классической булевой логике.

1.2.2 Синтез функций непрерывной логики заданных таблично

Функция нечёткой логики Заде всегда принимает значение одного из своих аргументов либо его отрицания. Таким образом, функцию нечёткой логики можно задать таблицей выбора, в которой перечислены все варианты упорядочения аргументов и отрицаний, и для каждого варианта указано значение функции.

Однако не любая произвольная таблица выбора задаёт функцию нечёткой логики. В одной работе был сформулирован критерий, позволяющий определить, является ли функция, заданная таблицей выбора, функцией нечёткой логики, и предложен простой алгоритм синтеза, основанный на концепциях конституента минимума и максимума. Функция нечёткой логики представляет собой дизъюнкцию конституент минимума, где конституента максимума — это конъюнкция переменных текущей области, больших либо равных значению функции в этой области (справа от значения функции в неравенстве, включая значение функции).

1.2.3 Теория приближённых вычислений

В общем, основное понятие нечёткой логики можно описать как использование нечётких множеств, которые определяются через обобщённую характеристическую функцию. Это позволяет работать с нечёткими отношениями, объединениями, пересечениями и дополнениями множеств. Важным элементом является лингвистическая переменная.

Для применения нечёткой логики в некоторых сферах достаточно этого минимального набора определений. Однако для большинства случаев требуется также определить правила вывода и оператор импликации.

1.2.4 Нечёткая логика и нейронные сети

Основная идея нечёткой логики в широком смысле заключается в использовании нечётких множеств, которые определяются через обобщённую характеристическую функцию. Путем введения операций объединения, пересечения и дополнения множеств (через характеристическую функцию) а также концепции нечётких отношений и лингвистических переменных, создается база для приложения нечёткой логики в различных областях.

Важно отметить, что для некоторых применений этого подхода достаточно указанных минимальных определений. Однако для большинства случаев необходимо определить правила вывода и оператор импликации.

Кроме того, поскольку нечёткие множества могут быть представлены функциями принадлежности, а t -нормы и k -нормы являются обычными математическими операциями, возможно представление нечётких логических рассуждений в виде нейронной сети. Здесь функции принадлежности интерпретируются как функции активации нейронов, передача сигналов как связи, а логические t -нормы и k -нормы как специальные виды нейронов, выполняющие соответствующие математические операции. Существует разнообразие подобных нейро-нечётких сетей, включая ANFIS (Adaptive Neuro Fuzzy Inference System) - адаптивную нейро-нечеткую систему вывода. О ней чуть расскажу чуть позже.

1.2.5 Байесовская вероятность

Связь между нечёткой логикой и байесовской вероятностью выражается через байесовскую логико-вероятностную модель нечёткого вывода. Эта модель трансформирует нечёткие продукции в вероятностные функции, определяющие апостериорное распределение на множестве гипотез, соответствующих значениям выходной лингвистической переменной. После этого происходит дефазификация для получения чёткого значения выходной переменной.

Неонечеткие продукции состоят из правил типа "ЕСЛИ ..., ТО ... где посылка и заключение содержат лингвистические переменные, а лингвистические переменные имеют терм-множества с функциями принадлежности.

Байесовская логико-вероятностная модель нечёткого вывода рассматривает значения выходной переменной как гипотезы и определяет их априорные вероятности. Новая информация поступает в виде значений входных переменных, которые служат свидетельствами в пользу или против гипотез.

Условные вероятности заключений при данных посылках определяются через функции принадлежности нечётких множеств.

Используя теорему Байеса, априорные вероятности обновляются с учётом новой информации, что позволяет получить апостериорное распределение вероятностей на множестве гипотез.

Для дефазификации можно использовать различные методы, такие как метод максимума апостериорной вероятности (MAP) или метод среднего значения апостериорной вероятности (MEP).

1.3 Понятие адаптивной системы нейро-нечеткого вывода

Адаптивная система нейро-нечеткого вывода (ANFIS) представляет собой разновидность искусственной нейронной сети, основанной на системе нечеткого вывода Такаги-Сугено. Разработанная в начале 1990-х годов, эта методика объединяет в себе нейронные сети и принципы нечеткой логики,

что дает ей потенциал для использования преимуществ обоих в одной структуре.

В ANFIS система вывода соответствует набору нечетких правил "ЕСЛИ–ТО способных к обучению для аппроксимации нелинейных функций. Следовательно, ANFIS считается универсальным оценщиком. Для более эффективного и оптимального использования ANFIS можно воспользоваться наилучшими параметрами, полученными с использованием генетического алгоритма. Эта технология находит применение в интеллектуальных системах управления энергопотреблением.

ANFIS представляет собой разновидность искусственной нейронной сети, базирующейся на нечеткой системе вывода Такаги-Сугено. Разработанная в начале 1990-х годов, эта методика объединяет в себе нейронные сети и принципы нечеткой логики, что дает ей потенциал для использования преимуществ обоих в одной структуре.

Архитектура ANFIS В структуре сети можно выделить две части: исходную и последующую. Архитектура состоит из пяти уровней:

1. Уровень фазификации: Принимает входные значения и определяет функции принадлежности, принадлежащие им. Степени принадлежности вычисляются на основе исходных параметров;
2. Уровень правил: Генерирует сильные стороны для правил на основе вторичных параметров;
3. Уровень нормализации: Нормализует вычисленную силу срабатывания путем деления каждого значения на общую силу срабатывания;
4. Уровень следствия: Принимает нормализованные значения и параметры следствия. Возвращает дефазифицированные значения;
5. Выходной уровень: Получает дефазифицированные значения и возвращает конечный результат.

ANFIS позволяет аппроксимировать нелинейные функции, делая его универсальным оценщиком с использованием генетического алгоритма для оптимизации его параметров. Эта технология находит применение в интеллектуальных системах управления энергопотреблением.

1.3.1 Слой фазификации

Первый уровень в сети ANFIS представляет собой ключевое отличие от обычной нейронной сети. Обычно нейронные сети работают с этапом предварительной обработки данных, на котором признаки преобразуются в нормализованные значения от 0 до 1. Однако в ANFIS нет необходимости в использовании сигмоидальной функции. Вместо этого происходит преобразование числовых значений в нечеткие.

Давайте рассмотрим пример: предположим, у нас есть сеть, которая получает на вход расстояние между двумя точками в 2D-пространстве. Это расстояние измеряется в пикселях и может принимать значения от 0 до 500 пикселей. Преобразование числовых значений в нечеткие выполняется с использованием функций принадлежности, которые состоят из семантических описаний, таких как "близко" "средне" и "далше". Каждое из этих лингвистических значений соответствует отдельному нейрону. Например, нейрон "близко" срабатывает с некоторым значением от 0 до 1, если расстояние попадает в категорию "близко". Точно так же нейрон "средне" срабатывает, если расстояние соответствует этой категории. Таким образом, входное значение "расстояние в пикселях" разбивается на три разных нейрона для каждой категории: "близко" "средне" и "далше".

2 Техническое задание

2.1 Основание для разработки

Основанием для разработки является задание на выпускную квалификационную работу бакалавра «Интеллектуальная система распознавания объектов по цветовым характеристикам на основе нечетких нейронных сетей».

2.2 Цель и назначение разработки

Основной задачей выпускной квалификационной работы является разработка интеллектуальной системы распознавания объектов на основе их цветовых характеристик с использованием нечетких нейронных сетей. Система должна обеспечивать высокую точность распознавания объектов различных классов на основе их цветовых параметров.

Задачами данной разработки являются:

- сбор набора данных, содержащего изображения объектов различных классов;
- предварительная обработка изображений для выделения цветовых характеристик объектов;
- проектирование архитектуры нечеткой нейронной сети;
- обучение нейронной сети на подготовленных данных;
- создание удобного поиска по сайту;
- оптимизация параметров сети для достижения максимальной точности распознавания;
- создание программного интерфейса для взаимодействия с разработанной нейронной сетью.

2.3 Требования пользователя к интерфейсу приложения

Приложение должно включать в себя:

- графический интерфейс пользователя;
- возможность загрузки изображений объектов для их распознавания;

- отображение результатов распознавания с указанием класса объекта и уверенности в распознавании;
- возможность обучения системы на пользовательских данных для улучшения качества распознавания;
- интерфейс для добавления новых классов объектов и их обучения;
- возможность экспорта результатов распознавания в формате, подходящем для дальнейшей обработки или анализа.

Композиция шаблона приложения представлена на рисунке 2.1.



Рисунок 2.1 – Композиция шаблона приложения

2.4 Моделирование вариантов использования

Для моделирования вариантов использования разрабатываемого приложения была использована диаграмма вариантов использования, представленная на рисунке 2.2.

На диаграмме представлены основные варианты использования приложения, включая загрузку изображений для распознавания, обучение системы, добавление новых классов объектов и экспорт результатов распознавания.



Рисунок 2.2 – Диаграмма вариантов использования

ния. Данная диаграмма помогает понять основные функциональные возможности приложения и взаимодействие с пользователями.

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

1. Распознание объекта на изображении по его цветовым характеристикам;
2. Сохранение результатов распознания для дальнейшего использования;
3. Обучение и переобучение нейро-нечёткой сети;
4. Сохранение результатов обучения для дальнейшего распознания.

2.5 Требования к оформлению документации

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

3 Технический проект

3.1 Общая характеристика организации решения задачи

Этот проект направлен на разработку системы, которая использует нечеткие нейронные сети для распознавания объектов на основе цветовых характеристик. Система будет способна анализировать изображения и выделять объекты, соответствующие заданным цветовым параметрам.

Основная цель - создание эффективной и точной системы распознавания объектов. Задачи включают:

- Разработка алгоритма нечеткой нейронной сети;
- Создание базы данных для обучения и тестирования системы.

3.2 Обоснование выбора технологии проектирования

Нечеткие нейронные сети сочетают принципы нечеткой логики и нейронных сетей, что позволяет системе обрабатывать нечеткие и неточные данные, характерные для реальных изображений.

3.2.1 Python и его библиотеки

Python является предпочтительным языком программирования благодаря своей читаемости, простоте и обширной экосистеме библиотек, подходящих для работы с данными и машинным обучением:

- NumPy: Используется для эффективной работы с массивами и матрицами, что критично для обработки изображений и численных вычислений;
- Pandas: Предоставляет удобные структуры данных для анализа и манипуляции данными;
- Matplotlib/Seaborn: Библиотеки для визуализации данных, которые помогают в анализе результатов и представлении данных;
- OpenCV: Открытая библиотека для работы с компьютерным зрением, которая может использоваться для предварительной обработки изображений;

- TensorFlow/Keras: Популярные фреймворки для глубокого обучения, которые предоставляют инструменты для создания, обучения и тестирования нейронных сетей;
- Scikit-learn: Библиотека для машинного обучения, предоставляющая различные алгоритмы классификации, регрессии и кластеризации.

3.2.2 Архитектура нечёткой нейронной сети

Архитектура нечеткой нейронной сети включает в себя 6 слоев с различными функциями:

Схема архитектуры нейронной сети представлена на рисунке 3.1.

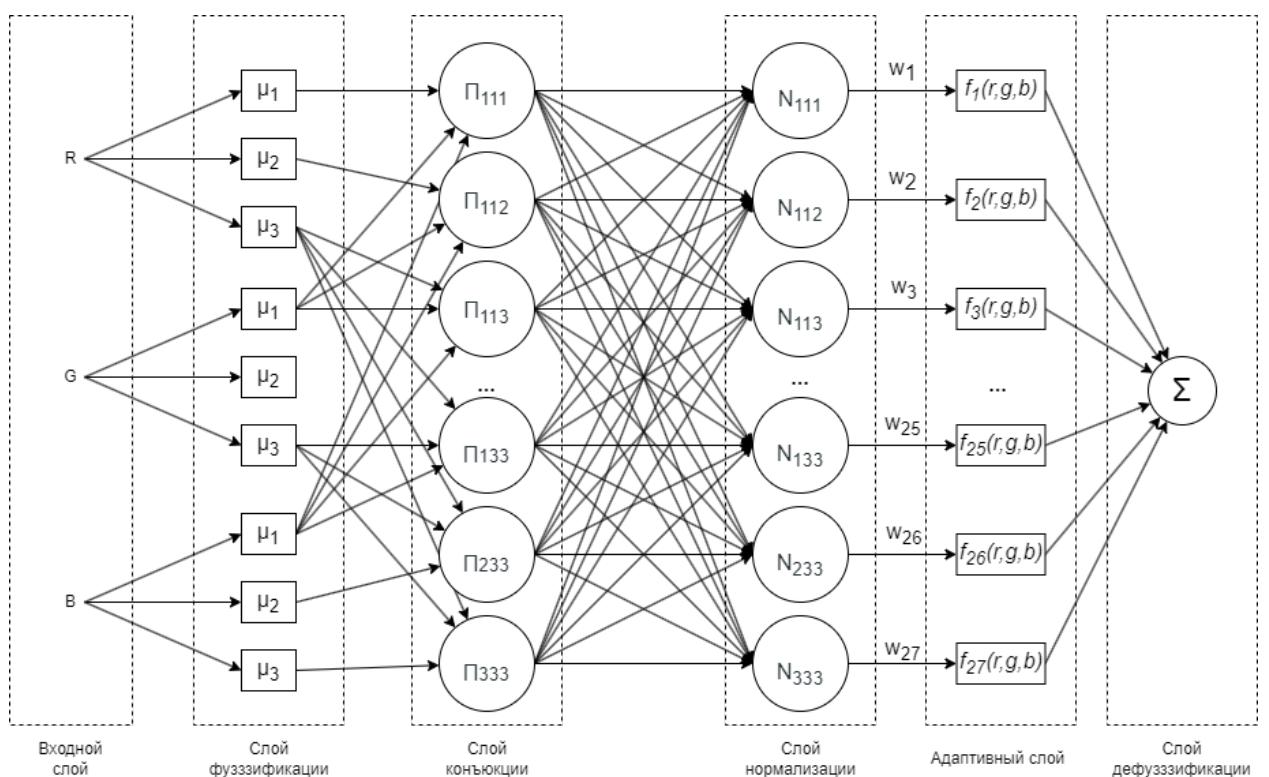


Рисунок 3.1 – Схема архитектуры нейронной сети

Входной слой: В этом слое содержатся входные переменные, представленные 3 цветовыми каналами изображения. Эти переменные предоставляют исходную информацию для последующей обработки.

Слой фурзификации: Он отвечает за преобразование значений яркости изображения в степени принадлежности. Здесь представлены функции принадлежности для каждой переменной, их количество равно 3 (яркие, средние

или темные оттенки цвета), что в сумме создает 27 комбинаций. Функции принадлежности представлены гауссианами.

Вид функции Гаусса представлен на рисунке 3.2.

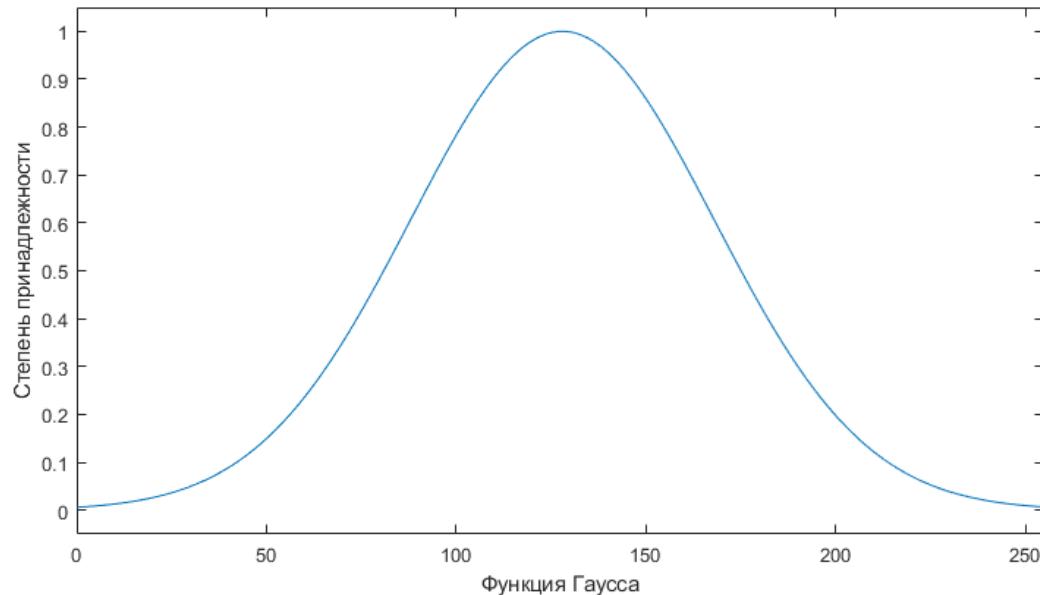


Рисунок 3.2 – Функция Гаусса

Слой конъюнкции: На этом этапе происходит перемножение функций для каждой комбинации функций принадлежности по 3 цветовым каналам.

Слой нормализации: Здесь происходит нормализация полученных произведений.

Адаптивный слой: На этой стадии добавляются веса к нормализованным данным.

Слой дефузрификации: Здесь происходит суммирование всех функций для получения результата - яркости выходного пикселя.

3.2.3 Описание базы данных

База данных будет хранить информацию о различных моделях обучения нейронной сети. У каждой модели будет своя таблица, в которой будут храниться данные об одном наборе обучения нейронной сети и одной таблице с уже обученными значениями. Каждый набор обучения будет содержать

четыре столбца: три столбца для входных значений, соответствующих трем цветовым каналам изображения, и один столбец для выходных значений.

ERD диаграмма структуры базы данных представлена на рисунке 3.3.

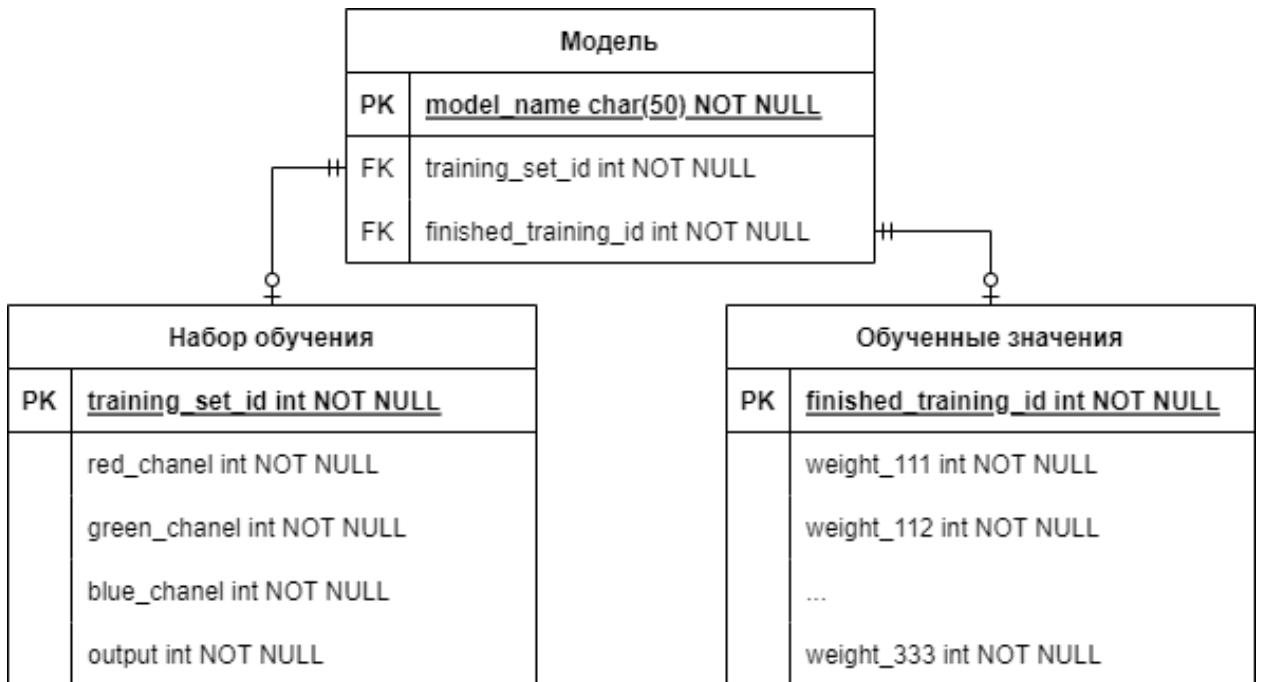


Рисунок 3.3 – Диаграмма структуры базы данных

3.3 Диаграмма компонентов

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

3.3.1 Структура компонентов

На диаграмме компонентов изображены основные блоки системы, такие как:

- Графический интерфейс пользователя: Модуль, отвечающий за взаимодействие с пользователем, представление результатов и получение входных данных;
- Модуль предварительной обработки данных: Отвечает за подготовку данных к анализу, включая фильтрацию шума и нормализацию изображений;

- Модуль нечеткой нейронной сети: Ядро системы, реализующее алгоритмы обучения и распознавания объектов;
- База данных: Хранит обучающий и тестовый наборы данных, а также результаты работы системы;
- Модуль анализа данных: Производит анализ данных, классификацию и предоставляет статистику по результатам.

3.3.2 Взаимодействие компонентов

Компоненты системы взаимодействуют друг с другом следующим образом:

1. Пользователь загружает изображение через графический интерфейс пользователя;
2. Интерфейс передает изображение в модуль предварительной обработки данных;
3. После обработки данные передаются в модуль нечеткой нейронной сети для распознавания объектов;
4. Результаты распознавания сохраняются в базе данных;
5. Модуль анализа данных извлекает результаты из базы данных и представляет их пользователю через графический интерфейс пользователя.

Диаграмма компонентов представлена на рисунке 3.4.

3.4 Содержание информационных блоков. Основные сущности

Исходя из описания, можно выделить пять основных сущностей:

1. Графический интерфейс: Этот компонент отвечает за взаимодействие пользователя с нейронной сетью. Он представляет собой интерфейс, через который пользователь может отправлять данные на обработку, управлять обучающими данными нейронной сети и получать результаты;
2. Предварительная обработка: Эта сущность отвечает за предварительную обработку данных, поступающих от пользователя, чтобы подготовить их для дальнейшей работы нейронной сети. Здесь могут проводиться



Рисунок 3.4 – Диаграмма компонентов системы

различные операции, такие как нормализация данных, фильтрация шума и преобразование формата;

3. Нейронная сеть: В данном участке происходит распознавание объектов на обработанном изображении с использованием выбранной модели или проведение обучения новой модели на выбранном наборе обучающих данных;

4. База данных: В этой части системы хранится информация о наборах обучающих данных и обученных моделях нейронной сети. База данных предоставляет доступ к данным для обучения и позволяет сохранять результаты работы нейронной сети для последующего использования;

5. Анализ данных: Этот компонент отвечает за анализ данных, полученных из нейронной сети. Здесь могут проводиться различные вычисления и визуализация результатов работы нейронной сети, а также их сохранение и представление пользователю.

3.4.1 Структура сущности графический интерфейс

Графический интерфейс хранит в себе методы:

1. Запрос файла изображения у пользователя;
2. Выбор нужного набора обучения, для обучения нейронной сети;
3. Выбор нужной модели для распознания объектов;
4. Запуск обучения, который вызовет метод обучения из сущности нейронной сети;
5. Запуск распознания, который вызовет метод обработки из сущности предварительной обработки;
6. Запрос методов анализа данных, полученных из нейронной сети из сущности анализа данных.

3.4.2 Структура сущности предварительная обработка

Предварительная обработка осуществляет преобразование входящего изображения в формат, необходимый для работы нейронной сети.

3.4.3 Структура сущности нейронная сеть

Нейронная сеть строится в соответствии со структурой, описанной выше, и включает в себя два основных метода: обучение и распознавание объектов.

3.4.4 Структура сущности база данных

База данных предоставляет информацию для обучения и распознавания объектов нейронной сети, а также для сохранения новых обученных моделей.

3.4.5 Структура сущности анализ данных

Анализ данных включает в себя различные методы преобразования полученных от нейронной сети значений в полезные данные, такие как центры кластеров, количественное содержание цветов на изображении и другие.

4 Рабочий проект

4.1 Классы, используемые при разработке сайта

Список классов и методов, которые были использованы при создании приложения представлены на таблице 4.1.

Таблица 4.1 – Описание классов, используемых в приложении

Название класса	Модуль, к которому относится класс	Описание класса	Методы
1	2	3	4
ANFIS	Нейронная сеть	ANFIS – класс, в котором осуществляется идентификация объектов с использованием нечеткой нейросети.	нормализацияКаналов(ПутьФайла) Происходит загрузка изображения из файла, последующее его разделение на цветовые каналы, а затем преобразование этих каналов, представленных в виде матриц, в массивы. После этого происходит нормализация значений яркости. АнфисаРаспознать(Путь, ИмяМодели) В рамках данной функции осуществляется прием данных, обработанных методом нормализацияКаналов.

Продолжение таблицы 4.1

1	2	3	4
			<p>Эти данные, совмещенные с оптимизированной моделью искусственной нейронной сети, подвергаются дальнейшей обработке в модуле anfisReady, который разработан на основе программного обеспечения MATLAB. Завершающим этапом является генерация черно-белой маски, которая визуализирует распознанные объекты и имеет разрешение 720x720 пикселей</p> <p>АнфисаТренировать(Набор, Размер, Имя-Модели)</p> <p>Инициируя процесс обучения, функция использует предоставленный набор данных и его объем для конфигурации модуля anfis, реализованного в среде MATLAB. На основании этих данных функция строит и оптимизирует модель нечеткой нейронной сети. После успешного обучения, сформированная модель сохраняется в базу данных с наименованием, указанным в параметрах функции.</p>

Продолжение таблицы 4.1

1	2	3	4
			<pre>function out = AnfisReady(data, name) Данная функция, реализованная на языке программирования MATLAB и использующая среду выполнения MATLAB Runtime, осуществляет передачу данных в нейронную сеть на пиксельном уровне. В соответствии с выбранной моделью нейросети, функция вычисляет степень соответствия каждого пикселя заданной модели и возвращает количественную оценку этого соответствия.</pre> <pre>function out = Anfis(data, name) Программный модуль, реализованный на языке программирования MATLAB и эксплуатирующий среду выполнения MATLAB Runtime, предназначен для приема набора данных, представляющих собой тренировочную выборку. Данный модуль осуществляет процесс обучения нечеткой нейронной сети, используя предоставленные данные.</pre>

Продолжение таблицы 4.1

1	2	3	4
			В результате выполнения, модуль выдает модель нечеткой нейронной сети, прошедшую процедуру обучения.
DB	База данных	DB – Класс для работы с базой данных	<p>сохранить(ИмяМодели, Набор)</p> <p>Данная функция реализует процедуру архивации тренировочного датасета и соответствующей обученной нейросетевой модели в структурированное хранилище данных. Это обеспечивает сохранность исходных обучающих данных и параметров модели для последующего использования и анализа.</p> <p>загрузитьСписокНаборов()</p> <p>Функциональный модуль предназначен для извлечения данных из базы данных, содержащей информацию о тренировочных наборах и соответствующих обученных нейронных моделях.</p>

Продолжение таблицы 4.1

1	2	3	4
			<p>Он выполняет операцию чтения и последующего формирования перечня доступных тренировочных наборов данных и ассоциированных с ними моделей.</p> <p>загрузитьНабор(Название) Эта функция осуществляет операцию извлечения из базы данных специфического обучающего набора, состоящего из трёх цветовых каналов и целевого значения.</p>
main	Графический интерфейс	main – Класс для взаимодействия с пользователем	<p>загрузитьИзображение() Данная функция выполняет загрузку изображения из файла, а затем проводит его масштабирование до определённых размеров 720x720 пикселей, что обеспечивает его корректное отображение в заданном графическом интерфейсе пользователя.</p>

Продолжение таблицы 4.1

1	2	3	4
			<p>распознать() Функция инициирует процесс передачи ранее загруженного изображения в архитектуру нейронной сети, после чего активирует интерфейс для выбора предварительно обученной модели из доступных вариантов, что позволяет пользователю взаимодействовать с системой машинного обучения.</p> <p>выбрать(ОкноВыбора) Данная функция осуществляет интеграцию предварительно обученной модели в структуру нейронной сети, после чего производит деактивацию интерфейса выбора.</p> <p>сохранитьРезультатРаспознания Функция инициирует активацию диалогового интерфейса, предназначенно-го для сохранения выходных данных нейронной сети в файловую систему, обеспечивая тем самым персистентность результатов вычислений.</p>

Продолжение таблицы 4.1

1	2	3	4
			<p>обучать()</p> <p>Функция активирует пользовательский интерфейс в виде диалогового окна, которое предоставляет возможность выбора набора данных для обучения нейронной сети.</p> <p>подтвердить(ОкноВыбора)</p> <p>Функция инициирует передачу выбранного или вновь созданного датасета для обучения нейронной сети, после чего активируется интерфейс в виде диалогового окна, предоставляющего пользователю опцию присвоения наименования новой модели.</p> <p>отправить(ОкноНазвания)</p> <p>Эта функция осуществляет трансмиссию наименования модели в архитекттуру нейронной сети, инициируя этим процедуру обучения.</p>

Продолжение таблицы 4.1

1	2	3	4
			По завершении процесса, модель систематически регистрируется и архивируется в соответствующей базе данных, что обеспечивает её доступность для дальнейшего использования и интеграции в различные прикладные задачи машинного обучения.

4.2 Модульное тестирование разработанного приложения

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

```
1 import os
2 import unittest
3 from PIL import Image
4 import anfis
5 import numpy as np
6 import matlab
7 from random import shuffle
8 from ANFIS import нормализацияКаналов
9 from ANFIS import АнфисаРаспознать
10
11 class TestНормализацияКаналов(unittest.TestCase):
12     def test_нормализация_существующего_файла(self):
13         результат = нормализацияКаналов('test_image.png')
14         self.assertIsNotNone(результат, "Функция должна возвращать не None
15                               для существующего файла")
16         self.assertTrue((результат >= 0).all() and (результат <= 1).all(), "
17                         Все значения должны быть в диапазоне от 0 до 1")
18     def test_нормализация_несуществующего_файла(self):
19         результат = нормализацияКаналов('несуществующий_файл.jpg')
20         self.assertIsNone(результат, "Функция должна возвращать None для
21                           несуществующего файла")
```

Рисунок 4.1 – Модульный тест метода нормализацияКаналов класса ANFIS

```

1 import os
2 import unittest
3 from PIL import Image
4 import anfis
5 import numpy as np
6 import matlab
7 from random import shuffle
8 from ANFIS import нормализацияКаналов
9 from ANFIS import АнфисаРаспознать

10
11 class TestАнфисаРаспознать(unittest.TestCase):
12     def setUp(self):
13         self.путь = 'test_image.png'
14         self.имяМодели = 'test_model_name'
15     def test_распознавание_изображения(self):
16         результат = АнфисаРаспознать(self.путь, self.имяМодели)
17         self.assertIsInstance(результат, Image.Image, "Функция должна
18             возвращать объект изображения")
19     def test_сохранение_изображения(self):
20         результат = АнфисаРаспознать(self.путь, self.имяМодели)
21         self.assertTrue(os.path.isfile('output_image.png'), "Файл изображения
22             должен быть сохранён")

```

Рисунок 4.2 – Модульный тест метода АнфисаРаспознать класса ANFIS

```

1 import os
2 import unittest
3 from PIL import Image
4 import anfis
5 import numpy as np
6 import matlab
7 from random import shuffle
8 from ANFIS import нормализацияКаналов
9 from ANFIS import АнфисаРаспознать

10
11 class TestНормализацияКаналов(unittest.TestCase):
12     def test_нормализация_существующего_файла(self):
13         результат = нормализацияКаналов('test_image.png')
14         self.assertIsNotNone(результат, "Функция должна возвращать не None
15             для существующего файла")
16         self.assertTrue((результат >= 0).all() and (результат <= 1).all(), "
17             Все значения должны быть в диапазоне от 0 до 1")
18     def test_нормализация_несуществующего_файла(self):
19         результат = нормализацияКаналов('несуществующий_файл.jpg')
20         self.assertIsNone(результат, "Функция должна возвращать None для
21             несуществующего файла")

```

Рисунок 4.3 – Модульный тест метода нормализацияКаналов класса ANFIS

Модульные тесты для класса DB из модели данных представлены на рисунках 4.4-4.6.

```

1 from DB import сохранить
2 from DB import загрузитьСписокНаборов
3 from DB import загрузитьНабор
4
5 class TestСохранить(unittest.TestCase):
6     def setUp(self):
7         self.ИмяМодели = 'test_model'
8         self.Набор = [(1, 2, 3, 4), (5, 6, 7, 8)]
9         self.Подключение = sqlite3.connect(':memory:')
10        self.Курсор = self.Подключение.cursor()
11        self.Курсор.execute(''CREATE TABLE Готовые_данные (Адрес_файла TEXT)
12        ''')
13        self.Курсор.execute(''CREATE TABLE Тренировочный_набор (
14            Красный_канал INTEGER, Синий_канал INTEGER, Зелёный_канал INTEGER,
15            Выходные_данные INTEGER)''')
16        self.Курсор.execute(''CREATE TABLE Модель (Имя_модели TEXT,
17            ИД_Тренировочного_набора INTEGER, ИД_Готовых_данных INTEGER)''')
18    def test_подключение_к_базе(self):
19        self.assertIsNotNone(self.Подключение, "Должно быть установлено
20            подключение к базе данных")
21    def test_добавление_в_готовые_данные(self):
22        сохранить(self.ИмяМодели, self.Набор)
23        self.Курсор.execute("SELECT * FROM Готовые_данные WHERE Адрес_файла =
24            ?", (self.ИмяМодели,))
25        результат = self.Курсор.fetchone()
26        self.assertIsNotNone(результат, "Модель должна быть добавлена в
27            таблицу Готовые_данные")
28    def test_добавление_в_тренировочный_набор(self):
29        сохранить(self.ИмяМодели, self.Набор)
30        for данные in self.Набор:
31            self.Курсор.execute("SELECT * FROM Тренировочный_набор WHERE
32                Красный_канал = ? AND Синий_канал = ? AND Зелёный_канал = ?
33                AND Выходные_данные= ?", данные)
34            результат = self.Курсор.fetchone()
35            self.assertIsNotNone(результат, "Данные должны быть добавлены в
36                таблицу Тренировочный_набор")
37    def test_закрытие_подключения(self):
38        self.Подключение.close()
39        self.assertRaises(sqlite3.ProgrammingError, self.Курсор.execute,
40            "SELECT * FROM Готовые_данные")
41    def tearDown(self):
42        self.Подключение.close()

```

Рисунок 4.4 – Модульный тест метода загрузитьНабор класса DB

```
1 from DB import сохранить
2 from DB import загрузитьСписокНаборов
3 from DB import загрузитьНабор
4
5 class TestЗагрузитьСписокНаборов(unittest.TestCase):
6     def setUp(self):
7         self.Подключение = sqlite3.connect(':memory:')
8         self.Курсор = self.Подключение.cursor()
9         self.Курсор.execute('''CREATE TABLE Готовые_данные (Адрес_файла TEXT)
10           ''')
11        self.Курсор.executemany("INSERT INTO Готовые_данные (Адрес_файла)
12          VALUES (?)", [('file1',), ('file2',), ('file3',)])
13
14    def test_загрузка_списка(self):
15        ожидаемый_список = ['file1', 'file2', 'file3']
16        результат = загрузитьСписокНаборов()
17        self.assertEqual(результат, ожидаемый_список, "Список наборов должен
18                      соответствовать ожидаемому")
19
20    def test_закрытие_подключения(self):
21        загрузитьСписокНаборов()
22        self.assertRaises(sqlite3.ProgrammingError, self.Курсор.execute, "
23          SELECT * FROM Готовые_данные")
24
25    def tearDown(self):
26        self.Подключение.close()
```

Рисунок 4.5 – Модульный тест метода загрузитьСписокНаборов класса DB

```

1 from DB import сохранить
2 from DB import загрузитьСписокНаборов
3 from DB import загрузитьНабор
4
5 class TestЗагрузитьНабор(unittest.TestCase):
6     def setUp(self):
7         Название = 'test_model'
8         self.Подключение = sqlite3.connect(':memory:')
9         self.Курсор = self.Подключение.cursor()
10        self.Курсор.execute('''CREATE TABLE Модель (Имя_модели TEXT,
11                           ИД_Тренировочного_набора INTEGER)''')
12        self.Курсор.execute('''CREATE TABLE Тренировочный_набор (
13                           ИД_тренировочного_набора INTEGER, Красный_канал INTEGER,
14                           Синий_канал INTEGER, Зелёный_канал INTEGER, Выходные_данные
15                           INTEGER)'''')
16        self.Курсор.execute("INSERT INTO Модель (Имя_модели,
17                           ИД_Тренировочного_набора) VALUES (?, ?)", (Название, 1))
18        self.Курсор.execute("INSERT INTO Тренировочный_набор (
19                           ИД_тренировочного_набора, Красный_канал, Синий_канал,
20                           Зелёный_канал, Выходные_данные) VALUES (1, 255, 0, 0, 1)")
21
22    def test_загрузка_набора(self):
23        ожидаемый_выход = [(255, 0, 0, 1)]
24        результат = загрузитьНабор('test_model')
25        self.assertEqual(результат[0], ожидаемый_выход, "Загруженный набор
26        должен соответствовать ожидаемому")
27
28    def test_формат_выходных_данных(self):
29        результат = загрузитьНабор('test_model')
30        self.assertEqual(результат[1], (1, 4), "Формат выходных данных должен
31        быть кортежем с размерностью набора")
32
33    def test_закрытие_подключения(self):
34        загрузитьНабор('test_model')
35        self.assertRaises(sqlite3.ProgrammingError, self.Курсор.execute,
36                         "SELECT * FROM Модель")
37
38    def tearDown(self):
39        self.Подключение.close()

```

Рисунок 4.6 – Модульный тест метода загрузитьНабор класса DB

4.3 Системное тестирование разработанного приложения

На рисунке 4.7 представлен интерфейс программы.

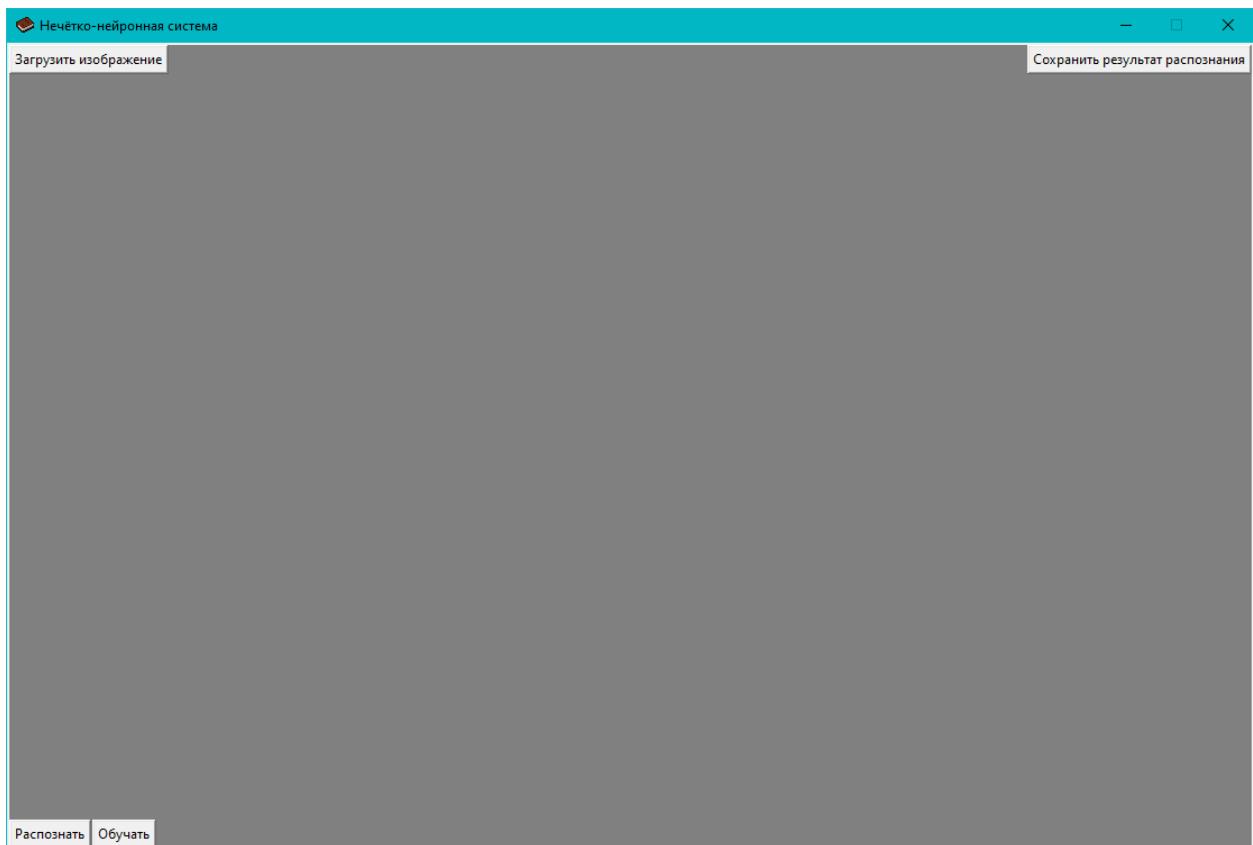


Рисунок 4.7 – Диаграмма вариантов использования

На рисунках 4.8-4.12 представлен полный путь распознания объекта на изображении.

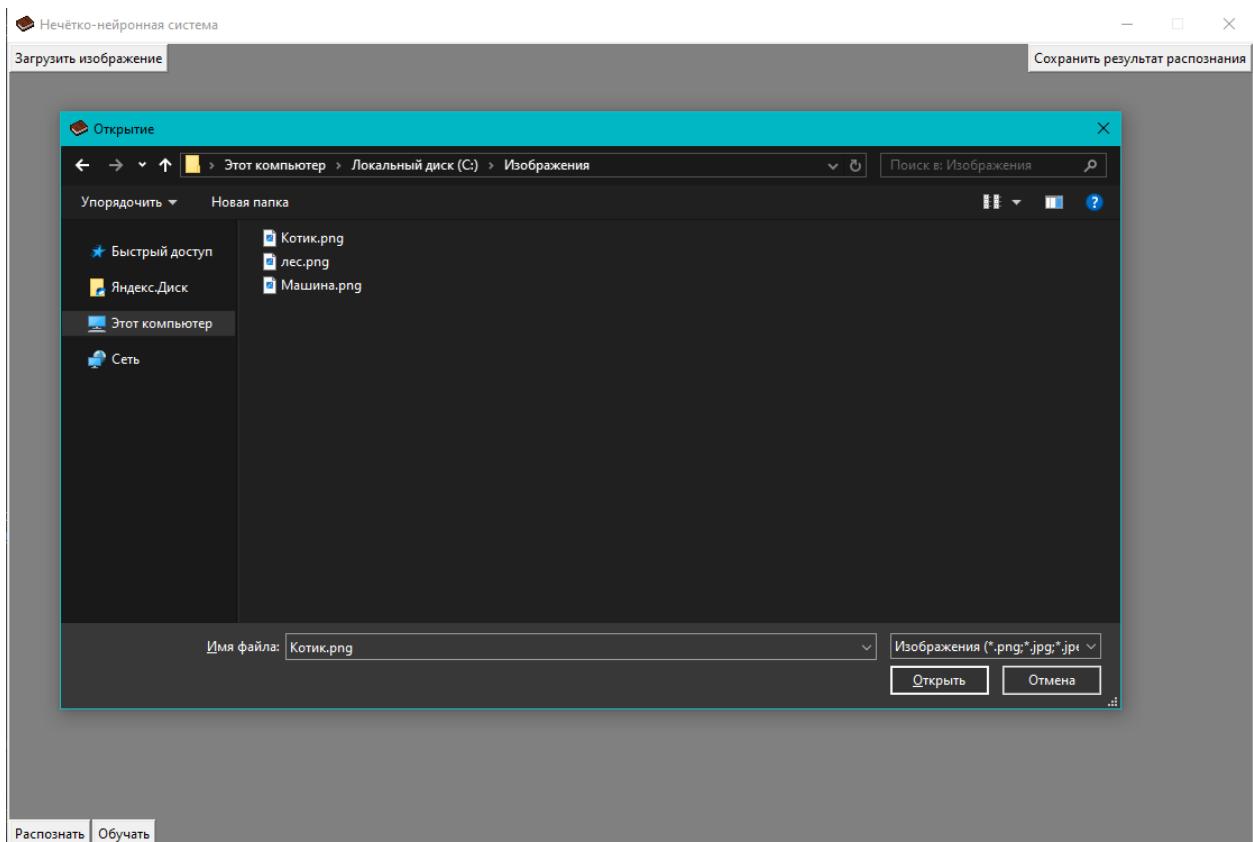


Рисунок 4.8 – Диалоговое окно загрузки файла Машина.png

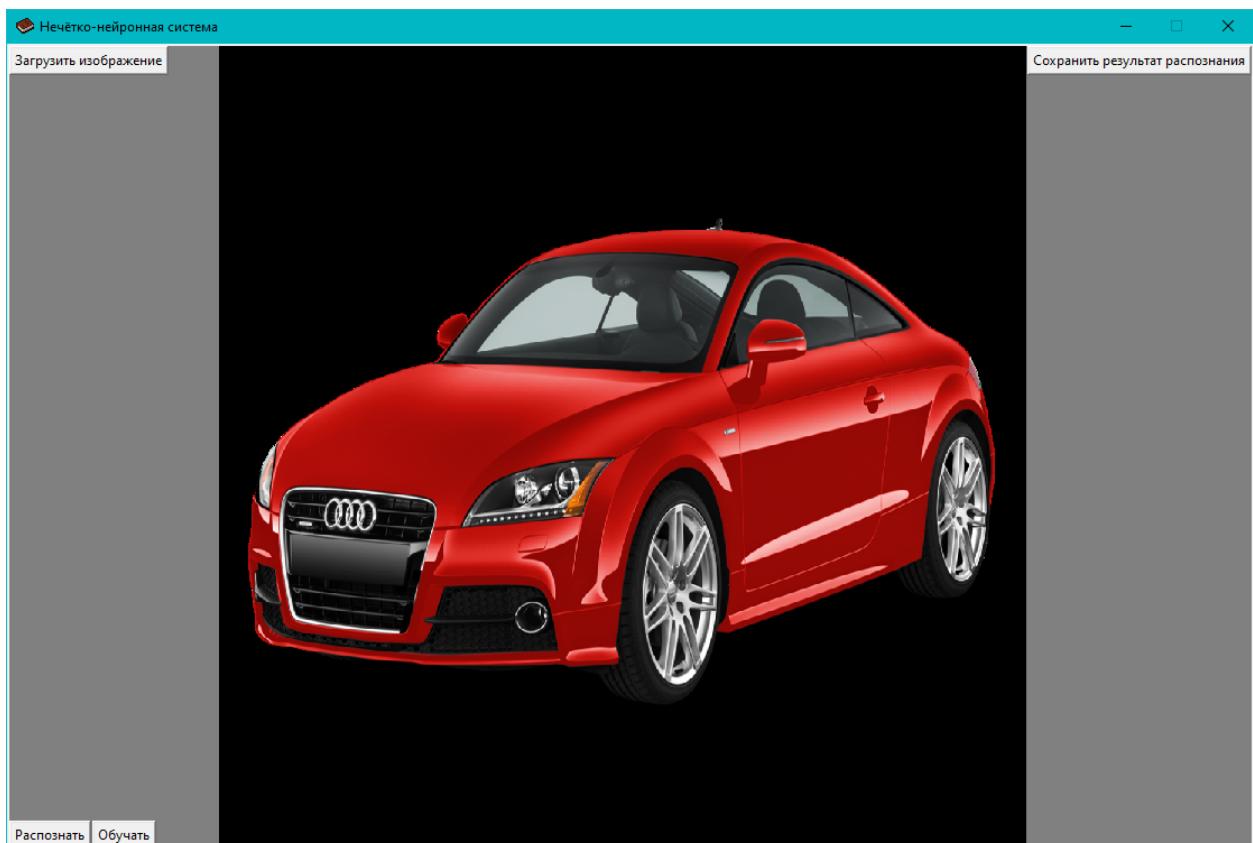


Рисунок 4.9 – Изображение отображено в интерфейсе программы

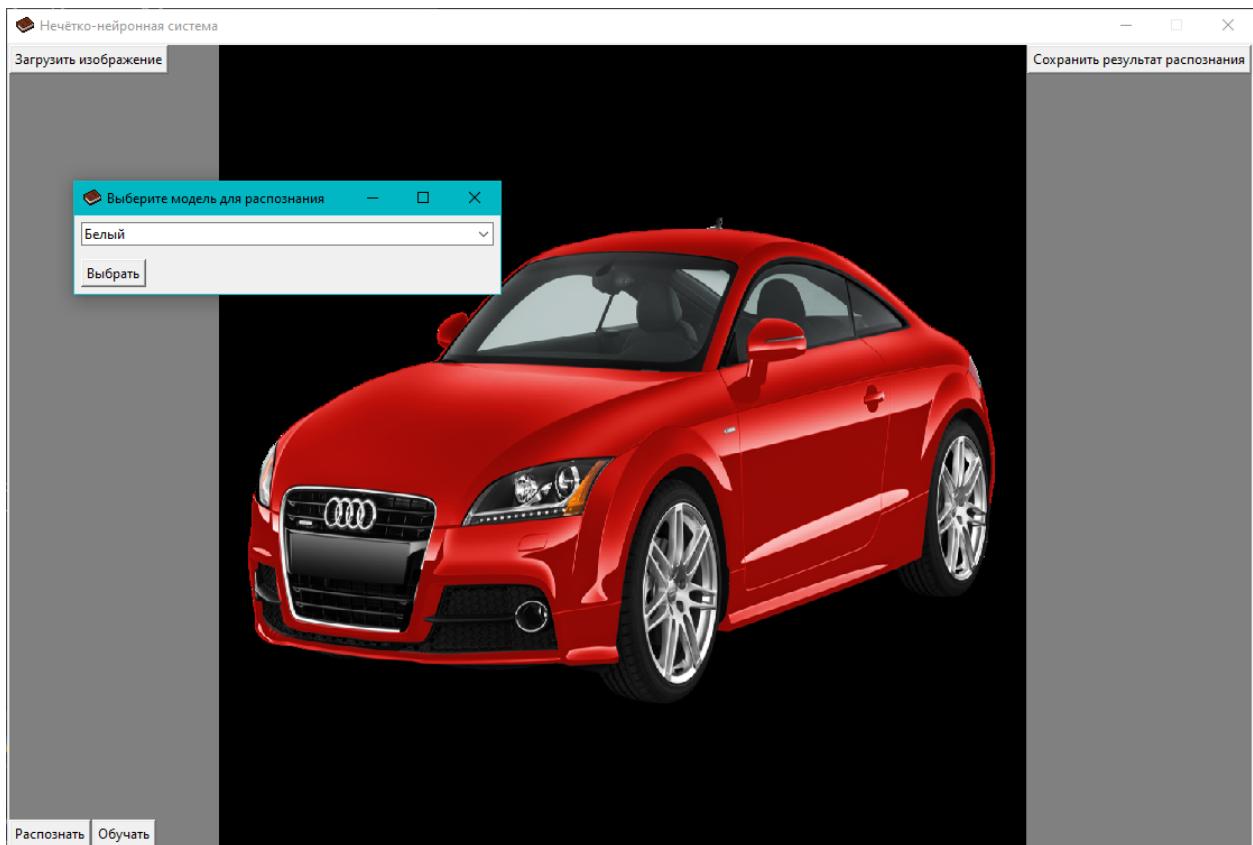


Рисунок 4.10 – Окно выбора модели для распознания

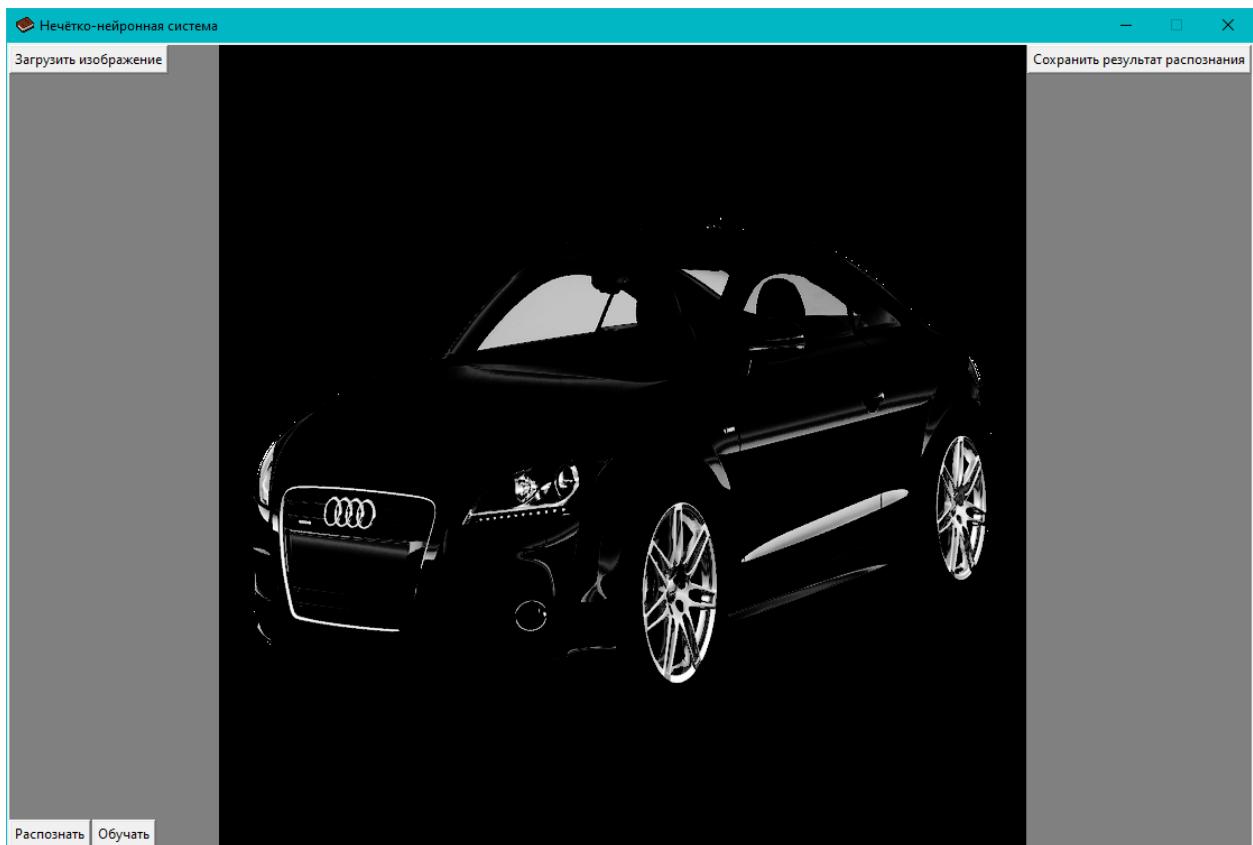


Рисунок 4.11 – Распознанный объект отображен на интерфейсе

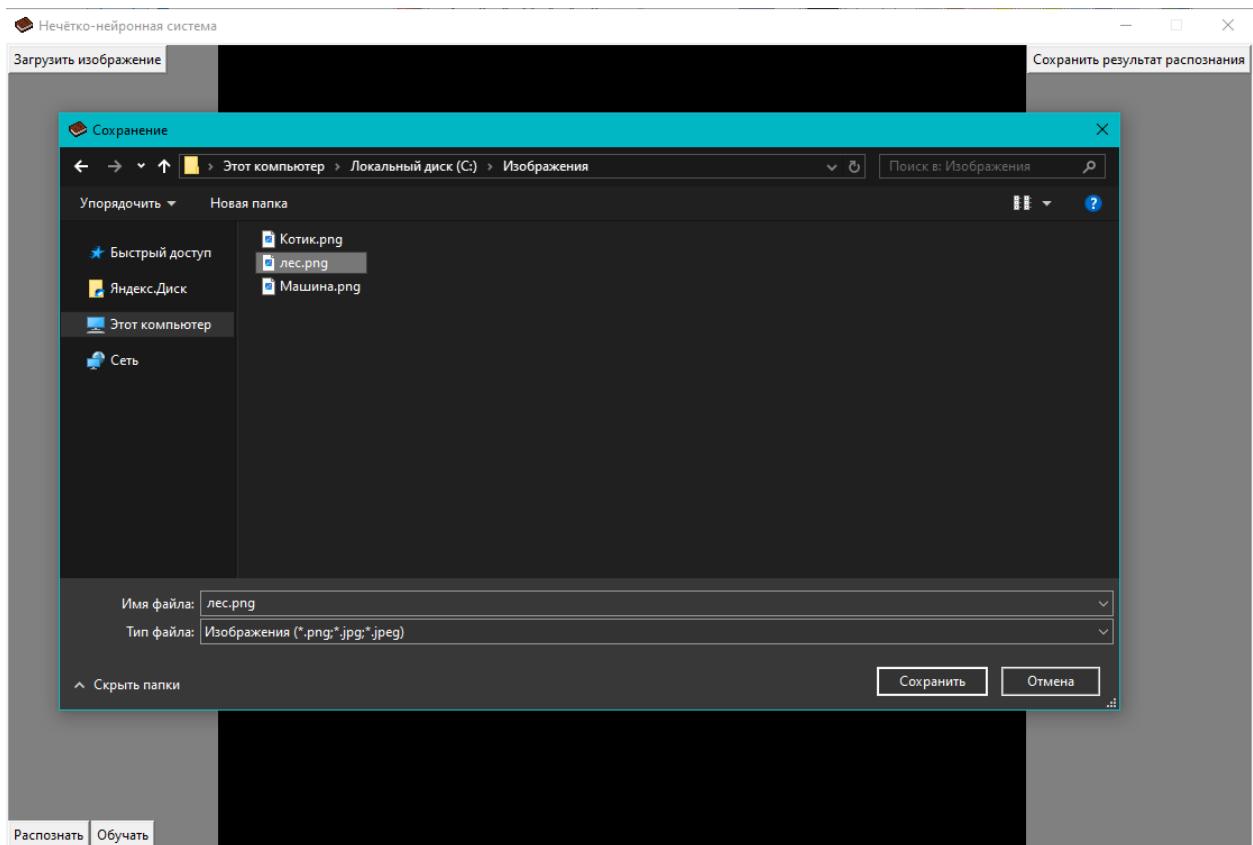


Рисунок 4.12 – Диалоговое окно сохранения результата

На рисунках 4.13-4.15 представлены все варианты обучения нейронной сети.

На рисунке 4.13 была нажата кнопка обучать, после чего открылось окно выбора набора.

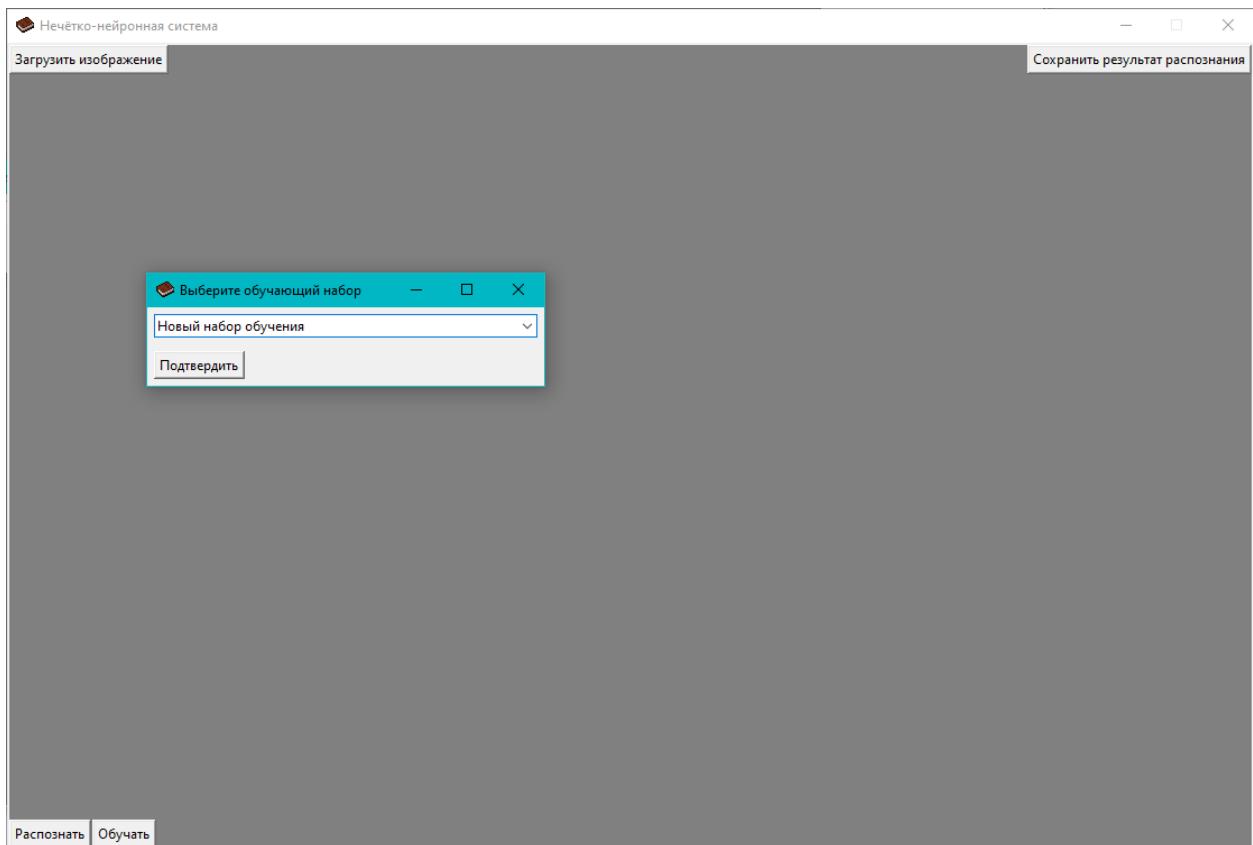


Рисунок 4.13 – Окно выбора обучающего набора

После выбора нового набора появилось окно, предлагающее назвать модель и обучить её.

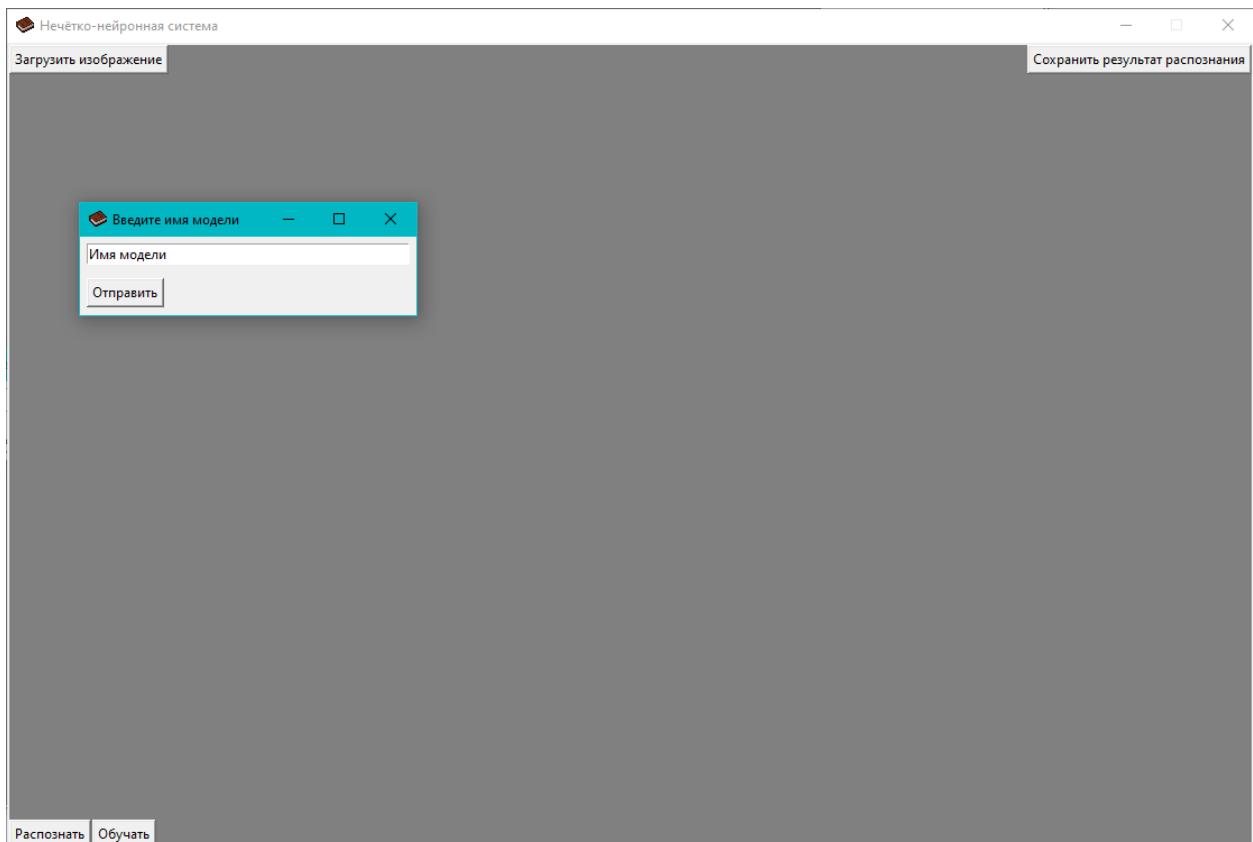


Рисунок 4.14 – Окно выбора названия новой модели

Вместо создания нового набора, после нажатия кнопки обучать, можно выбрать заготовленный набор и обучить нейронную сеть по нему.

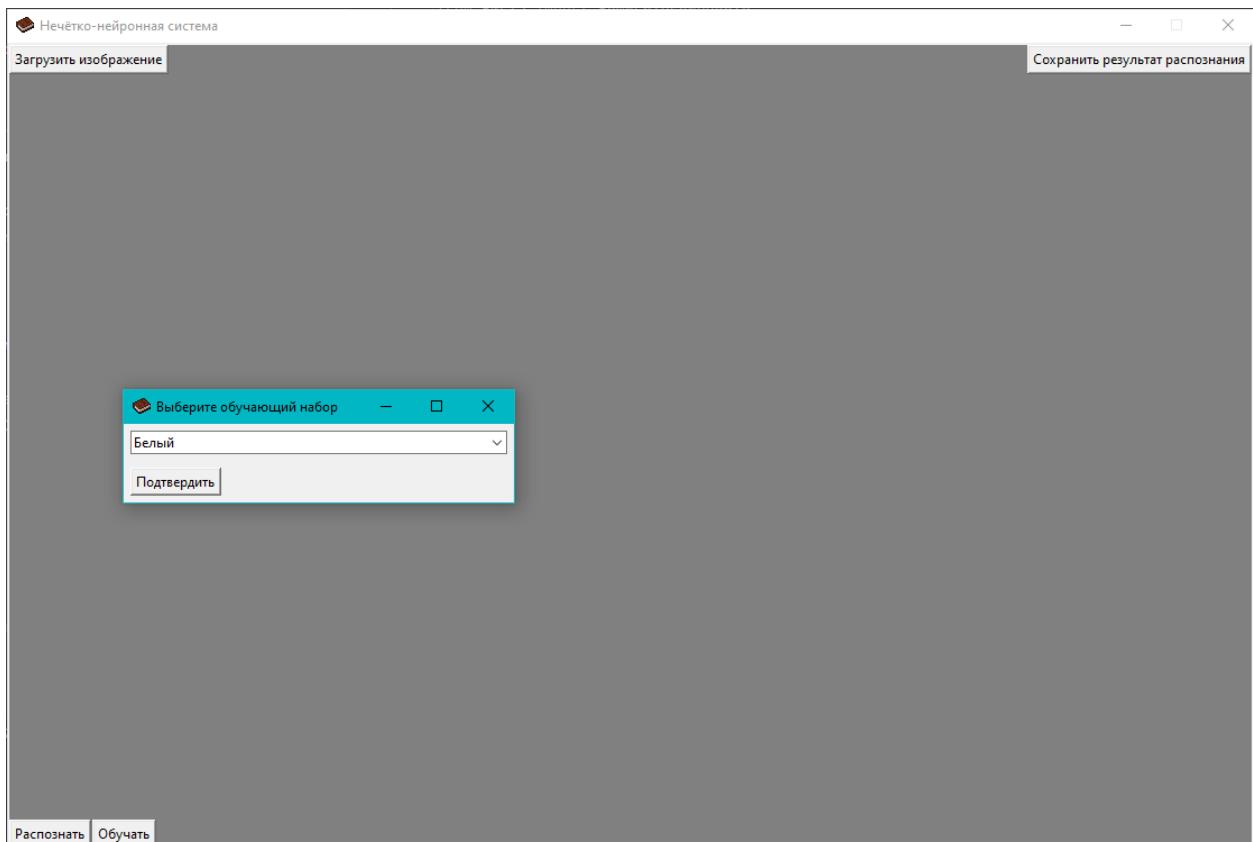


Рисунок 4.15 – Выбор обучения по набору ”Белый”

На рисунке 4.16 было загружено изображение кошки.

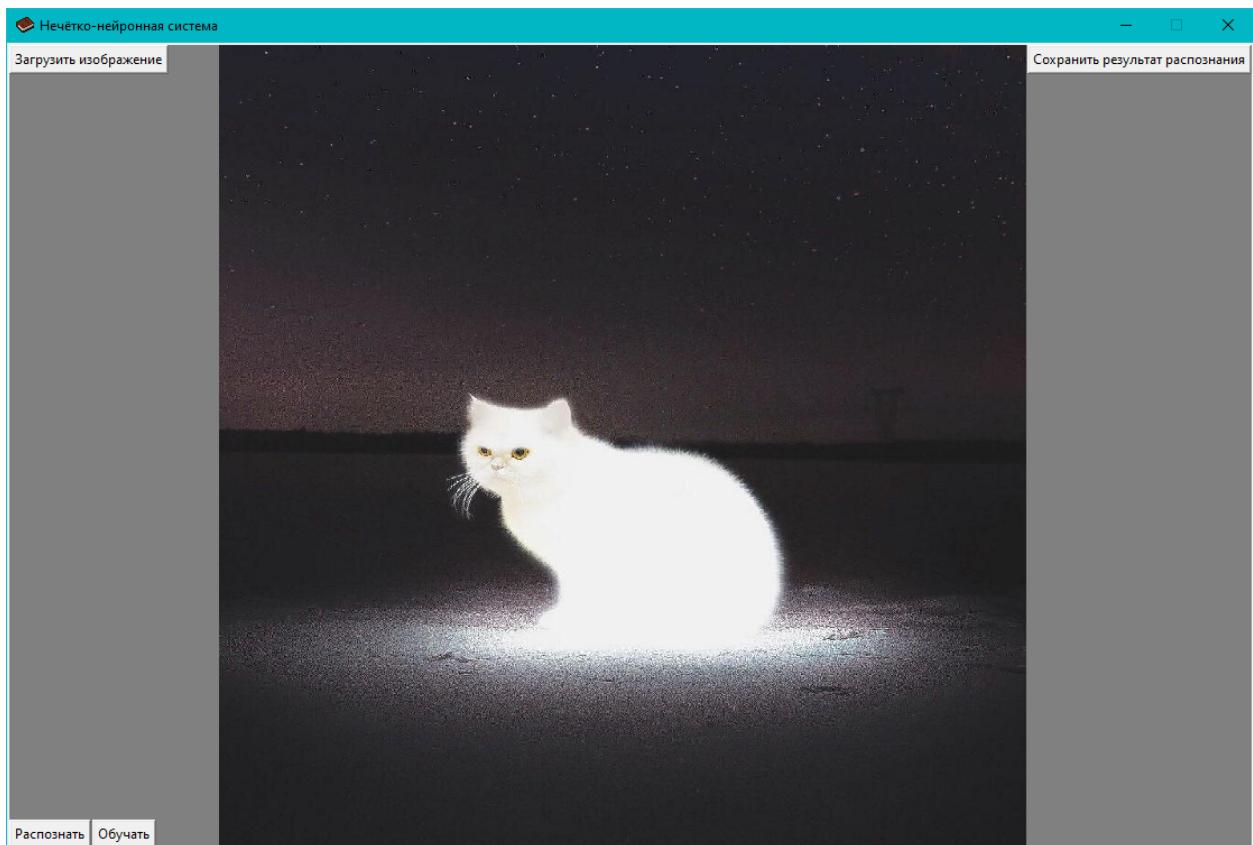


Рисунок 4.16 – Интерфейс с изображением кошки

На рисунке 4.17 изображение кошки было обработано по модели "Белый".

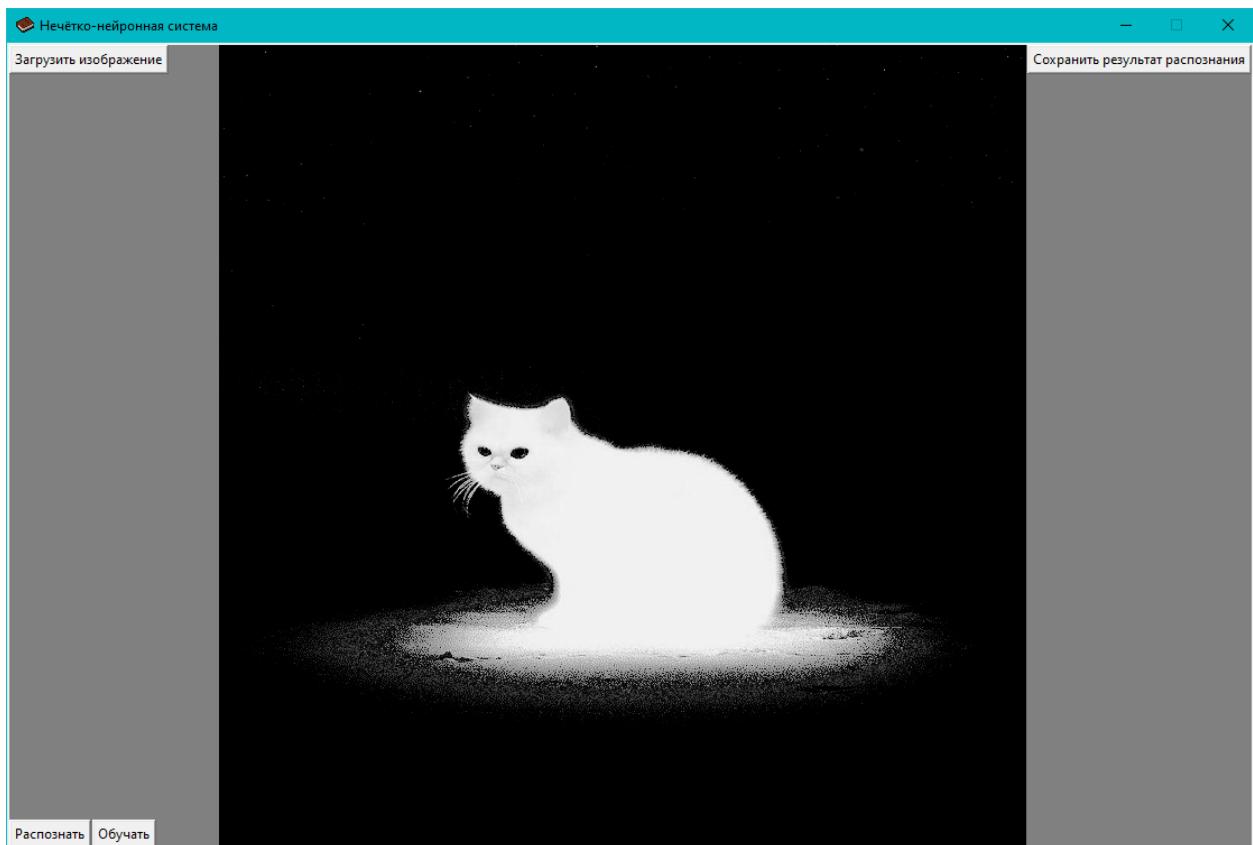


Рисунок 4.17 – Распознанные объекты на изображении кошки

На рисунке 4.18 было загружено изображение лилии.

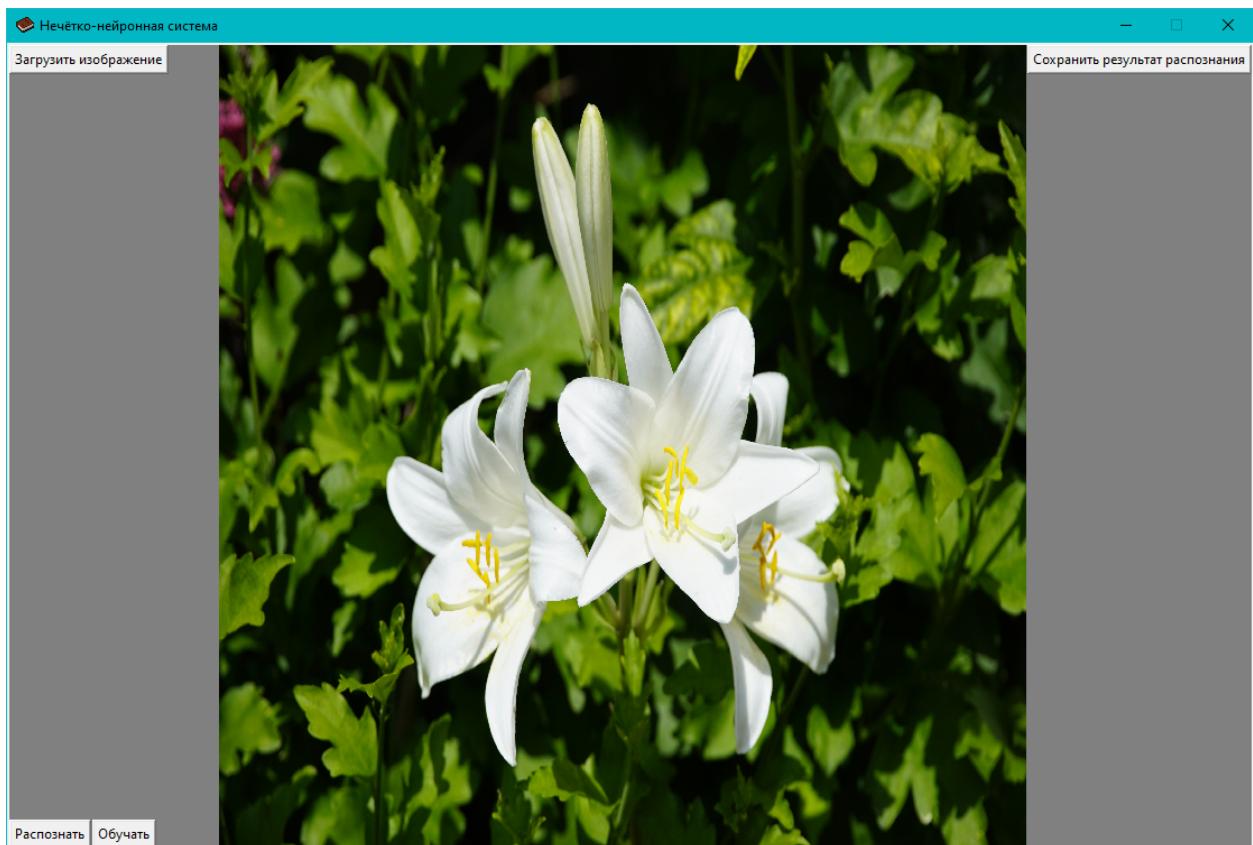


Рисунок 4.18 – Интерфейс с изображением лилии

На рисунке 4.19 изображение лилии было обработано по модели "Белый".

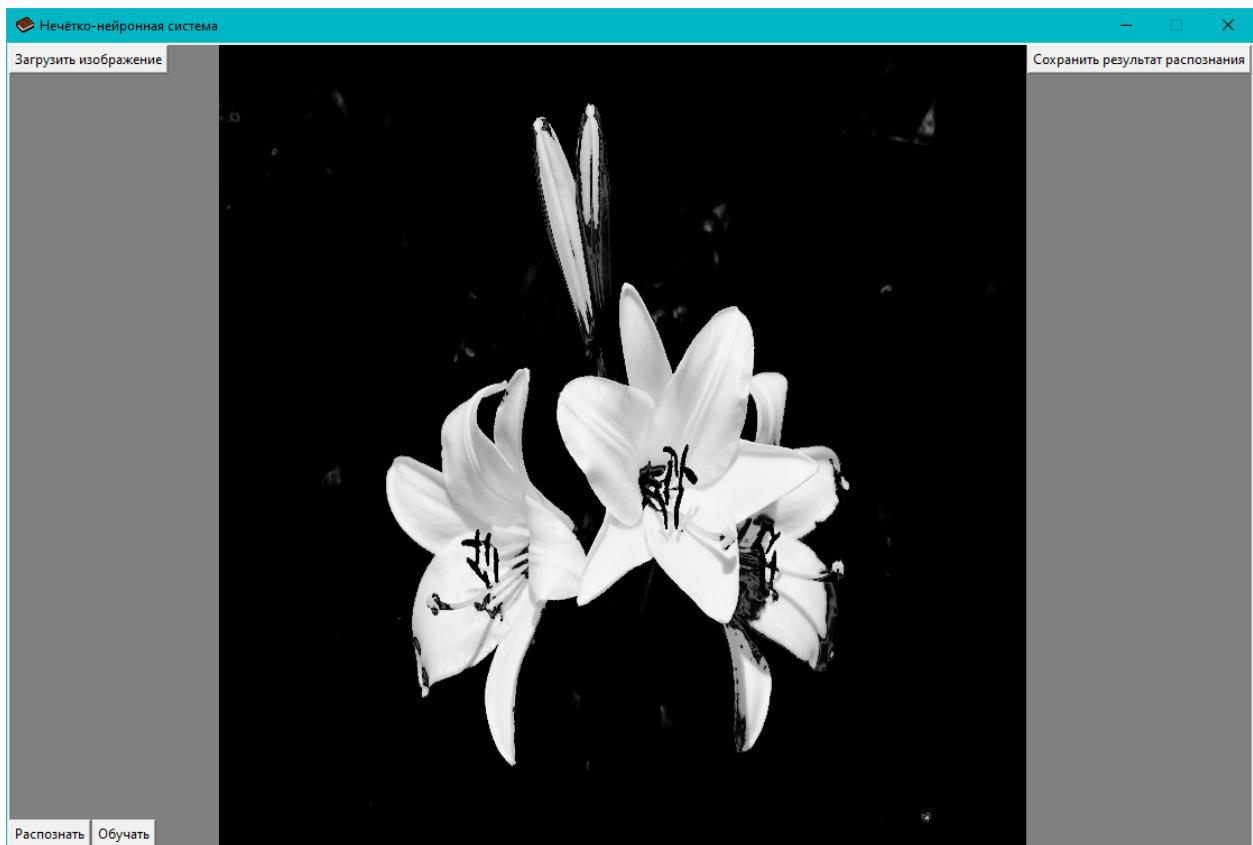


Рисунок 4.19 – Распознанные объекты на изображении лилии

На рисунке 4.20 было загружено изображение леса.

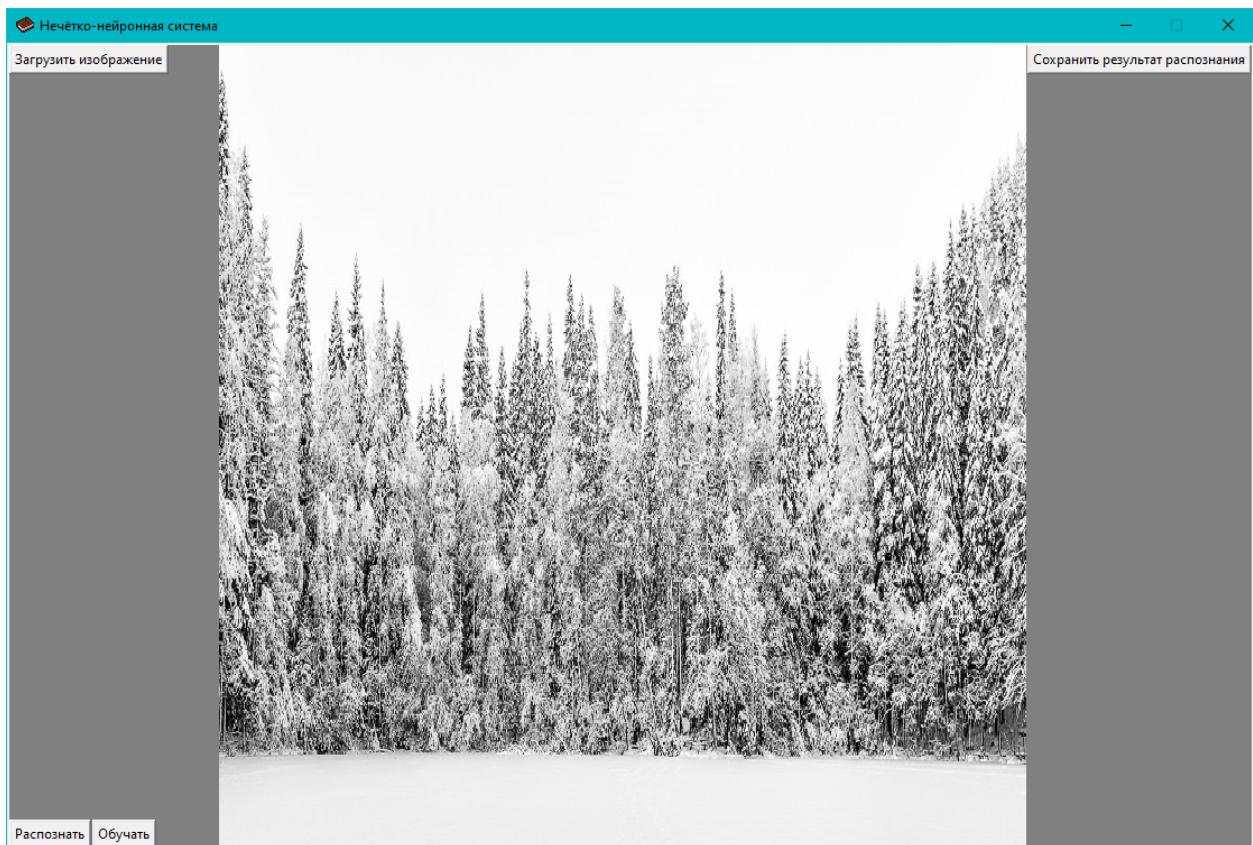


Рисунок 4.20 – Интерфейс с изображением леса

На рисунке 4.21 изображение леса было обработано по модели "Белый".

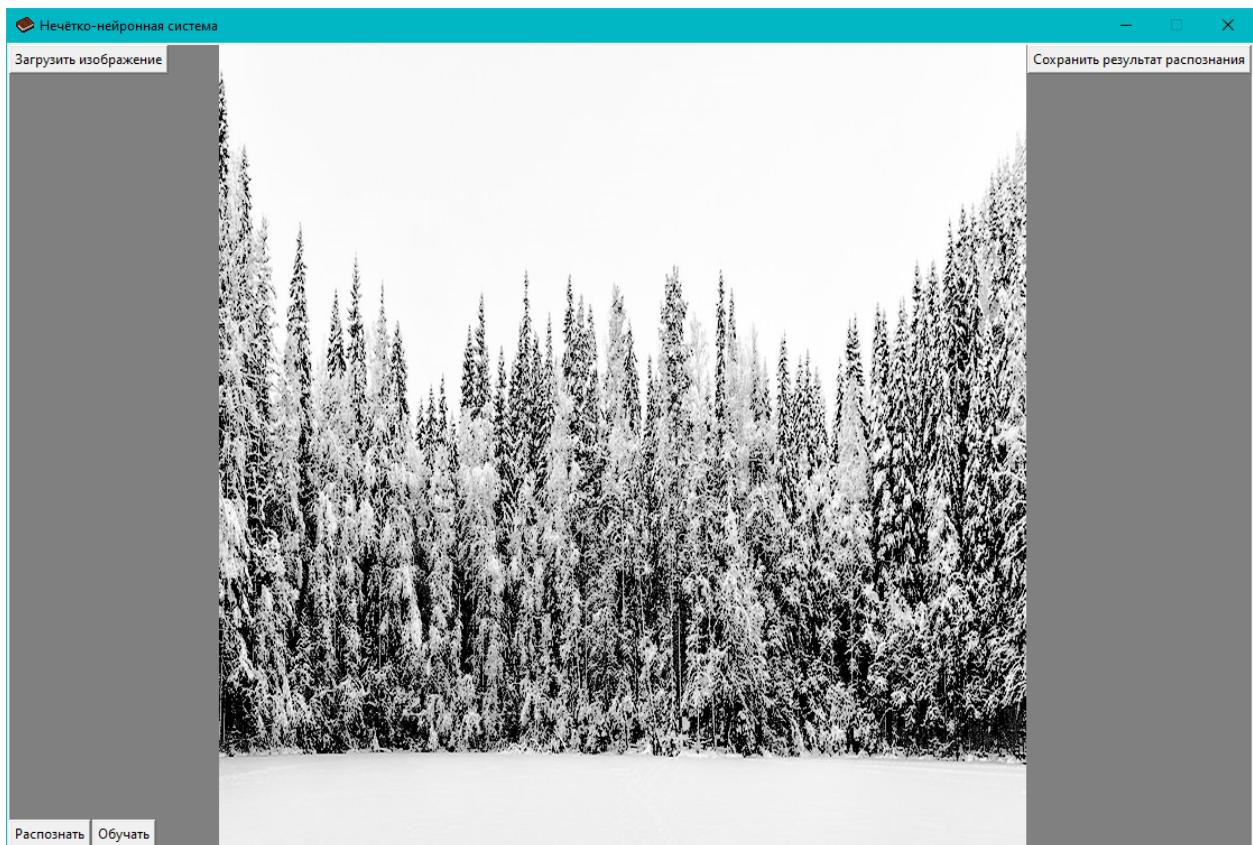


Рисунок 4.21 – Распознанные объекты на изображении леса

На рисунке 4.22 было загружено изображение футбольного мяча.

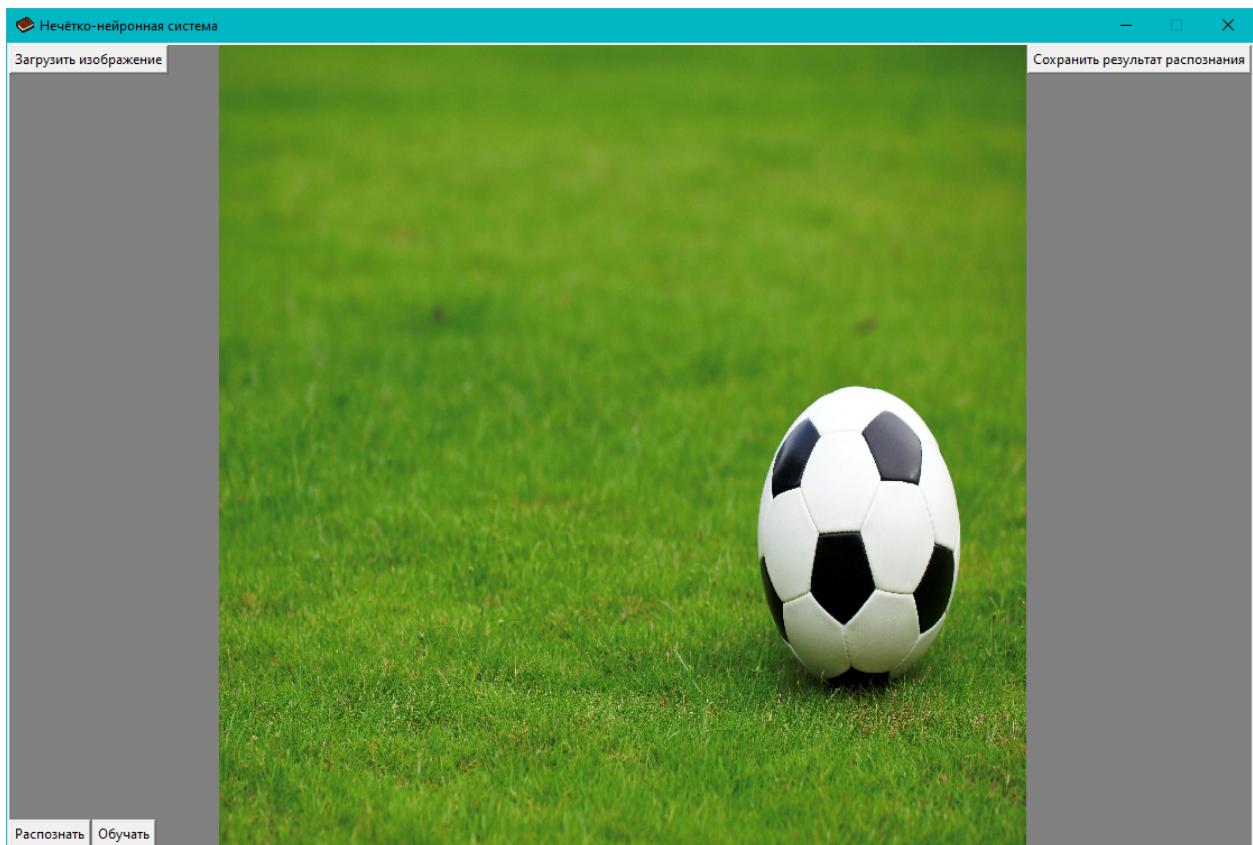


Рисунок 4.22 – Интерфейс с изображением футбола

На рисунке 4.23 изображение футбола было обработано по модели "Белый".

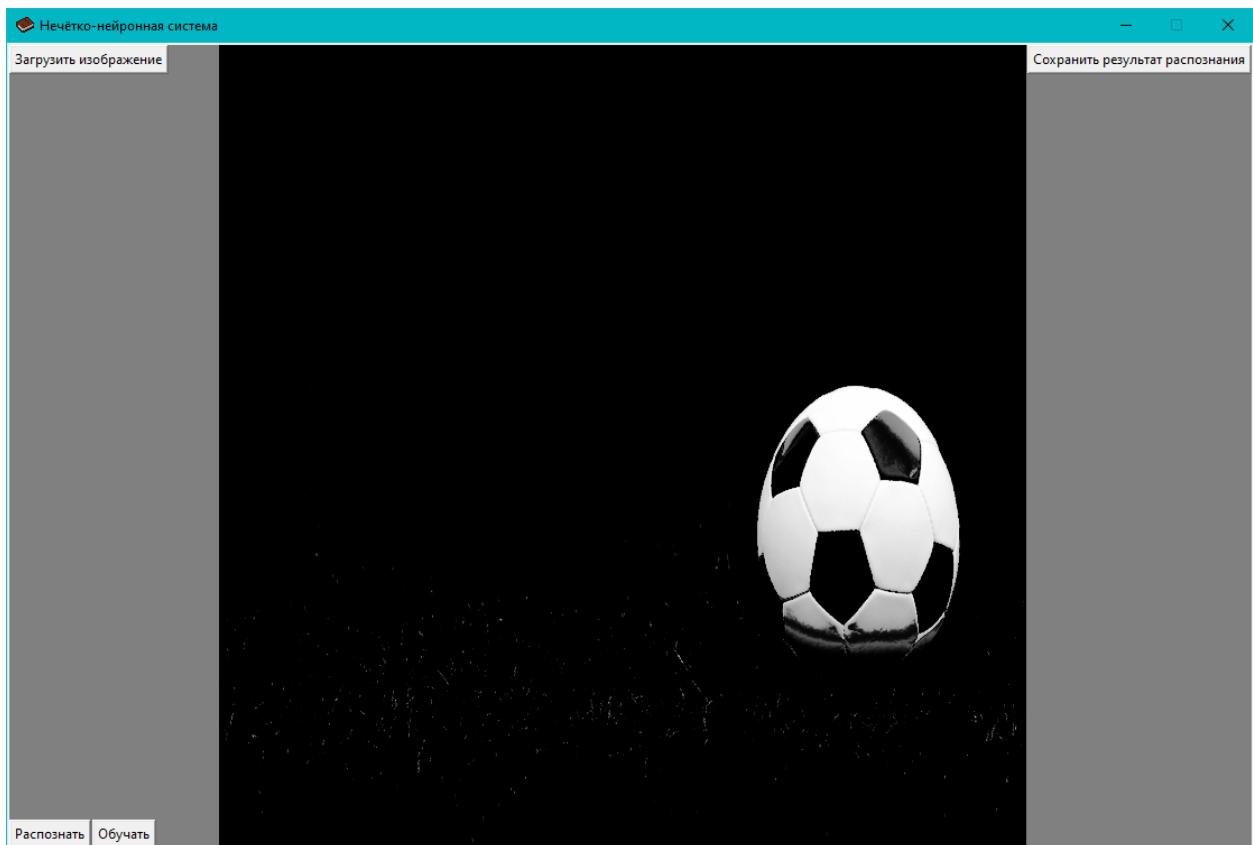


Рисунок 4.23 – Распознанный объекты на изображении футбола

На рисунке 4.24 было загружено изображение платья.

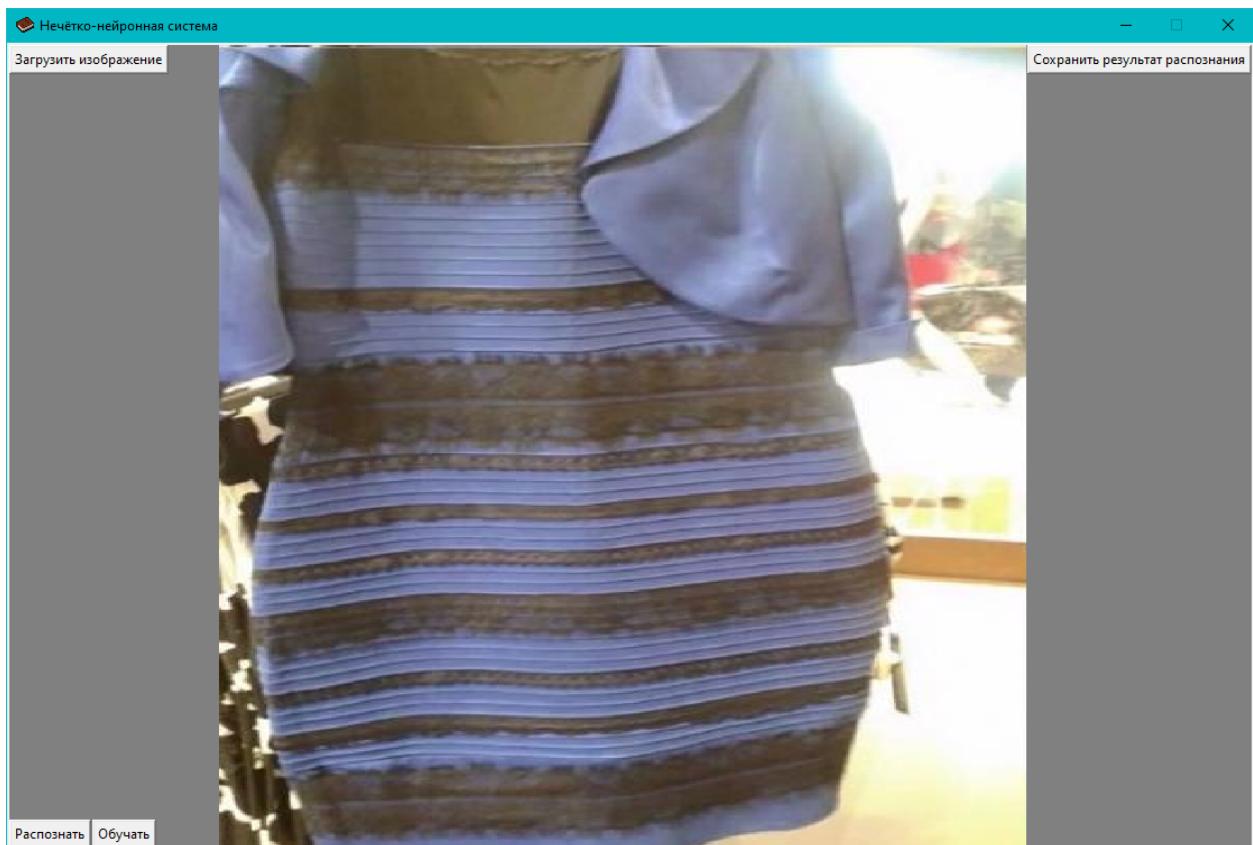


Рисунок 4.24 – Интерфейс с изображением платья

На рисунке 4.25 изображение платья было обработано по модели "Белый".

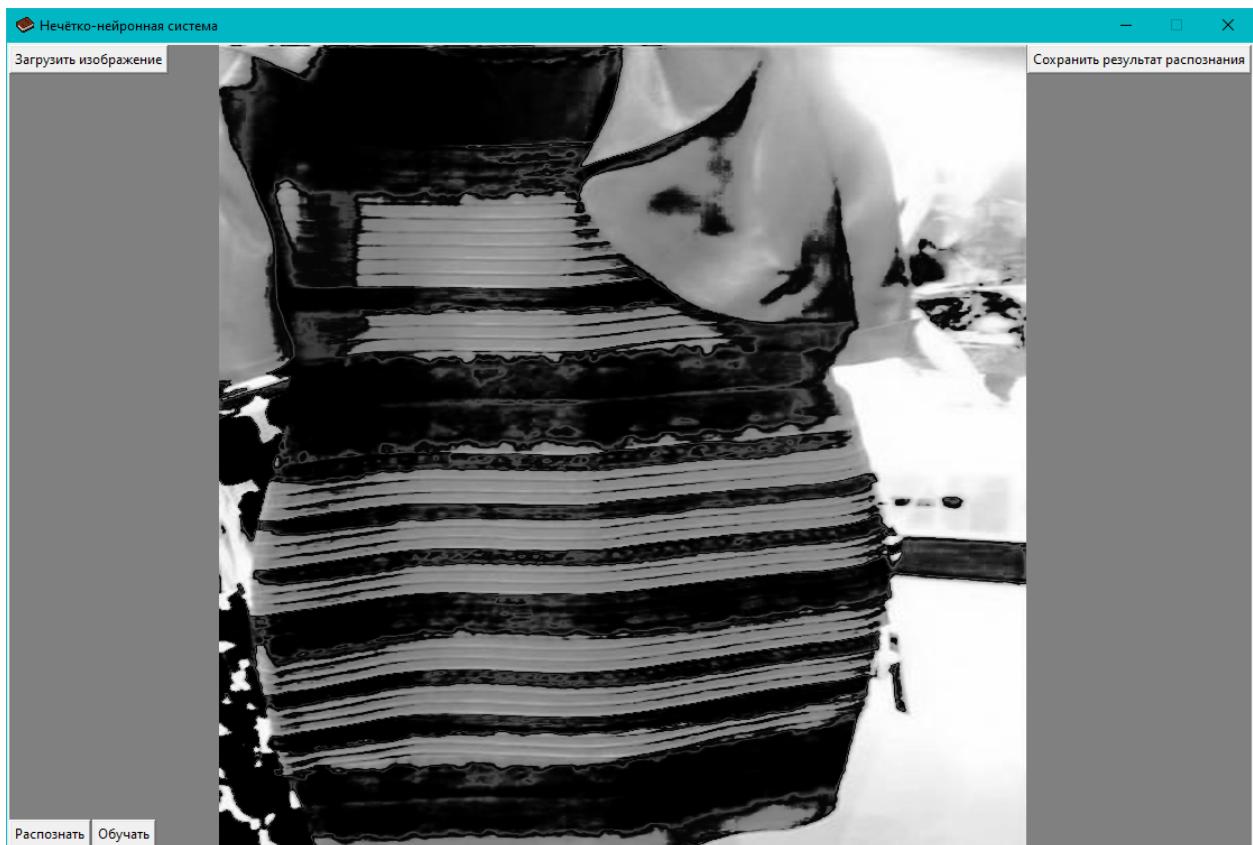


Рисунок 4.25 – Распознанные объекты на изображении платья

На рисунке 4.26 было загружено изображение берёзы.

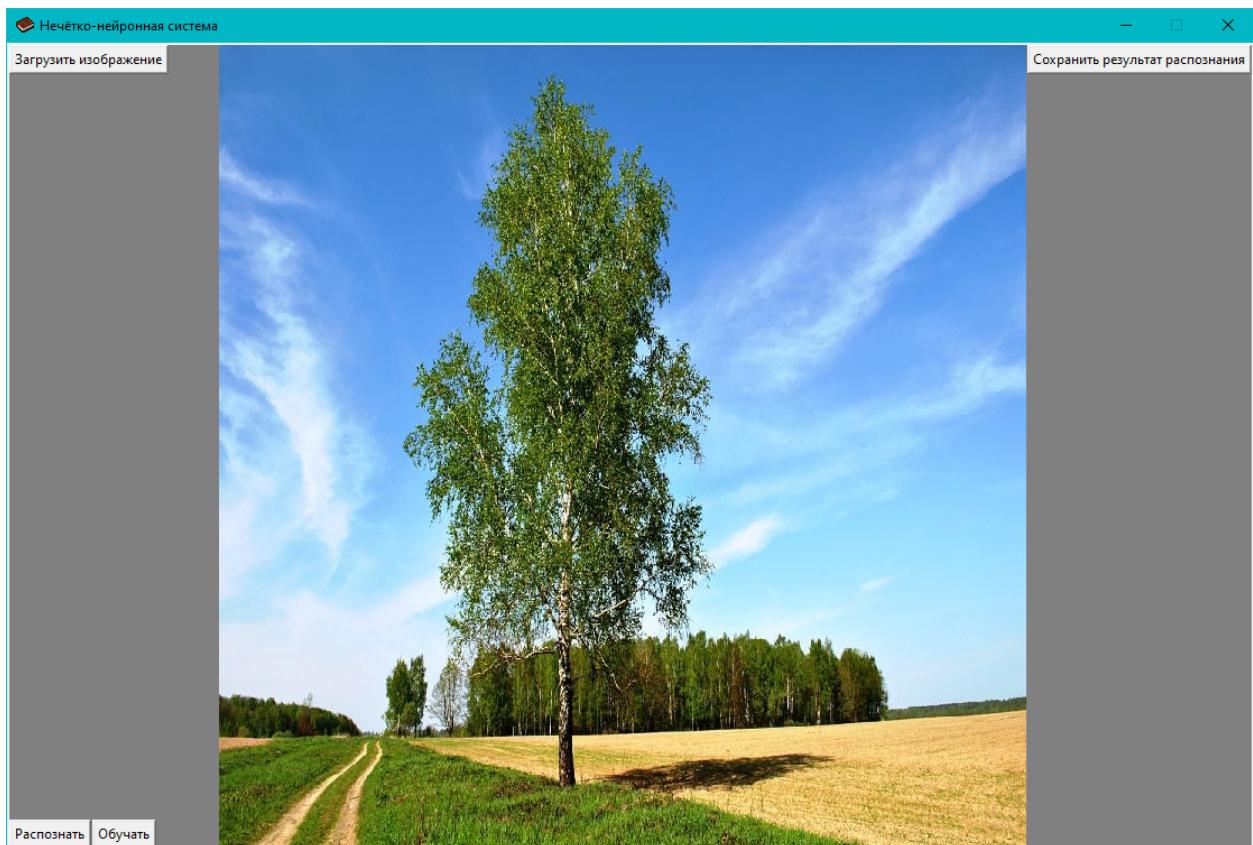


Рисунок 4.26 – Интерфейс с изображением берёзы

На рисунке 4.27 изображение берёзы было обработано по модели "Белый".

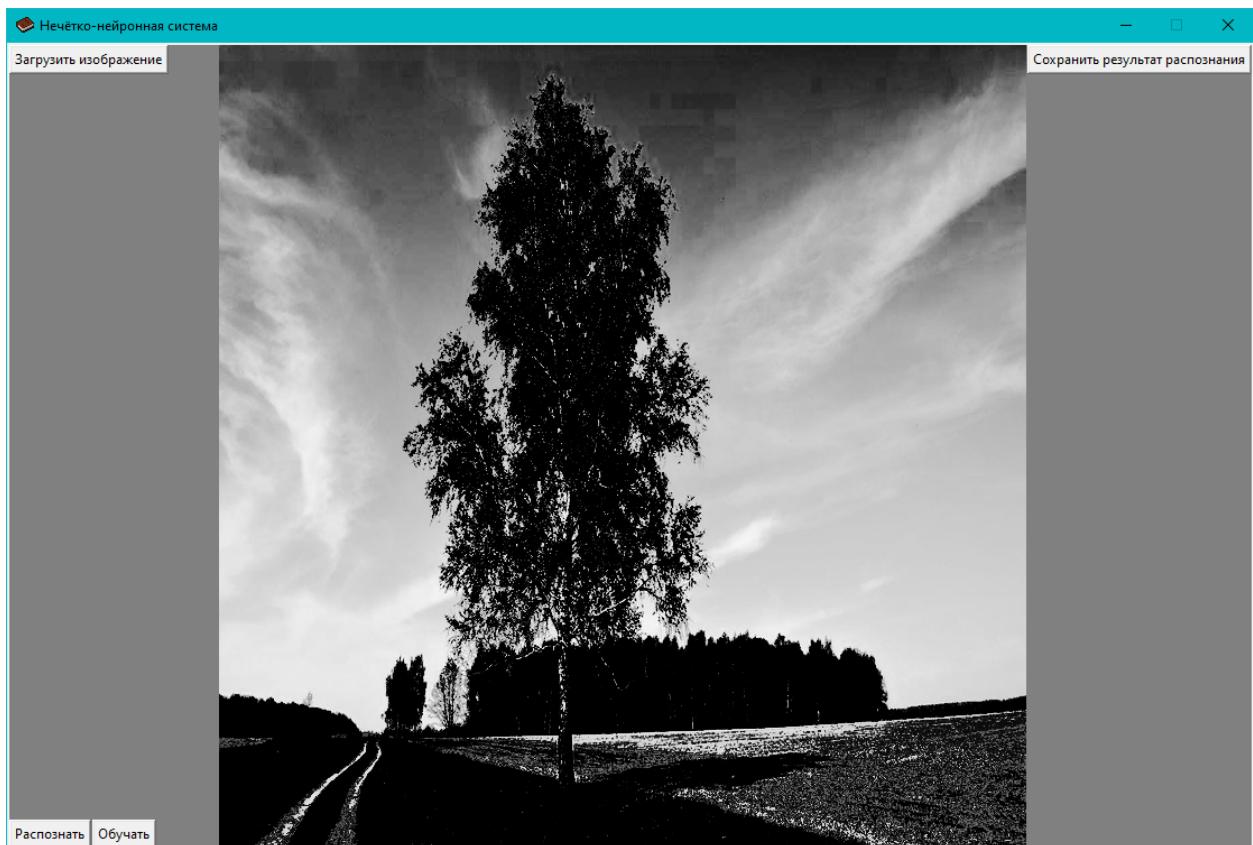


Рисунок 4.27 – Распознанный объекты на изображении берёзы

На рисунке 4.28 было загружено изображение двери.

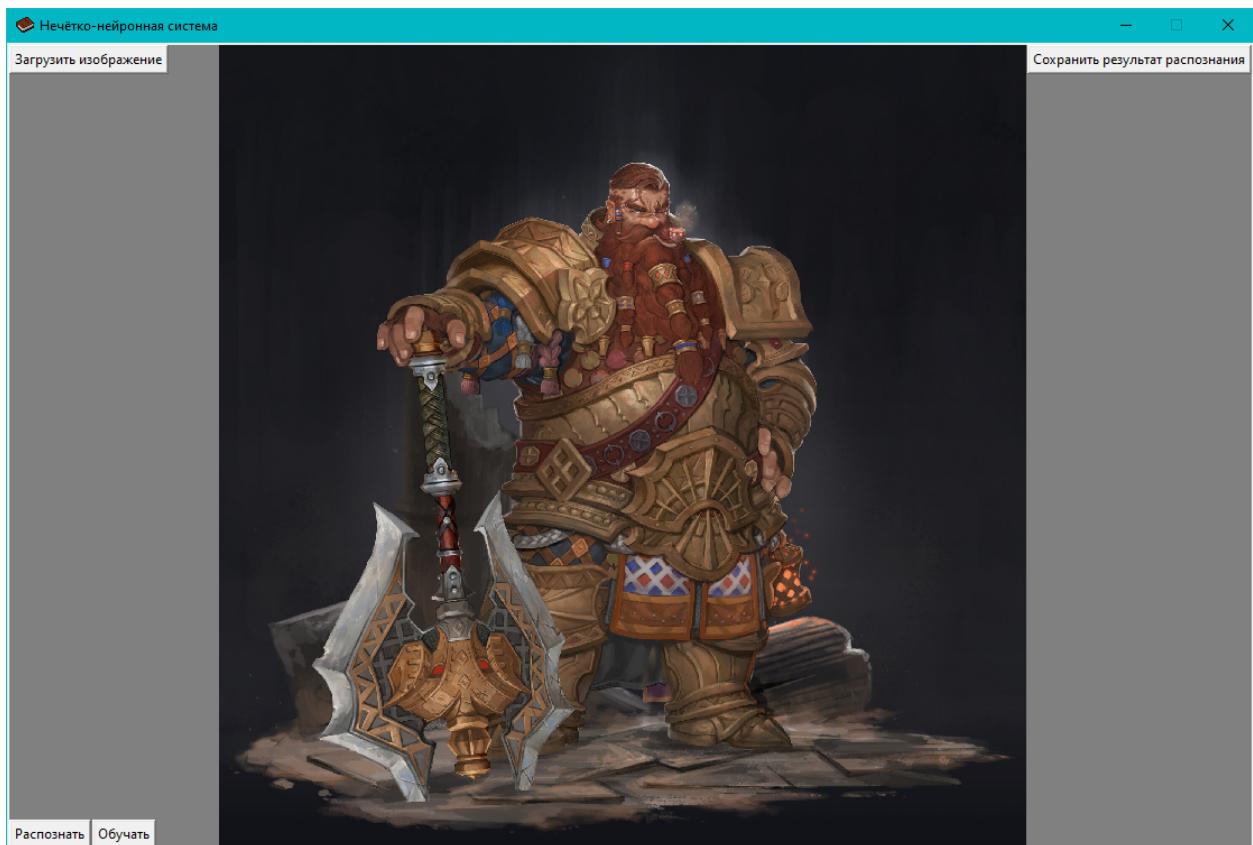


Рисунок 4.28 – Интерфейс с изображением дварфа

На рисунке 4.29 изображение дварфа было обработано по модели "Белый".

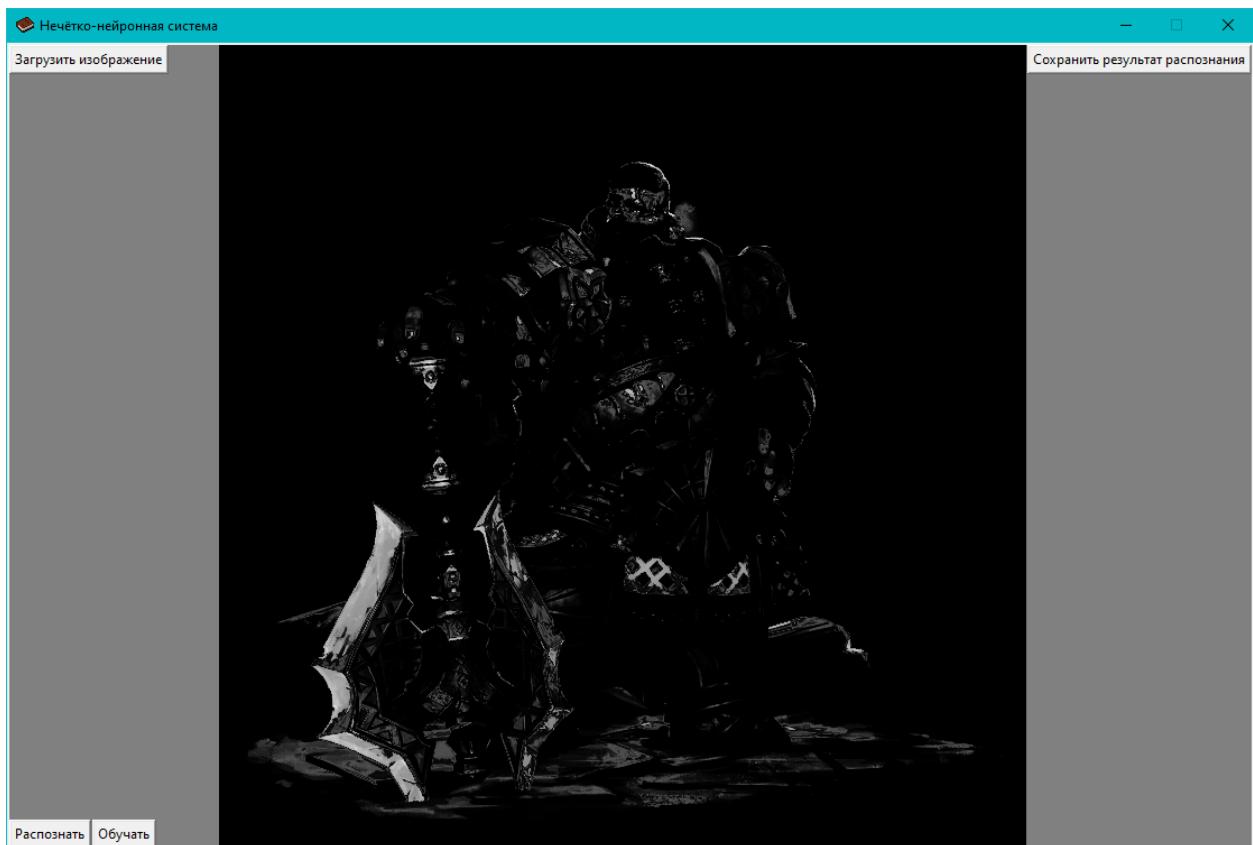


Рисунок 4.29 – Распознанные объекты на изображении дварфа

На рисунке 4.30 было загружено изображение Чебурашки.

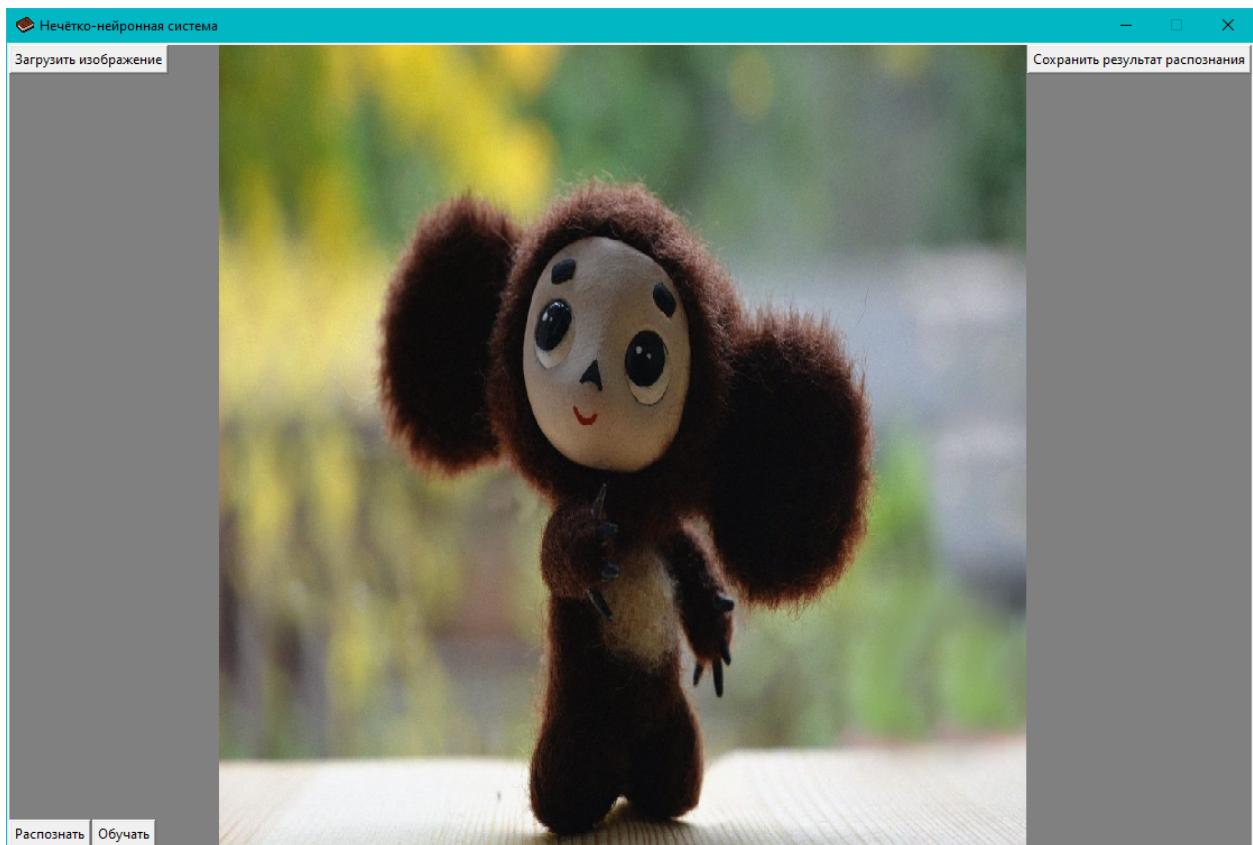


Рисунок 4.30 – Интерфейс с изображением Чебурашки

На рисунке 4.31 изображение Чебурашки было обработано по модели “Белый”.

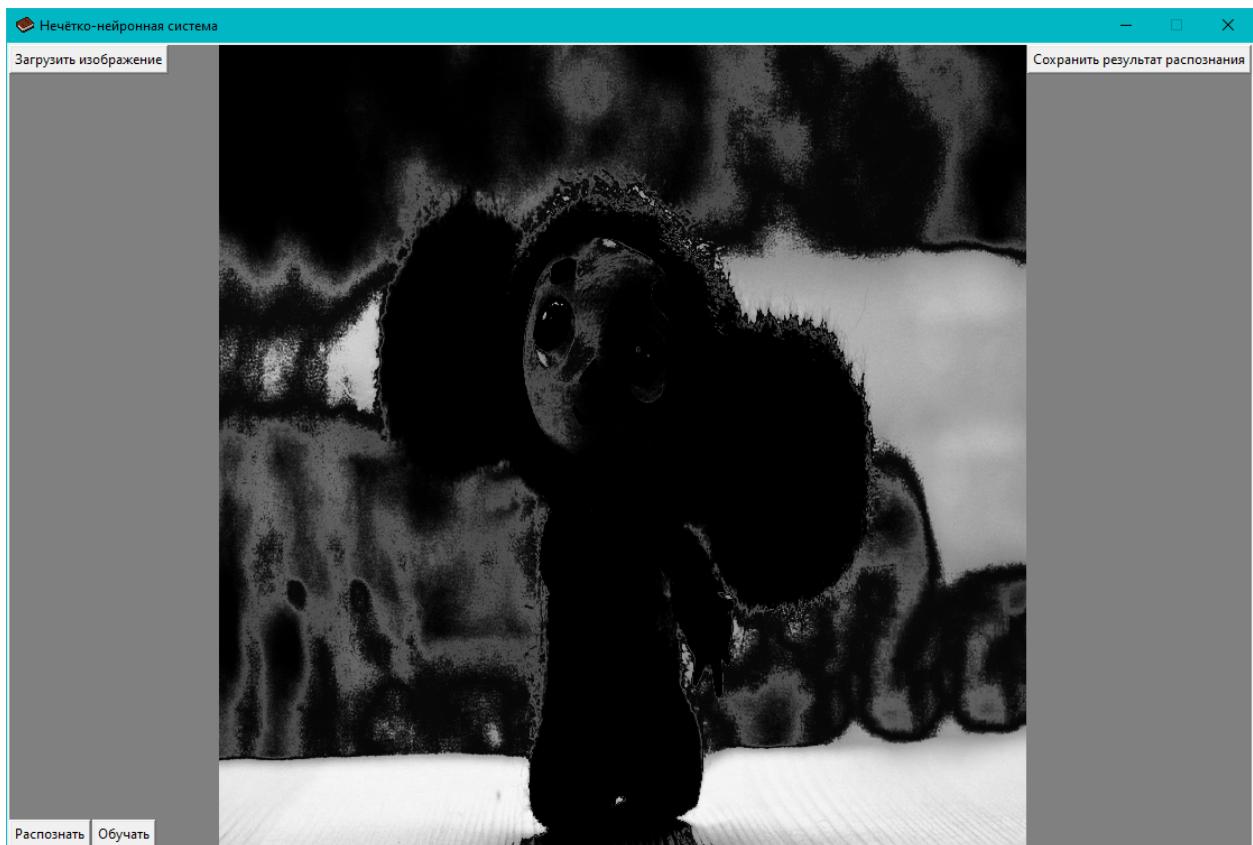


Рисунок 4.31 – Распознанные объекты на изображении Чебурашки

ЗАКЛЮЧЕНИЕ

В заключение, проект по созданию интеллектуальной системы распознавания объектов по цветовым характеристикам на основе нечётких нейронных сетей демонстрирует значительный прогресс в области компьютерного зрения. Эта разработка не только подтверждает возможности искусственного интеллекта и машинного обучения в решении сложных задач, но и открывает новые перспективы для их применения в различных сферах жизнедеятельности человека.

Использование адаптивных нечётких нейронных сетей позволяет системе эффективно адаптироваться к изменяющимся условиям и обеспечивает высокую точность распознавания объектов. Таким образом, данное приложение может стать незаменимым инструментом в медицинской диагностике, обеспечении безопасности и других ключевых областях.

Проект также иллюстрирует важность синергии между технологическими инновациями и потребностями общества. Результаты, полученные в ходе работы над проектом, могут служить основой для дальнейших исследований и разработок, направленных на улучшение качества жизни и безопасности людей.

Таким образом, проект подчёркивает роль искусственного интеллекта как ключевого фактора в продвижении научно-технического прогресса и открывает новые горизонты для инновационных исследований в будущем.

Основные результаты работы:

1. Проведен анализ предметной области. Выявлена необходимость использовать язык Python и библиотеки MATLAB;
2. Разработана концептуальная модель приложения. Разработана модель данных системы. Определены требования к системе;
3. Осуществлено проектирование приложения. Разработана архитектура нейронной сети и базы данных. Разработан пользовательский интерфейс приложения;

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

Все требования, объявленные в техническом задании, были полностью реализованы, все задачи, поставленные в начале разработки проекта, были также решены.

Готовый рабочий проект представлен приложением.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лутц, М. Изучаем Python, 5-е издание / М. Лутц. – Санкт-Петербург : Питер, 2019. – 1584 с. – ISBN 978-5-4461-0705-9. – Текст : непосредственный.
2. Гудфеллоу, И., Бенджио, Ю., Курвилль, А. Глубокое обучение / И. Гудфеллоу, Ю. Бенджио, А. Курвилль. – Москва : ДМК Пресс, 2017. – 652 с. – ISBN 978-5-97060-487-9. – Текст : непосредственный.
3. Хайкин, С. Нейронные сети: полный курс, 2-е издание / С. Хайкин. – Москва : Вильямс, 2018. – 1104 с. – ISBN 978-5-8459-2101-0. – Текст : непосредственный.
4. Росс, Т. Дж. Нечеткая логика с приложениями в инженерных науках / Т. Дж. Росс. – Москва : Мир, 2016. – 800 с. – ISBN 978-5-03-004474-5. – Текст : непосредственный.
5. Мерфи, К. Машинное обучение: вероятностный подход / К. Мерфи. – Москва : ДМК Пресс, 2018. – 704 с. – ISBN 978-5-97060-212-7. – Текст : непосредственный.
6. Рашка, С., Мирджалили, В. Python и машинное обучение / С. Рашка, В. Мирджалили. – Москва : ДМК Пресс, 2018. – 418 с. – ISBN 978-5-97060-310-0. – Текст : непосредственный.
7. Жолковский, Е. К. TensorFlow для профессионалов / Е. К. Жолковский. – Москва : ДМК Пресс, 2019. – 480 с. – ISBN 978-5-97060-746-7. – Текст : непосредственный.
8. Чоллет, Ф. Глубокое обучение на Python / Ф. Чоллет. – Москва : ДМК Пресс, 2018. – 304 с. – ISBN 978-5-97060-409-1. – Текст : непосредственный.
9. Клейн, Р. Нечеткие системы в Python / Р. Клейн. – Москва : ДМК Пресс, 2020. – 320 с. – ISBN 978-5-97060-758-0. – Текст : непосредственный.

10. Бейдер, Д. Python Tricks: A Buffet of Awesome Python Features / Д. Бейдер. – Москва : ДМК Пресс, 2021. – 300 с. – ISBN 978-5-97060-999-7. – Текст : непосредственный.
11. Герон, О. Практическое машинное обучение с Scikit-Learn и TensorFlow / О. Герон. – Москва : ДМК Пресс, 2019. – 572 с. – ISBN 978-5-97060-524-1. – Текст : непосредственный.
12. Нильсен, М. Нейронные сети и глубокое обучение / М. Нильсен. – Москва : ДМК Пресс, 2021. – 250 с. – ISBN 978-5-97060-777-1. – Текст : непосредственный.
13. Джеймс, Д. Нечеткие системы и Python: практическое руководство / Д. Джеймс. – Москва : ДМК Пресс, 2022. – 340 с. – ISBN 978-5-97060-888-4. – Текст : непосредственный.

ПРИЛОЖЕНИЕ А

Представление графического материала

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

Сведения о ВКРБ

Минобрнауки России
Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА ПО ПРОГРАММЕ БАКАЛАВРИАТА

«Интеллектуальная система распознавания объектов по цветовым характеристикам на основе нечетких нейронных сетей»

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

Автор ВКРБ
студент группы ПО-026
Тягунов Владимир Вячеславович

Рисунок А.1 – Сведения о ВКРБ

Цель и задачи разработки

Цель работы – разработка приложения на базе адаптивной нечёткой нейронной сети для распознавания и классификации объектов на изображениях по цветовым характеристикам.

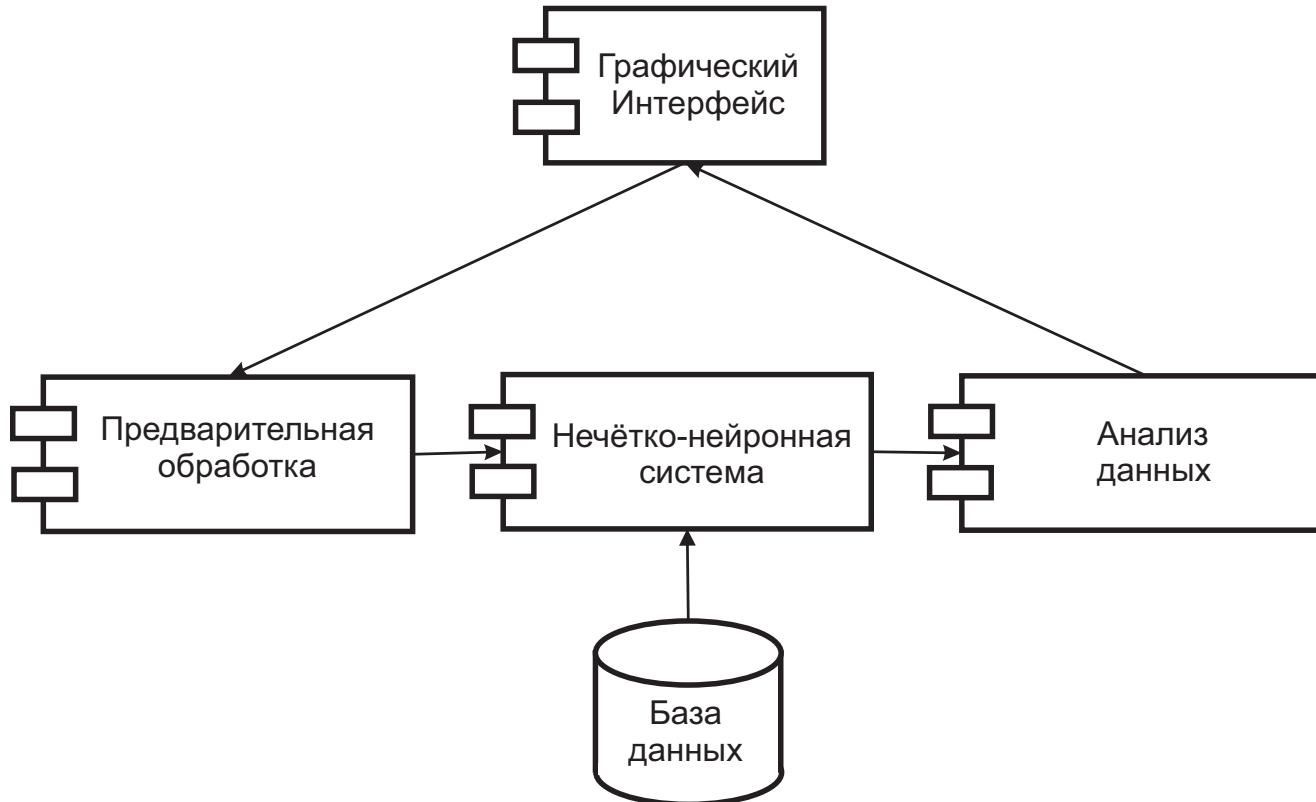
Для достижения поставленной цели необходимо решить следующие задачи:

- проводить анализ предметной области;
- разработать концептуальную модель приложения;
- спроектировать приложение;
- реализовать приложение.

Фамилия И. О. Имя И. И. Отчество И. И.		VKRB 20060149.09.03.0424.010
Автор работы Чагунов В.В.		Сведения об VKRB
Группа ЧИ-10		Лицо Номер Номер
Нормандоль Чемигин А.А.		Лист 1 Лист 5
		Авторская группа: Учебно-исследовательский лаборатория бакалавров
		ЮЗГУ №-026

Рисунок А.2 – Цель и задачи разработки

Диаграмма компонентов



	ВКРБ 20060149.09.03.04.24.010	Ни Имя Фамилия
	Сведения об ВКРБ	
Автор/работы	Фамилия И.О.	Литер/Инициалы
Руководитель	Григорьев В.В.	
Координатор	Ивановская Г.Р.	
Внешность	Чепалова А.А.	
	Адрес электронной почты admin@kuban.kareva.ru	
	ЮЗГУ №-020	

Рисунок А.3 – Диаграммы компонентов

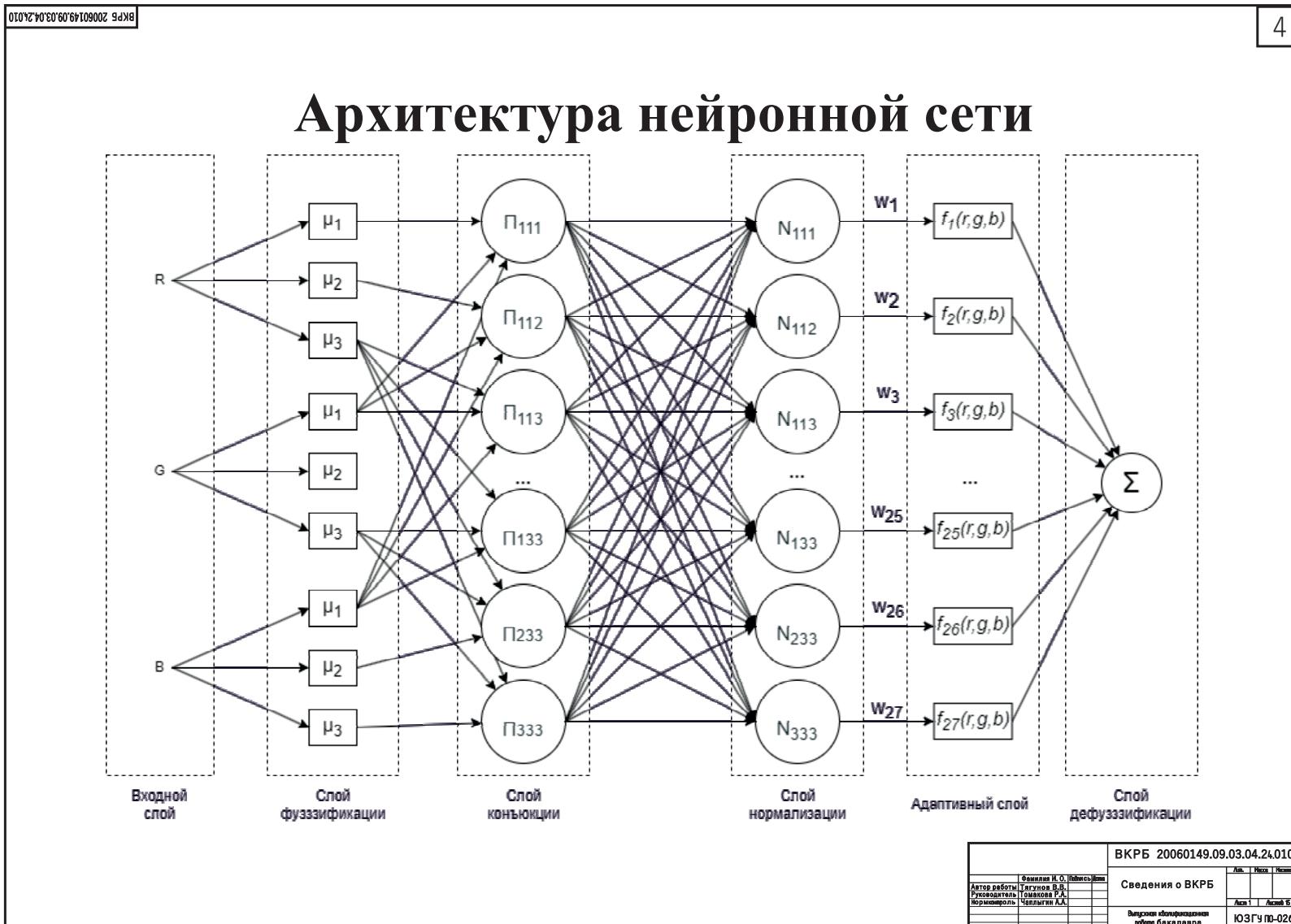


Рисунок А.4 – Архитектура нейронной сети

Функция принадлежности

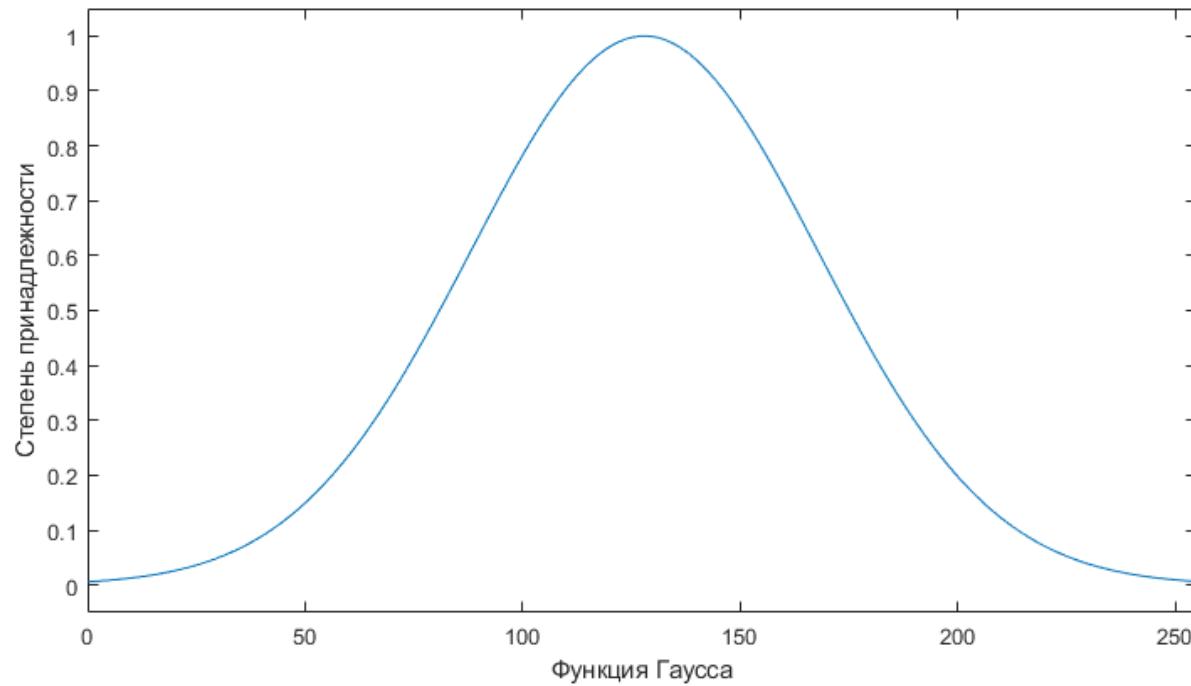


Рисунок А.5 – Функция принадлежности

Диаграмма архитектуры базы данных

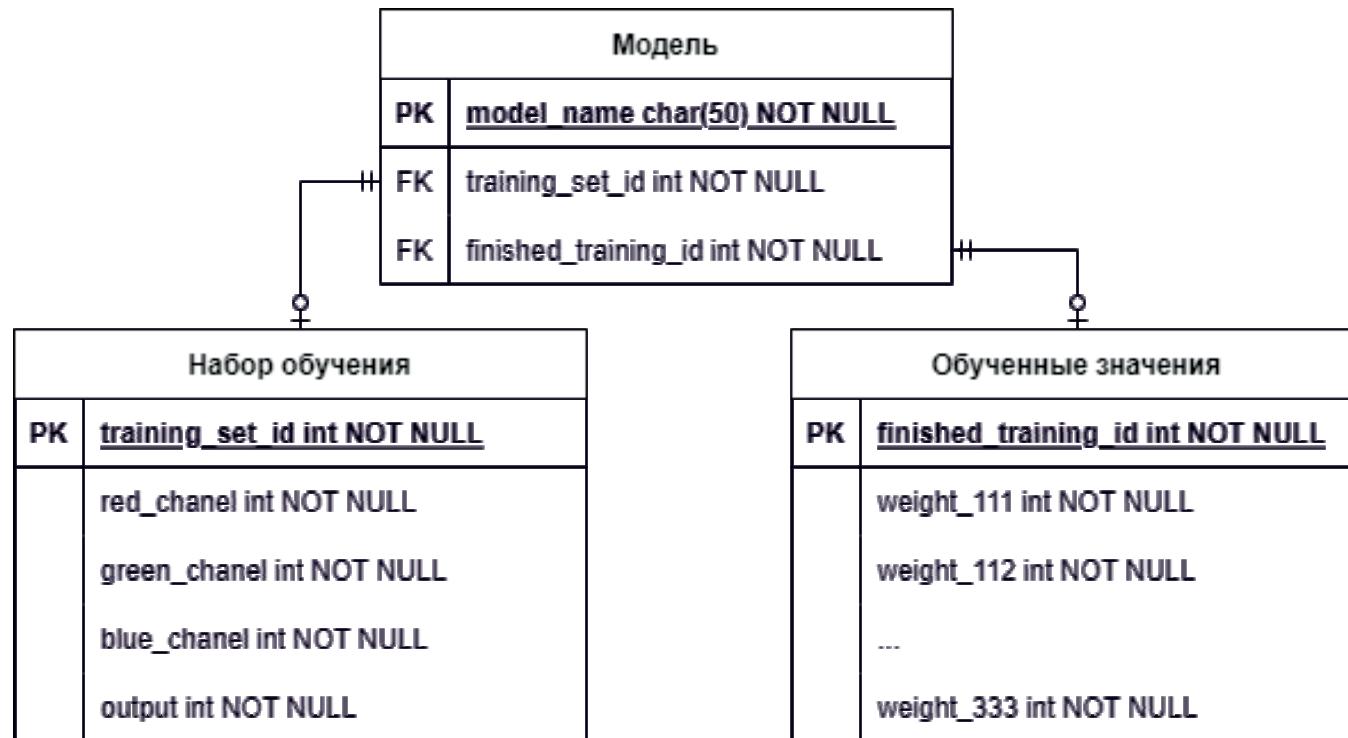


Рисунок А.6 – База данных

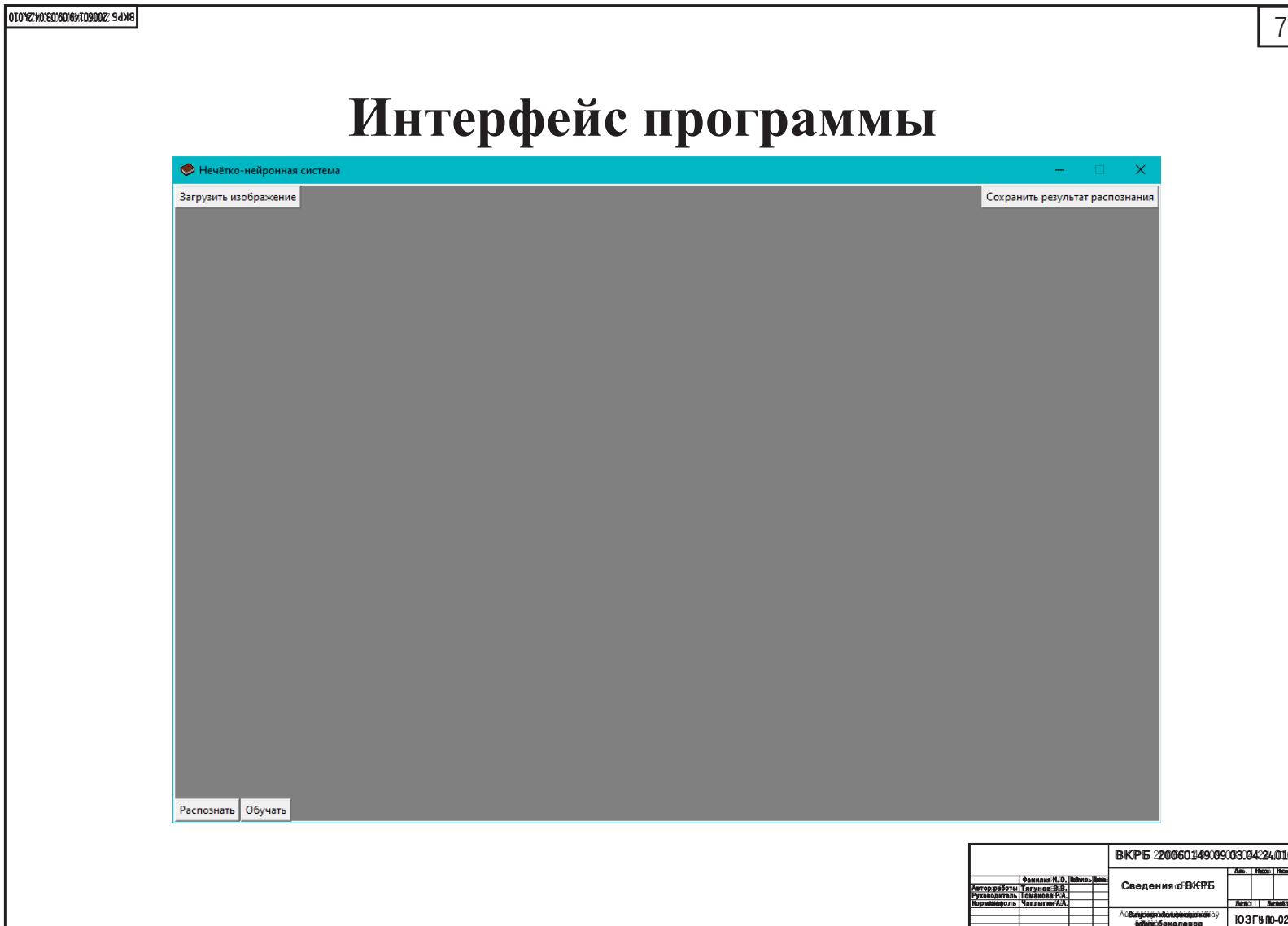


Рисунок А.7 – Интерфейс приложения

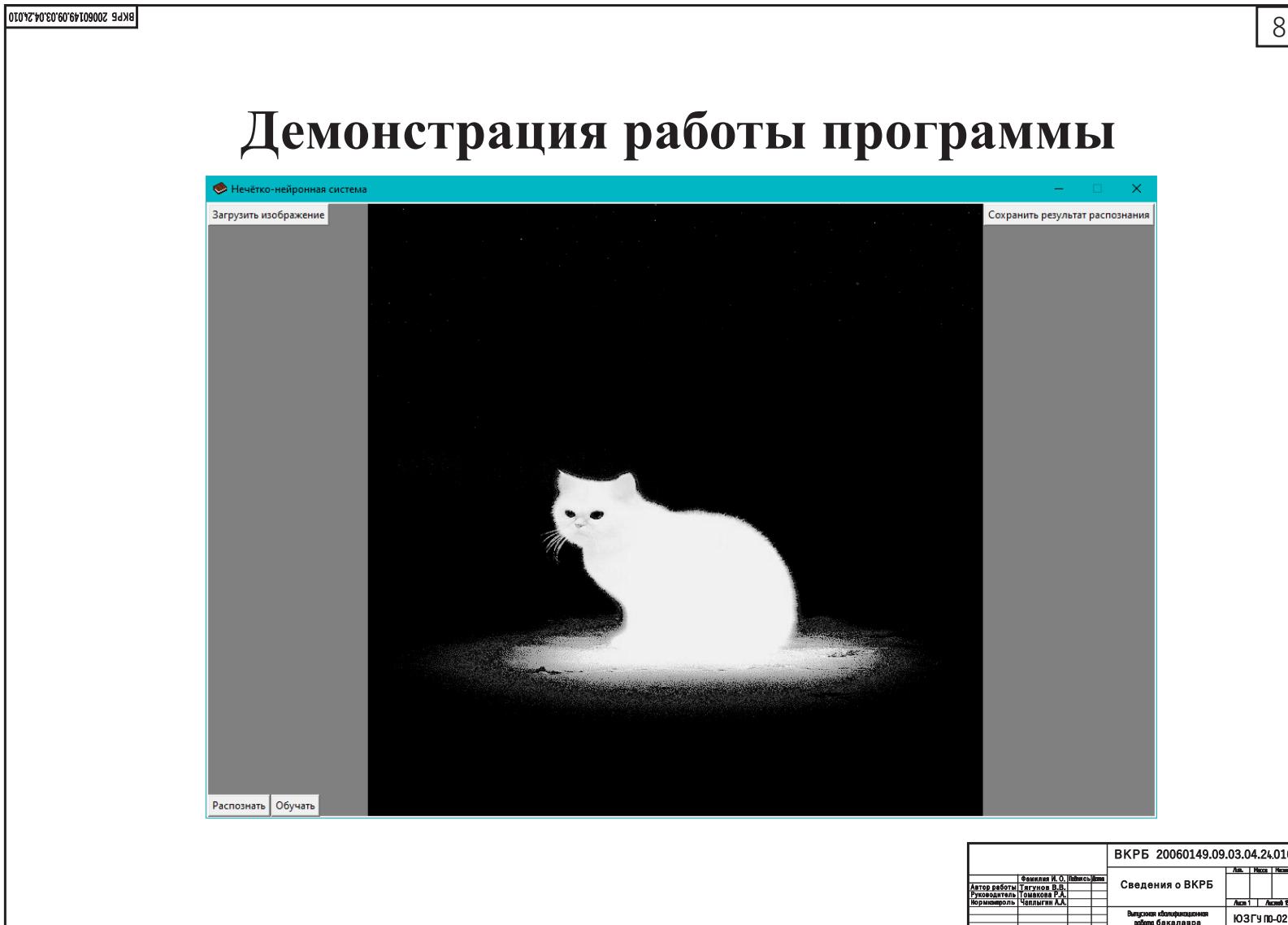


Рисунок А.8 – Демонстрация работы программы

Заключение

Основные результаты работы:

Проведен анализ предметной области. Выявлена необходимость использовать язык Python и библиотеки MATLAB;

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

Осуществлено проектирование приложения. Разработана архитектура нейронной сети и базы данных. Разработан пользовательский интерфейс приложения;

Реализовано и протестировано приложение. Проведено модульное и системное тестирование.

		ВКРБ 20060149.09.03.04.24.010
		Сведения о ВКРБ
Автор работы	Фамилия И. О. Ильинский	
Автор работы	Фамилия В. В.	
Руководитель	Фамилия Р. А.	
Координатор	Фамилия А. А.	
Выпускное квалификационное		
документ бакалавра		
ЮЗГУ №-026		

Рисунок А.9 – Заключение

ПРИЛОЖЕНИЕ Б

Фрагменты исходного кода программы

main.py

```
1 from tkinter import *
2 from tkinter import filedialog
3 from tkinter.ttk import Combobox
4 from ANFIS import АнфисаРаспознать
5 from ANFIS import АнфисаТренировать
6 from DB import сохранить
7 from DB import загрузитьСписокНаборов
8 from DB import загрузитьНабор
9 from PIL import Image
10
11 global Набор
12 Окно = Tk()
13 Окно.geometry("1112x720")
14 Окно.resizable(False, False)
15 Окно.title("Нечётко-нейронная система")
16 Окно.iconbitmap(default="Иконка.ico")
17 ПолеИзображения = Canvas(bg="Gray", height=1080, width=720)
18 ПолеИзображения.pack(fill=BOTH)
19 НижняяГраница = 720
20 ПраваяГраница = 1112
21 ВысотаКнопки = 25
22 Интервал = 2
23 def загрузитьИзображение():
24     global Изображение
25     global Путь
26     Путь = filedialog.askopenfilename(filetypes=[("Изображения", "*.png;*.jpg
27 ;*.jpeg")])
28     if Путь:
29         Изображение = PhotoImage(file=Путь)
30         Ширина = Изображение.width()
31         Высота = Изображение.height()
32         НовоеИзображение = PhotoImage(width=720, height=720)
33         for x in range(720):
34             for y in range(720):
35                 x = int(x * Ширина / 720)
36                 y = int(y * Высота / 720)
37                 rgb = '#%02x%02x%02x' % Изображение.get(x, y)
38                 НовоеИзображение.put(rgb, (x, y))
39         Изображение = НовоеИзображение
40         ПолеИзображения.create_image(Интервал + 187, Интервал, anchor=NW,
41                                         image=Изображение, tags='Старое')
42     def распознать():
43         try:
44             if Путь:
45                 ОкноВыбораМодели = Toplevel()
46                 ОкноВыбораМодели.title("Выберите модель для распознания")
47                 ОкноВыбораМодели.geometry("300x70")
48                 СписокМоделей = []
49                 Список = загрузитьСписокНаборов()
```

```

48     for Элемент in Список:
49         СписокМоделей.append(Элемент)
50     global ВыбранныйЭлементСписка
51     ВыбранныйЭлементСписка = StringVar(value=СписокМоделей[0])
52     Список = Combobox(ОкновыбораМодели, values=СписокМоделей,
53                         textvariable=ВыбранныйЭлементСписка)
54     Список.pack(anchor=NW, padx=6, pady=6, fill='x')
55     КнопкаПодтверждения = Button(ОкновыбораМодели, text="Выбрать",
56                                   command=lambda: выбрать(ОкновыбораМодели))
57     КнопкаПодтверждения.pack(anchor=NW, padx=6, pady=6)
58     ОкновыбораМодели.grab_set()
59 except Exception as e:
60     print('Нет пути файла:\n{}'.format(e))
61
60 def выбрать(Окновыбора):
61     global Изображение
62     АнфисаРаспознать(Путь, ВыбранныйЭлементСписка.get())
63     ПолеИзображения.delete('Старое')
64     Изображение = PhotoImage(file='output_image.png')
65     ПолеИзображения.create_image(Интервал + 187, Интервал, anchor=NW, image=
66                                   Изображение)
66     Окновыбора.grab_release()
67     Окновыбора.destroy()
68
69 def сохранитьРезультатРаспознания():
70     global Путь
71     Путь = filedialog.asksaveasfile(mode='w', filetypes=[("Изображения", "*.
72     png;*.jpg;*.jpeg")])
73     Изображение = Image.open('output_image.png')
74     Изображение.save(Путь.name)
75     Изображение.close()
76
77
78 def обучать():
79     ОкновыбораНабора = Toplevel()
80     ОкновыбораНабора.title("Выберите обучающий набор")
81     ОкновыбораНабора.geometry("300x70")
82     СписокНаборов = []
83     Список = загрузитьСписокНаборов()
84     for Элемент in Список:
85         СписокНаборов.append(Элемент)
86     СписокНаборов.append('Новый набор обучения')
87     global ВыбранныйЭлементСписка
88     ВыбранныйЭлементСписка = StringVar(value=СписокНаборов[0])
89     Список = Combobox(ОкновыбораНабора, values=СписокНаборов, textvariable=
90                         ВыбранныйЭлементСписка)
91     Список.pack(anchor=NW, padx=6, pady=6, fill='x')
92     КнопкаПодтверждения = Button(ОкновыбораНабора, text="Подтвердить",
93                                   command=lambda: подтвердить(ОкновыбораНабора))
94     КнопкаПодтверждения.pack(anchor=NW, padx=6, pady=6)
95     ОкновыбораНабора.grab_set()
96
95 def подтвердить(Окновыбора):

```

```

96     global Путь
97     global Данные
98     global Размер
99     global НазваниеНабора
100    ОкноВыбора.grab_release()
101    ОкноВыбора.destroy()
102    НазваниеНабора = ВыбранныйЭлементСписка.get()
103    if НазваниеНабора == 'Новый набор обучения':
104        Путь = filedialog.askopenfilename(filetypes=[("Набор обучения", "*.
105            txt;*.json")])
106        Данные = []
107        with open(Путь, "r") as Файл:
108            for Список in Файл:
109                Данные.append(Список.split())
110        Размер = []
111        Размер.append(len(Данные))
112        Размер.append(len(Данные[0]))
113        ОкноНазвания = Toplevel()
114        ОкноНазвания.title("Введите имя модели")
115        ОкноНазвания.geometry("300x70")
116        global ИмяМодели
117        ИмяМодели = StringVar()
118        ПолеВвода = Entry(ОкноНазвания, textvariable=ИмяМодели)
119        ПолеВвода.insert(0, 'Имя модели')
120        ПолеВвода.pack(anchor=NW, padx=6, pady=6, fill='x')
121        КнопкаОтправки = Button(ОкноНазвания, text="Отправить", command=
122            lambda: отправить(ОкноНазвания))
123        КнопкаОтправки.pack(anchor=NW, padx=6, pady=6)
124        ОкноНазвания.grab_set()
125    else:
126        (Данные, Размер) = загрузитьНабор(НазваниеНабора)
127        АнфисаТренировать(Данные, Размер, НазваниеНабора)
128        сохранить(НазваниеНабора, Данные)
129
130    def отправить(ОкноНазвания):
131        ОкноНазвания.grab_release()
132        АнфисаТренировать(Данные, Размер, ИмяМодели.get())
133        сохранить(ИмяМодели.get(), Данные)
134        ОкноНазвания.destroy()
135
136    КнопкаЗагрузкиИзображения = Button(ПолеИзображения, text=f"Загрузить
137        изображение", command=загрузитьИзображение)
138    ПолеИзображения.create_window(Интервал, Интервал, anchor=NW, window=
139        КнопкаЗагрузкиИзображения)
140
141    КнопкаНачалаРаспознания = Button(ПолеИзображения, text=f"Распознать", command=
142        распознать)
143    ПолеИзображения.create_window(Интервал, (НижняяГраница - ВысотаКнопки -
144        Интервал), anchor=NW, window=КнопкаНачалаРаспознания)
145    КнопкаНачалаОбучения = Button(ПолеИзображения, text=f"Обучать", command=
146        обучать)
147    ПолеИзображения.create_window(Интервал + 74, (НижняяГраница - ВысотаКнопки -
148        Интервал), anchor=NW, window=КнопкаНачалаОбучения)

```

```

142
143 КнопкаСохраненияДанныхРаспознания = Button(ПолеИзображения, text="Сохранить
144     результат распознания", command=сохранитьРезультатРаспознания)
145 ПолеИзображения.create_window(ПраваяГраница - Интервал - 200, Интервал,
146     anchor=NW, window=КнопкаСохраненияДанныхРаспознания)
147
148 Окно.mainloop()

```

ANFIS.py

```

1 import numpy as НП
2 from PIL import Image
3 import anfis
4 import anfisReady
5 import matlab
6 from random import shuffle as перемешать
7
8 def нормализацияКаналов(ПутьФайла):
9     try:
10         Изображение = Image.open(ПутьФайла)
11         Изображение = Изображение.resize((720, 720), Image.LANCZOS)
12     except IOError:
13         print("Ошибка при чтении файла изображения.")
14         return None
15     МассивИзображения = НП.array(Изображение)
16     КрасныйКанал = МассивИзображения[:, :, 0].flatten() / 255.0
17     ЗелёныйКанал = МассивИзображения[:, :, 1].flatten() / 255.0
18     СинийКанал = МассивИзображения[:, :, 2].flatten() / 255.0
19     return
20
21 def АнфисаРаспознать(Путь, ИмяМодели):
22     Анфиса = anfisReady.initialize()
23     Изображение = Image.open(Путь)
24     Изображение = Изображение.resize((720, 720), Image.LANCZOS)
25     МассивИзображения = НП.array(Изображение)
26     КрасныйКанал = МассивИзображения[:, :, 0].flatten() / 255.0
27     ЗелёныйКанал = МассивИзображения[:, :, 1].flatten() / 255.0
28     СинийКанал = МассивИзображения[:, :, 2].flatten() / 255.0
29     ОбучающийНабор = matlab.double(КрасныйКанал.tolist() + ЗелёныйКанал.
30                                         tolist() + СинийКанал.tolist(),
31                                         size=(518400, 3))
32     Результат = Анфиса.AnfisReady(ОбучающийНабор, ИмяМодели)
33     Мaska = НП.empty((720, 720, 4), dtype=НП.uint8)
34     for i in range(720):
35         for j in range(720):
36             if Результат[i * 720 + j][0] > 0:
37                 Мaska[i][j][0] = НП.uint8(Результат[i * 720 + j][0] * 255)
38                 Мaska[i][j][1] = НП.uint8(Результат[i * 720 + j][0] * 255)
39                 Мaska[i][j][2] = НП.uint8(Результат[i * 720 + j][0] * 255)
40             else:
41                 Мaska[i][j][0] = 0
42                 Мaska[i][j][1] = 0
43                 Мaska[i][j][2] = 0
44                 Мaska[i][j][3] = 255
45     Файл = Image.fromarray(Мaska, mode='RGBA')

```

```

45     Файл.save( 'output_image.png' )
46     Файл.close()
47     return Image.fromarray(Маска)
48
49 def АнфисаТренировать(Набор, Размер, ИмяМодели):
50     try:
51         Анфиса = anfis.initialize()
52     except Exception as e:
53         print('Ошибка инициализации пакета Анфисы:\n{}'.format(e))
54         exit(1)
55     try:
56         перемешать(Набор)
57         print(Набор)
58         Анфиса.Anfis(matlab.double(НП.array(Набор, НП.double).flatten() .
59             tolist(), size=Размер), ИмяМодели)
60     except Exception as e:
61         print('Ошибка во время обучения:\n{}'.format(e))
62     Анфиса.terminate()
63     return

```

DB.py

```

1 import sqlite3
2
3
4
5 def сохранить(ИмяМодели, Набор):
6     Подключение = sqlite3.connect('Based_data.db')
7     Курсор = Подключение.cursor()
8     Курсор.execute(f"SELECT EXISTS(SELECT * FROM Готовые_данные WHERE
9         Адрес_файла = ?)", (ИмяМодели, ))
10    НаличиеМодели = Курсор.fetchone()[0]
11    if not НаличиеМодели:
12        Курсор.execute(f"INSERT INTO Готовые_данные (Адрес_файла) VALUES (?)"
13                      , (ИмяМодели, ))
14        Подключение.commit()
15    Курсор.execute(f"SELECT * FROM Готовые_данные WHERE Адрес_файла = ?",
16                  (ИмяМодели, ))
17    ИДМодели = Курсор.fetchone()[0]
18    for И in range(len(Набор)):
19        Курсор.execute(f"SELECT EXISTS(SELECT * FROM Тренировочный_набор
20            WHERE Красный_канал = ? AND Синий_канал = ? AND Зелёный_канал = ?
21            AND Выходные_данные= ?)", (Набор[И][0], Набор[И][1], Набор[И][2],
22            Набор[И][3]))
23        НаличиеНабора = Курсор.fetchone()[0]
24        if not НаличиеНабора:
25            Курсор.execute(f"INSERT INTO Тренировочный_набор (Красный_канал,
26                            Синий_канал, Зелёный_канал, Выходные_данные) VALUES (?, ?, ?, ?
27                            )", (Набор[И][0], Набор[И][1], Набор[И][2], Набор[И][3]))
28            Подключение.commit()
29            Курсор.execute(f"SELECT seq FROM sqlite_sequence WHERE name= ?",
30                          ('Тренировочный_набор', ))
31        Курсор.execute(f"SELECT Ид_тренировочного_набора FROM
32            Тренировочный_набор WHERE Красный_канал = ? AND Синий_канал = ?"

```

```

        AND Зелёный_канал = ? AND Выходные_данные= ?" , (Набор[И][0], Набор
        [И][1], Набор[И][2], Набор[И][3]))
23 ИДНабора = Курсор.fetchone()[0]
24 Курсор.execute(f"SELECT EXISTS(SELECT * FROM Модель WHERE Имя_модели
        = ? AND ИД_Тренировочного_набора = ? AND ИД_Готовых_данных = ?)" ,
        (ИмяМодели, ИДНабора, ИДМодели))
25 НаличиеМодели = Курсор.fetchone()[0]
26 if not НаличиеМодели:
27     Курсор.execute(f"INSERT INTO Модель (Имя_модели,
        ИД_Тренировочного_набора, ИД_Готовых_данных) VALUES (?, ?, ?)"
        , (ИмяМодели, ИДНабора, ИДМодели))
28     Подключение.commit()
29     Подключение.close()
30
31 def загрузитьСписокНаборов():
32     Подключение = sqlite3.connect('Based_data.db')
33     Курсор = Подключение.cursor()
34     Курсор.execute(f"SELECT Адрес_файла FROM Готовые_данные")
35     Список = Курсор.fetchall()
36     for A in range(len(Список)):
37         Список[A] = Список[A][0]
38     Подключение.close()
39     return Список
40
41 def загрузитьНабор(Название):
42     Подключение = sqlite3.connect('Based_data.db')
43     Курсор = Подключение.cursor()
44     Курсор.execute(f"SELECT ИД_Тренировочного_набора FROM Модель WHERE
        Имя_модели = ?" , (Название,))
45     Список = Курсор.fetchall()
46     for И in range(len(Список)):
47         Курсор.execute(f"SELECT * FROM Тренировочный_набор WHERE
        ИД_тренировочного_набора = ?" , Список[И])
48     Выход = Курсор.fetchone()
49     Список[И] = (Выход[1], Выход[2], Выход[3], Выход[4])
50     Подключение.close()
51     return (Список, (len(Список), 4))

```

Место для диска