МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-сервиса для распознавания рукописного текста с использованием нейросетевых технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-308.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ Я.А. Краева

Автор работы,
студент группы КЭ-401
_____ А.В. Послыхалин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-401
Послыхалину Александру Владимировичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка веб-сервиса для распознавания рукописного текста с использованием нейросетевых технологий.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow. Концепции, инструменты. – Диалектика, 2020. – 1040 с.
3.2. Набор изображений рукописных слов и словосочетаний на русском языке. [Электронный ресурс] URL: https://www.kaggle.com/datasets/constantinwerner/cyrillic-handwriting-dataset (дата обращения: 01.02.2024 г.).
3.3. Николенко С.И., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. – Питер, 2018. – 480 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ предметной области и выполнить обзор существующих аналогов и алгоритмов распознавания рукописного текста на основе нейронных сетей.
4.2. Спроектировать архитектуру веб-сервиса для распознавания рукописного текста с фотографии.
4.3. Спроектировать и реализовать нейросетевую модель для распознавания рукописного текста.

4.4. Разработать веб-сервис, предоставляющий возможность распознать рукописный текст с фотографии.
4.5. Провести вычислительные эксперименты по исследованию эффективности нейронной сети.
4.6. Выполнить тестирование веб-сервиса.
5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП Я.А. Краева

Задание принял к исполнению А.В. Послыхалин

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Информация и различное с ней взаимодействие является очень крупной частью жизни большинства людей. Значительная часть информации находится в письменном виде на бумажных носителях, в том числе и в виде рукописных текстов. Однако мир крайне быстро цифровизируется [4], поэтому в условиях нынешнего времени, когда большая часть информации и коммуникации переносится в онлайн, актуальность разработки веб-сервиса для распознавания рукописного текста с фотографии становится достаточно значительной для разных категорий пользователей. Такой инструмент предлагает уникальную возможность значительно облегчить и ускорить процесс ввода и обработки текстовой информации для людей самых разных профессий и интересов, включая студентов, исследователей, писателей, в общем, всех тех, кто работает с текстовой информацией.

Например, для студента веб-сервис для распознавания рукописного текста с фотографии может стать незаменимым помощником в учебном процессе. С помощью такого сервиса можно быстро перевести записи с лекций, конспекты, заметки с бумаги в цифровой формат, что облегчит их дальнейшую обработку, редактирование и организацию. Такая возможность особенно важна в условиях современного образовательного процесса, где эффективное управление информацией и доступ к ней в любое время и в любом месте является ключевым фактором успеха.

Также веб-сервис может служить прекрасным инструментом для людей, которые ведут личные дневники или рабочие записные книжки в бумажном виде. Возможность быстро перенести свои мысли и идеи в цифровую среду открывает перед ними широкие возможности для хранения, сортировки и поиска необходимой информации. Это не только экономит время, но и способствует лучшей организации личного пространства и информационных потоков.

Для любителей чтения и коллекционеров книг веб-сервис для распознавания текста с фотографии может стать способом цифровизации редких или старинных изданий, которые недоступны в электронном виде. Это позволит не только сохранить содержание книг для будущих поколений, но и обеспечит удобный доступ к ним в любое время.

Таким образом, веб-сервис для распознавания рукописного текста с фотографии может предоставлять простое и доступное решение для интеграции традиционных методов записи информации в современные цифровые технологии, облегчая повседневные задачи и улучшая качество жизни.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-сервиса для распознавания рукописного текста с использованием нейросетевых технологий. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области и выполнить обзор существующих аналогов и алгоритмов распознавания рукописного текста на основе нейронных сетей;

2) спроектировать архитектуру веб-сервиса для распознавания рукописного текста с фотографии;

3) спроектировать и реализовать нейросетевую модель для распознавания рукописного текста;

4) разработать веб-сервис, предоставляющий возможность распознать текст с фотографии;

5) провести вычислительные эксперименты по исследованию эффективности нейронной сети;

6) выполнить тестирование веб-сервиса.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 43 страницы, объем списка литературы – 21 источник.

В рамках первой главы описывается проведение анализа предметной области, а именно рассматриваются существующие аналоги разрабатываемого сервиса, приводится обзор различных алгоритмов и методов распознавания рукописного текста с изображений с помощью нейросетевых технологий.

Вторая глава посвящена проектированию системы, где описываются функциональные и нефункциональные требования к системе, приводится диаграмма вариантов использования. Также в рамках данной главы осуществляется выделение отдельных компонентов системы и их описание, а также построение на этой основе диаграммы компонентов и диаграммы деятельности системы для визуализации и лучшему понимаю связей между компонентами.

В третьей главе приводятся используемые в разработке программные средства, описывается используемый для обучения и тестирования нейронной сети набор данных, приводятся детали реализации различных компонентов системы распознавания, предоставляются листинги программы и пояснения к ним. Также описывается процесс реализации веб-сервиса и демонстрируется конечный вид клиентской части веб-приложения.

Четвертая глава посвящена тестированию разработанной системы. В ней приводятся вычислительные эксперименты, исследующие эффективность работы нейронной сети для распознавания текста, а также результаты функционального тестирования разработанного веб-сайта.

В заключении были приведены основные результаты, полученные при выполнении выпускной квалификационной работы, а также планы по дальнейшей доработке и усовершенствованию проекта.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор существующих аналогов

На сегодняшний день существует достаточно обширное количество сервисов, предоставляющих возможность распознавание рукописного текста с изображения, среди которых присутствуют и веб-сервисы.

Среди зарубежных сервисов можно выделить сервис Pen to Print [14], который представляет из себя веб-сайт для распознавания рукописного текста на английском языке, который позволяет пользователям преобразовывать фотографии с рукописных заметок в цифровой текст. Сервис имеет несколько режимов работы, благодаря чему, помимо обычного сканирования текста с фотографии и получения цифрового текста, присутствует возможность сканировать сразу несколько рукописных заметок с конвертацией их в единый PDF-документ, а также возможность сканирования заполненных вручную форм и преобразование в форматы Word и Excel для дальнейшего редактирования и анализа данных. Сервис является платным, но доступно 10 пробных распознаваний. Главная страница сайта, страница загрузки файла с изображением, а также страница с результатом распознавания представлены на рисунках 1–3.

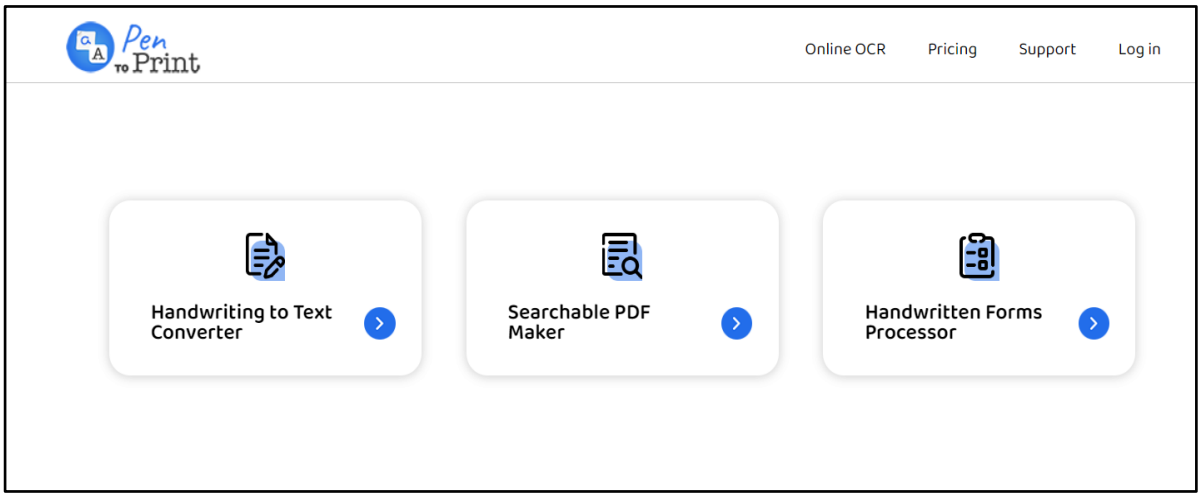

Рисунок 1 – Главная страница сайта Pen to Print

Рисунок 2 – Страница для загрузки файла сайта Pen to Print

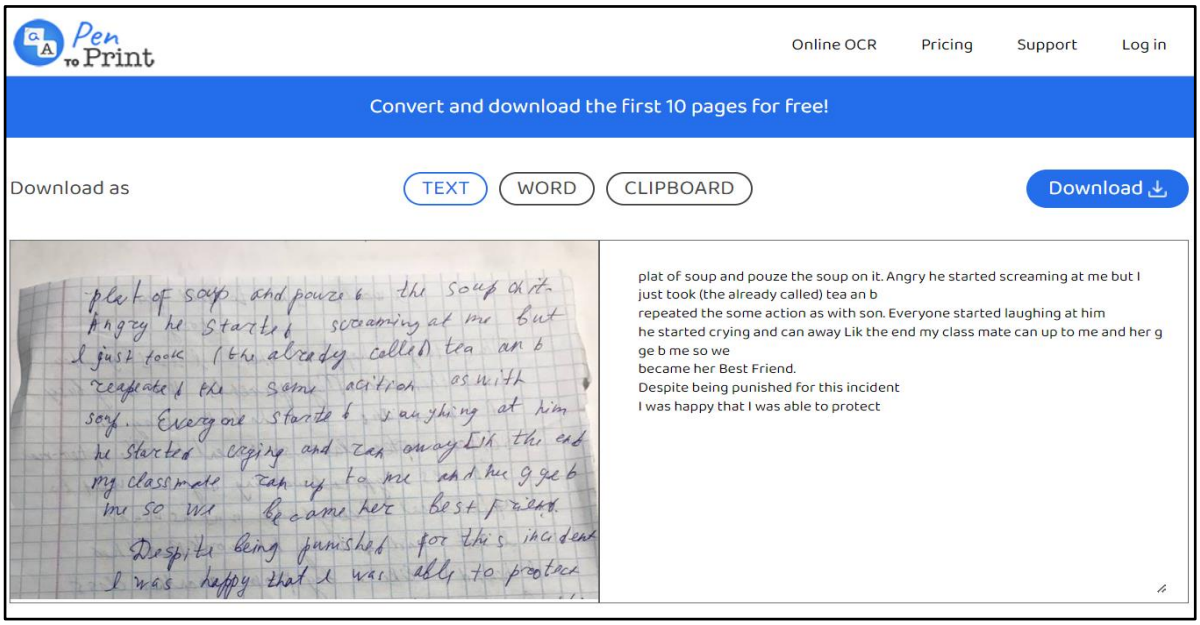

Рисунок 3 – Страница с результатом распознавания сайта Pen to Print

Среди отечественных сервисов наибольшее внимание привлекает сервис rehand.ru [16]. В отличие от сервиса Pen to print сервис rehand.ru не имеет режима, адаптированного для распознавания текстовых форм и сразу нескольких изображений одновременно, зато помимо рукописного на сайте также доступен режим распознавания и печатного текста. Также вместе с английским доступна опция распознавания и русского текста. Rehand.ru

обеспечивает высокую точность распознавания благодаря современным алгоритмам и предоставляет доступ к API для интеграции с другими приложениями. Платформа также является платной и предлагает различные тарифные планы для удовлетворения потребностей как индивидуальных пользователей, так и крупных организаций. Вид главной страницы сайта после загрузки изображения представлен на рисунке 4.

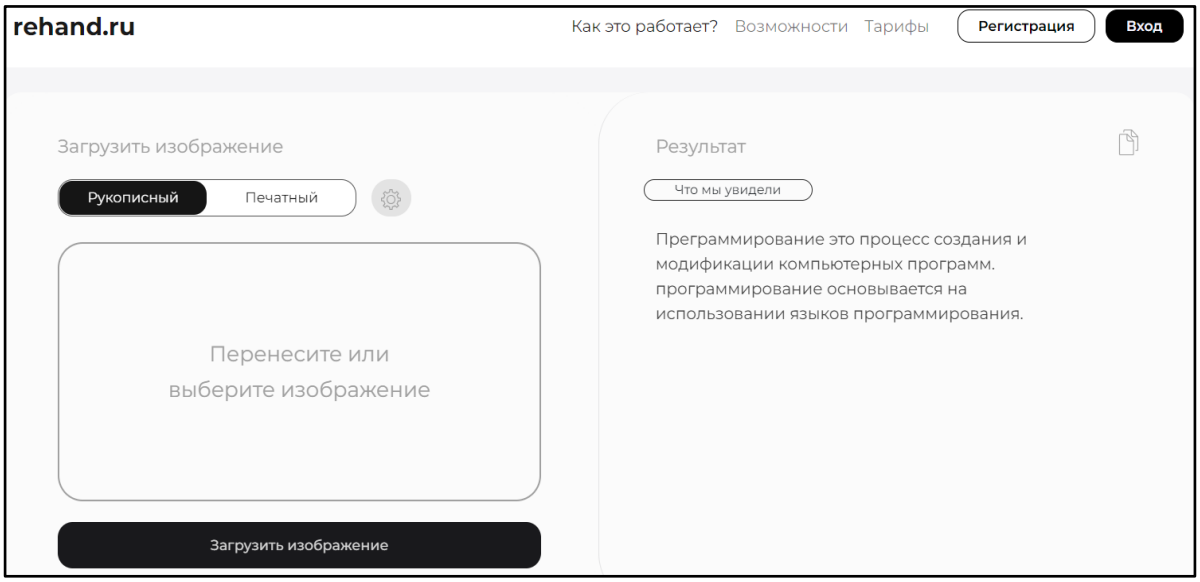

Рисунок 4 – Главная страница сайта rehand.ru после загрузки изображения

1.2. Обзор алгоритмов распознавания рукописного текста с использованием нейронных сетей

Распознавание рукописного текста – это область, которая претерпела значительные изменения с появлением технологий машинного обучения и глубокого обучения. Этот процесс включает в себя преобразование изображений рукописного текста в машинно-читаемый формат, что представляет собой сложную задачу из-за вариативности стилей письма, качества изображений и других факторов.

Первоначальные методы распознавания текста сосредоточивались на обработке печатного текста с использованием достаточно простых алгоритмических подходов. Они ограничивались распознаванием стандартизиро-

ванных шрифтов и форматов. Но их никак нельзя было назвать эффективными для работы с рукописным текстом, так как они не могли адекватно обрабатывать вариативность и нечеткость рукописного написания.

С появлением нейронных сетей [20] картина изменилась. Сверточные нейронные сети (Convolutional Neural Network, CNN) [1] стали основой для извлечения визуальных признаков из изображений, а рекуррентные нейронные сети (Recurrent Neural Network, RNN) [6], в частности, нейросеть с долгой краткосрочной памятью (Long Short-Term Memory, LSTM) [2], позволили эффективно работать с последовательностями данных, что идеально подходит для обработки текста.

Современные решения часто используют комбинацию CNN для обработки изображений и RNN/LSTM [3] для последовательного анализа распознанных символов. Такие комбинированные подходы показали значительные успехи в распознавании рукописного текста благодаря способности адаптироваться к различным стилям письма и качеству изображений.

Однако, даже современные методы, которые основываются на достаточно сложных нейросетевых моделях, сталкиваются с рядом проблем. Вариативность рукописного текста, качество изображений, шумы и искажения все равно остаются вызовами. Создание, обучение и использование таких нейросетевых моделей требует огромного количества вычислительных мощностей, а особенно – обучающих данных, что во многом и является ограничивающим фактором в настоящее время.

Вывод по первой главе

В рамках данной главы был произведен анализ предметной области, а именно обзор уже существующих аналогов разрабатываемого проекта, а также обзор существующих алгоритмов распознавания рукописного текста.

2. ПРОЕКТИРОВАНИЕ

2.1. Требования к системе

Благодаря анализу предметной области были выделены следующие функциональные и нефункциональные требования к системе.

Функциональные требования

Функциональные требования определяют ключевые операции и возможности, которые система должна предоставить пользователю. Функциональные требования к разрабатываемой системе представлены ниже.

1. Система должна позволять загрузить изображение, текст которого должен быть распознан.
2. Система должна позволять распознать рукописный текст с изображения.

Нефункциональные требования

Нефункциональные требования определяют конкретные технологии, которые должны быть использованы в проекте, и служат гарантией совместимости, эффективности и удобства дальнейшей поддержки и развития системы. Нефункциональные требования к разрабатываемой системе представлены ниже.

1. Система должна быть реализована на языке программирования Python 3.
2. Нейросетевая модель распознавания текста должна быть реализована посредством использования фреймворка Keras, базирующегося на платформе TensorFlow.
3. Веб-сайт должен быть разработан при помощи технологий HTML, CSS и JavaScript.

2.2. Диаграмма вариантов использования

Для проектирования нейронной сети был использован язык UML (Unified Modeling Language). Для отражения взаимодействия пользователя с

системой была построена диаграмма вариантов использования. Данная диаграмма представлена на рисунке 5.

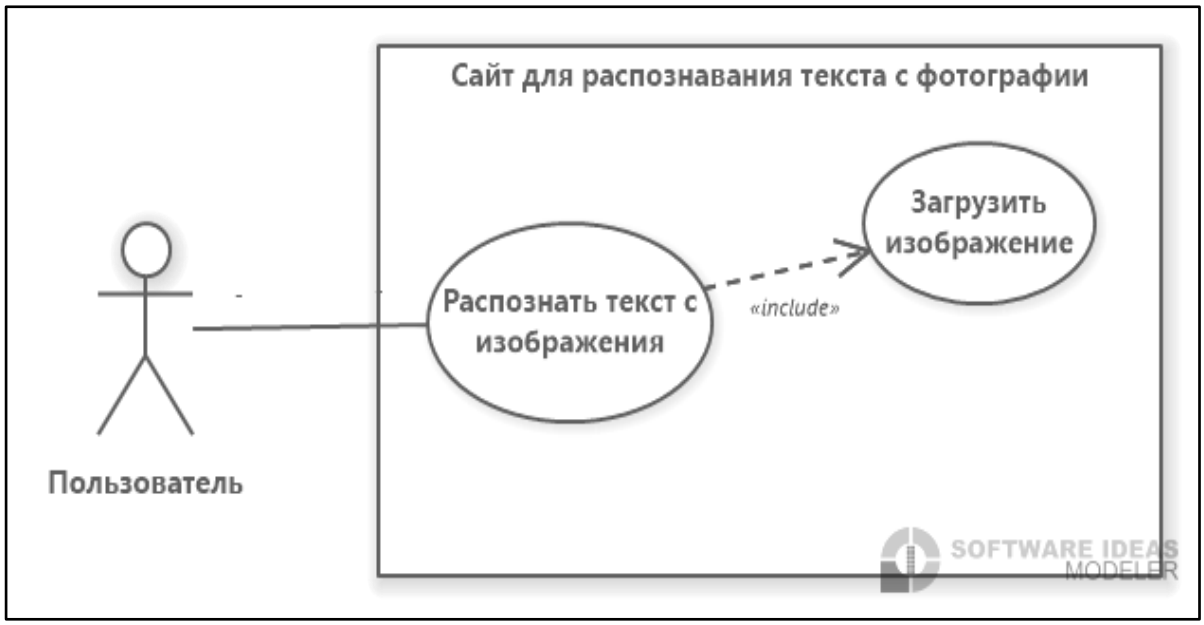

Рисунок 5 – Диаграмма вариантов использования

На данной диаграмме изображен актер «Пользователь», который может взаимодействовать с системой «Сайт для распознавания текста с фотографии» с помощью варианта использования «Распознать текст с изображения». Данный вариант использования связан исходящим отношением include «Загрузить изображение», так как загрузка изображения является обязательной частью процедуры распознавания текста с изображения.

2.3. Проектирование архитектуры системы

Система состоит из двух ключевых компонентов, а именно компонента создания и обучения модели нейронной сети, который заключается в проектировании нейросетевой модели и ее обучении на предобработанных данных, и компонента клиент-серверного приложения, который необходим для возможности использовать обученную модель нейросети пользователем, что позволяет ему загружать изображение с рукописным текстом через сайт и затем получать результат распознавания нейросети. В свою очередь каждый из этих двух компонентов состоит из более мелких компонентов.

Компонент создания и обучения модели нейронной сети состоит из следующих компонентов.

1. Компонент загрузки и обработки данных для обучения, который отвечает за загрузку данных из обучающего набора данных изображений рукописных слов, предобработку этих данных, формирования на основе этих данных наборов, состоящих из прошедших предобработку изображений и их меток для обучения и тестирования нейросети.

2. Компонент проектирования нейронной сети, который охватывает разработку архитектуры нейронной сети, то есть выбор количества и типов ее слоев, функций активации, определения гиперпараметров, таких как скорость обучения, функция потерь, оптимизатор, а также реализацию и подключение пользовательской метрики определения точности предсказаний нейросети и пользовательской метрики функции потерь.

3. Компонент обучения нейронной сети, который включает в себя обучение спроектированной модели нейросети на подготовленном наборе данных, проверку эффективности обучения нейросети на валидационном наборе данных в конце каждой эпохи обучения.

4. Компонент тестирования качества обучения нейронной сети, который состоит из тестирования уже обученной на прошлом этапе нейронной сети на тестовом наборе данных, который не участвовал в процессе обучения.

Компонент клиент-серверного приложения для возможности использовать обученную модель нейросети пользователем состоит из следующих компонентов.

1. Компонент сегментации изображения текста, который ответственен за предобработку и сегментацию изображения рукописного текста на изображения отдельных слов этого текста.

2. Компонент предсказания сегментированных частей, который включает в себя предобработку и передачу сегментированных частей изображения текста, то есть изображений отдельных слов, в уже обученную сеть

и объединение полученных предсказаний в итоговый результат в виде последовательности текстовый символов, то есть итогового текста, который является финальным результатом работы программы.

3. Компонент веб-сервиса, который включает в себя пользовательский интерфейс, а также серверную логику для возможности пользователя удобным образом загружать в систему изображения текста и получать результат распознавания.

Для наглядного представления архитектуры системы была построена диаграмма компонентов, представленная на рисунке 6.

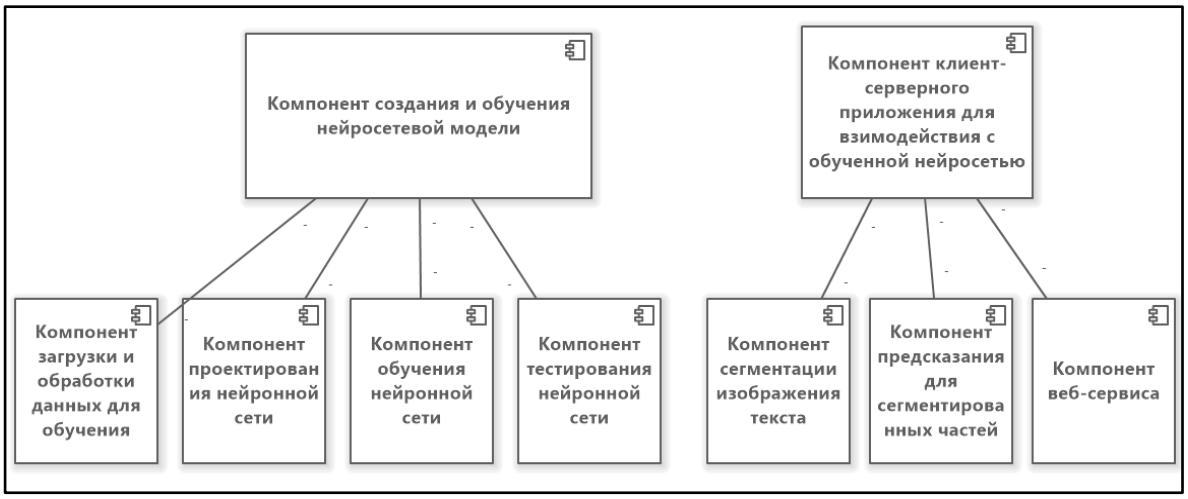

Рисунок 6 – Диаграмма компонентов системы

2.4. Проектирование алгоритма процесса создания и обучения нейросетевой модели

Для того, чтобы иметь представление о последовательности работы различных компонентов программы во время создания и обучения нейросетевой модели, была разработана диаграмма деятельности, которая отражает последовательность действий, начиная от загрузки обучающих данных и заканчивая тестированием обученной нейросетевой модели. Данная диаграмма деятельности изображена на рисунке 7.

Рисунок 7 – Диаграмма деятельности создания и обучения нейросетевой модели

2.5. Проектирование алгоритма работы клиент-серверного приложения

Для того, чтобы иметь представление также о последовательности работы различных компонентов клиент-серверного приложения, была разработана диаграмма деятельности, которая отражает последовательность действий, совершаемых программой, начиная с загрузки изображения с сайта и заканчивая выводом на сайт результата распознавания. Данная диаграмма деятельности изображена на рисунке 8.

Рисунок 8 – Диаграмма деятельности клиент-серверного приложения

Вывод по второй главе

В данной главе был описан процесс проектирования системы. Были сформулированы функциональные и нефункциональные требования к системе, построена диаграмма вариантов использования системы, описана схема базы данных. Также были описаны компоненты системы, построена диаграмма компонентов системы и диаграмма деятельности.

3. РЕАЛИЗАЦИЯ

3.1. Программные средства

В процессе разработки и обучения нейросетевой модели для распознавания рукописного текста были использованы следующие программные средства.

1. Python [15] – высокоуровневый язык программирования, который обеспечивает удобство разработки и широкие возможности для создания различных приложений. Python предоставляет богатый выбор библиотек и инструментов, необходимых для разработки систем машинного обучения.

2. Pandas [13] – библиотека для обработки и анализа данных. Она предоставляет удобные структуры данных, такие как DataFrame, для эффективного манипулирования и агрегирования табличных данных. Pandas также обладает функциональностью для чтения и записи данных из различных источников.

3. TensorFlow [18] – мощная платформа для машинного обучения от Google, поддерживающая разработку и тренировку глубоких нейронных сетей. Она предлагает гибкие инструменты для проектирования и выполнения сложных вычислительных операций, что делает ее подходящей для широкого спектра задач в области ИИ.

4. Keras [10] – высокоуровневая библиотека глубокого обучения на базе TensorFlow, которая облегчает создание и обучение нейронных сетей. Keras предоставляет простой и интуитивно понятный интерфейс для определения архитектуры нейронных сетей, выбора оптимизатора и функции потерь, а также обучения модели.

5. Scikit-learn [17] – библиотека машинного обучения, предоставляющая различные инструменты для предобработки данных, выбора и оценки моделей, а также применения различных алгоритмов машинного обучения. Scikit-learn включает в себя функциональность для масштабирования данных, разделения на обучающую и тестовую выборки, а также расчета метрик оценки моделей.

6. Flask [5] – микрофреймворк для разработки веб-приложений на языке Python. Flask обеспечивает простоту и гибкость в создании веб-приложений, позволяя легко определить маршруты и обработчики запросов.

7. HTML и CSS [7] – языки разметки и стилей, используемые для создания визуального представления веб-страницы. HTML определяет структуру страницы, а CSS позволяет управлять внешним видом элементов.

8. JavaScript [8] – язык программирования, который широко применяется в веб-разработке. Он обеспечивает возможности для создания интерактивных веб-страниц и динамического взаимодействия с пользователем. JavaScript позволяет добавлять функциональность, обрабатывать события, валидировать данные и изменять содержимое веб-страницы без необходимости ее перезагрузки.

9. Jinja2 [9] – шаблонизатор для языка Python.

10. Matplotlib [11] – библиотека Python, предназначенная для построения графиков.

11. OpenCV [12] – библиотека компьютерного зрения, предназначенная в том числе для сегментации изображений.

3.2. Подготовка данных для обучения нейросети

Описание набора данных

В качестве данных для обучения использовался набор данных Cyrillic Handwriting Dataset [19] с сайта Kaggle. Данный набор данных состоит из множества фотографий с русскоязычными рукописными словами и словосочетаниями. Загруженный набор данных уже размечен и разбит на обучающую и тестовую выборки посредством его представления в виде двух папок с изображениями train (72 286 изображений) и test (1 544 изображения), а также двух табличных tsv файлов train.tsv и test.tsv, в каждом из которых первый столбец представляет из себя названия файлов изображений в соот-

ветствующей папке, а второй столбец – метки, то есть строки с текстом, соответствующие каждому изображению. Вид части содержимого папки train и файла train.tsv представлены на рисунках 9 и 10 соответственно.

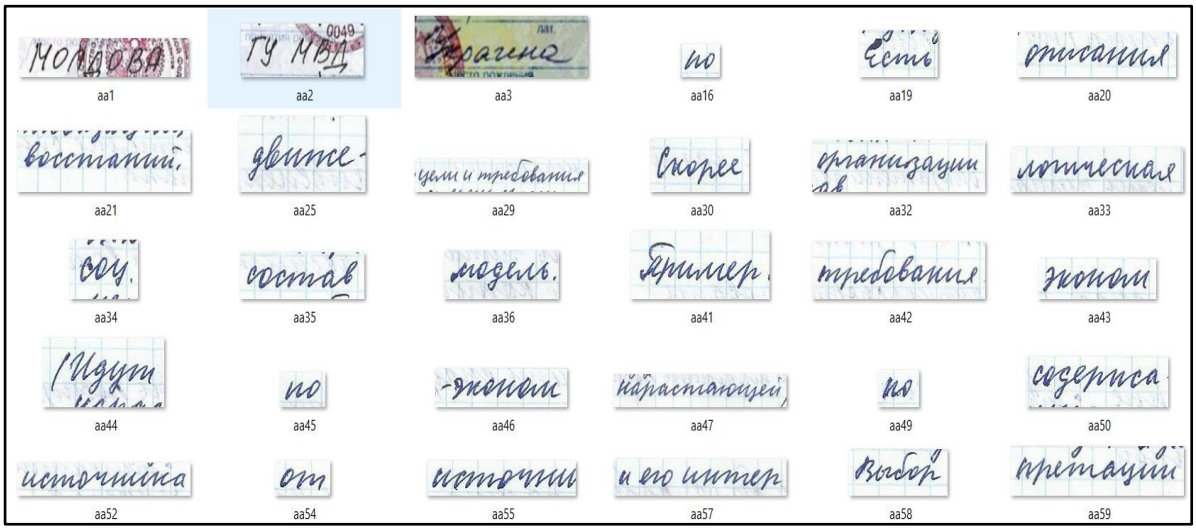

Рисунок 9 – Вид части содержимого папки train

```
1    aa1.png Молдова
2    aa1007.png  продолжила борьбу
3    aa101.png   разработанные
4    aa1012.png  Плачи
5    aa1013.png  Гимны богам
6    aa1017.png  (вспомнить
7    aa1018.png  миф
8    aa1019.png  созда-
9    aa102.png   Даже работа
10   aa1021.png  людей).
```

Рисунок 10 – Вид части содержимого файла train.tsv

Загрузка данных

Изначально происходит подключение Google диска к среде Google Colab, после чего архив с набором данных, располагающийся на диске, разархивируется прямо в файловое пространство виртуальной машины внутри среды Google Colab. Код представлен в листинге 1.

Листинг 1 – Загрузка данных

```
drive.mount('/content/drive')
!unzip "/content/drive/My Drive/RusHandwrittingTextFromImage/Kiril-
licHandwritingDataset.zip" -d ./
```

Функция предобработки изображений

Для того, чтобы привести изображение в удобный для обработки нейросетью вид, применяется предобработка изображений. Этап предобработки включает в себя серию операций, целью которых является приведение всех изображений к единообразному виду. Код функции предобработки изображения представлен в листинге 2.

Листинг 2 – Предобработка изображения

```
def preprocess(img):
  for func in [resize_n_rotate, add_adaptiveThreshold]:
    img = func(img)
  return img
def resize_n_rotate(img, shape_to=(64, 400)):
  if img.shape[0] > shape_to[0] or img.shape[1] > shape_to[1]:
    shrink_multiplayer = min(math.floor(shape_to[0] / img.shape[0] * 100) /
100,
                             math.floor(shape_to[1] / img.shape[1] * 100) /
100)
  img = cv2.resize(img, None,
                   fx=shrink_multiplayer,
                   fy=shrink_multiplayer,
                   interpolation=cv2.INTER_AREA)

  img = cv2.copyMakeBorder(img, math.ceil(shape_to[0]/2) -
math.ceil(img.shape[0]/2),
                           math.floor(shape_to[0]/2) -
math.floor(img.shape[0]/2),
                           math.ceil(shape_to[1]/2) -
math.ceil(img.shape[1]/2),
                           math.floor(shape_to[1]/2) -
math.floor(img.shape[1]/2),
                           cv2.BORDER_CONSTANT, value=255)
  return cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
def add_adaptiveThreshold(img):
  return cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY_INV, 11, 2).astype('bool')
```

Функция `preprocess` является основным входом в процесс предобработки. Она последовательно применяет к изображению две функции: `resize_n_rotate` и `add_adaptiveThreshold`.

Функция `resize_n_rotate` выполняет две задачи. Сначала она проверяет, нужно ли изменять размер изображения, чтобы соответствовать заданным параметрам `shape_to`. Если изображение больше этих размеров, оно уменьшается. Затем функция добавляет рамку вокруг изображения так, чтобы оно точно соответствовало заданному размеру, и поворачивает изображение на 90 градусов по часовой стрелке.

Функция `add_adaptiveThreshold` применяет адаптивное пороговое значение к изображению для его преобразования в черно-белый формат. Этот метод выбирает пороговое значение для каждого пикселя на основе среднего значения из небольшого окна пикселей вокруг него. Это делает метод более гибким в сравнении с обычным пороговым преобразованием, поскольку он адаптируется к различным условиям освещения и контрастности в разных частях изображения. Также данный метод осуществляет нормализацию изображения, приравнивая каждый пиксель значению 1 или 0 в зависимости от того превышает ли яркость пикселя вычисленный для него порог.

Считывание изображений и их меток

Эта часть программы предназначена для считывания данных из файлов и их подготовки, которая необходима для обучения нейросети. Основная задача заключается в извлечении изображений и соответствующих им меток из предоставленных файлов, их предобработке для приведения к единому формату и последующем разделении на обучающую и тестовую выборки. Код представлен в листинге 3.

Листинг 3 – Считывание изображений и их меток

```
def load_images_and_labels(images_folder, labels_df):
    images = []
    labels = []
    for index, row in labels_df.iterrows():
        image_path = os.path.join(images_folder, row['image_name'])
        img = cv2.imread(image_path, 0)
        if img is not None:
            img = preprocess(img)
            images.append(img)
            labels.append(row['label'])
        else:
            print(f"Warning: Image not found - {image_path}")
    return np.array(images), labels
train_labels_df = pd.read_csv(train_tsv_path, sep='\t', header=None,
names=['image_name', 'label'])
test_labels_df = pd.read_csv(test_tsv_path, sep='\t', header=None,
names=['image_name', 'label'])
train_X, train_y = load_images_and_labels('train', train_labels_df)
test_X, test_y = load_images_and_labels('test', test_labels_df)
```

Функция `load_images_and_labels` играет важную роль в подготовке данных для обучения нейросети, автоматизируя процесс загрузки и предобработки изображений с их текстовыми метками. Чтение информации из TSV файлов приводит к созданию объектов `DataFrame`, которые служат основой для дальнейшей обработки: для каждого изображения строится путь, осуществляется его загрузка и применяется предобработка с помощью ранее описанной функции `preprocess`. Успешно обработанные изображения собираются в массивы, которые затем используются для формирования обучающих и тестовых наборов данных. Результатом работы функции являются готовые к использованию наборы `train_X` и `train_y`, `test_X` и `test_y`, содержащие изображения и соответствующие им метки, подготовленные для обучения и оценки модели.

Кодирование меток

В дополнение к предобработке изображений была также проведена предобработка меток, а именно преобразование текстовых данных в числовые, которые могут быть использованы для обучения нейросети. В контексте распознавания рукописного текста, где каждая метка представляет собой строку текста, необходимо создать универсальный способ представления этих строк в виде последовательности чисел, что и достигается путем кодирования текста с использованием алфавита всех уникальных символов, среди символов, представленных как в обучающем, так и в тестовом наборе данных. Код представлен в листинге 4.

Листинг 4 – Кодирование меток

```
def encode_text(texts, alphabet):
  def _label_to_num(label, alphabet):
    label_num = []
    for ch in label:
      label_num.append(alphabet.find(ch))
    return np.array(label_num)
  nums = np.ones([len(texts), max([len(text) for text in texts])],
dtype='int64') * len(alphabet)
  for i, text in enumerate(texts):
    nums[i][:len(text)] = _label_to_num(text, alphabet)
  return nums
train_texts = [str(label) for label in train_y]
test_texts = [str(label) for label in test_y]
combined_texts = train_texts + test_texts
```

```
alphabet = ''.join(sorted(pd.Series(combined_texts).apply(list).ap-
ply(pd.Series).stack().unique()))
train_y = encode_text(train_texts, alphabet)
test_y = encode_text(test_texts, alphabet)
```

Функция `encode_text` принимает массив текстов и алфавит, возвращая массив числовых представлений меток. Внутри этой функции, для каждого текста создается массив, где каждый символ заменяется на соответствующий индекс из алфавита. В результате получается структурированное числовое представление текстов, которое можно использовать для обучения нашей нейронной сети.

Для того, чтобы получить алфавит, происходит объединение всех меток, включая как обучающий, так и тестовый набор. Объединенный список меток преобразуется в одну длинную строку, в которой последовательно собраны все символы из каждой метки. Это позволяет рассматривать весь набор текстовых данных как единую последовательность символов. Затем, из этой объединенной строки извлекаются все уникальные символы для формирования алфавита.

В конечном итоге выполнения данной части программы, `train_y` и `test_y` трансформируются из массивов текстовых строк в массивы числовых представлений.

Отделение валидационного набора

Для более качественного обучения модели следует выполнить разделение исходного тренировочного набора данных так, чтобы выделить из него валидационный набор. Валидационный набор необходим для оценки модели в процессе обучения таким образом, чтобы видеть прогресс в способности модели выполнять предсказания на основе данных, позволяющих протестировать обобщающую способность модели и оценивать риски переобучения, на каждой эпохе, а не только на отдельном тестовом наборе, оценка на котором будет произведена только после прохождения всех эпох обучения. Код представлен в листинге 5.

Листинг 5 – Отделение валидационного набора

```
train_X, val_X, train_y, val_y  = train_test_split(train_X, train_y,
test_size=0.2, random_state=42)
```

Функция `train_test_split` разделяет исходные обучающие данные и их метки на новый обучающий и валидационный наборы в соотношении 80:20 соответственно. Указание `random_state` обеспечивает одинаковое разделение при каждом запуске.

3.3. Реализация и обучение нейронной сети

Реализация метрики для оценки производительности нейросети

Для того чтобы оценить эффективность обучения нейронной сети, используется метрика Character Error Rate (CER), которая измеряет процент ошибочно распознанных символов и является стандартом в оценке систем распознавания текста. Реализация этой метрики важна для мониторинга и улучшения точности распознавания текста, так как она дает понимание о том, насколько хорошо модель справляется с задачей распознавания на символьном уровне. Код представлен в листинге 6.

Листинг 6 – Метрика для оценки производительности нейросети

```
class CERMetric(tf.keras.metrics.Metric):
    def __init__(self, name='CER_metric', **kwargs):
        super(CERMetric, self).__init__(name=name, **kwargs)
        self.cer_accumulator = self.add_weight(name="total_cer", initial-
izer="zeros")
        self.counter = self.add_weight(name="cer_count", initializer="ze-
ros")
    def update_state(self, y_true, y_pred, sample_weight=None):
        input_shape = K.shape(y_pred)
        input_length = tf.ones(shape=input_shape[0]) * K.cast(in-
put_shape[1], 'float32')
        decode, log = K.ctc_decode(y_pred, input_length, greedy=True)
        decode = K.ctc_label_dense_to_sparse(decode[0], K.cast(in-
put_length, 'int32'))
        y_true_sparse = K.ctc_label_dense_to_sparse(y_true, K.cast(in-
put_length, 'int32'))
        y_true_sparse = tf.sparse.retain(y_true_sparse,
tf.not_equal(y_true_sparse.values, tf.math.reduce_max(y_true_sparse.val-
ues)))
        decode = tf.sparse.retain(decode, tf.not_equal(decode.values, -1))
        distance = tf.edit_distance(decode, y_true_sparse, normalize=True)
        self.cer_accumulator.assign_add(tf.reduce_sum(distance))
        self.counter.assign_add(K.cast(len(y_true), 'float32'))
    def result(self):
        return tf.math.divide_no_nan(self.cer_accumulator, self.counter)
```

```
    def reset_state(self):
        self.cer_accumulator.assign(0.0)
        self.counter.assign(0.0)
```

В этом коде реализован класс `CERMetric`, который наследует функциональность от `tf.keras.metrics.Metric` для вычисления `Character Error Rate (CER)`, критической метрики в задачах распознавания текста. В конструкторе класса инициализируются переменные: `cer_accumulator` для хранения суммарного количества ошибок распознавания символов и `counter` для подсчета общего количества рассмотренных символов.

Метод `update_state` использует расстояние Левенштейна, также известное как редакционное расстояние, которое измеряет минимальное количество односимвольных операций (вставок, удалений или замен), необходимых для преобразования одной строки текста в другую. Это расстояние служит основой для подсчета ошибок в распознавании текста, позволяя оценить, насколько предсказанный текст отличается от истинного. Таким образом, функция `update_state` применяет алгоритмы TensorFlow для декодирования предсказанных последовательностей и сравнивает их с истинными значениями, вычисляя расстояние Левенштейна между ними. Затем она обновляет `cer_accumulator` и `counter` для отражения новых результатов.

Метод `result` вычисляет окончательный CER, деля общее накопленное расстояние ошибок на общее количество рассмотренных символов, что дает среднюю оценку ошибок на символ. Метод `reset_state` используется для сброса метрики к начальному состоянию, что необходимо при старте новой эпохи обучения или при оценке модели, обеспечивая чистый подсчет для каждого нового цикла оценки или обучения.

Реализация функции потерь для обучения модели

Так как в случае обучения модели для распознавания текста длины входных и целевых данных часто не совпадают, критически важно использовать подходящую функцию потерь, которая может эффективно оценить

разницу между предсказанными и истинными значениями. В контексте распознавания рукописного текста часто применяется функция потерь под названием Connectionist Temporal Classification (CTC), которая оптимально подходит для задач, где длина выходной последовательности не обязательно равна длине входной. Код представлен в листинге 7.

Листинг 7 – Реализация функции потерь

```
def CTCLoss(y_true, y_pred):
    batch_len = tf.cast(tf.shape(y_true)[0], dtype="int64")
    input_length = tf.cast(tf.shape(y_pred)[1], dtype="int64")
    label_length = tf.cast(tf.shape(y_true)[1], dtype="int64")
    input_length = input_length * tf.ones(shape=(batch_len, 1),
dtype="int64")
    label_length = label_length * tf.ones(shape=(batch_len, 1),
dtype="int64")
    loss = K.ctc_batch_cost(y_true, y_pred, input_length, label_length)
    return loss
```

Функция `CTCLoss` предназначена для вычисления значения потерь во время обучения модели на основе механизма CTC. Первые строки функции определяют размеры батча и длины входных и целевых последовательностей. Эти длины необходимы, поскольку CTC требует знания длин всех входных и целевых последовательностей для корректного вычисления потерь.

Важным элементом является функция `ctc_batch_cost`, которая является частью библиотеки Keras. Она принимает на вход истинные метки (`y_true`), предсказания модели (`y_pred`), а также длины входных и целевых последовательностей. Функция `ctc_batch_cost` вычисляет потери CTC для каждого элемента батча, позволяя модели обучаться на основе сравнения предсказанных последовательностей с истинными.

Реализация архитектуры нейросети

Ключевым фактором в разработке нейросетевой модели для распознавания рукописного текста с изображений является выбор архитектуры, благодаря который модель будет способна эффективно обрабатывать как пространственные, так и последовательные зависимости данных. Использова-

ние сверточных нейронных сетей (CNN) в начальных слоях позволяет извлекать важные визуальные признаки из изображений, такие как линии, углы и текстурные паттерны. Эти признаки имеют решающее значение для идентификации символов и слов на изображении.

После сверточных слоев применяются рекуррентные нейронные сети, а именно слои LSTM, которые обрабатывают данные как последовательности. Это позволяет модели улавливать контекстуальные связи между символами и словами, что необходимо для точного распознавания рукописного текста. Код представлен в листинге 8.

Листинг 8 – Реализация архитектуры нейросети

```
model = Sequential()
model.add(Conv2D(64, (5, 5), padding='same', activation=LeakyReLU(al-
pha=0.01), input_shape=(400, 64, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (5, 5), padding='same', activation=LeakyReLU(al-
pha=0.01)))
model.add(MaxPooling2D((1, 2)))
model.add(Conv2D(128, (3, 3), padding='same', activation=LeakyReLU(al-
pha=0.01)))
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding='same', activation=LeakyReLU(al-
pha=0.01)))
model.add(Conv2D(256, (3, 3), padding='same', activation=LeakyReLU(al-
pha=0.01)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(512, (3, 3), padding='same', activation=LeakyReLU(al-
pha=0.01)))
model.add(MaxPooling2D((1, 2)))
model.add(BatchNormalization())
model.add(Conv2D(512, (3, 3), padding='same', activation=LeakyReLU(al-
pha=0.01)))
model.add(MaxPooling2D((1, 2)))
model.add(Reshape((50, 512)))
model.add(Bidirectional(LSTM(256, return_sequences=True)))
model.add(Bidirectional(LSTM(256, return_sequences=True)))
model.add(Dense(len(alphabet) + 1, activation='softmax')) # +1 for ctc
blank
model.summary()
```

Сначала создается модель `Sequential`, к которой поочередно добавляются различные слои: сверточные слои `Conv2D`, обеспечивающие извлечение визуальных признаков из изображений, и слои подвыборки `MaxPooling2D`, снижающие размерность данных, сохраняя при этом важнейшие особенности, что позволяет уменьшить время обучения, не снижая

эффективности. Использование функции активации `LeakyReLU` помогает избежать проблем с исчезающими градиентами, в то время как слои `BatchNormalization` ускоряют обучение и повышают его эффективность за счет нормализации входных данных слоев.

После сверточной части данные преобразуются с помощью слоя `Reshape`, подготавливая их к обработке рекуррентными слоями. Два слоя Bidirectional LSTM анализируют полученные из прошлых слоев данные, обрабатывая информацию как в прямом, так и в обратном направлениях для лучшего понимания контекста. Завершает модель линейный слой `Dense`, который отвечает за классификацию символов, где каждый нейрон соответствует определенному символу в алфавите уникальных символов. Важно отметить, что в этот слой включен дополнительный нейрон, представляющий «CTC blank» – специальный символ, используемый в алгоритме CTC для обозначения пробела между символами или отсутствия символа.

Конструкция модели последовательная, что облегчает ее понимание и реализацию. На каждом этапе от входа к выходу данные последовательно проходят через все слои, трансформируясь и обогащаясь необходимыми для итогового распознавания признаками.

Компиляция и обучение модели

После реализации архитектуры необходимо скомпилировать, а после чего и обучить нейронную сеть. Код представлен в листинге 9.

Листинг 9 – Компиляция и обучение модели

```
model.compile(optimizer=Nadam(learning_rate=0.001, clipnorm=1.0),
loss=CTCLoss, metrics=[CERMetric()])
history = model.fit(
    train_X, train_y,
    validation_data=(val_X, val_y),
    epochs=75,
    batch_size=64,
    callbacks=[
        EarlyStopping(patience=10, restore_best_weights=True, moni-
tor='val_CER_metric'),
        model_checkpoint_callback  # Добавляем ModelCheckpoint
    ],
    verbose=1
)
```

В этом процессе модель компилируется с использованием оптимизатора Nadam, который сочетает в себе преимущества методов Adam и Nesterov, что позволяет адаптивно корректировать скорость обучения для эффективного снижения ошибок. Этот оптимизатор очень широко применяется в машинном обучении из-за своей устойчивости и эффективности. В качестве функции потерь и метрики используем уже описанные выше `CTCLoss` и `CERMetric` соответственно.

Обучение происходит на обучающем наборе данных с проверкой в конце каждой эпохи на валидационном наборе, чтобы контролировать переобучение и иметь возможность убеждаться, что модель хорошо обобщает информацию, а не просто запоминает конкретные учебные примеры без возможности экстраполировать закономерности обучающих данных на ранее неизвестные входные данные. Количество эпох и размер пакетов данных выбраны исходя из размера набора данных и вычислительных возможностей, чтобы обеспечить баланс между скоростью и качеством обучения. Использование коллбэков `EarlyStopping` и `ModelCheckpoint` помогает оптимизировать процесс обучения, автоматически останавливая его при отсутствии прогресса, то есть улучшения показателя метрики на указанном количестве итераций, и сохраняя наилучшее состояние модели для дальнейшего использования.

Декодирование полученных числовых представлений

Для того, чтобы превратить числовые данные, являющиеся результатом работы нейронной сети, в текст, состоящий из символов алфавита обучающего и тестового наборов, необходимо применить процесс, обратный уже использованному нами ранее кодированию текстовых меток, то есть декодирование. Этот шаг необходим для того, чтобы преобразовать результаты работы модели в читаемый и понятный человеку формат. Код представлен в листинге 10.

Листинг 10 – Декодирование полученных числовых предсказаний

```
# Decode label for single image
def num_to_label(num, alphabet):
    text = ""
    for ch in num:
        if ch == len(alphabet): # ctc blank
          break
        else:
          text += alphabet[ch]
    return text
# Decode labels for softmax matrix
def decode_text(nums):
  values = get_value(
      ctc_decode(nums, input_length=np.ones(nums.shape[0])*nums.shape[1],
                 greedy=True)[0][0])
  texts = []
  for i in range(nums.shape[0]):
    value = values[i]
    texts.append(num_to_label(value[value >= 0], alphabet))
    return texts
```

Функция `num_to_label` преобразует числовой массив, представляющий собой индексы символов в алфавите, в строку текста. Она перебирает каждый элемент массива, используя его как индекс для выбора соответствующего символа из уже полученного нами ранее алфавита. Когда функция встречает индекс, равный длине алфавита (индекс CTC blank), она прекращает добавление символов в строку текста.

Функция `decode_text` применяется для массива предсказаний, где каждое предсказание – это числовая последовательность, выданная моделью после декодирования CTC. Функция `ctc_decode` разделяет выходные данные модели на отдельные экземляры, каждый из которых соответсвует своему конкретному экземляру входных данных, то есть используется для преобразования этих последовательностей в более понятный формат, который затем переводится в текст с помощью ранее описанной функции `num_to_label`.

3.4. Реализация механизма сегментации текста

Для поддержки возможности распознавания текста, состоящего не из одного слова или словосочетания, а из нескольких слов и возможно даже

строк, был реализован механизм для сегментации изображения текста на изображения отдельных строк и слов, чтобы затем выполнить предсказания поочередно для каждого полученного изображения отдельного слова.

Предварительная обработка изображения

Перед тем как выполнять сегментацию изображения текста на строки и слова следует выполнить приведение изображения в более удобных для выполнения данных действий вид, чтобы улучшить различимость текста от фона, таким образом, чтобы иметь облегчить задачу сегментации. Код представлен в листинге 11.

Листинг 11 – Предварительная обработка изображения

```
img = cv2.imread(image, cv2.IMREAD_GRAYSCALE)
    def bradley_threshold(image):
        height, width = image.shape
        S = width // 8
        s2 = S // 2
        t = 0.15
        res = np.zeros_like(image)
        integral_image = np.zeros((height, width), dtype=np.int64)
        for i in range(width):
            sum_ = 0
            for j in range(height):
                sum_ += image[j, i]
                if i == 0:
                    integral_image[j][i] = sum_
                else:
                    integral_image[j][i] = integral_image[j][i - 1] + sum_
        for i in range(width):
            for j in range(height):
                x1 = max(i - s2, 0)
                x2 = min(i + s2, width - 1)
                y1 = max(j - s2, 0)
                y2 = min(j + s2, height - 1)
                count = (x2 - x1) * (y2 - y1)
                sum_ = integral_image[y2][x2] - integral_image[y1][x2] -
integral_image[y2][x1] + integral_image[y1][x1]
                if image[j, i] * count < sum_ * (1.0 - t):
                    res[j, i] = 255  # Белый цвет для текста
                else:
                    res[j, i] = 0  # Черный цвет для фона
        return res
    thresh_img = bradley_threshold(img)
```

Вначале загружается изображение в оттенках серого, для дальнейшей работы с функциями библиотеки cv2. Функция `bradley_threshold` предназначена для бинаризации изображения по методу Брэдли, который является разновидностью адаптивной пороговой обработки. Этот метод использует

интегральное изображение (где каждый пиксель содержит сумму значений яркости всех пикселей выше и слева от данного, что позволяет быстро вычислять среднее значение яркости в любом заданном окне) для быстрого вычисления среднего значения в локальных окнах изображения. Функция начинает с создания интегрального изображения, которое позволяет эффективно вычислять суммы яркости пикселей в любой прямоугольной области. Далее, для каждого пикселя изображения функция сравнивает его яркость со средним значением в окне, центрированном на данном пикселе, и устанавливает белый или черный цвет в зависимости от того, является ли пиксель темнее или светлее среднего на заданный порог. Это позволяет адаптивно реагировать на различные условия освещенности и контраста текста по отношению к фону, что особенно важно при обработке рукописных текстов на различных типах бумаги.

Сегментация на строки

Для того, чтобы корректно сегментировать изображение текста на изображения слов, а затем вернуть пользователю корректный текст с сохранением переносов строк, сначала выполним сегментацию на строки. Код представлен в листинге 12.

Листинг 12 – Сегментация на строки

```
kernel = np.ones((3, 80), np.uint8)
dilated = cv2.dilate(thresh_img, kernel, iterations=1)
contours, _ = cv2.findContours(dilated.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
sorted_contours_lines = sorted(contours, key=lambda ctr: cv2.bound-
ingRect(ctr)[1])
```

Сначала применяем дилатацию, используя метод `cv2.dilate`, для расширения областей контуров текста на изображении, что облегчает их выявление за счет того, что области слов в строках при выбранном нами ядре (3x80) сливаются друг с другом, при этом между областями строк по-прежнему остаются промежутки. Затем функция `cv2.findContours` применяется для обнаружения этих усиленных контуров, представляющих строки

текста. Обнаруженные контуры сортируются по их вертикальному расположению с помощью `sorted`, чтобы обеспечить правильную последовательность текстовых строк.

Сегментация на слова и формирование итогового текста

Следующим шагом после сегментации на строки является сегментация изображения каждой строки на изображения слов, каждое из которых передается в функцию предсказания модели, за счет чего происходит формирования итогового распознанного текста. Листинг представлен в листинге 13.

Листинг 13 – Сегментация на слова и формирование итогового текста

```
kernel = np.ones((3, 15), np.uint8)
    dilated_words = cv2.dilate(thresh_img, kernel, iterations=1)
    recognized_text = []
    for line in sorted_contours_lines:
        line_text = []
        x, y, w, h = cv2.boundingRect(line)
        roi_line = dilated_words[y:y+h, x:x+w]
        contours, _ = cv2.findContours(roi_line.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
        sorted_contour_words = sorted(contours, key=lambda ctr: cv2.bound-
ingRect(ctr)[0])
        for word in sorted_contour_words:
            if cv2.contourArea(word) < 400:
                continue
            x2, y2, w2, h2 = cv2.boundingRect(word)
            word_img = img[y + y2:y + y2 + h2, x + x2:x + x2 + w2]
            predicted_word = word_predict(word_img)
            line_text.append(predicted_word)
        recognized_text.append(' '.join(line_text))
        recognized_text.append('\n')
    final_text = ''.join(recognized_text)
    print(final_text)
    return final_text
```

Сначала изображение текста каждой строки подвергается операции дилатации с использованием функции `cv2.dilate`, которая помогает объединить близлежащие символы в слова. В этот раз параметры ядра дилатации выбраны как (3, 15), что позволяет соединять лишь символы внутри слов, не соединяя отдельные слова или строки друг с другом с целью выделить части изображений, соответствующие именно отдельным словам. После дилатации текст каждой строки разделяется на отдельные слова с помощью функции `cv2.findContours`, а затем контуры сортируются с учетом

горизонтальной координаты для обеспечения правильного порядка слов с целью дальнейшего корректного формирования итогового текста. Для каждого слова, выделенного контуром, вызывается функция `word_predict`, которая использует уже ранее обученную нейронную сеть для предсказания рукописных слов. Результаты предсказания слов добавляются в итоговый текстовый список. В процессе обхода контуров текста в цикле for, в список также добавляется символ переноса строки (\n) для каждой новой строки, обеспечивая правильное форматирования текста.

3.5. Реализация веб-сервиса

Веб-сайт, реализованный при помощи технологий HTML, CSS, JavaScript, имеет одну веб-страницу, которая связана с остальной частью системы посредством фрейморвка Flask, с помощью которого создается единое веб-приложение, и шаблонизатора Jinja2, с помощью которого осуществляется вывод результата работы нейросети в соответствующие элементы веб-страницы. Загрузка изображения и передача его на сервер осуществляется с помощью ajax-запроса.

При попадании на сайт пользователь видит заголовок с названием «Распознавание рукописного текста с изображения», форму для загрузки файла изображения, а также кнопку «Загрузить», после нажатия на которую данные передаются на сервер. После того как данные обрабатываются на сервере и возвращаются клиенту, в области с результатами выводится само загруженное изображение и распознанный с него текст. Страница сайта до и после загрузки корректного файла с изображением рукописного текста изображена на рисунке 11 и рисунке 12 соответственно.

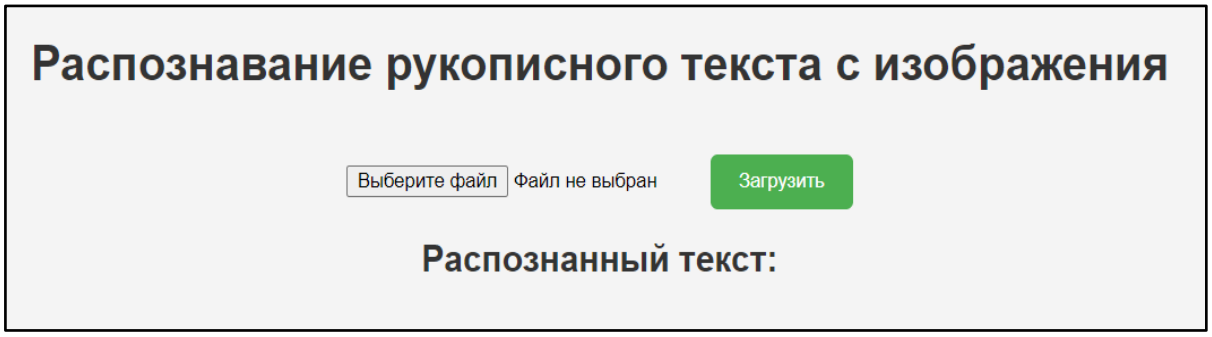

Рисунок 11 – Вид страницы сайта до загрузки изображения

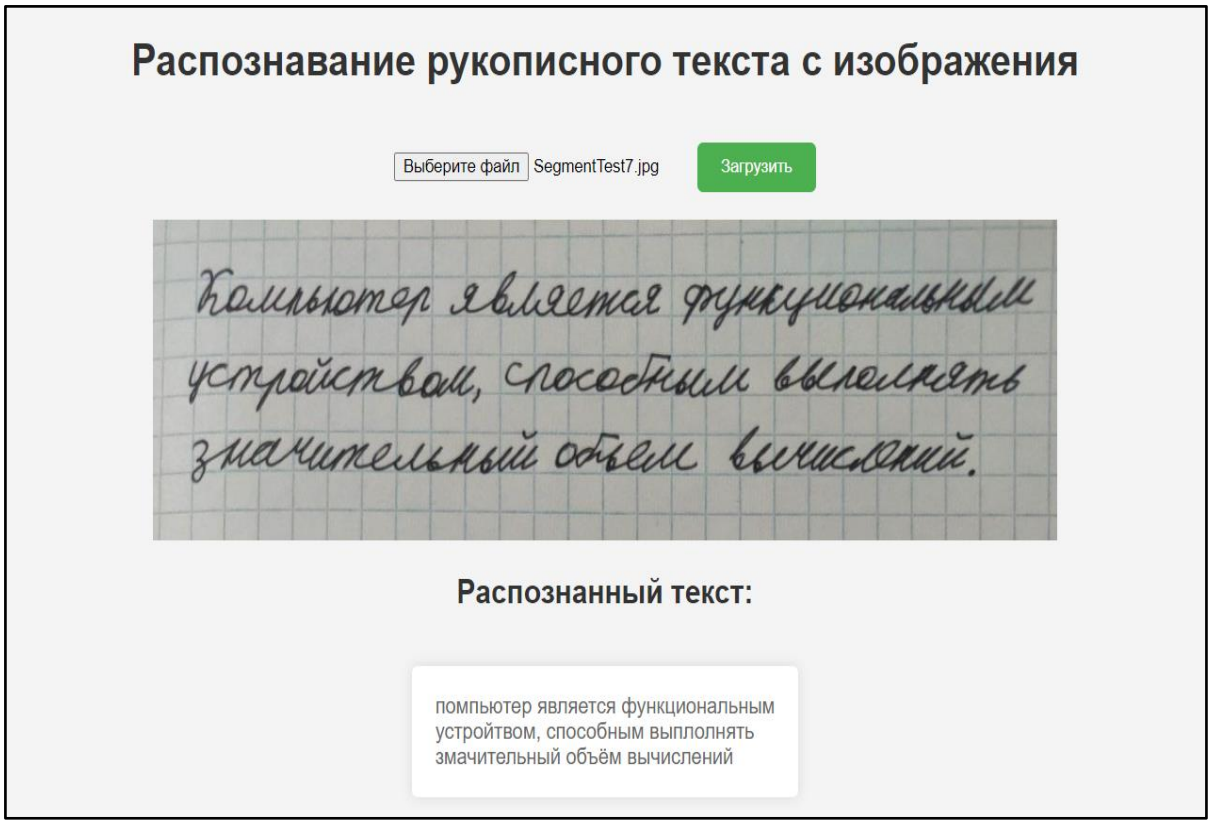

Рисунок 12 – Вид страницы сайта после загрузки изображения

Вывод по третьей главе

В данной главе были описаны используемые в разработке программные средства, используемый для обучения нейросетевой модели набор данных. Далее был приведен и описан код программной реализации подготовки данных и обучения нейросети, а также реализации механизма сегментации текста. Также был описан процесс реализации веб-сервиса и представлен конечный вид веб-сайта.

4. ТЕСТИРОВАНИЕ

4.1. Вычислительные эксперименты точности распознавания слов

Финальная оценка точности распознавания нейросети проводится на не участвующих в обучении данных, то есть на данных из тестового набора, с помощью метода `model.evaluate`. Итоговая показатель CER_metric составил 0,2628, что говорит о том, что правильно распознаны было большинство, а именно 73,72% символов.

Также для большей наглядности была реализована визуализации, чтобы отобразить предварительно обработанную версию входного изображения, предсказание нейросети и реальную метку для выбранных элементов тестовых данных. Примеры представлены ниже (рисунок 13).

Рисунок 13 – Визуализация результатов тестовых предсказаний

4.2 Тестирование механизма сегментации

Для оценки точности сегментации изображения текста на изображения отдельных слов было подготовлено 10 собственноручно написанных разными цветами ручки на разных типах бумаги небольших однострочных и многострочных рукописных текстов, содержащий суммарно 180 слов. В результате ручного подсчета было определено, что верно выделено 177 слов, количество ложных или некорректных выделений составило 8.

Таким образом Precision (точность), то есть доля правильно сегментированных слов от всех сегментированных слов составила 95,6%, в том время как Recall (полнота), то есть доля всех правильно сегментированных слов от всех слов в тексте составила 98,3%.

Также для большей наглядности на примере нескольких текстов была реализована визуализация результатов сегментации (рисунок 14).

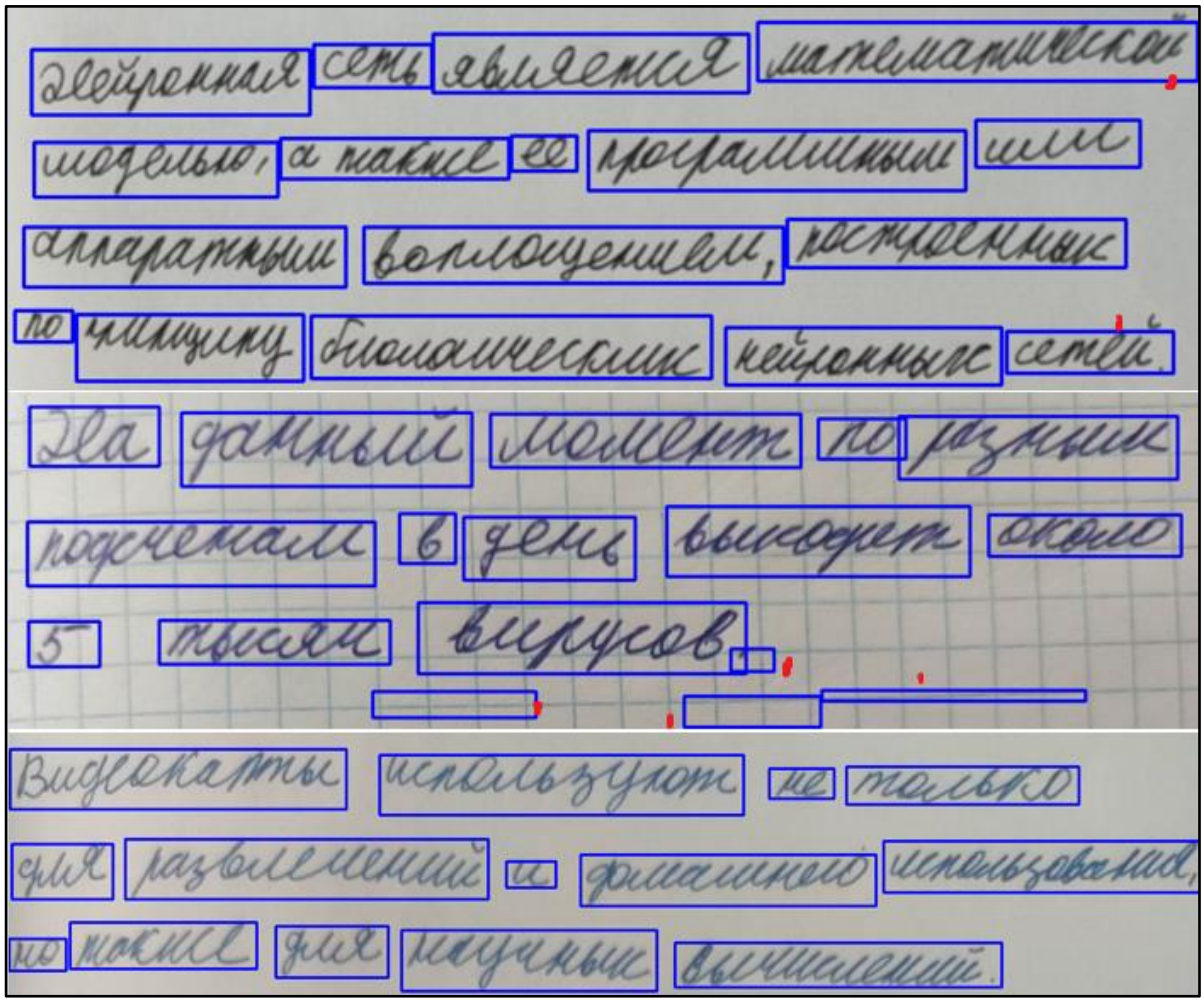

Рисунок 14 – Визуализация части результатов сегментации

При данной визуализации границы областей сегментации были обведены синей рамкой, рядом с некорректными областями сегментации были сделаны пометки в виде красных точек.

4.3. Функциональное тестирование веб-сервиса

Проведение функционального тестирование позволило оценить работоспособность сайта, его правильное функционирование, то есть его способность предоставить пользователю возможность загрузить файл изображения и получить результат распознавания, а в случае ошибочных действий пользователя корректно уведомить его об ошибке. Ниже представлены проведенные функциональные тесты, которые отражают различные сценарии поведения пользователя на сайте.

Тест №1

Отображение элементов интерфейса (Заголовка, формы для загрузки файла изображения, кнопки «Загрузить»).

Входные данные: пользователь заходит на сайт.

Ожидаемый результат: отображается заголовок, форма для загрузки файла изображения, кнопка «Загрузить».

Полученный результат: совпадает с ожидаемым.

Тест №2

Отображение загруженного изображения и распознанного текста после выбора файла с корректным расширением и нажатия кнопки «Загрузить».

Входные данные: пользователь выбирает файл с корректным расширением, нажимает на кнопку «Загрузить».

Ожидаемый результат: в области с результатами отображается загруженное изображение и весь распознанный с него текст.

Полученный результат: совпадает с ожидаемым.

Тест №3

Отображения сообщения об ошибке после выбора файла с некорректным расширением и нажатия кнопки «Загрузить».

Входные данные: пользователь выбирает файл с некорректным расширением и нажимает на кнопку «Загрузить».

Ожидаемый результат: выводится уведомление о неправильно выбранном файле, вывод в области результатов отсутствует.

Полученный результат: совпадает с ожидаемым.

Тест №4

Отображения сообщения об ошибке после нажатия кнопки «Загрузить», которому не предшествовал выбор файла.

Входные данные: пользователь нажимает на кнопку «Загрузить», не выбирая файл перед этим.

Ожидаемый результат: выводится уведомление о необходимости выбрать файл, вывод в области результатов отсутствует.

Полученный результат: совпадает с ожидаемым.

Вывод по четвертой главе

Результаты тестирования разработанной системы показали успешное выполнение вычислительных экспериментов для проверки точности предсказаний нейронной сети. В этой части были продемонстрированы конкретные результаты предсказаний из тестовой выборки, включая визуализацию правильных и ошибочных предсказаний. Также в этой главе были представлены результаты функционального тестирования веб-сервиса, которые включали проверку различных пользовательских сценариев на сайте. Эти тесты показали, что веб-сервис работает корректно, с совпадением ожидаемых и полученных результатов, подтверждая способность системы обрабатывать запросы пользователей, причем как в случаях корректных, так и некорректных действий пользователей.

Таким образом, тестирование подтвердило надежность и точность системы, ее соответствие требованиям.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был разработан веб-сервис для распознавания текста с фотографии на основе нейросетевых технологий. Поэтапно были решены поставленные задачи.

1. Проведен анализ предметной области и выполнен обзор существующих аналогов и алгоритмов распознавания рукописного текста на основе нейронных сетей.
2. Спроектирована архитектура веб-сервиса для распознавания рукописного текста на фотографии.
3. Спроектирована и реализована нейросетевая модель для распознавания рукописного текста.
4. Разработан веб-сервис, предоставляющий возможность распознавать рукописный текст с фотографии.
5. Проведены вычислительные эксперименты по исследованию эффективности нейронной сети.
6. Выполнено тестирование веб-сервиса.

В дальнейшем планируется доработка и усовершенствование проекта путем добавления режима для распознавания печатного текста, внедрение способности распознавания текста на английском языке, увеличения точности распознавания и сегментации текста, а также усовершенствование функциональности веб-сайта в соответствии с новыми возможностями распознавания.

ЛИТЕРАТУРА

1. Ahlawat S., Choudhary A., Nayyar A., Singh S., Yoon B. Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN). // Sensors. – 2020. – Vol. 20. – No. 12. – P. 3344.

2. Amirgaliyev Y., Cherikbayeva L. Improving OCR Accuracy for Kazakh Handwriting Recognition Using GAN Models. // Applied Sciences. – 2023. – Vol. 13. – No. 9. – P. 5677.

3. An Automatic Diagnosis of Arrhythmias Using a Combination of CNN and LSTM Technology. [Электронный ресурс] URL: https://www.nature.com/articles/s41598-023-41537-z.pdf (дата обращения: 01.02.2024 г.)

4. Digital Progress and Trends Report 2023. – The World Bank, 2024. – 149 p. – DOI: 10.1596/978-1-4648-2049-6.

5. Flask. [Электронный ресурс] URL: https://palletsprojects.com/p/flask/ (дата обращения: 01.02.2024 г.).

6. Graves A., Fernández S., Gomez F., Schmidhuber J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. // Proceedings of the 23rd International Conference on Machine Learning (ICML), 2006. – P. 369–376.

7. HTML, CSS. [Электронный ресурс] URL: https://metanit.com/web/html5/ (дата обращения: 01.02.2024 г.).

8. JavaScript. [Электронный ресурс] URL: https://www.javascript.com/ (дата обращения: 01.02.2024 г.).

9. Jinja2. [Электронный ресурс] URL: https://palletsprojects.com/p/flask/ (дата обращения: 01.02.2024 г.).

10. Keras. [Электронный ресурс] URL: https://keras.io/ (дата обращения: 01.02.2024 г.).

11. Matplotlib. [Электронный ресурс] URL: https://matplotlib.org/ (дата обращения: 01.02.2024 г.).

12. OpenCV. [Электронный ресурс] URL: https://opencv.org/ (дата обращения: 01.02.2024 г.).

13. Pandas. [Электронный ресурс] URL: https://pandas.pydata.org (дата обращения: 01.02.2024 г.).

14. Pen to print. [Электронный ресурс] URL: https://pen-to-print.com (дата обращения: 01.02.2024 г.).

15. Python. [Электронный ресурс] URL: https://www.python.org/ (дата обращения: 01.02.2024 г.).

16. Rehand.ru. [Электронный ресурс] URL: https://rehand.ru/ (дата обращения: 01.02.2024 г.).

17. Scikit-Learn. [Электронный ресурс] URL: https://scikitlearn.org/stable/ (дата обращения: 01.02.2024 г.).

18. TensorFlow. [Электронный ресурс] URL: https://www.tensorflow.org/ (дата обращения: 01.02.2024 г.).

19. Датасет изображений рукописных слов и словосочетаний на русском языке. [Электронный ресурс] URL: https://www.kaggle.com/datasets/constantinwerner/cyrillic-handwriting-dataset (дата обращения: 01.02.2024 г.).

20. Николенко С.И., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. – Питер, 2018. – 480 с.

21. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow. Концепции, инструменты. – Диалектика, 2020. – 1040 с.