МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой, д.т.н., с.н.с.
/ Б.М. Суховилов /
«_____»_____20____ г.

Нейронная сеть. Распознавание математических формул

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.03.2021.031.ВКР

Руководитель, к.т.н., доцент
/ Г.А. Поллак /
«_____»_____2021 г.

Автор
студент группы ЭУ – 402
_____/ А.Д. Максимов /
«_____» _____2021 г.

Нормоконтролер, доцент
_____/Е.А. Конова/
«_____»_____2021 г.

Челябинск 2021

АННОТАЦИЯ

Максимов А.Д. Нейронная сеть. Распознавание математических формул – Челябинск: ЮУрГУ, ЭиУ-402, 53 с., 24 ил., 1 табл., библиогр. список – 15 наим., 1 прил.

В дипломном проекте проведен сравнительный анализ сетевых-приложений, распознающих математические символы. По рассмотренным существующим решениям, сделан вывод, на основе которого построен план реализации десктоп-приложения в определенной темой предметной области.

С помощью библиотек Tesseract, OpenCV, NumPy на языке программирования Python версии 3.7, дистрибутива Anaconda, реализовано десктоп-приложение «Распознавание математических символов», позволяющие пользователям преобразовывать нарисованный символ или загруженное изображение с математическим выражением, в редактируемый текстовый формат.

Сделан экономический расчёт по вычислению затрат на реализацию прикладной программы.

Протестирована и определена точность распознавания символов разработанного приложения.

.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель выпускной квалификационной работы – разработать систему, способную распознавать математические символы и рукописный текст.

Задача – с учётом развития направления нейронных сетей в информационном окружении, достаточно актуальна и интересна, т.к. программное обеспечение систем распознавания или определения некоторых символьных наборов, широко распространяется в общественных организациях, малых и больших бизнесах, в научных и учебных организациях.

Задачи работы:

– выполнить сравнительный анализ аналогов – систем\сервисов распознавания с целью определения наиболее распространенных архитектур, необходимого функционала, принципов;

– рассмотреть характеристики доступных ресурсов для работы с системами искусственного интеллекта;

– определить функционал разрабатываемого десктоп-приложения;

– выбрать тип архитектуры приложения, платформу для разработки, ресурсы обработки и разработать дизайн приложения;

– разработать приложение с целью дальнейшего его апгрейда и поиска бизнес реализации;

– отладить и протестировать разработанное приложение.

Объектом выпускной квалификационной работы является исследование возможности реализации оптического распознавания символов посредством нейросетей. Предмет – разработка настольного приложения, реализующего распознавание математических символов и рукописного текста.

1 СРАВНЕНИЕ И АНАЛИЗ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА, НАПРАВЛЕННЫХ НА РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ

1.1 Теоретическое введение

Специалисты по компьютерным технологиям выделяют в отраслях искусственного интеллекта различные типы систем. Каждый тип определяет собой направление, в котором решаются определенные наборы задач.

Типы систем искусственного интеллекта.

- **логические выводы с использованием нечётких множеств** – тип системы подразумевает логические выводы на основе заранее определенного понимания бытовых, повседневных, технических, логистических задач, а также помогает решить задачи характера, когда нужно отталкиваться от временной ситуации и окружающего момента. Примером могут служить системы умного дома, которые в зависимости от окружающего состояния, меняют настройки характеристик в доме;
- **машинное обучение** – тип систем в большей степени направлен на то, чтобы машина с заранее заданным набором данных самостоятельно обучалась посредством решения нужных человеку задач;
- **эвристический поиск** – тип систем направлен на решение задач путём перебора всех возможных вариантов решения задачи и выбором самого оптимального варианта, при котором конечное состояние системы, в которой решается задача, стане лучше, в сравнении с другими ответами. Примером может быть анализ хорошего хода в любой настольной игре.

Одним из самых известных примеров использования искусственного интеллекта машины является **оптическое распознавание символов** (англ. Optical Character Recognition – OCR). Это особая система, способная преобразовывать образы рукописных символов, отсканированные файлы, изображения, видеофрагменты в текст, воспринимаемый машиной и преобразованный для

восприятия человеком. Эта технология распознавания машиной того, что пишет человек на бумаге. Данная технология применяется во многих отраслях жизни человека, в распознавании огромного количества документов, поиске нужных данных в огромном потоке информации. Технология по сути своей считается «зрением машины». До её появления единственным методом оцифровки бумажных носителей являлась повторная ручная печать текста, что занимало огромное количество времени и человеческих ресурсов. Такие системы строятся на основе искусственных нейронных сетей[2,6].

Искусственная нейронная сеть – это громадный распределенный параллельный процессор, состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и предоставляющий их для последующей обработки. Нейронная сеть сходна с человеческим мозгом с двух точек зрения:

– знания поступают в нейронную сеть из окружающей среды и используются в процессе обучения;

– для накопления знаний применяются связи между нейронами, называемые синоптическими весами.

Для того чтобы добиться высокой производительности, нейронные сети используют множество взаимосвязей между элементарными ячейками вычислений – нейронами. Нейронная сеть обычно реализуется с помощью электронных компонентов или моделируется программой, выполняемой на цифровом компьютере[1].

Структурная составляющая данной области представлена на Рисунок 1 – составляющие ИИ 1.

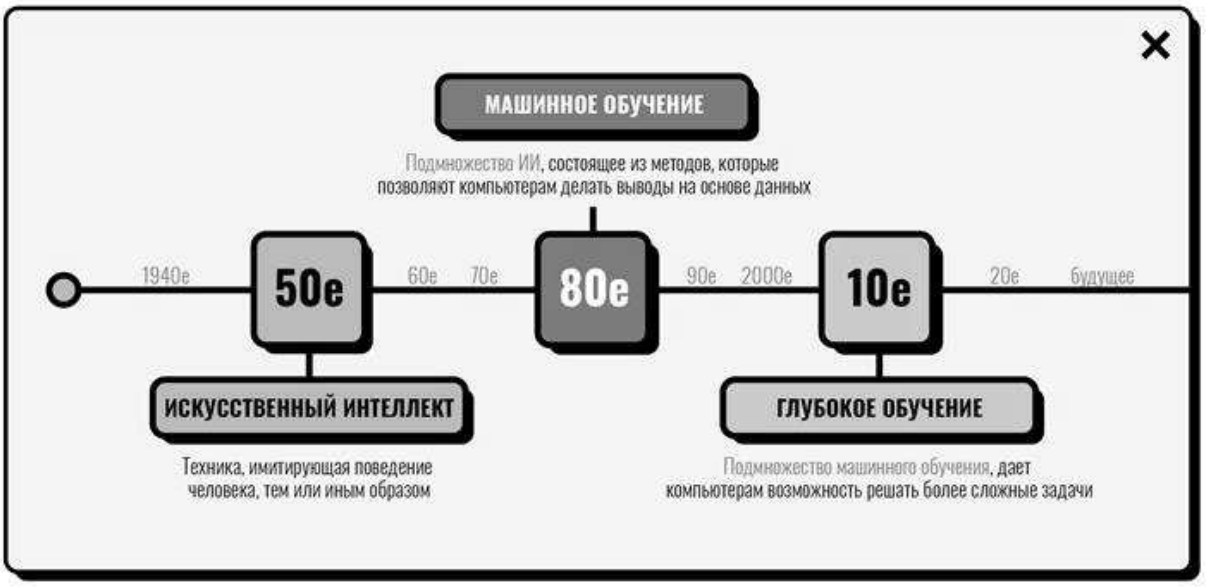

Рисунок 1 – составляющие ИИ

ИИ(AI, Artificial Intelligence) – означает, что компьютер тем или иным образом имитирует поведение человека.

Машинное обучение – подмножество искусственного интеллекта, подразумевающее создание машины, способной самообучаться и находить закономерности в данных, представляемых ей для обучения. Основной принцип машинного обучения – машины получают данные и на их основе обучаются. Такие системы позволяют быстро применять знания, полученные при обучении на больших наборах данных[3,5].

Цель машинного обучения – автоматизировать решение сложных аналитических задач.

Составляющие машинного обучения:

– набор данных – информация с помощью которой обучается машина;

– область применения – прикладное направление обучаемой системы;

– алгоритм – выбор метода решения поставленных задач обучаемой системы.

Виды машинного обучения:

– обучение с учителем – участие учителя рассматривается как набор знаний об окружающей среде, представленный в виде вход-выход. На основе этих знаний учитель формирует и передаёт нейронной сети желаемый отклик. Параметры сети настраиваются с учётом обучающего вектора и сигналов ошибки. Данная форма обучения является обучение на основе коррекции ошибок;

– обучение без учителя – название данного метода обучения подчеркивает отсутствие руководителя, контролирующего процесс настройки весовых коэффициентов. При использовании такого метода обучения не существует маркированных примеров, по которым проводится обучение сети. Существует только некоторая независимая от задач мера качества представления, которой должна научиться нейронная сеть, и свободные параметры сети оптимизируются по отношению к этой мере;

– обучение с подкреплением – такая система обучения предполагает обучение с отложенным подкреплением, где отложенное подкрепление означает сначала признак самого подкрепления, а потом само подкрепление. Это значит, что система получает из внешней среды последовательность сигналов векторов состояния, которые приводят к выполнению заранее запланированных исходов.

Глубокое обучение – подмножество машинного обучения, использующее сложные алгоритмы для обучения модели. Глубокое обучение показывает более высокие показатели точности результатов, если сравнивать с традиционным машинным обучением[13].

1.2 **Сравнение существующих систем в области распознавания символов**

Внедрение систем искусственного интеллекта во все области жизни человека говорит об актуальности темы разработки программных продуктов в этом направлении.

Основные направления развития ИИ:

– компьютерное зрение;

– распознавание речи;
– понимание логики текстов;
– аналитика принятия решений;
– полностью автоматизированные предприятия.

Программные продукты в области компьютерного зрения очень востребованы в разработке и реализации. Возможности их применения затрагивают огромный пласт задач, связанных с визуальными источниками, от определения человека по его внешности, до определения языка текста на изображении.

Эффективность этих программных продуктов или системных инструментов в реализации задач, связанных с автоматизацией и внедрением искусственного интеллекта, с каждым годом растёт.

Распознавание рукописного текста одна из областей в огромном мире программных реализаций ИИ в направлении компьютерного зрения. Рынок информационных продуктов имеет огромный набор инструментов, начиная с голого кода нейронной сети, которую можно создать, обучить и использовать благодаря открытым сервисам корпорации Google, заканчивая полными программными продуктами коммерческого плана, такими как ABBYY Fine Reader 14 и Readiris. Реализация полностью зависит от конечной тематики использования.

Огромное количество мировых языков, надписей систематического назначения, математических выражений пользуются спросом в реализации. Математические формулы и выражения считаются достаточно сложной тематикой в мире компьютерного зрения. Помимо того, что нужно распознать не только отдельные символы, но и саму структуру математической формулы.

Определение рукописного текста с изображения считается одним из основных направлений в компьютерном зрении. Проблема реализации этого процесса состоит во-первых: в реализации системы нейронных сетей, а во-

вторых: в форматировании исходного изображения для повышения точности распознавания нейронной сетью.

Для того чтобы провести анализ сравнения систем ИИ, требуется рассмотреть не только готовые системы, но, и инструменты, и сервисы, без которых невозможна адекватная разработка программного обеспечения и конечного продукта в сфере компьютерного зрения.

В качестве объектов сравнения, выбраны следующие системы.

Google Docs

OCR-система по распознаванию и поиску отдельных символов, нарисованных в реальном времени. Интерфейс сервиса представлен на Рисунок 2.

Рисунок 2 – Google Docs

Система встроена в сервисы Google Docs, способна распознать не только математические символы.

Detexify

Онлайн-сервис, посимвольного распознавания. Интерфейс сервиса представлен на Рисунок 3.

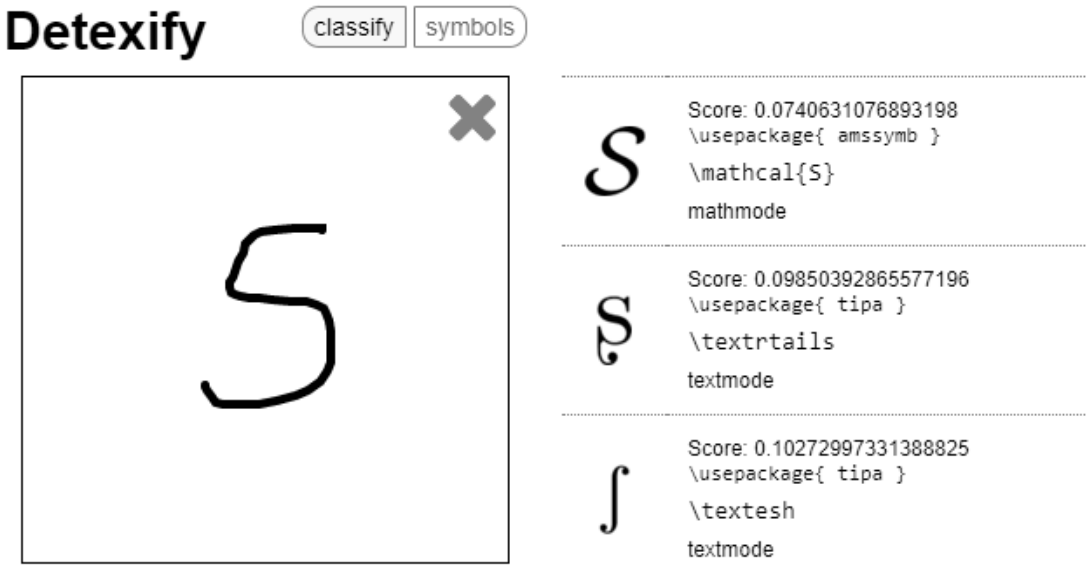

Рисунок 3 – Detexify

Сервис Detexify предназначен для распознавания единичных математических символов и представления их в формате LaTeX.

OmniPage

Программа на основе OCR-системы для оптического распознавания символов, поддерживает более 120 различных языков и имеет возможность распознавания математических символов. Интерфейс приложения представлен на Рисунок 4.

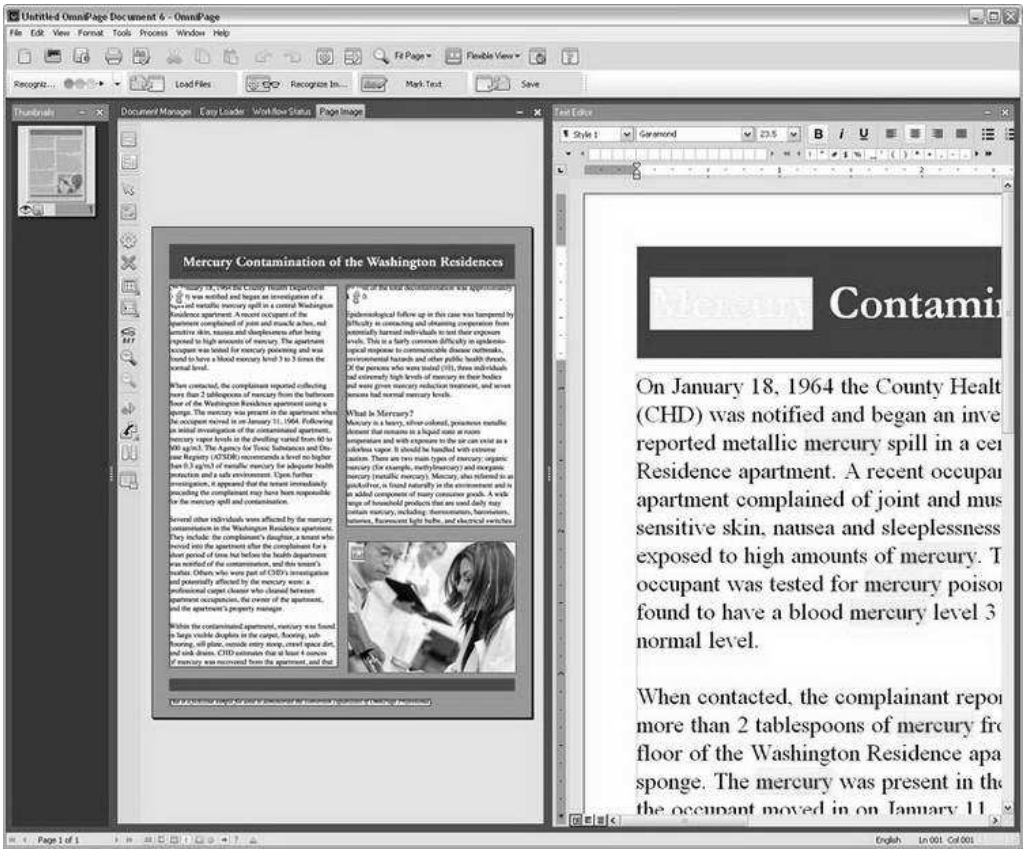

Рисунок 4 – OmniPage

Приложение предназначено для распознавания фрагментов текста и предоставляет возможность чтения Pdf файлов.

Shapecatcher

Онлайн-сервис, посимвольного распознавания. Интерфейс сервиса представлен на Рисунок 5.

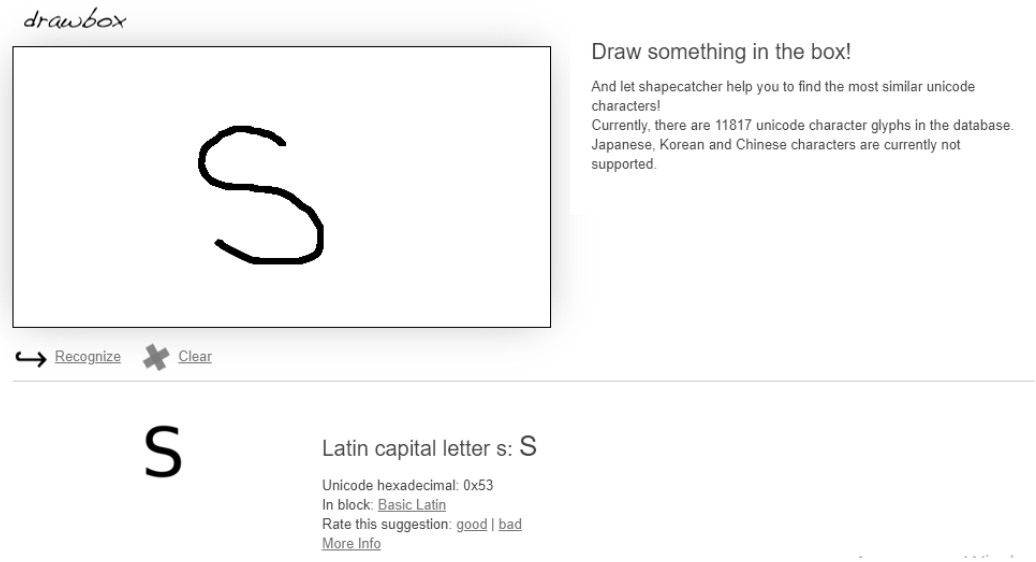

Рисунок 5 – Shapecatcher

Приложение предназначено для распознавания отдельных математических символов и представления их в формате Unicode.

Free-OCR

Веб-сервис оптического распознавания символов. Интерфейс сервиса представлен на Рисунок 6.

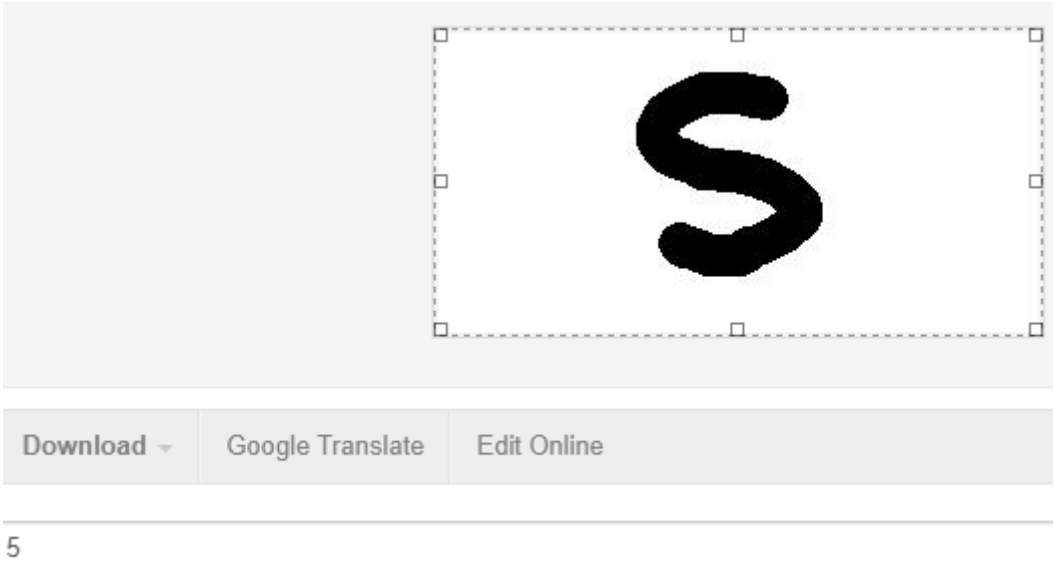

Рисунок 6 – Free-OCR

Сервис предназначен для распознавания отдельных математических символов в формате online.

Задачи сравнения существующих приложений:

– определить критерии сравнения выбранных систем и приложений;

– провести оценку систем и приложений по данным критериям;

– выявить сильные и слабые стороны реализации анализируемых систем и приложений;

– сделать выводы, на основе чего взять лучшие черты каждого из сервисов и представить их в своём десктоп-приложении.

Критерии сравнения характеристик систем и приложений по распознаванию математических формул и символов:

– возможность распознавания нескольких символов;

– возможность распознавания дробных выражений и степеней;

– доступность и свободное использование;

– реализации offline;

– удобность интерфейса и функционал.

Условные обозначение:

– «+» – реализация решения удовлетворяет критерию;

– «–» – реализация решения не удовлетворяет критерию.

Результаты исследования основных характеристик систем и приложений распознавания математических символов приведены в Таблица 1.

Таблица 1 – Сравнение существующих систем

Название системы или приложения	Возможность распознавания нескольких символов	Возможность распознавания дробных выражений и степеней	Доступность и свободное использование	Реализации offline	Удобность интерфейса и функционал
Google Docs	–	–	+	–	+
Detexify	–	–	+	–	+
OmniPage	+	–	–	+	–
Shapecatcher	–	–	+	–	+
Free-OCR	+	–	+	–	–

После сравнения существующих программных продуктов выявлены факты.

Формат реализации приложений. Практически все системы реализованы в online режиме, т.е. веб-сервисы. Вид реализации не работает без подключения к сети интернет и является не автономным.

Из рассмотренных приложений четыре из пяти являются общедоступными и бесплатными в своём использовании.

Возможность распознавать множество символов отмечена только в 2-х приложениях.

Сложность некоторых распознаваемых формул весьма велика для распознавания.

Удобность интерфейса и функционал в рассмотренных сервисах и приложениях достаточно просты и не выделяются. Весь функционал в основном

распределяется на распознавание нарисованного символа или же распознавание по уже готовому изображению.

Выбрав все сильные стороны, обозначив отрицательные моменты, следует обратить внимание на функциональность программы в таких направлениях как:

– возможность распознавать символы в зависимости от их количества на исходном изображении;

– возможность распознавания не только загружаемых изображений, но и рисуемых пользователем в реальном времени;

– интерфейс не должен быть очень понятным и функциональным;

– следует обратить внимание на возможность распознавания сложных математических символов. Для этого потребуется не только программно распознать символы на изображении, но и определить его структурное положение относительно других символов. По программным продуктам, которые были представлены в сравнении выше, можно понять, что данная задача является достаточно трудным вопросом, а значит актуальность этой проблемы говорит об актуальности тематики распознавания математических символов.

1.3 **Инструменты и сервисы для создания программного продукта**

На данный момент большое количество разработок и проектов по тематике. От всевозможных датасетов, содержащих наборы изображений для нейронной сети, до целых библиотек, которые разработаны для непосредственной работы и настройки. Все эти ресурсы не ограничиваются только инструментарием, они дополнены объяснениями и примерами, концепциями разработки нейронных сетей.

Огромную нишу в этой области занимает корпорация Google. Уже не первое десятилетие Google создаёт исследовательские проекты, направленные на полную адаптацию и изучение искусственного интеллекта. Исследовательский проект Google Brain по изучению искусственного интеллекта на основе глубокого обучения или Google-AI – подразделение Google, занимающееся искусственным

интеллектом. Эти подразделения имеют формат открытых исследований в области машинного обучения. Помимо исследовательских работ Google на рынке информационных продуктов присутствует огромное количество открытых ресурсов и инструментов для работы с машинным обучением. Их перечень предоставлен ниже.

TensorFlow – открытая программная библиотека, разработанная компанией Google. Решает задачи построения и обучения нейронных сетей, с отличным достигаемым конечным качеством точности и работоспособности нейросетей[15].

Keras – открытый высокоуровневый фронтэнд глубокого обучения для TensorFlow. Также является надстройкой над такими фреймворками, как Deeplearning4j, TensorFlow и Theano. Keras содержит в себе несколько обучающих датасетов, они уже готовы к работе. Обладает большими возможностями в препроцессинге изображений, текстов и других типов данных. Позволяет очень просто создавать модели нейронных сетей и работать над ними[12].

Theano – библиотека численных вычислений в Python с открытым исходным кодом. Содержит в себе компилятор математических выражений.

Tesseract – очень популярная библиотека для OCR с открытым исходным кодом, разработанная Google, которая позволяет получать высокие результаты точности в распознавании символов. Использует языковые модели и словари. Языковая модель содержит настройки нейронной сети и другие данные обучения[11].

Caffe – является библиотекой с открытым исходным кодом для разработки глубинного обучения. Свёрточная архитектура для извлечения признаков. Поддерживает большое количество типов машинного обучения, нацеленных на классификацию и сегментацию визуальных представлений.

OpenCV – библиотека компьютерного зрения с открытым исходным кодом, предназначенная для анализа и классификации изображений[7].

NumPy – библиотека с открытым исходным кодом. Представляющая возможность хранить данные в многомерных массивах, по сути своей альтернатива MATLAB. Всё его преимущество в простоте использования и адаптивности с другими библиотеками.

Выводы по первому разделу

Проведено сравнение существующих систем распознавания математических символов.

Сделаны выводы на основе которых разрабатывалось приложение.

Описаны и выбраны библиотеки и ресурсы, направленные на разработку приложения.

2 РАЗРАБОТКА ПРИЛОЖЕНИЯ

2.1 **Проектирование программы**

Под технологией оптического распознавания символов подразумевают, программный продукт, который позволяет преобразовывать изображения, документы, фотографии в формат, удобный для редактирования человеком. Таким образом, в проектировании десктоп-приложения следует ориентироваться на удобство использования получаемого преобразованного ответа. Достигается это доступным и лаконичным пользовательским интерфейсом.

Чтобы быстро считывать информацию, необходимо, чтобы она была представлена как можно проще для восприятия. Это помогает сосредоточить внимание пользователя на выполняемой задаче и позволяет сократить время выполнения операции.

В реализуемом проекте используются кроссплатформенная библиотека Tkinter для разработки графического интерфейса, написанная на языке программирования Tcl. Дополнительно подключено расширение к Tkinter (Tk), называемое Themed Tk (Ttk). Данное расширение улучшает визуальное отображение виджетов, подключенной библиотеки, и позволяет задавать тему интерфейса[8].

2.2 **Выбор языка программирования**

Для разработки информационной системы выбран язык программирования Python версии 3.7. Решение обусловлено такими параметрами как:

- простота;
- множество библиотек, подключаемых для разработки в предметной области;
- универсальность;
- Интерпретируемость;
- гибкость.

Ключевым фактором в выборе языка является его расширяемость – возможность подключить все полезные для разработки библиотеки[6]. Язык программирования установлен менеджером пакетов для программного обеспечения «Conda». Менеджер пакетов облегчает работу с установкой, обновлением, удалением пакетов Pythona. Используется «Anaconda» – дистрибутив Python, разработанный Continuum Analytics[9]. Он представляет все необходимые решения задач по анализу и обработке данных. Это набор таких систем, как Scipy, NumPy, Pandas. В качестве интегрированной среды разработки выбрана среда PyCharm. Подключение всех используемых инструментов и библиотек представлено на Листинг А.1.

2.3 **Описание используемых технологий**

Оптическое распознавание символов как процесс обычно состоит из нескольких подпроцессов, которые должны выполняться с максимальной точностью:

- предварительная обработка изображения;
- текстовая локализация;
- сегментация текста;
- распознавание текста;
- постобработка.

Приведённый список подпроцессов имеет возможность изменяться и модифицироваться в зависимости от поставленных задач. В программной реализации оптического распознавания символов основная цель – идентифицировать и фиксировать все уникальные слова с использованием разных языков из письменных текстовых символов.

В течение почти двух десятилетий традиционные системы оптического распознавания символов широко использовались для обеспечения автоматизированного перевода бумажного текста в цифровую форму. Такие системы столкнулись с большими трудностями в распознавании различных

шрифтов и форматов страниц. Любые отклонения в формате распознаваемого объекта приводили к огромному количеству ошибок трансформации информации в цифровую форму.

В механизмах оптического распознавания текста нового поколения стали использовать и применять новейшие исследования в области глубокого обучения, что привело к решению проблем систем распознавания старого поколения. С использованием моделей глубокого обучения и общедоступных наборов данных точность распознавания текста достигла высочайшего уровня. В настоящее время широко используются генерация синтетических данных с различными шрифтами, что позволяет решать проблему распознавания большого количества шрифтов[14].

Для реализации основного вопроса: распознавания математических формул и символов выбрана библиотека Tesseract, написанная на языке программирования C++. Эта библиотека представляет движок распознавания символов, использующий нейронную сеть долгой краткосрочной памяти (англ. Long short-term memory; LSTM). Подключение корневой директории представлено в Листинг А.2.

Оптическое распознавание символов так и остаётся очень сложной задачей, если распознаваемый текст искажён или имеет сложный фон. Реализация оптического распознавания символов в приложении представлена на Рисунок 77.

Рисунок 7 – Процесс оптического распознавания

Входное изображение после предварительной обработки сегментируется посимвольно с помощью библиотеки Tesseract и представляется в виде текстового формата.

Чтобы распознать изображение, содержащее один символ, обычно используется свёрточная нейронная сеть. Текст произвольной длины – это последовательность символов, и такие проблемы решаются с помощью рекуррентной нейронной сети, а LSTM – популярная форма рекуррентной нейронной сети.

Рекуррентная нейронная сеть (англ. Recurrent Neural Networks, RNN) – это нейронная сеть, сохраняющая информацию и имеющая обратные связи. Наличие обратных связей в нейронных сетях оказывает непосредственное влияние на способность таких сетей к обучению и на их производительность. Обратная связь подразумевает использование элементов единичной задержки (обеспечивает задержку входного сигнала на один шаг эталонного времени), что приводит к нелинейному динамическому поведению. Идея RNN в выполнении одной и той же задачи для каждого элемента последовательности, в конченом итоге выход будет завесить от предыдущих вычислений. Получается, что в RNN сетях нейроны обмениваются информацией между собой таким образом, что вместе с входящими данными нейроны также получают информацию о прошлом

состоянии сети, таким образом в сети образуется «память», что позволяет анализировать любые последовательности данных, в которых важно в каком порядке идут значения[1,10]. Пример представлен на Рисунок 88.

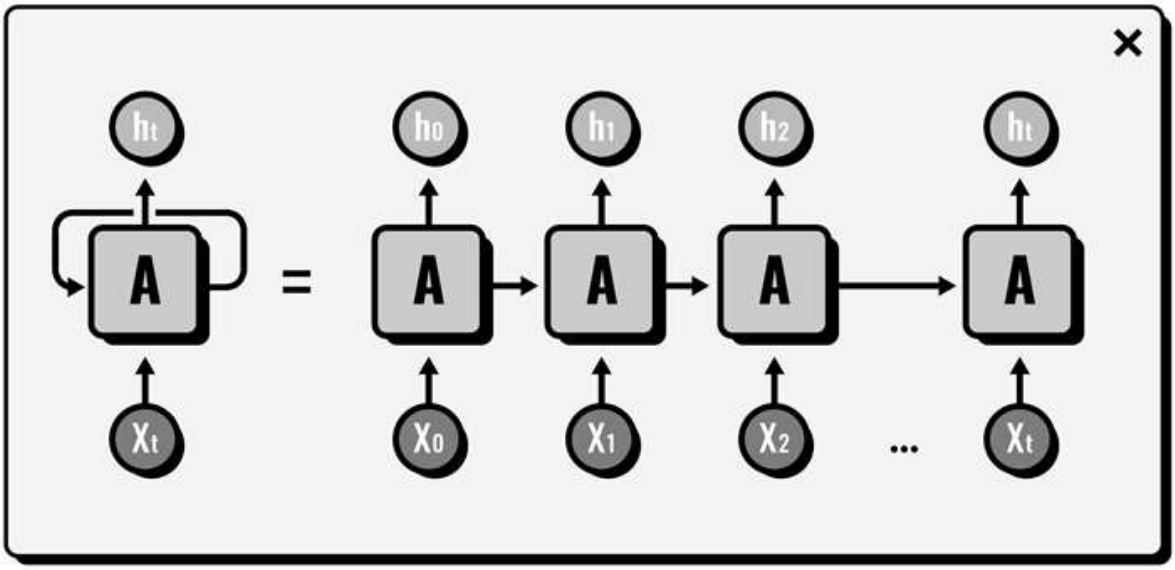

Рисунок 8 – Рекуррентная нейронная сеть

Цикл передачи входных данных из одного внутреннего слоя нейронов X и информации о состоянии предыдущего внутреннего слоя A, на основании чего генерируется ответ h на каждом новом состоянии.

Основная проблема рекуррентной сети - её нейроны хорошо «помнят» недавно полученную информацию, но не имеют возможности надолго сохранить в памяти что-то, что было обработано много циклов назад, какой бы важной та информация ни была.

Нейронная сеть долгой краткосрочной памяти (LSTM) – необычная модификация рекуррентной нейронной сети. В этой нейронной сети решена основная проблема рекуррентной нейронной сети, она может запоминать информацию о состоянии предыдущих внутренних слоёв намного дольше, что позволяет ей иметь больший потенциал.

Этот тип нейронной сеть оборудован системой передачи информации («Ворота») и имеет клеточное состояние, которое и представляет краткосрочную память. «Ворота» определяют, какая информация попадёт в клеточное состояние,

какая сотрется, а какая повлияет на результат на данном шаге. Пример представлен на Рисунок 99.

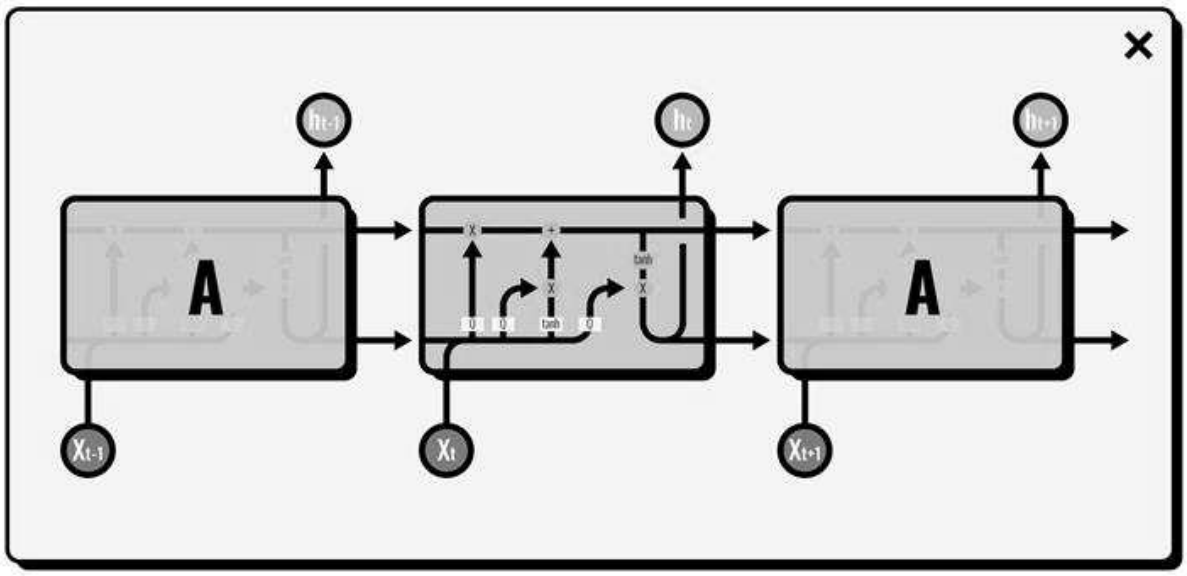

Рисунок 9– Нейронная сеть долгой краткосрочной памяти

Нейроны внутренних слоёв могут считывать и изменять состояние ячейки. Передача входные данных внутреннего слоя нейронов X и модифицированная информация прошлого клеточного слоя, переданная «воротами» нужному слою клеточного состояния A, после чего генерируется ответ h.

Библиотека Tesseract применяется в такой последовательности:

– исходное изображение обрабатывается и приводится к наилучшему формату с помощью библиотеки OpenCV;

– подготовленное изображение посимвольно сегментируется подаётся на движок Tesseract;

– символы распознаются с помощью обученной на языковой модели нейронной сети долгой краткосрочной памяти;

– распознанные символы преобразуются в текстовую строку.

Error! Reference source not found.наглядно показывает принцип работы приложения.

Рисунок 10 – Принцип распознавания

Посимвольное определение и последующая постобработка с выводом информации в текстовый формат с помощью нейронной сети долгой краткосрочной памяти.

2.4 **Предварительная обработка изображения**

Перед применением библиотеки Tesseract нужно подготовить исходное изображение. Форматирование исходного изображения проводится с целью улучшения результата распознавания. Для данных преобразований используется OpenCV – библиотека компьютерного зрения с открытым исходным кодом, предоставляющая набор инструментов для обработки изображений.

Шаги предварительной обработки изображения.

1. Загрузка изображения в программу и применение фильтра градации серого. Результат преобразования представлен на Рисунок 11. Метод реализации представлен в Листинг А.6.

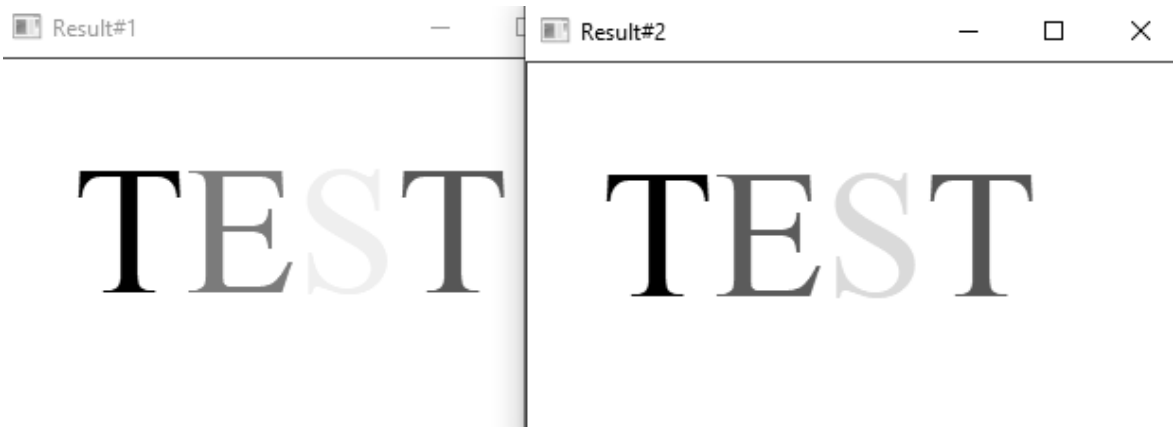

Рисунок 11 – Загрузка и применение фильтра градации серого

Метод применяется для последующего преобразования в чёрно-белый формат и загрузки исходного изображения в программу.

2. Изменение размера исходного изображения. Результат представлен на Рисунок 12. Метод реализации представлен в Листинг А.5.

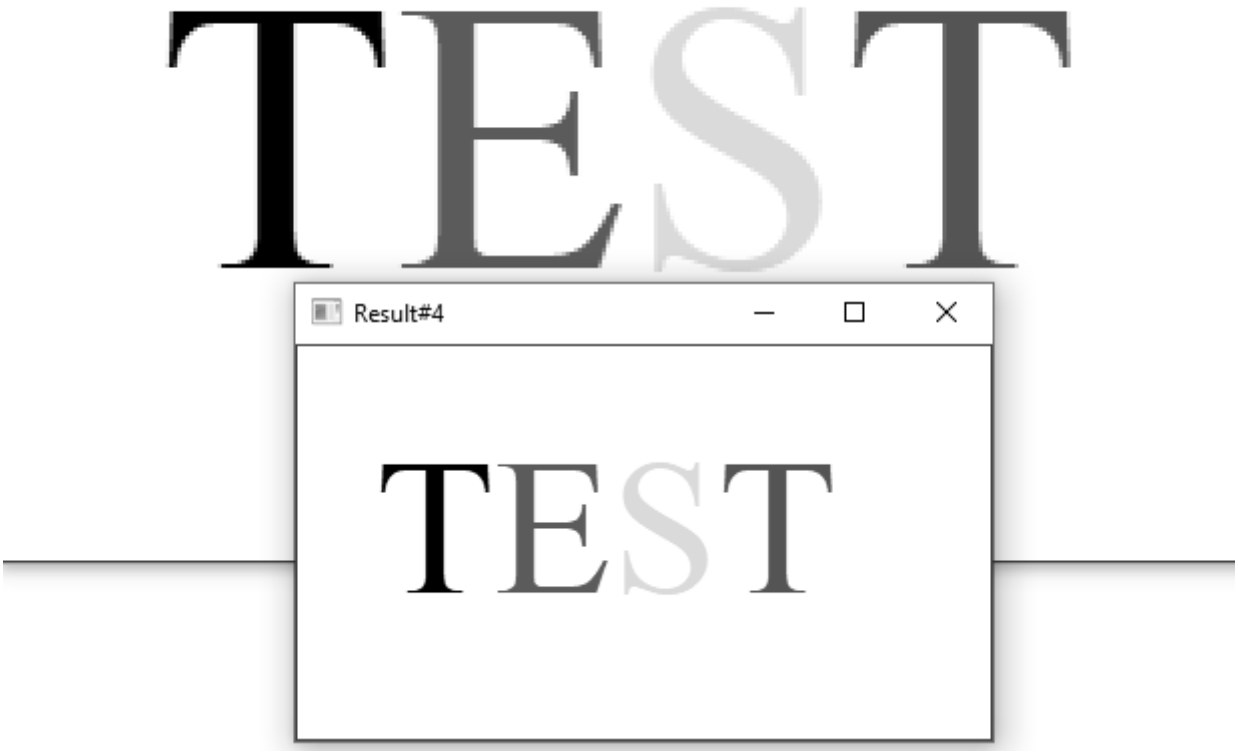

Рисунок 12 – Уменьшение размера изображения

Метод применяется для того, чтобы добиться большей чёткости изображения путём уменьшения изображения. Преобразование помогает добиться

меньшей пикселизации распознаваемых символов, что повышает уровень определения контура каждого символа.

3. Бинаризация изображения в чёрно-белый формат. Результат представлен на Рисунок 13. Метод реализации представлен в Листинг А.8.

Рисунок 13 – Бинаризация изображения

Преобразование используется для установления пороговых значений, где чёрный цвет имеет значение 0, а белый значение 1.

4. Применение размытия по Гауссу. Результат представлен на Рисунок 14. Метод реализации представлен в Листинг А.7.

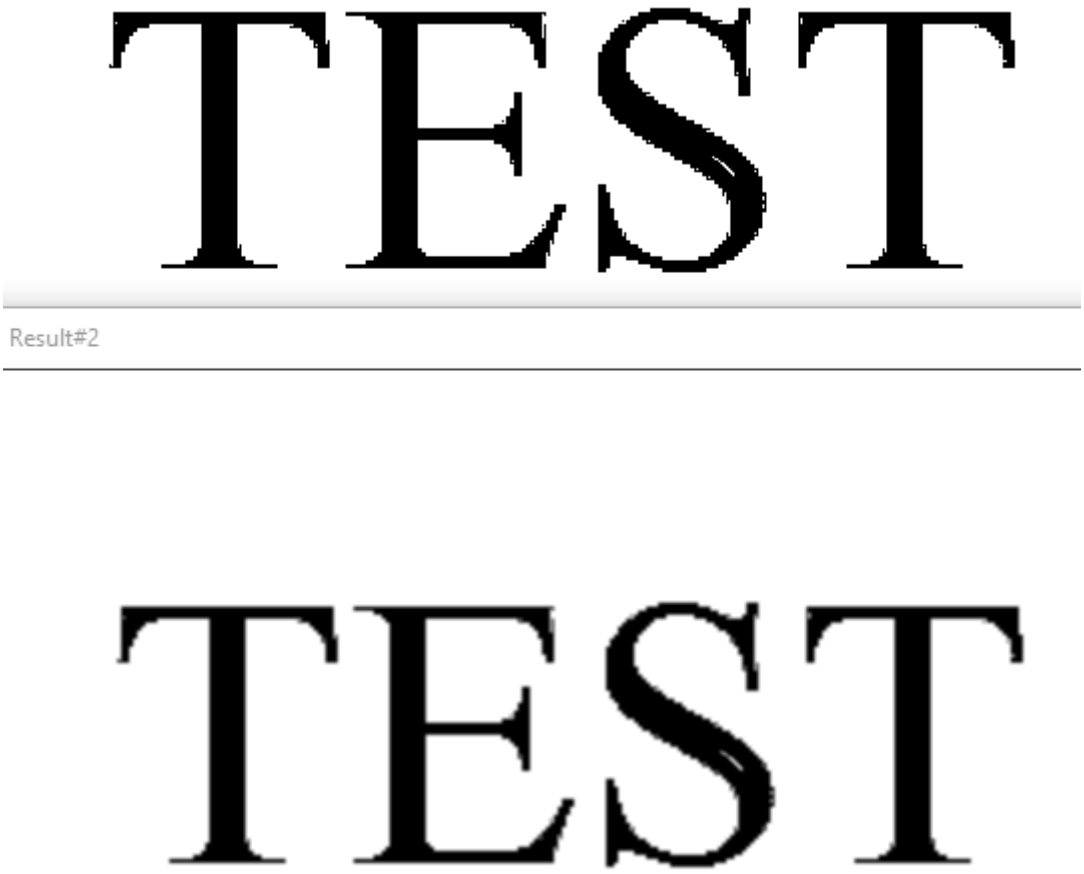

Рисунок 14 – Применение размытия по Гауссу

Применение размытия по Гауссу для избавления от лишних шумов на фоне и сглаживания контура распознаваемых символов.

5. Посимвольная сегментация с помощью цветовой сегментации. Для наглядности на Рисунок 1515 контур каждого символа выделен прямоугольниками. Метод реализации представлен в Листинг А.10.

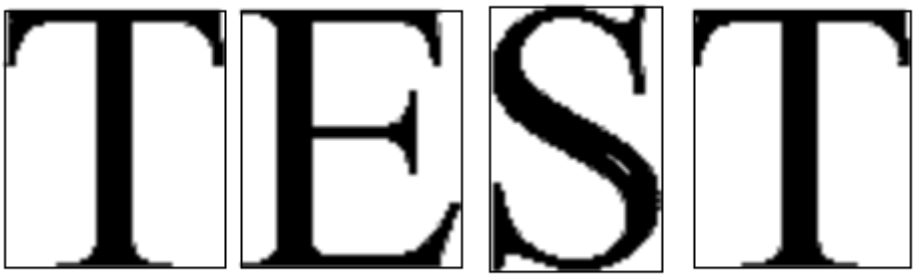

Рисунок 15– Посимвольное распознавание по контуру

Посимвольная сегментация по контуру каждого символа изображения используется для дальнейшего распознавания отсегментированных символов.

Предварительная обработка изображения происходит в фоновом режиме, после чего обработанное и отсегментированное изображение поступает на распознавание библиотекой Tesseract. Реализация представлена в Листинг А.14.

2.5 **Разработка языковой модели распознавание**

2.5.1 Создание языковой модели

Точность и объем распознаваемых символов полностью зависит от языковой модели, разрабатываемой для сети долгой краткосрочной памяти. Языковая модель – файл, содержащий в себе данные обучения и значения параметров модели нейронной сети.

Подготовка данных для обучения нейронной сети начинается с составления подробного набора подходящих под задачу символов. Составляется карта символов, где карта символов – это изображение формата .tif, с символами, составляющими данные для обучения. На Рисунок 16 представлена карта символов, использованная в создании набора данных для обучения нейронной сети.

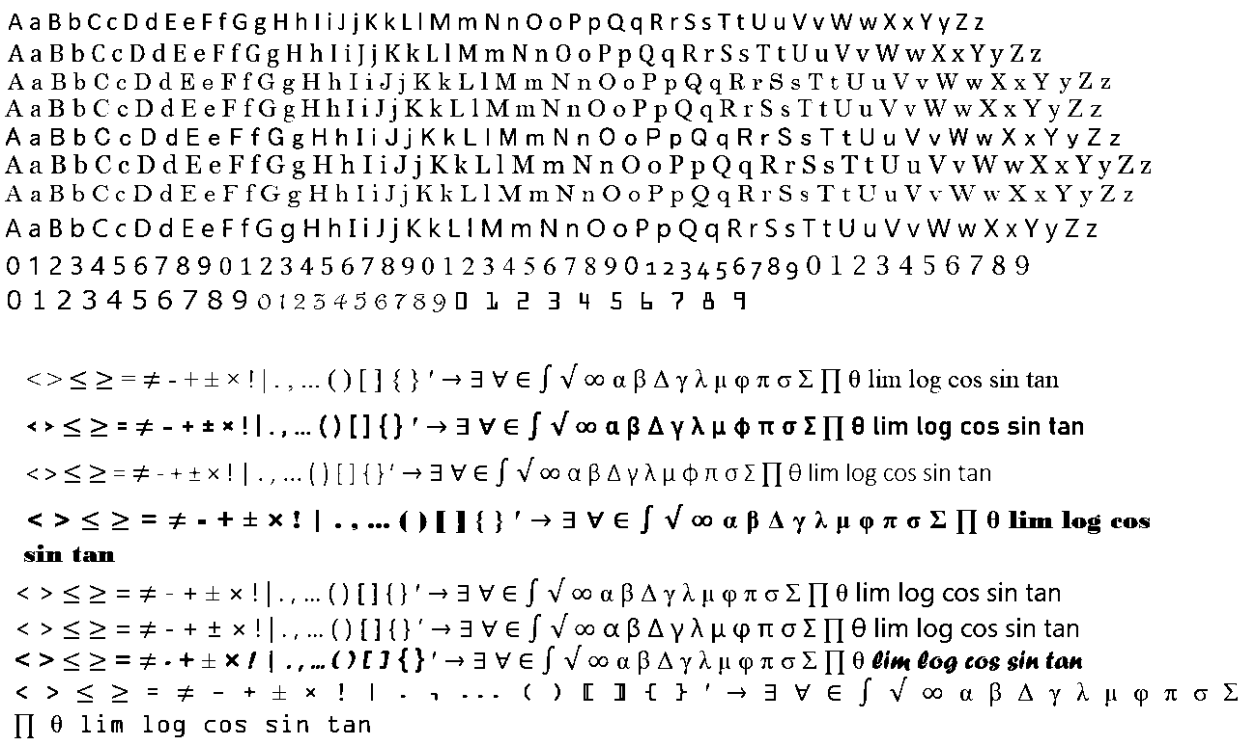

Рисунок 16 – Карта символов

Для того чтобы сегментировать данные по каждому из представленных символов следует создать box-файл, содержащий информацию по каждому символу из карты символов. Создание box-файла производится путём использования утилит «makebox» и «batch.nochop» библиотеки Tesseract, расположенных в папке, куда была установлена библиотека Tesseract. Все преобразования над картой символов с помощью утилит выполняются в командной строке Microsoft Windows 10. Пример работы с консолью представлен на Рисунок 17.

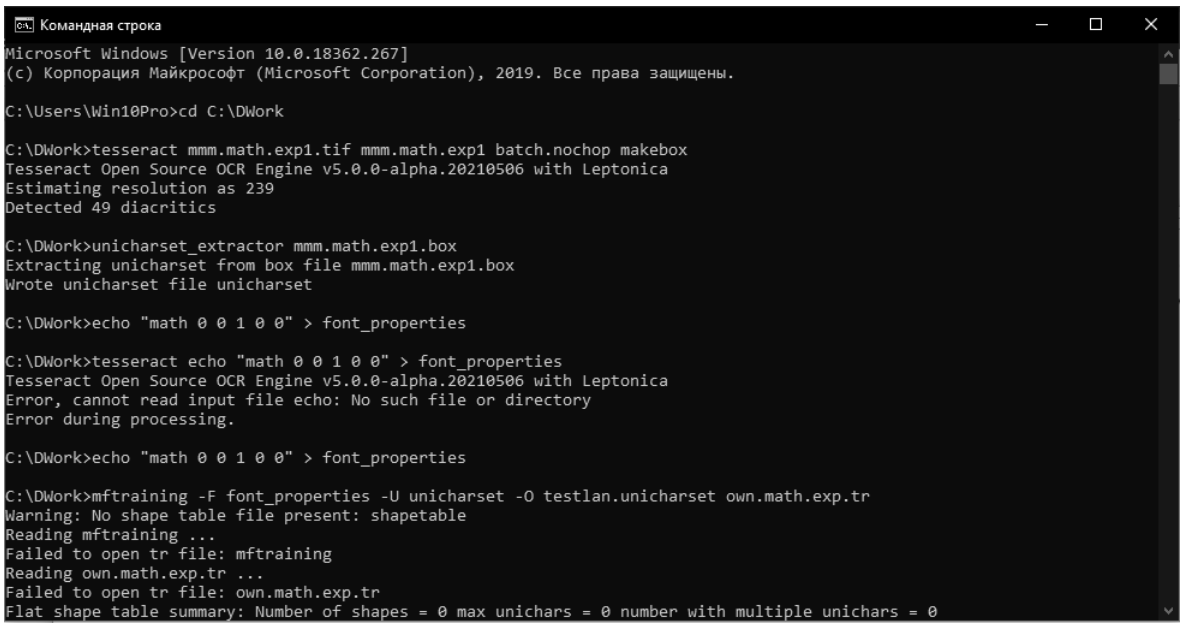

Рисунок 17 – Работа с Tesseract в консоли

Для настройки box-файла применяется программа jTessBoxEditor. Функционал программного продукта направлен на работу с box-файлами. В jTessBoxEditor выполняется настройка сегментации каждого символа на символьной карте, выставление категории каждого символа. На Рисунок 188 продемонстрирована настройка сегментации символов карты символов.

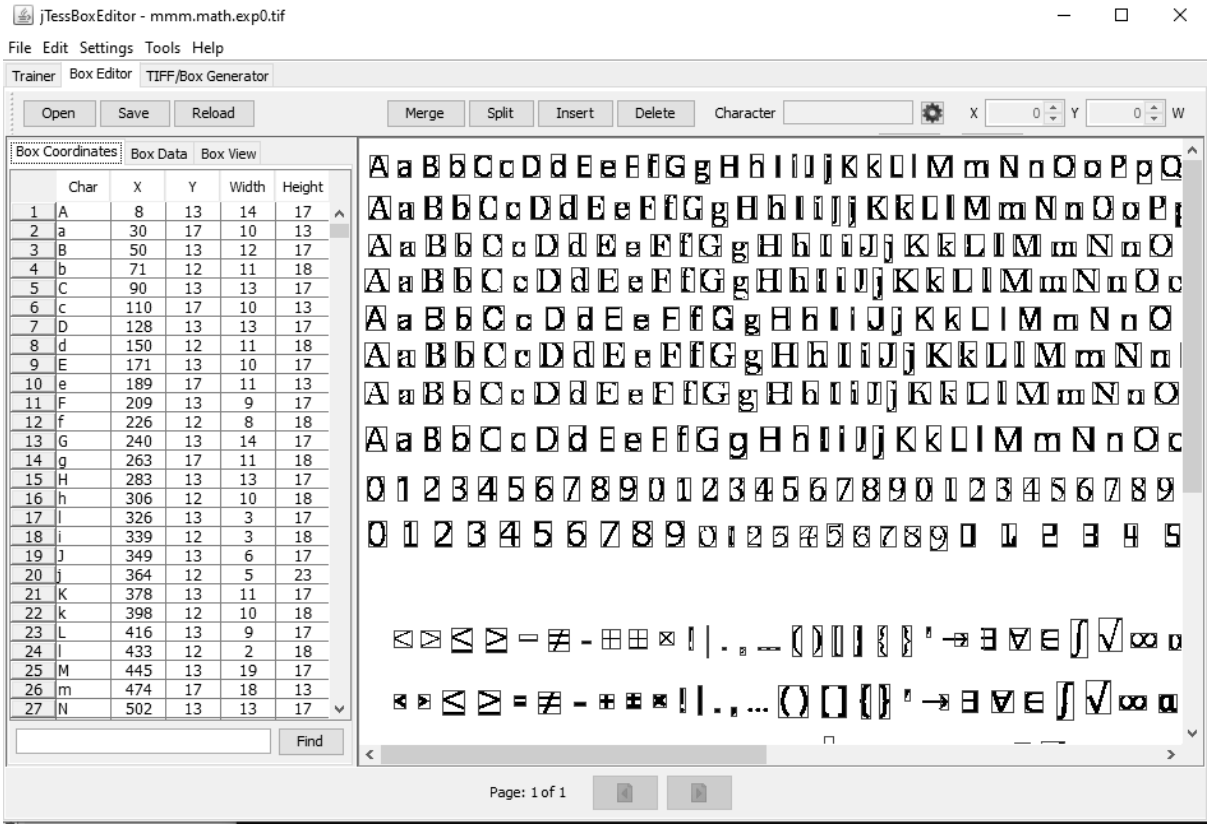

Рисунок 18– Настройка box'ов символов в jTessBoxEditor

Карта символов содержит 542 разновидности символов:

– 416 разновидностей букв английского алфавита;

– 80 разновидностей цифр;

– 46 разновидностей математических символов, включающих набор символов греческого алфавита.

Обучение нейронной сети библиотеки Tesseract проводится путём использования выполнения надстроек утилитами Tesseract. Надстройки производимые для обучения нейронной сети:

– с помощью утилиты train.box в командной строке запустить обучение. Обучение построено на настройках отсегментированных символов box-файла. Файл формата TR в директории карты символов и box-файла, является файлом надстройки обученной нейронной сети;

– после чего с помощью утилиты «unicharset_extractor» задаётся файл набора символов unicharset_extractor.txt, где указывается информация по символам, которые являются набором данных обученной нейронной сети;

– описать стиль шрифта, путём создания в командной строке файла формата «font_properties»;

– создать файл применения описанного шрифта к файлу надстройки обучения на box-файле в командной строке, с помощью утилиты «mftraining»;

– создать в командной строке конченый файл обучения нейронной сети, используя утилиту «cntraining»;

– скомпилировать все полученные файлы в языковую модель утилитой «combine_tessdata».

Скомпилированный файл – это языковая модель, содержащая информацию по набору данных обученной нейронной сети, которая используется движком библиотеки уже для непосредственного распознавания символов с обработанных библиотекой изображений.

2.5.2 Точность распознавания нейронной сети библиотеки Tesseract на созданной языковой модели

Показатели точности программ оптического распознавания достаточно разнятся из-за большой зависимости от качества распознаваемого изображения, поэтому принято рассчитывать точность распознавания по обработанным и готовым источникам.

Расчёт точности производится по сравнению количества символов в наборе данных к количеству символов, которые программа распознала.

Формула по расчёту точности распознавания (1):

$$\text{Точность} = \frac{\text{Количество правильно распознаных символов}}{\text{Количество символов набора данных}} \cdot \mathbf{100\%}, \quad (1)$$

Подсчёт точности распознавания в разработанной прикладной программе после десяти тестовых изображений 52 букв английского алфавита и 47 символов математического формата, отображаемые в различных шрифтах составил:

$$\text{Точность} = \frac{94}{99} \cdot 100\% = 94{,}9\%$$

Где, количество правильно распознанных символов равняется среднему значению десяти проведённых подсчётов тестовых изображений с отображением 52 букв английского алфавита и 47 символов математического формата.

2.6 **Разработка десктоп-приложения**

Диаграмма десктоп-приложения представлена на Рисунок 1919.

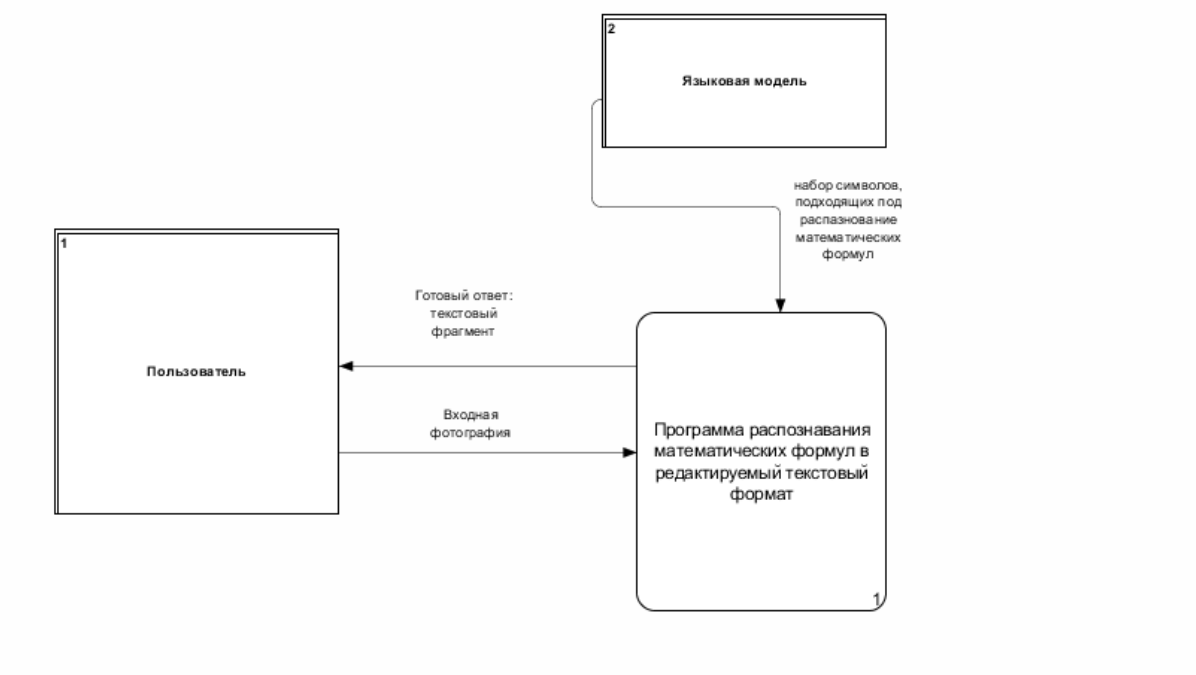

Рисунок 19 – Структура десктоп-приложения

Пользователь загружает входную фотографию в приложение по распознаванию математических символов в текстовый формат. Языковая модель содержит информацию об обученной модели нейронной сети и наборе данных, что позволяет программе распознать исходное изображение.

Строение внутреннего слоя прикладного приложения представлено на Рисунок 2020.

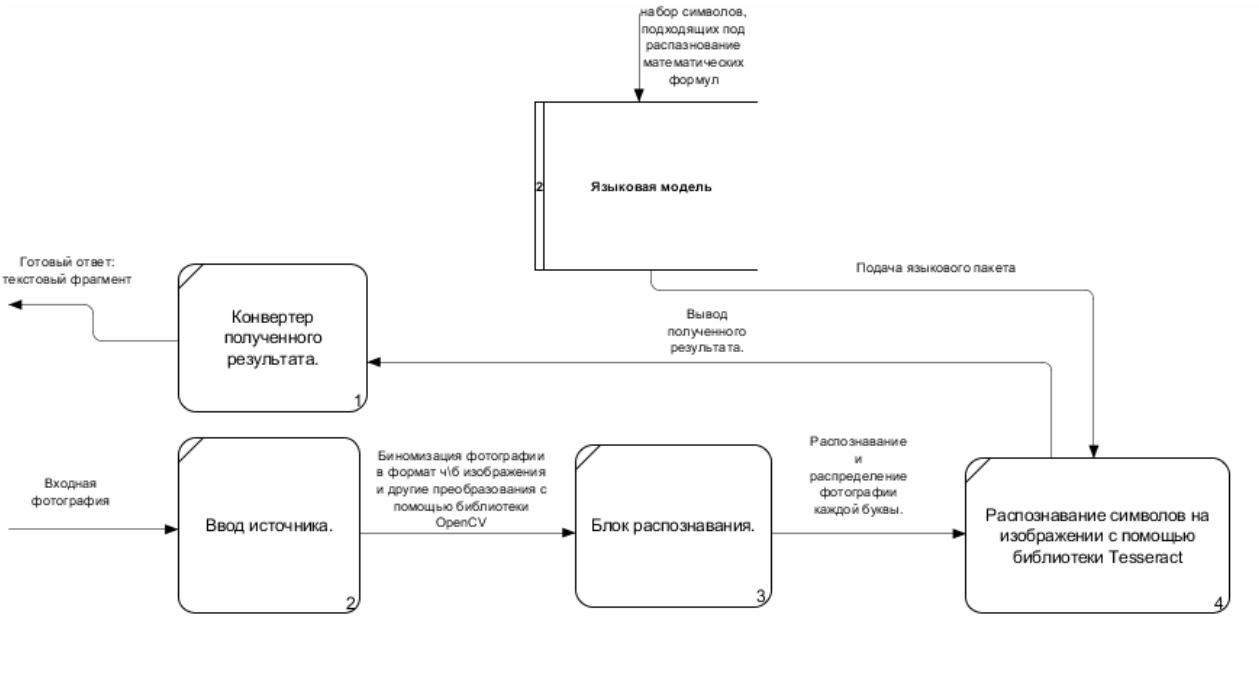

Рисунок 20 – Схема внутреннего слоя программы

Входное изображение поступает в блок распознавания и с помощью библиотеки OpenCV форматируется до нужного формата, после чего отформатированное изображение распознается с помощью библиотеки Tesseract и подаётся пользователю в текстовом виде.

Исходя из представленных схем и перечисленных выше библиотек, разработано десктоп-приложение, способное распознавать математические символы с загружаемых в программу изображений или же рисуемых в реальном времени математических символов. Подключенные виджеты и тема представлены на Листинг А.3.

Интерфейс десктоп-приложения представляет из себя форму с графическим холстом для рисования в реальном времени и файловый диалог, для подгрузки отсканированных документов или изображений. Интерфейс представлен на Рисунок 2121

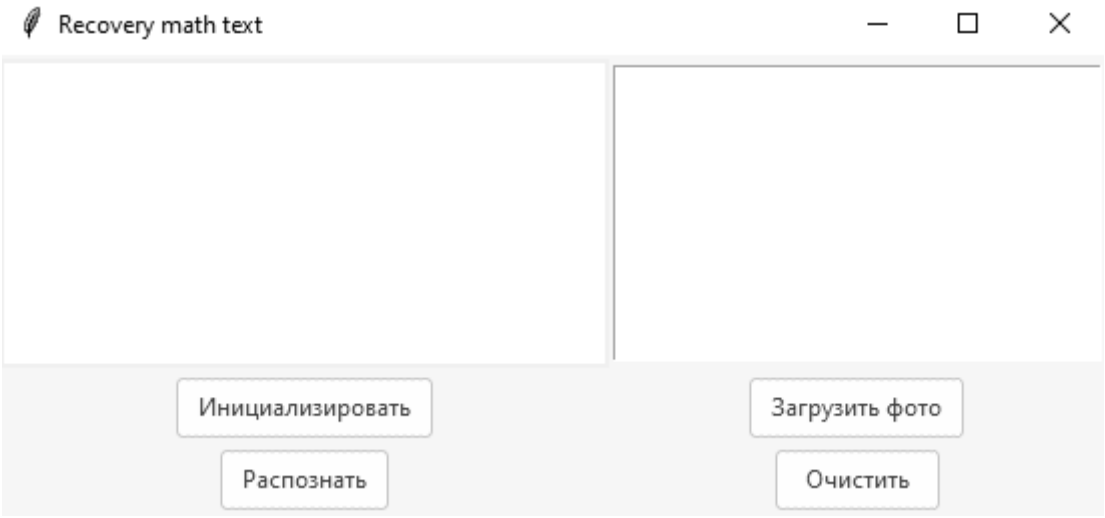

Рисунок 21 – Интерфейс десктоп-приложения

Чтобы распознать нарисованный символ в реальный момент времни, нужно нажать на кнопку «Инициализировать». Реализация пера рисования на холсте представлена в Листинг А.11.

После чего, изображение символа загрузится в модуль распознавания и останется лишь нажать кнопку «Распознать», после чего в правой стороне формы появится результат, как на Рисунок 2222. Реализация классификации изображения холста представлена в Листинг А.12.

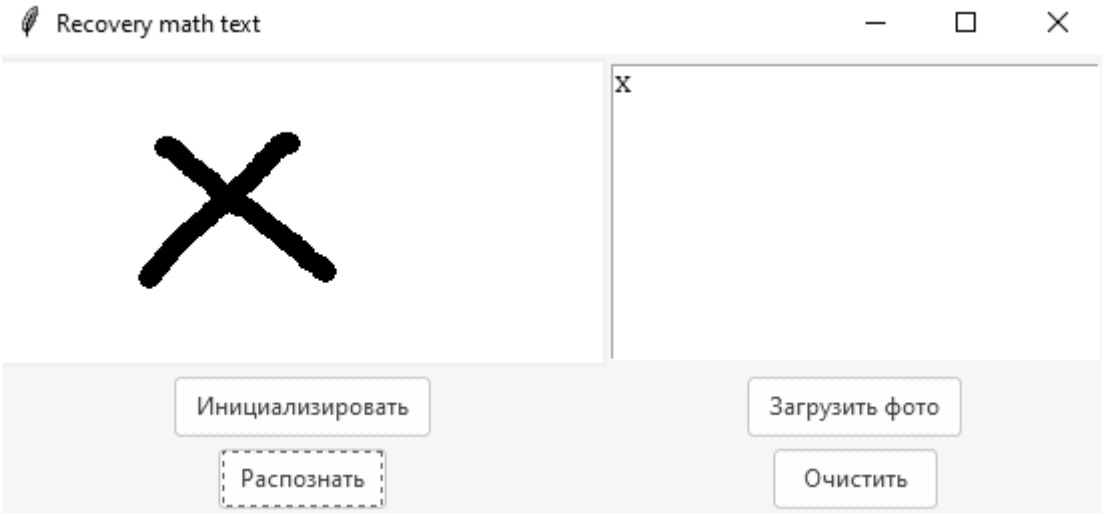

Рисунок 22– Главная форма, распознавание нарисованного символа

Для работы с загружаемым изображением нужно нажать на кнопку «Загрузить изображение». После чего откроется файловый диалог, как на Рисунок 2323.

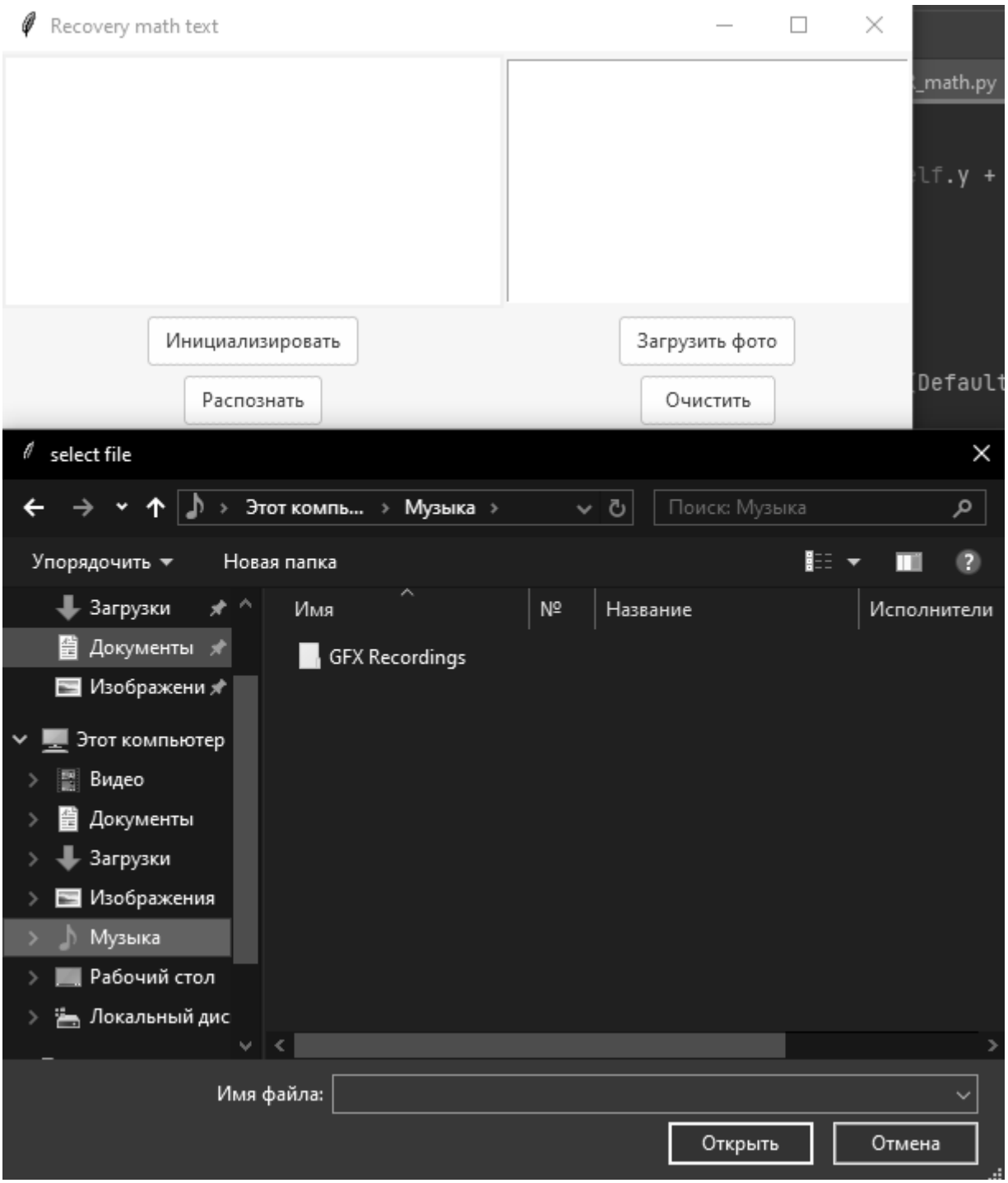

Рисунок 23 – Файловый диалог, загружаемого изображения

В открытом файловом диалоге следует выбрать интересующее изображение, после чего нажать на кнопку «Открыть». После чего на главной форме с правой стороны появятся распознаные символы. Пример представлен на Рисунок 244.

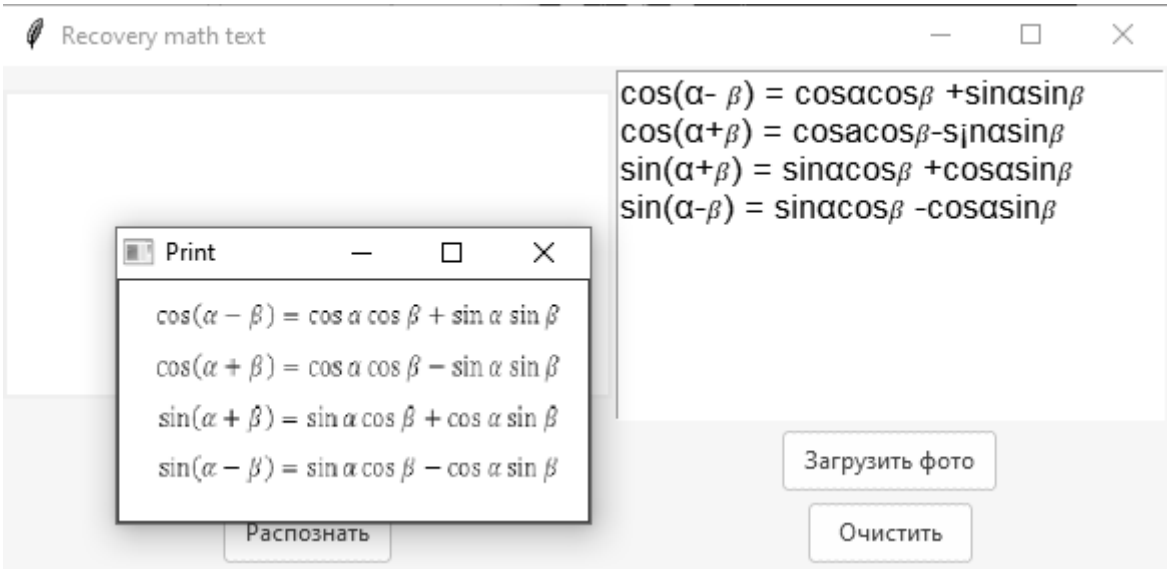

Рисунок 24 – Главная форма, распознавание изображения

Для очистки текстового окна и холста приложения используется кнопка «Очистить». Реализация очистки информации представлена в Листинг А.13

Выводы по второму разделу

Изучены и выбраны инструменты и библиотеки.

Показана предобработка изображения с помощью библиотеки OpenCV.

Проведено обучение языковой модели нейронной сети библиотеки Tesseract и вычислена точность распознавания.

Разработано десктоп приложение.

3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

3.1 **Цели и задачи, решаемые в экономической части**

Экономическая составляющая этого проекта –составляющие электронно-вычислительную машину и программное обеспечение, обеспечивающее приемлемую скорость обработки информации.

Цель решаемая в экономической части – расчет затрат на разработку десктоп-приложения. В результате расчета подсчитывается себестоимость прикладной программы.

Вычисляемые параметры:

- амортизационные отчисления на полное восстановление технических средств и программного обеспечения;
- оплата труда программиста;
- доплаты и надбавки к заработной плате;
- затраты на электроэнергию;
- накладные расходы;
- единый социальный налог.

3.2 **Расчет амортизационных отчислений**

В разработке прикладной программы используются следующие технические комплектующие:

- системный блок с настроенной архитектурой IBM PC;
- монитор viewsonic VA2223-H;
- клавиатура Logitech k280e;
- компьютерная мышь Genius.

Закупочная цена по каждому из компонентов:

- Системный блок – 22000 рублей;
- монитор – 8000 рублей;

– клавиатура – 1000 рублей;

– компьютерный манипулятор – 700 рублей.

Норма амортизации на технические средства при линейном методе расчёта имеет вид $\text{Норма амортизации} = \frac{100\%}{\text{Срок полезного действия}}$ (2):

$$\text{Норма амортизации} = \frac{100\%}{\text{Срок полезного действия}} \quad (2)$$

$$\text{Норма амортизации} = \frac{100\%}{5 \text{ лет}} = 20\%$$

Общая стоимость технических средств в рублях рассчитывается по формуле (3):

$$\text{Цена технических средств} = \text{Цком} + \text{Цмон} + \text{Цклав} + \text{Цм} \quad (3)$$

где Цком – цена IBM совместимого компьютера, Цмон – цена монитора, Цклав – цена клавиатуры, Цм – цена компьютерного манипулятора.

$$\text{Цена технических средств} = 22000 + 8000 + 1000 + 700 = 31700$$

Программное обеспечение, применяющееся в создании прикладной программы:

– платформа Windows 10 – 5000 рублей;

– IDE среда PyCharm 2021 – 9 870 рублей.

Общая стоимость программного обеспечения в рублях рассчитывается по формуле (4):

$$\text{Цена программного обеспечения} = \text{Цплат} + \text{Цide} \quad (4)$$

где Цплат – цена платформы windows 10, Цide –цена IDE среды PyCharm 2021.

$$\text{Цена программного обеспечения} = 5000 + 9870 = 14\,870.$$

Общая стоимость технических средств и программного обеспечения рассчитывается по формуле (5):

$$\text{Цо} = \text{Цтс} + \text{Цпо}, \quad (5)$$

где Цо – Общая стоимость, Цтс – стоимость технических средств, Цпо – стоимость программного обеспечения.

$$Цо = 31700 + 14\,870 = 46\,570$$

Годовые амортизационные отчисления на полное восстановление технических средств и программного обеспечения в рублях рассчитываются по формуле (6):

$$Ао = Цо \cdot Норма\ амортизации, \quad (6)$$

где Ао –годовые амортизационные отчисления на полное восстановление.

$$Ао = 46\,570 \cdot 0.2 = 9\,314$$

Амортизационные отчисления за период разработки прикладной программы в рублях рассчитывается по формуле $Ап = \frac{Ао \cdot Кдз}{Крд}$ (7):

$$Ап = \frac{Ао \cdot Кдз}{Крд} \quad (7)$$

где Ап – Амортизационные отчисления за период разработки прикладной программы, Кдз – Количество дней затраченных на работу, Крд – Количество рабочих дней в году.

Кдз равно 30-и дням, Крд равно 280-и дням.

$$Ап = \frac{9\,314 \cdot 30}{280} = 997{,}9$$

3.3 **Расчёт расходов на энергопотребление**

Электронно-вычислительная машина на которой разработана прикладная программа, является потребителем электрической энергии сети переменного тока, напряжением 220 В.

Согласно технической документации, суммарная мощность потребляемая компьютером и монитором, составляет 250 Вт·ч.

Расход денежных средств, связанный с энергопотреблением технических средств в рублях можно найти по формуле (8):

$$Рэл = Кдз \cdot Враб \cdot Мсум \cdot Цэ, \quad (8)$$

где Рэл – расход денежных средств, связанный с энергопотреблением, Враб – продолжительность рабочего времени, в часах, Мсум – мощность

потребляемая техническими средствами, в кВт час, Цэ – стоимость электроэнергии по тарифу, в кВт час.

Враб равно 6-ти часам,а Цэ равно 1,5-а рублям за кВт-час.

$$Рэл = 30 \cdot 6 \cdot 0{,}25 \cdot 1{,}5 = 67{,}5$$

3.4 **Расчёт заработной платы программиста**

Исходя из 30 отработанных шестичасовых дней, количество фактически отработанного времени в часах рассчитывается по формуле **Error! Reference source not found.**):

$$Тоф = Кдз \cdot Враб, \quad (9)$$

где Тоф – Количество фактически отработанного времени.

$$Тоф = 30 \cdot 6 = 180$$

Минимальная часовая тарифная ставка программиста составляет 150 рублей, получим основную заработную плату в рублях по формуле$Зосн = Тоф \cdot$ Средняя часовая тарифная ставка (10) :

$$Зосн = Тоф \cdot \text{Средняя часовая тарифная ставка} \quad (10)$$

где Зосн – основная заработная плата.

$$Зосн = 180 \cdot 150 = 27000$$

Для определения общей суммы расходов на оплату труда следует учесть доплаты и надбавки. Удельный вид доплат и надбавок в рублях установим в размере 15% от основной заработной платы.

$$\text{Удельный вид доплат и надбавок} = 27000 \cdot 0{,}15 = 4050.$$

Таким образом общий расход на оплату труда составит:

$$\text{Оплата труда} = 27\,000 + 4\,050 = 31\,500$$

Социальный налог в размере 25% от расходов на оплату труда в рублях составит:

$$\text{Социальный налог} = 31500 \cdot 0.25 = 7762$$

3.5 **Расчёт общих затрат на создание прикладной программы**

Расчёт общих затрат на создание прикладной программы составит сумму всех затрат в разработке программного продукта, формула $Озатр = Ап + Рэл + Оплата\ труда + социальный\ налог,$ (11):

$$Озатр = Ап + Рэл + Оплата\ труда + социальный\ налог, \quad (11)$$
$$Озатр = 997{,}9 + 67{,}5 + 31500 + 7762 = 40327\ рублей$$

Выводы по третьему разделу

Поставлены цели задачи экономического расчёта.

Подсчитана общая стоимость технических средств и программного обеспечения.

Произведен расчет амортизационных отчислений на полное восстановление технических средств и программного обеспечения.

Произведён подсчёт амортизационного отчисления за период разработки прикладной программы.

Произведен расчет общих затрат на создание прикладной программы.

ЗАКЛЮЧЕНИЕ

Разработано приложение, способное распознавать математические символы в двух вариантах, с помощью загружаемого изображения и изображения, рисуемого в реальном времени.

Выполнен сравнительный анализ пяти сервисов и программных решений. С учетом полученной в ходе анализа информации, сделан выбор в пользу разработки десктоп-приложения, работающего в режиме offline, имеющего возможность распознавать, как изображения, нарисованные в ручную, так и изображения, готового образца.

Рассмотрены характеристики библиотек и механизмов, обеспечивающих возможность работать с машинным обучением.

С помощью библиотек Tesseract, OpenCV, NumPy, на языке программирования Python версии 3.7, дистрибутива Anaconda, реализовано десктоп-приложение распознавания математических символов и выражений, позволяющее пользователям перевести данные на изображении в редактируемый формат.

Таким образом, цель работы достигнута и все поставленные задачи выполнены. Результат работы может быть применен на практике в любом направлении, где потребуется реализация компьютерного зрения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Саймон Хайкин. Нейронные сети. Полный курс / Саймон Хайкин – Вильямс, 2019. – 1104 с.

2 Солдатова О.П. Применение сверточной нейронной сети для распознавания рукописных цифр / Солдатова О.П., Гаршин А. А. – Самарский государственный аэрокосмический университет имени академика С. П. Королева 2010 г. – 8 с.

3 Anh Nguyen. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images / Anh Nguyen, Jason Yosinski, Jeff Clune – Piscataway: Institute of Electrical and Electronics Engineers, 2015. – 20 с.

4 Eric W. Brown. Применение нейронных сетей для распознавания символов / Eric W. Brown, 2012. – 7 с.

5 Geoffrey E. Hinton. A Fast Learning Algorithm for Deep Belief Nets / Geoffrey E. Hinton, Simon Osindero, Yee-Whye The. – Toronto: Department of Computer Science, 2006. – 28 с.

6 Библиотека репозитория Python. – https://www.python.org/

7 Виктор Ерухимов. Компьютерное зрение и библиотека OpenCV. – https://www.lektorium.tv/course/22800

8 Информация по библиотеке TkInter для Python. – https://wiki.python.org/moin/TkInter.

9 Anaconda, библиотека статей. – https://www.anaconda.com/library

10 Christopher Olah. Глубокое обучение, обработка естественного языка и представление данных. – http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

11 Google, egorpugin. Tesseract. – https://github.com/tesseract-ocr/tesseract

12 Keras, руководства. – https://keras.io/guides/

13 Michael Nielsen. Neural Networks and Deep Learning. – http://neuralnetworksanddeeplearning.com/

14 Shalakhmetov A., Aubakirov S. Optical character recognition with neural networks. – https://bm.kaznu.kz/index.php/kaznu/article/view/572/460

15 TensorFlow, руководства. – https://www.tensorflow.org/tutorials

ПРИЛОЖЕНИЕ А

Функции, методы и библиотеки, используемые в программе

Листинг А.1– Подключенные и используемые библиотеки.

```
import cv2
import pytesseract as tess
from tkinter import *
import tkinter as tk
import win32gui
from PIL import ImageGrab, Image, ImageDraw
import numpy as np
import os
from tkinter import filedialog as fd
import tkinter.ttk as ttk
from PIL import Image, ImageTk
import matplotlib.pyplot as plt
from ttkthemes import ThemedStyle
```

Листинг А.2 – Подключение корневой папки движка Tesseract.

```
 path = (r"C:\Program Files\Tesseract-OCR\tesseract.exe")
tess.pytesseract.tesseract_cmd = path
```

Листинг А.3 – Функция инициализации виджетов и темы используемого интерфейса.

```
def __init__(self):
    tk.Tk.__init__(self)
    self.title("Recovery math text")
    self.x = self.y = 0
    #тема интерфейса
    Style = ThemedStyle(self)
    Style.set_theme("yaru")
    #Style.set_theme("equilux")
    bg = Style.lookup('TLabel', 'background')
    fg = Style.lookup('TLabel', 'foreground')
    self.configure(bg=Style.lookup('TLabel', 'background'))
    #--------------------
    # Создание элементов
    self.canvas = tk.Canvas(self, width=300, height=150, bg="white",
cursor="cross")
    self.label = ttk.Label(self, font=("Helvetica", 1))
    #self.entry = tk.Entry(self,width=10,font=("Helvetica", 12))
    self.text = tk.Text(self, width=30, height=9)
#font=("Helvetica", 12))
    self.classify_btn = ttk.Button(self, text="Инициализировать",
command=self.classify_handwriting)
    self.button_clear = ttk.Button(self, text="Очистить",
```

```
command=self.clear_all)
    self.button_initial = ttk.Button(self,text = "Распознать",
command=self.intial)
    self.button_fd = ttk.Button(self, text = "Загрузить фото",
command=self.select_file)
    # Сетка окна
    self.canvas.grid(row=0, column=0, pady=2, sticky=W)
    self.label.grid(row=0, column=1, pady=2, padx=2)
    #self.entry.grid(row=0, column=1, pady =2, padx=2)
    self.text.grid(row=0, column=1, pady=2, padx=2)
    self.classify_btn.grid(row=1, column=0, pady=2, padx=2)
    self.button_clear.grid(row=2, column=1, pady=2)
    self.button_initial.grid(row=2,column=0,pady=2)
    self.button_fd.grid(row=1, column=1,pady=2)
    self.canvas.bind("<B1-Motion>", self.draw_lines)
```

Листинг А.4 – Методы сегментации распознаваемого изображения.

```
    # Page segmentation method
    config1 = '--psm 1'
    config2 = '--psm 2'
    config3 = '--psm 3'
    config4 = '--psm 4'
    config5 = '--psm 5'
    config6 = '--psm 6'
    config7 = '--psm 7'
    config8 = '--psm 8'
    config9 = '--psm 9'
    config10 = '--psm 10'
    config11 = '--psm 11'
    config12 = '--psm 12'
    config13 = '--psm 13'
```

Листинг А.5 – Метод уменьшения входного изображения, реализуемый библиотекой OpenCV.

```
image = image.resize((300,150),Image.ANTIALIAS)
```

Листинг А.6 – Метод чтения изображения и применения фильтра градации серого, реализуемый библиотекой OpenCV.

```
img = cv2.imread(str(File_path), cv2.IMREAD_GRAYSCALE)
```

Листинг А.7 – Метод Гауссового размытия изображения, реализуемый библиотекой OpenCV.

```
blur = cv2.GaussianBlur(img, (3, 3), 0)
```

Листинг А.8 – Метод адаптивной бинаризации изображения, реализуемый библиотекой OpenCV.

```
gray = cv2.adaptiveThreshold(blur, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 31, 1)
```

Листинг А.9 – Метод размытия фонов, реализуемый библиотекой OpenCV.

```
dilate_img = cv2.dilate(gray, np.ones((1,1)))
```

Листинг А.10 – Метод определения контура распознаваемых изображений, реализуемый библиотекой OpenCV.

```
h, w, c = img.shape
boxes = tess.image_to_boxes(img)
for b in boxes.splitlines():
    b = b.split(' ')
    img2 = cv2.rectangle(img, (int(b[1]), h - int(b[2])),
(int(b[3]), h - int(b[4])), (0, 255, 0), 0)
```

Листинг А.11 – Функция реализации пера рисования на холсте формы.

```
def draw_lines(self, event):
    self.x = event.x
    self.y = event.y
    r = 5
    self.canvas.create_oval(self.x - r, self.y - r, self.x + r,
self.y + r, fill='black')
```

Листинг А.12 – Функция классификации изображения на холсте.

```
def classify_handwriting(self):
    HWND = self.canvas.winfo_id()
    rect = win32gui.GetWindowRect(HWND)
```

Листинг А.13 – Функция очистки результатов.

```
def clear_all(self):
    self.canvas.delete("all")

  path = os.path.join(os.path.abspath(os.path.dirname(__file__)),
'1111.png')
    os.remove(path)
    self.text.delete(1.0, END)
```

Листинг А.14 – Метод, реализуемый с помощью библиотеки Tesseract, для использования нейронной модели и метода сегментации изображения.

```
tess.image_to_string(blur,lang='math',  config='--psm 10 --oem 3')
```