Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
Институт радиоэлектроники и информационных технологий – РТФ
Школа профессионального и академического образования

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

Директор ШПиАО
_____Д.В. Денисов_
(подпись) (Ф.И.О.)
«_____»_____2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПРОЕКТИРОВАНИЕ ЛАБОРАТОРНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

Научный руководитель: Кислицын Евгений Витальевич _____
к.э.н., доцент подпись

Нормоконтролер: Огуренко Егор Владимирович _____
подпись

Студент группы: РИМ-220962 Искужин Ирамаль Рафикович _____
подпись

Екатеринбург
2024

РЕФЕРАТ

Выпускная квалификационная работа магистра содержит 57 с., 15 рис., 3 табл., 41 источника.

Ключевые слова: КОМПЬЮТЕРНОЕ ЗРЕНИЕ, МАШИННОЕ ОБУЧЕНИЕ, АНАЛИЗ ТЕКСТА, ПРОЕКТИРОВАНИЕ ЛАБОРАТОРНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ, ОПТИЧЕСКОЕ РАСПОЗНАВАНИЕ СИМВОЛОВ

Цель работы: проектирование лабораторной информационной системы (ЛИС) с использованием алгоритмов машинного обучения.

Объект исследования: проектирование лабораторной информационной системы.

Предмет исследования: интеграция алгоритмов машинного обучения в ЛИС с целью решения задач, связанных с областью лабораторных исследований.

В процессе исследования проводились:

– анализ существующих подходов к проектированию лабораторных информационных систем;

– выбор и обоснование архитектуры системы, включающей клиентскую, серверную части и микросервис для обработки OCR;

– выбор алгоритмов машинного обучения, подходящих для задач распознавания текста в строительной отрасли;

– разработка и внедрение методов генерации синтетических данных для обучения моделей распознавания текста;

– проведение экспериментов по дообучению модели на кириллическом наборе данных и оценка ее производительности.

Область практического применения:

– автоматизация рутинных операций в лабораториях строительной отрасли;

– оптимизация процессов анализа и обработки данных;

– улучшение качества и скорости распознавания текстовой информации в документах, что способствует повышению общей эффективности работы лабораторий.

Разработанный микросервис для распознавания текста OCR для лабораторной информационной системы демонстрирует высокую эффективность распознавания и производительность. Применение модели SVTR в PaddleOCR позволило значительно улучшить результаты распознавания текста, что делает данную систему перспективным решением для автоматизации и оптимизации лабораторных процессов в строительной отрасли. Точность распознавания составляет более 90%.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время информационные технологии и методы машинного обучения становятся все более важными и широко применяемыми в различных областях. В контексте работы с данными, информационные системы играют ключевую роль в обеспечении эффективного получения, регистрации, обработки и анализа информации.

Цель данной магистерской работы состоит в проектировании лабораторной информационной системы для использования на промышленных предприятиях и в строительных лабораториях. Эта система будет оснащена алгоритмами машинного обучения, направленными на оптимизацию процессов анализа данных и принятия решений в лабораторной практике. Применение алгоритмов машинного обучения позволяет автоматизировать и улучшить процессы обработки и интерпретации данных, что приводит к повышению эффективности работы лабораторий и качества исследований.

В современных предприятиях существуют производственные лаборатории, которые играют ключевую роль в анализе и оценке качественных показателей сырьевых компонентов, полуфабрикатов и готовой продукции. Обычно структура производственной лаборатории включает центральную лабораторию и около 6-8 самостоятельных подразделений, таких как цеховые подразделения, лаборатории сертификации готовой продукции, экологические лаборатории, полевые лаборатории и другие. На крупных предприятиях численность лабораторий может достигать от 100 до 300 работников, а общее количество анализов качества, проводимых за год, может составлять сотни тысяч.

Исследование [1] выделяет три направления компьютеризации лабораторий: внедрение ЛИМС, специализированных общелабораторных ПО, использование не специализированных «офисных» или кустарных программ.

По мнению автора, «офисные» и «кустарные» программы не позволяют в полной мере решать задачи обеспечения качества лабораторного контроля.

В 1980-е годы началось развитие промышленных систем автоматизации производственных лабораторий, так называемых систем управления лабораторной информацией (LIMS – Laboratory Information Management System). Сегодня существует несколько десятков фирм, производящих подобные системы, и их внедрение на предприятиях всех технологических отраслей является распространенным явлением. Однако, внедрение существующих LIMS недостаточно для удовлетворения современных требований к лабораториям, которые должны обеспечить технологов и операторов необходимой информацией для точного соблюдения заданных значений качественных показателей производства. Таким образом, необходимо развивать существующий уровень автоматизации производственных лабораторий, чтобы они соответствовали современным требованиям к их работе [2].

Это вызов не только обозначает важность совершенствования информационных систем в лабораторных условиях, но и подчеркивает необходимость интеграции современных методов машинного обучения для оптимизации процессов анализа и интерпретации данных. В связи с этим, данная магистерская работа не только затронет тему разработки новой лабораторной информационной системы, но и будет исследовать ее возможность в улучшении эффективности работы лабораторий и повышении качества проводимых исследований.

Кроме того, в данной работе уделяется внимание техническим аспектам проектирования лабораторной информационной системы с использованием алгоритмов машинного обучения. Это включает в себя выбор подходящих методов обработки и анализа данных, архитектуру системы, выбор и оптимизацию алгоритмов машинного обучения под конкретные потребности лабораторной практики.

Исследование также будет включать в себя анализ существующих подходов и решений, а также оценку их эффективности и применимости для разрабатываемой лабораторной информационной системы. Результаты данного исследования позволят выделить наиболее перспективные методы и подходы для дальнейшей реализации.

В итоге, разработка лабораторной информационной системы с использованием алгоритмов машинного обучения представляет собой важное направление в современной науке и технологиях, которое имеет потенциал значительно улучшить процессы анализа и интерпретации данных в лабораторной практике.

Актуальность работы обусловлена необходимостью создания новой лабораторной информационной системы, базирующейся на использовании алгоритмов машинного обучения для обработки данных.

Целью данной работы является проектирование лабораторной информационной системы с использованием алгоритмов машинного обучения.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) Провести анализ существующих подходов к проектированию информационных систем;

2) Исследовать гипотезы применения машинного обучения в рамках информационной системы;

3) Разработать микросервис для оптического распознавания символов (OCR) и подготовить его для интеграции в лабораторную информационную систему.

Объектом исследования является проектирование лабораторной информационной системы.

Предметом исследования является интеграция алгоритмов машинного обучения в лабораторную информационную систему для автоматизации процессов обработки данных.

1. АНАЛИЗ СУЩЕСТВУЮЩИХ ПОДХОДОВ К ПРОЕКТИРОВАНИЮ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1 Изучение и анализ предметной области

Информационные технологии играют ключевую роль там, где требуется автоматизация большого объема рутинных операций. В лабораториях предприятий производится значительное количество испытаний, результаты которых оформляются и хранятся в специальных журналах. Этот процесс частично или полностью автоматизируется с помощью лабораторных информационных менеджмент систем (ЛИМС). ЛИМС выполняют различные функции, такие как планирование работы лаборатории, регистрация проб, формирование паспортов продукции и другие промежуточные задачи. Кроме того, часто ЛИМС интегрируются в общую информационную систему предприятия.

ЛИМС (Laboratory Information Management System, LIMS) - представляет собой категорию программных продуктов, разработанных для сбора, анализа, отчетности, хранения, управления и обработки данных, полученных в лабораторных условиях [3].

Различные организации, такие как специализированные метрологические лаборатории, судебно-медицинские учреждения и лаборатории производственных предприятий, внедряют ЛИМС для разнообразных целей. Каждый из этих типов организаций имеет уникальные особенности в проведении лабораторных исследований, что отражается в их индивидуальных требованиях к информационному обеспечению процессов.

Несмотря на доступность различных лабораторных информационных менеджмент систем (ЛИМС) на российском рынке, производственные процессы, требующие лабораторного контроля, демонстрируют значительное разнообразие в зависимости от сферы применения. Это создает вызов в

разработке универсального ЛИМС, способного эффективно удовлетворять потребности различных отраслей промышленности и учитывать их специфику. Например, в отличие от химической промышленности и фармацевтических компаний, добывающие, металлургические и строительные отрасли демонстрируют более ограниченный спектр производственных операций, подлежащих контролю. Однако, помимо внутренних процессов производства, лаборатории этих предприятий занимаются также измерениями влияния на окружающую среду и контролем поступающих материалов. Следовательно, общее количество разнообразных лабораторных операций, требующих информационного сопровождения, может достигать значительных объемов, измеряемых сотнями или даже тысячами [4].

Строительная сфера, будучи одной из старейших отраслей, характеризуется интенсивным использованием трудовых и технических ресурсов. С точки зрения индустриализации, строительство является уникальной производственной отраслью, где возводимые объекты, такие как здания и конструкции, создаются путем последовательного выполнения различных процессов. Однако сложность процесса строительства оказала замедляющее влияние на его промышленную эволюцию. В частности, в развивающихся странах строительная отрасль до сих пор придерживается традиционных трудоемких практик с высоким энергопотреблением, неблагоприятным воздействием на окружающую среду, угрозами безопасности и низкой эффективностью при выполнении проектов.

Индустрия 4.0 представляет новые возможности для строительной отрасли, включая применение интернета вещей, анализа больших данных, облачных вычислений и искусственного интеллекта. Эти технологии постепенно внедряются в строительство для оптимизации процессов проектирования, оценки производительности, управления ресурсами, мониторинга рисков, энергосбережения и реализации проектов. Однако

строительная отрасль в этом контексте все еще находятся на начальном этапе и отстают от других отраслей промышленности [5].

На российском рынке присутствуют разработчики ЛИМС, которые специализируются на создании инструментов, адаптированных к специфике строительной отрасли. Эти компании осуществляют значительные усилия в разработке и совершенствовании ЛИМС, предназначенных для обеспечения эффективного управления лабораторными процессами в строительстве. Их продукты обладают достаточным набором функциональных возможностей, включая управление жизненным циклом проб, обеспечение наличия и актуальности нормативно-технической документации, управление оборудованием лаборатории и генерацию необходимой документации [6].

Однако, несмотря на разнообразие ЛИМС на российском рынке, проведенный анализ показывает, что данные системы не включают в свою функциональность алгоритмы машинного обучения. Применение таких алгоритмов могло бы существенно оптимизировать процессы сбора и обработки данных в лабораторной среде, а также способствовать прогнозированию событий и выявлению закономерностей.

1.2 Проектирование информационных систем

Проектирование информационных систем представляет собой важный этап в разработке. В данной разделе главы рассматриваются основные принципы и методы проектирования информационных систем, а также их роль и значение в контексте оптимизации лабораторных процессов. Особое внимание уделяется аспектам, связанным с интеграцией лабораторных информационных менеджмент систем (ЛИМС) и использованием передовых технологий, включая алгоритмы машинного обучения, для достижения оптимальной эффективности и точности в управлении данными и процессами в лабораторной среде.

Проектируемая система, должна обеспечить выполнение следующих задач [6]:

– обеспечение регистрации и идентификации образцов;

– автоматический или ручное составление плана испытаний образов (основываясь на нормативно-технические документы);

– ручной и автоматический ввод результатов анализов для различных образцов;

– конвертация единиц измерения;

– проверка введенных результатов испытаний на соответствие диапазону допустимых значений согласно нормативным документам;

– автоматический расчет показателей качества;

– возможность просмотра результатов, их утверждения или отклонения;

– формирование отчета об отклонениях различных показателей качества от нормы с использованием статистических методов, таких как контрольные карты Шухарта;

– генерация различных форм отчетов, таких как паспорта качества и протоколы испытаний, журналы, рабочие листы испытаний.

Кроме того, в системе должно быть организовано разграничение прав пользователей.

Для успешной реализации описанной функциональности в информационной системе необходимо составить техническое задание, которое будет определять требования к системе. Этот процесс начинается с сбора требований, который включает в себя анализ потребностей пользователей, выявление основных функций и возможностей системы, а также определение ожидаемых результатов от ее использования. Сбор требований является важным этапом в разработке любой информационной системы, поскольку он определяет основные параметры и цели проекта, а также обеспечивает понимание ожиданий заказчика и конечных пользователей.

Для разработки предлагаемой системы были выбраны определенные технологии и инструменты. Для серверной части системы был выбран язык программирования Ruby и фреймворк Ruby on Rails, который отлично подходит для создания веб-приложений благодаря своей простоте и высокой производительности. Для клиентской части системы будет выбраны такие технологии как язык программирования JavaScript, который широко применяется для создания интерактивных пользовательских интерфейсов в веб-приложениях, язык гипертекстовой разметки HTML и каскадные таблицы стилей CSS. В качестве системы управления базами данных была выбрана PostgreSQL, известная своей надежностью, масштабируемостью и расширяемостью. Для проверки гипотез по машинному обучению и разработки микросервисов будет использоваться язык программирования Python и среда Jupyter Notebook, обеспечивающий удобную интерактивную работу с данными и анализ результатов.

В архитектуре данного проекта предусмотрено разделение на три основные компоненты: клиентскую часть, серверную часть и микросервис, ответственный за обработку очереди задач. Клиентская часть обеспечивает взаимодействие пользователей с системой через удобный и интуитивно понятный интерфейс. Серверная часть отвечает за обработку запросов клиентов, выполнение бизнес-логики и взаимодействие с базой данных. Микросервис, специализированный на обработке очереди задач, занимается обработкой отсканированных документов с последующим распознаванием и индексированием полученной информации. Такое разделение позволяет достичь оптимальной организации работы системы, обеспечивая высокую производительность и эффективность ее функционирования.

Архитектурные решения проекта были приняты с учетом требований к производительности, масштабируемости и безопасности системы.

1.3 Технологии и методы машинного обучения, применяемые в информационных системах

Информационные технологии и информационные системы, в частности становятся неотъемлемой частью в сферах жизнедеятельности предприятий. Устойчивые тенденции, связанные с развитием и интеграцией информационных технологий, обусловлены возможностью качественного улучшения и повышения эффективности различных бизнес-процессов. В результате практически все современные предприятия и организации вкладывают значительные ресурсы в создание новых технологических решений и интеграцию существующих в свою деятельность [7].

С бурным развитием искусственного интеллекта и машинного обучения появилась возможность интеграции их алгоритмов в такие информационные системы.

Так, [7] для эффективной защиты информационных систем могут быть использованы технологии искусственного интеллекта и машинного обучения. Применение методов машинного обучения, включая обнаружение аномалий в реальном времени, представляет собой важный аспект защиты информационных систем от активных кибератак. Использование искусственного интеллекта для автоматизации и обеспечения соблюдения нормативов безопасности способствует упрощению процесса контроля за безопасностью информационных систем и повышению их надежности. Это имеет особенно критическое значение в настоящее время в свете постоянно возрастающих угроз в области информационной безопасности. Кроме того, интеллектуальные технологии также применяются для решения разнообразных задач, направленных на защиту различных информационных систем: обнаружение и реагирование на кибератаки, обнаружение мошенничества в бизнес-процессах, управление доступом и аутентификация,

анализ поведения пользователей и устройств, обнаружение вредоносных программ, антифишинг и т д.

В настоящее время значительное количество оперативных и рутинных задач могут быть полностью автоматизированы. Система на основе искусственного интеллекта и машинного обучения может обрабатывать огромные объемы данных, предлагая последующие действия, выявляя возможные проблемы и улучшая эффективность труда. Системы поддержки принятия решений, использующие искусственный интеллект, способны улучшить результаты и привести к новым способам оптимизации, с высокой релевантностью и точностью прогнозируя вероятность исхода или риска, анализируя прошлые, текущие и новые данные, и выявляя, а также предлагая пользователю решения проблем безопасности [8].

В контексте лабораторных информационных систем технологии искусственного интеллекта и машинного обучения могут применяться для анализа временных рядов, оптимизации производственных процессов, прогнозирования качества материалов и создания рекомендательных систем.

Методы анализа временных рядов могут быть использованы для прогнозирования загруженности лабораторного оборудования и персонала, а также для прогнозирования расхода реактивов и материалов, необходимых для проведения анализа.

Оптимизация производственных процессов с применением алгоритмов машинного обучения основывается на данных об испытаниях материалов и процессах производства. Например, путем оптимизации параметров смешивания асфальтобетонной смеси можно достичь повышения эффективности производства.

С помощью алгоритмов машинного обучения также возможно прогнозировать качество получаемых материалов, основываясь на характеристиках и результатах испытаний исходных материалов. Например,

можно предсказать качество готовой асфальтобетонной смеси при использовании определенных марок щебня, песка и дорожного битума.

Для анализа состояния покрытий сооружений и конструкций, а также для обнаружения внешних дефектов может применяться компьютерное зрение. Этот процесс включает в себя распознавание и классификацию дефектов на конструктивных элементах сооружений, оценку объема повреждений и автоматическое создание отчетной документации с указанием типа, изображения дефекта, географических координат и характеристик.

При подборе оптимального состава строительных материалов, таких как строительные смеси, асфальтобетонные и бетонные смеси, рекомендательные системы могут играть значительную роль. На основе анализа требований к проекту, свойств различных материалов и результатов их использования в подобных проектах, рекомендательная система может предложить оптимальный состав материалов для конкретного объекта строительства. Эти рекомендации учитывают различные факторы, такие как климатические особенности места эксплуатации материалов, прочность, долговечность, стоимость и экологичность. Такие рекомендации способствуют оптимизации процесса выбора материалов, что позволяет построить качественные и долговечные конструкции, а также экономно использовать ресурсы.

В рамках данной работы будет проанализирована возможность интеграции библиотеки для оптического распознавания текста (OCR) на изображениях и документах в проектируемую лабораторную информационную систему.

Данная система может применяться для извлечения ценной информации из больших наборов данных, а полученные данные будут обрабатываться и накапливаться для последующего анализа, оценки и построения новых моделей машинного обучения. Со временем система может эволюционировать в полноценную платформу для интеллектуальной обработки документов (IDP) [9], способную автоматически анализировать и

обрабатывать разнообразные типы документов и изображений с использованием различных алгоритмов машинного обучения и искусственного интеллекта. Сравнение возможностей OCR и IDP приведено в таблице 1.1.

Таблица 1.1 – Сравнение возможностей OCR и IDP [10]

	OCR	IDP
Что делает	Читает сканы и изображения документов и преобразует их в машиночитаемые, доступные для поиска документы и/или текст.	Извлекает значимые данные из структурированных (форм), и неструктурированных документов
Как это работает	Применяет алгоритмы ИИ для преобразования бумажных документов в цифровой текст, включая структуру документа и сегментацию (текст, изображения, таблицы, штрихкоды, контрольные знаки, подписи, линии, символы, шрифты, размеры текста, язык и т.д.).	Использует полный текст, предоставленный OCR, как базовый вход и применяет ИИ, NLP, машинное обучение, регулярные выражения и правила для понимания информации в документе и извлечения отмеченных, значимых данных, которые могут быть переданы в бизнес-приложения для принятия решений.
Технология	Улучшение изображений, Обнаружение объектов, OCR / ICR	Классификация Deep Learning NLP Быстрое машинное обучение Правила извлечения Сегментация Распознавание именованных сущностей (NER)

Продолжение таблицы 1

Использование	Предоставляет доступ к машиночитаемому тексту как основе для: PDF-преобразования, цифрового архива, поиска по документам, расширенного поиска, анализа данных, интеллектуальной обработки документов (IDP).	Автоматизация любого процесса обработки документов: автоматизация расчетов по счетам, обработка клиентских анкет автоматизация транспортной и логистической документации

Оптическое распознавание текстов (OCR) представляет собой технологию, позволяющую преобразовывать изображения, содержащие текстовую информацию, в машинно-читаемые данные. Это важный инструмент для автоматизации ввода данных, извлечения информации и создания цифровых архивов.

Существуют различные технологии OCR, которые могут быть классифицированы по их методам обработки изображений и алгоритмам распознавания текста. Среди них можно выделить следующие:

Tesseract OCR – программное обеспечение с открытым исходным кодом. Изначально Tesseract был разработан в Бристольской лаборатории Hewlett-Packard и Hewlett Packard Co (США) в 1985 – 1994 годах. В 2005 году HP открыла исходной код и с 2006 по 2018 годы система поддерживалась и разрабатывалась компанией Google. Tesseract проходит по тексту дважды: в первый раз он распознает текст и одновременно дополнительно обучается, во второй раз распознает то, что не было распознано в первый раз. Tesseract предлагает три предобученные модели: быструю, точную и стандартную, которая поддерживается старыми версиями программы. Начиная с версии 3.0, Tesseract поддерживается русский язык [11].

ABBYY FineReader – является коммерческим программным решением, разработанным российской компанией ABBYY. Разработка первой версии

программы началась в 1992 году в связи с возникшей потребностью при создании комплекса программ Lingvo Systems. Внутренняя технология распознавания текста является коммерческой тайной. Программа продолжает поддерживаться и обновляться, регулярно выходят новые версии и улучшения [12]. С 2022 года в России доступен как Content Reader PDF [13].

EasyOCR – популярная OCR-библиотека, реализованная на языке Python и находящаяся в открытом доступе. Она поддерживает более 80 языков, включая кириллицу, и предоставляет простой интерфейс для интеграции OCR в различные приложения. Библиотека предоставляет готовые модели и возможность обучить собственную модель, полагаясь на pyTorch для реализации нейронных сетей. В качестве модели поиска текста в EasyOCR используется CRAFT (Character Region Awareness for Text Detection) [14], а для распознавания текста – CRNN (Convolutional Recurrent Neural Network), состоящую из трех основных компонентов: извлечение признаков (в настоящее время используется ResNet и VGG), маркировка последовательностей (sequence labeling) (LSTM) и декодирование (CTC) (рисунок 1.1). Основными преимуществами EasyOCR являются простой и понятный интерфейс, а также высокое качество распознавания текста. Однако главным недостатком является медленное время работы при использовании процессора вместо видеокарты с поддержкой CUDA, что ограничивает список устройств, на которых библиотека функционирует оптимально [15].

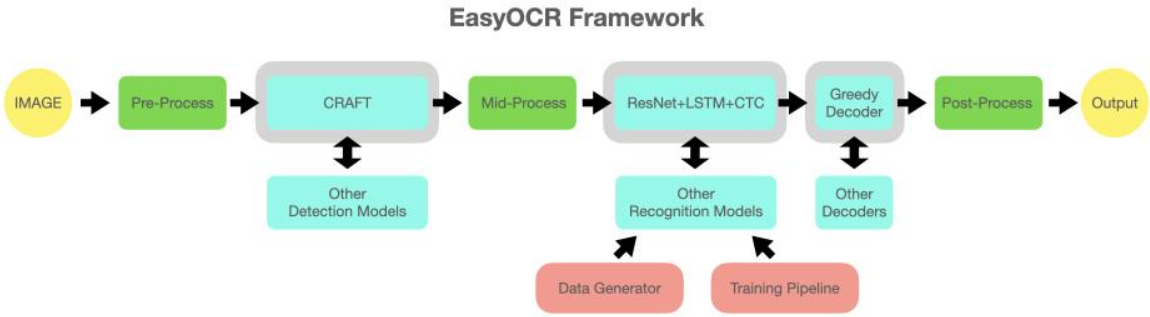

Рисунок 1.1 – Архитектура EasyOCR [16]

PaddleOCR – это библиотека для OCR, основанная на фреймворке PaddlePaddle, разработанном компанией Baidu. Она предназначена для распознавания текста на изображениях и поддерживает множество языков, включая китайский, английский и русский. Она считается одной из лучших библиотек OCR с открытым исходным кодом, доступных на данный момент [17; 18]. Схема PaddleOCR приведена на рисунке 1.2.

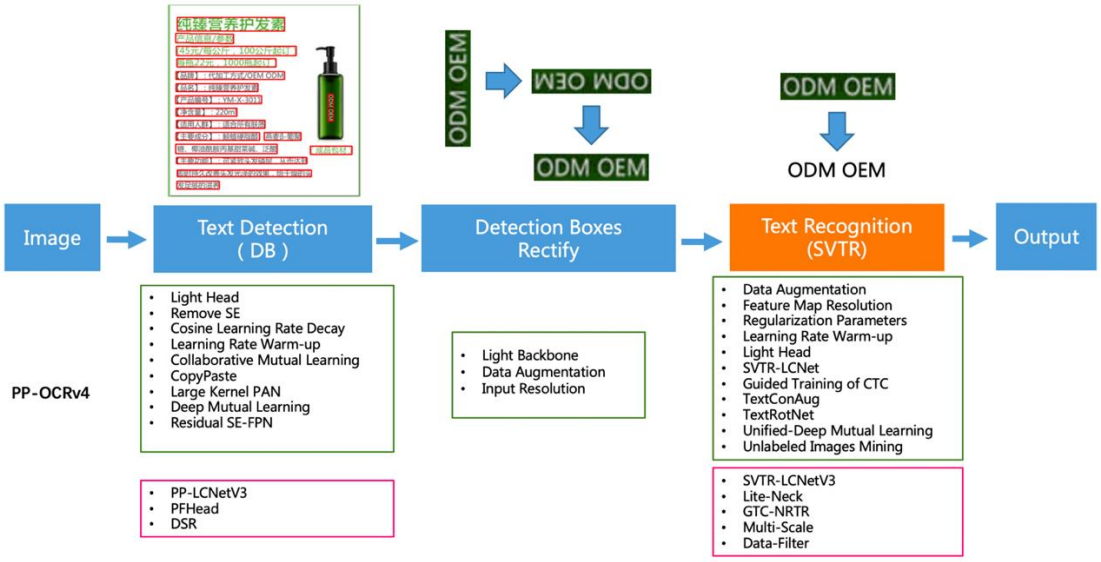

Рисунок 1.2 – Схема PaddleOCR

Программные обеспечения и облачные сервисы, такие как ABBYY Fine Reader (Content AI), Yandex Vision OCR [19], Amazon Textract, Microsoft Azure Computer Vision, Google Cloud Vision не будут рассматриваться в данной работе по нескольким причинам. Во-первых, эти сервисы являются платными и проприетарными. Во-вторых, серверы некоторых из них находятся за пределами Российской Федерации, что может представлять риск для безопасности данных. Наконец, документы организаций, которые будут обрабатываться ЛИС, могут содержать конфиденциальные данные и коммерческую тайну, что также ограничивает использование данных сервисов.

Список рассмотренных систем OCR не охватывает всё многообразие существующих решений. Существует множество других систем, которые не были рассмотрены в рамках данного исследования по ряду причин, включая прекращение поддержки, отсутствие поддержки кириллических символов, а также временные и ресурсные ограничения данного исследования.

1.4 Выводы по главе 1

В первой главе была проведена комплексная оценка существующих подходов к проектированию лабораторных информационных систем, а также рассмотрены возможности интеграции алгоритмов машинного обучения в эти системы.

Информационные технологии играют ключевую роль в автоматизации рутинных операций в лабораториях. Лабораторные информационные системы применяются для управления различными аспектами лабораторной деятельности, включая планирование работы, регистрацию проб и формирование отчетов. Представленные на российском рынке решения не охватывают весь спектр потребностей различных отраслей промышленности, что создает вызов в разработке современных решений. Кроме того, существующие информационные системы для строительной отрасли не включают алгоритмы машинного обучения, что ограничивает их потенциал.

Машинное обучение и искусственный интеллект могут значительно улучшить функциональность лабораторных информационных систем. Методы машинного обучения могут использоваться для анализа временных рядов, оптимизации производственных процессов, прогнозирования качества материалов и создания рекомендательных систем. Интеграция библиотеки OCR позволяет автоматизировать процесс извлечения и обработки данных, что способствует повышению производительности и качества лабораторных исследований.

В результате проведенного анализа выяснилось, что для оптимизации лабораторных процессов требуется спроектировать современную лабораторную информационную систему с интеграцией алгоритмов машинного обучения. Важным элементом такой системы является создание микросервиса для распознавания текста (OCR). Этот микросервис обеспечит автоматизацию извлечения текстовой информации из изображений и документов, ускорив обработку данных и снизив вероятность ошибок.

Проектирование и внедрение такой системы станет важным шагом в направлении повышения эффективности лабораторных исследований, усовершенствования сбора данных и оптимизации использования ресурсов.

2. АРХИТЕКТУРА ЛАБОРАТОРНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1. Выбор инструментов и стека технологий

Архитектура лабораторной информационной системы (ЛИС) играет ключевую роль в обеспечении её функциональности, надежности и масштабируемости. Система, предназначенная для строительной отрасли, должна эффективно справляться с обработкой большого объема данных, поддерживать высокую точность анализа и предоставлять удобный доступ к информации для различных пользователей. ЛИС должна быть способна интегрироваться с различными внешними системами и технологиями, включая системы оптического распознавания текстов (OCR) для автоматизации ввода данных из бумажных документов.

Создание ЛИС для строительной отрасли требует тщательного подхода к выбору технологий и архитектурных решений. Необходимо учесть особенности работы с данными строительных лабораторий, специфику анализа и обработки данных, а также требования к безопасности и производительности системы. В этой главе рассматриваются основные компоненты архитектуры ЛИС, обосновывается выбор Ruby on Rails для разработки основного приложения и использование микросервисной архитектуры для реализации OCR.

В настоящее время актуальным программным среди множества сред разработки обладает высокоуровневый фреймверк Ruby on Rails (RoR). Его возможности могут значительно повысить производительность веб-программирования по сравнению с такими технологиями как PHP и Perl. Язык Ruby, лежащий в основе фреймворка Ruby on Rails, имеет быстрый цикл разработки, реализован как интерпретатор и поддерживает нетипизированные переменные, которые не требуют объявлений. Это язык, в котором парадигма

функционального программирования сбалансирована принципами императивного программирования. Ruby on Rails позволяет получить высокую надежность и гибкость при разработке проектов любого размера. Фреймворк включает в себя тестовую подсистему, которая значительно экономит время разработки и улучшает качество веб-проекта [20].

Ruby on Rails – это фреймворк для веб-приложений, написанный на языке программирования Ruby. Он реализует архитектурный шаблон Model-View-Controller (MVC), обеспечивая интеграцию веб-приложений с веб-сервером и сервером баз данных. Ruby on Rails является открытым программным обеспечением и распространяется под лицензией MIT [21].

Множество сервисов мирового уровня разработаны с использованием языка программирования Ruby, включая такие как Shopify, Coinbase, GitHub, Airbnb, Kickstarter, Square, Twitch, Basecamp, HEY и другие [22]. Ruby входит в топ-10 языков программирования для серверной разработки, что подтверждает его популярность и эффективность в данной области [23; 24].

Использование Ruby on Rails обусловлено его возможностями для быстрой и эффективной разработки веб-приложений, поддержкой принципов модульности и масштабируемости.

Для лабораторной информационной системы Ruby on Rails был выбран по следующим причинам:

Эффективность разработки: RoR позволяет быстро разрабатывать и внедрять новые функции, что особенно важно в условиях быстро меняющихся требований. Он также является подходящим инструментом для создания MVP (Minimum Viable Product) приложений, что важно для проверки гипотез при разработке стартапов.

Надежность и устойчивость: RoR предоставляет множество инструментов для тестирования и отладки, что способствует созданию надежного и устойчивого приложения.

Масштабируемость: Возможность легко масштабировать приложение позволяет обеспечить его работу с увеличивающимися объемами данных и пользователей.

Микросервисная архитектура, в свою очередь, позволяет выделить функции OCR в отдельный сервис, что упрощает управление, обновление и масштабирование этой части системы без влияния на основное приложение.

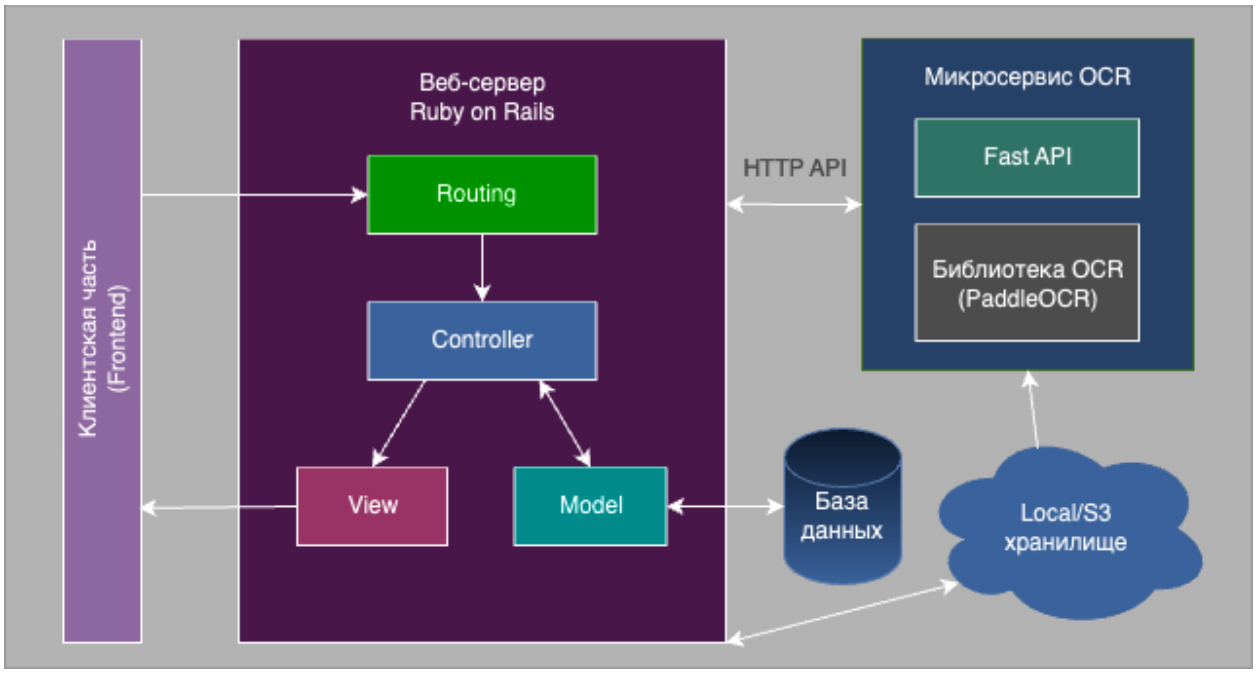

Рисунок 2.1 – Архитектура приложения

Основные компоненты архитектуры ЛИС и их функции.

1) Пользовательский интерфейс (UI/Frontend), HTML5, CSS3, JavaScript:

– Ввод данных и управление ими;

– Отображение результатов анализа и отчетов;

– Интерактивные графики и визуализации данных;

– Управление пользователями и правами доступа.

2) Серверное приложение (Backend), Ruby on Rails:

– Обработка запросов от пользовательского интерфейса;

– Управление бизнес-логикой;

– Взаимодействие с базой данных;

– Взаимодействие с сервисом OCR;

– Обработка данных и выполнение анализа.

3) База данных (Database), реляционная база данных PostgreSQL:

– Хранение результатов лабораторных испытаний;

– Хранение спецификаций материалов;

– Хранение технических отчетов и документов;

– Хранение информации о лабораторном оборудовании и реактивах для проведения испытаний;

– Хранение результатов, выданных от сервиса OCR;

– Хранение пользовательских данных и прав доступа.

4) Микросервис для OCR, python, библиотека PaddleOCR:

– Преобразование изображений в текст;

– Предварительная обработка изображений (удаление шумов, коррекция яркости и контрастности);

– Интеграция с основным серверным приложением через API;

API и интеграционные слои:

– Взаимодействие между пользовательским интерфейсом и серверным приложением;

– Обмен данными между серверным приложением и микросервисом для OCR;

– Интеграция с внешними системами и сервисами (например, системы управления проектами, бухгалтерские системы).

5) Потоки данных и управления в ЛИС организованы следующим образом:

– Пользователь вводит данные через интерфейс, данные отправляются на серверное приложение через API;

– Серверное приложение обрабатывает запросы, взаимодействует с базой данных для хранения и извлечения данных;

– Если требуется распознавание текста с изображения, серверное приложение направляет изображение в микросервис для OCR;

– Микросервис для OCR обрабатывает изображение, преобразует его в текст и возвращает результат серверному приложению;

– Серверное приложение обрабатывает полученные данные и обновляет базу данных;

– Обработанные данные и результаты анализа отображаются в пользовательском интерфейсе.

2.2. Микросервис для OCR

Микросервисная архитектура представляет собой подход к разработке программного обеспечения, при котором приложение разбивается на небольшие, независимые сервисы, каждый из которых отвечает за выполнение одной функции. Этот подход обеспечивает ряд значительных преимуществ, таких как модульность, масштабируемость, независимость и устойчивость к сбоям. Разделение приложения на небольшие сервисы упрощает разработку, тестирование и обслуживание каждого компонента. Масштабируемость достигается за счет возможности независимого масштабирования каждого микросервиса, что позволяет эффективно использовать ресурсы. Кроме того, микросервисы могут быть разработаны и развернуты независимо друг от друга, что упрощает управление изменениями и обновлениями. Устойчивость к сбоям обеспечивается тем, что сбой одного микросервиса не приводит к отказу всего приложения, повышая общую надежность системы. Важным преимуществом является также возможность разработки микросервисов на различных языках программирования, что позволяет использовать наиболее подходящие технологии для каждой конкретной задачи.

Использование микросервисной архитектуры для реализации OCR в лабораторной информационной системе (ЛИС) имеет ряд значительных

преимуществ. Во-первых, обработка изображений и распознавание текста требуют значительных вычислительных ресурсов. Выделение OCR в отдельный микросервис позволяет изолировать эти процессы и избежать влияния на производительность основного приложения. Во-вторых, микросервис для OCR можно легко обновлять и улучшать без необходимости изменения основного приложения, что обеспечивает гибкость и возможность быстрого внедрения новых функций и улучшений. Наконец, микросервисная архитектура позволяет интегрировать библиотеку PaddleOCR, которая обеспечивает высокую точность и производительность распознавания текста. Микросервисная архитектура для сервиса OCR была выбрана по причине того, что PaddleOCR разработан на Python, что упрощает разработку и поддержку этого сервиса.

Файлы изображений и документы, которые должны обрабатываться микросервисом OCR, могут храниться в нескольких местах, в зависимости от архитектурных решений и требований к системе.

Локальное хранилище на сервере – файлы сохраняются на локальном диске сервера, где находится серверное приложение или микросервис для OCR. Преимуществами такого подхода являются простота реализации и быстрота доступа. Однако существуют и недостатки, такие как ограниченное дисковое пространство и сложности с масштабированием.

Облачное S3-хранилище – файлы загружаются и хранятся в облачных хранилищах, таких как Yandex Object Storage, VK Cloud Storage, Amazon S3, Google Cloud Storage и т. д. Преимущества этого варианта включают масштабируемость, высокую доступность и надежность, а также обеспечение безопасности и управление доступом. Недостатками являются зависимость от интернет-соединения и возможные задержки при доступе к файлам.

База данных – файлы сохраняются в базе данных в виде BLOB (Binary Large Object). Преимуществами такого подхода являются централизованное управление данными и встроенные механизмы резервного копирования.

Однако увеличение размера базы данных может привести к проблемам с производительностью при работе с большими файлами.

Файловый сервер – файлы хранятся на отдельном файловом сервере, к которому имеют доступ серверное приложение и микросервис для OCR. Преимуществами данного решения являются разделение хранения и обработки данных, а также возможность масштабирования файлового хранилища. Недостатками являются необходимость настройки и администрирования файлового сервера, а также возможные задержки при доступе к файлам.

Для приложений начального уровня, как MVP можно использовать локальное хранилище, что обеспечивает простоту реализации и быстроту доступа, а также снижает затраты на покупку доступа к S3-хранилищам. Со временем, по мере роста системы и увеличения объема данных, можно перейти на использование собственного хранилища на базе S3 или облачного S3-хранилища, что обеспечит высокую стабильность, масштабируемость и надежность.

Для данной работы было выбрано локальное хранилище и основной пользовательский кейс будет выглядеть таким образом:

1) Загрузка файла:

– Пользователь загружает изображение или документ через веб-интерфейс;

– Файл отправляется на серверное приложение, которое сохраняет его на локальном диске сервера.

2) Обработка файла:

– Серверное приложение сохраняет путь к файлу в базе данных;

– Когда требуется распознавание текста, серверное приложение отправляет запрос микросервису для OCR, передавая путь к файлу по HTTP API;

– Микросервис загружает файл с локального диска, выполняет распознавание текста и возвращает результат серверному приложению по HTTP API в формате json.

3) Сохранение результатов:

– Серверное приложение сохраняет результаты распознавания в базе данных;

– Пользователь может просматривать результаты через веб-интерфейс.

Этот подход позволяет быстро разрабатывать и тестировать систему, а в будущем, по мере необходимости, легко перейти на более масштабируемое решение, такое как собственное или облачное S3-хранилище.

2.3. Библиотека распознавания текста PaddleOCR

Современные модели распознавания текстов в сценах обычно состоят из двух основных компонентов: визуальной модели для извлечения признаков и последовательной модели для транскрипции текста. Эта гибридная архитектура, хотя и точна, является сложной и менее эффективной. Одним из представителей таких сетей является CRNN, который был представлен еще в 2015 году. CRNN состоит из CNN блока, который распознает признаки, BiLSTM блока, который обрабатывает эти полученные признаки и с помощью блока CTC декодируется в текст. Архитектура CRNN сети продемонстрирована на рисунке 2.2.

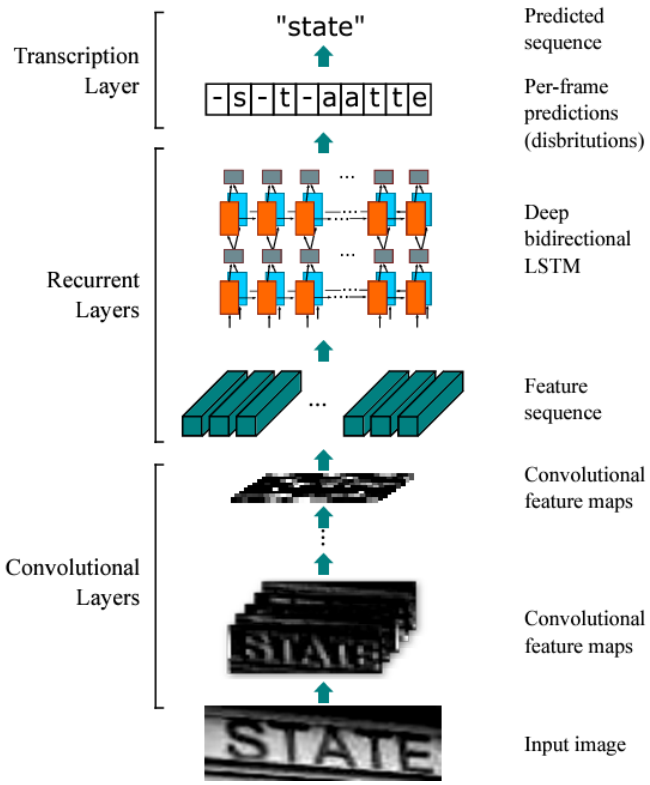

Рисунок 2.2 – Архитектура CRNN сети

В данном исследовании [25; 26] предлагают единую визуальную модель для распознавания текстов в сценах в рамках подхода токенизации изображения по компонентам, полностью исключая последовательное моделирование. Этот метод, названный SVTR, сначала разбивает текст изображения на небольшие компоненты, называемые компонентами символов. Затем, используя механизм self-attention между этими компонентами, модель извлекает важную информацию, опираясь как на локальные, так и на глобальные признаки. После этого, уменьшая размерность и объединяя признаки после блоков self-attention, модель формирует многогранное представление текста на изображении. В конечном итоге модель генерирует последовательность признаков, в которой текст уже закодирован.

Предложенная архитектура SVTR для распознавания текстов в сценах показывает значительные преимущества в точности и скорости по сравнению с традиционными гибридными моделями. Исключение последовательного

моделирования и использование подхода токенизации изображения по патчам позволили значительно упростить архитектуру, сохранив высокую производительность. Модели SVTR-L и SVTR-T демонстрируют свою эффективность в различных задачах распознавания текста, подтверждая потенциал предложенного подхода.

В архитектуре SVTR на входе расположен блок STN (Spatial Transformer Network) [27], который выпрямляет текст на исходном изображении (рисунок 2.3). Главное преимущество использования STN по сравнению с другими методами выпрямления текста, такими как аффинные преобразования, заключается в отсутствии необходимости ручной разметки, поскольку сеть обучается за счет градиентов, распространяющихся от сети распознавания.

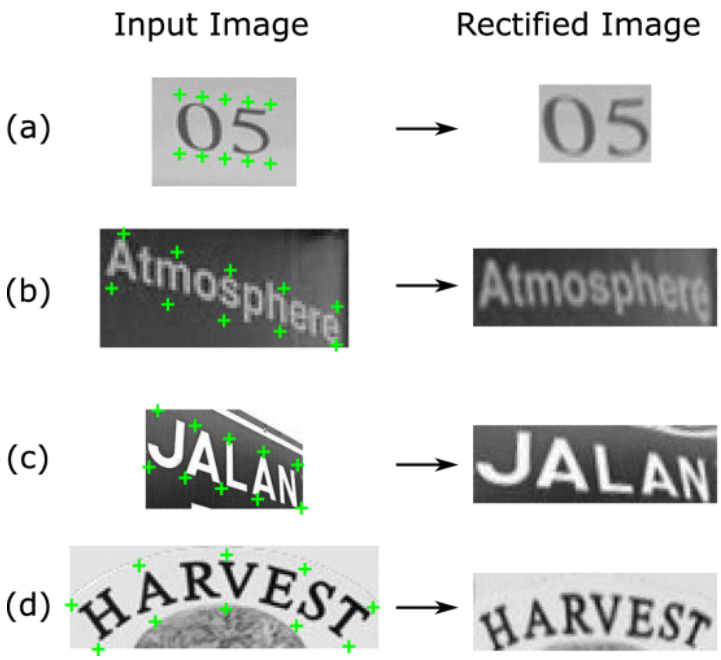

Рисунок 2.3 – Преобразование искаженного текста блоком STN

Далее, новое изображение с выпрямленным текстом подаётся на вход блоку Patch Embedding. Этот блок разбивает входное изображение на патчи (малые части одинакового размера) и представляет каждый патч в виде вектора. Такой подход был впервые предложен в исследовании от команды Google Research, Brain Team [28] и предполагал разбиение изображения на

непересекающиеся патчи размером (16, 16) с последующим применением линейного преобразования к каждому из них. Однако в SVTR для кодирования патчей размером (4, 4) было предложено использовать два свёрточных слоя размером (3, 3) с шагом 2. Этот подход несколько увеличивает вычислительные затраты, но даёт более детализированное представление о каждой части изображения.

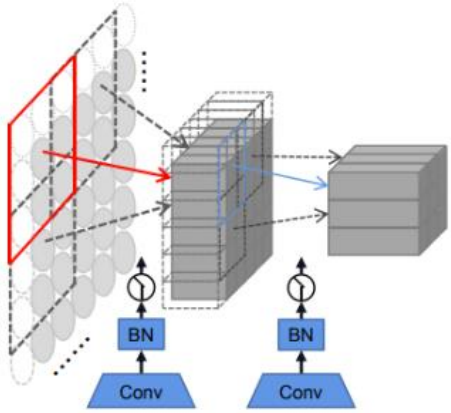

Рисунок 2.4 – Схема работы Patch Embedding

К выходу Patch Embedding добавляется position embedding для передачи информации о позиции каждого патча на изображении. В оригинальной статье каждый полученный эмбеддинг обозначается как character component, который для упрощения мы будем называть компонентом. Полученный набор компонентов представляет собой тензор размером (H/4, W/4, D_0), где D_0 — размер компонента (эмбеддинга), H и W — высота и ширина исходного изображения соответственно.

Одной из наиболее распространенных проблем при распознавании текста является неправильная классификация похожих символов, таких как цифра "0" и буква "O". Правильное определение класса символа значительно зависит от качества выделенных признаков из соответствующей ему области изображения. В существующих исследованиях выделенные признаки обычно представляют собой последовательность, в которой закодирован текст, и

каждый признак соответствует небольшой области изображения. Часто эти признаки содержат шум, особенно если текст расположен неровно. Основная сложность заключается в том, что обработка фокусируется на получении последовательности, в которую уже закодирован текст, что недостаточно для точного описания каждого символа отдельно.

Для решения данной проблемы авторы статьи предложили выделять два типа признаков: локальные и глобальные. Локальные признаки представляют собой паттерны, такие как штрихи, извлеченные из определенной области изображения. Эти признаки содержат морфологические характеристики и корреляции между компонентами конкретной части изображения, что помогает идентифицировать каждый отдельный символ. Глобальные признаки, в свою очередь, определяют зависимости между всеми компонентами изображения. Поскольку каждая часть изображения может относиться либо к тексту, либо к нетекстовой области, вычисление глобальных признаков позволяет установить долгосрочные зависимости между текстовыми компонентами, снижая влияние нетекстовых областей и придавая большее значение тексту.

Для извлечения локальных и глобальных признаков авторы разработали слои global mixing и local mixing. Global mixing реализован в виде multi-head self-attention слоя, который вычисляет зависимости между всеми компонентами. Local mixing вычисляет взаимосвязи только с соседними компонентами, используя маску, аналогичную операции свертки. Local mixing проходит скользящим окном размером (7, 11) по всему набору компонентов с шагом (3, 5), вычисляя зависимости на каждом шаге.

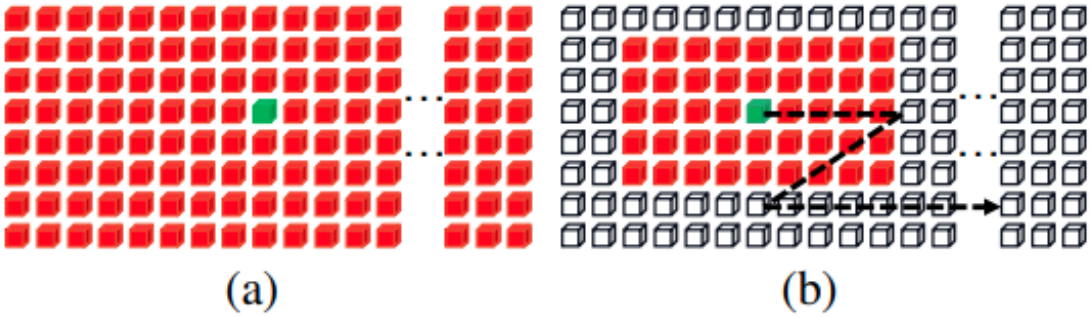

Рисунок 2.5 – Иллюстрация Global (a) и Local (b) mixing

Полученный от Patch Embedding тензор, размером (H/4, W/4, D_0) проходит три стадии обработки.

На каждой стадии обработки сначала изменяется размер входного тензора из (h,w,d) в (h*w,d), где,

h — количество компонентов по высоте,

w — количество компонентов по ширине,

d — размер компонента.

Затем применяется последовательность блоков смешивания (mixing blocks), которые состоят из нормализации слоя (layer normalization), локального или глобального смешивания (local или global mixing), нормализации слоя и MLP (многоуровневого персептрона) с добавлением shortcut connections. После применения нескольких блоков смешивания, полученный тензор снова преобразуется в тензор размером (h,w,d).

Поскольку обработка с постоянным пространственным разрешением является вычислительно затратной и приводит к избыточному представлению, полученный тензор проходит через блок объединения (merging), состоящий из слоя свёртки и нормализации слоя, который уменьшает размер тензора по высоте в 2 раза, увеличивая количество каналов, но оставляя ширину неизменной. Эта операция не только снижает вычислительные затраты, но и способствует изучению изображения в разном масштабе, не влияя на

пространственную информацию по ширине, что уменьшает вероятность кодирования соседних символов в один.

После прохождения двух стадий обработки через блоки смешивания и объединения, на третьей (заключительной) стадии блок объединения заменяется на блок combing. При входном изображении с высотой 32, на выходе третьего блока смешивания образуется тензор высотой 2, поэтому операция свёртки заменяется на слой pooling, за которым следует полносвязный слой, нелинейная функция активации и dropout. Таким образом, в конце третьей стадии обработки формируется тензор размером (1, W/4, D3).

После объединения признаков блоком combing следует последний блок — полносвязный слой, на выходе которого получается последовательность размером W/4, состоящая из компонентов размером N, где N – количество символов для предсказания. Предсказанная последовательность представляет собой закодированный текст, где каждый компонент, соответствующий одному символу, предсказывает один определенный класс, а компоненты, не относящиеся к тексту, предсказывают класс blank.

Архитектура SVRT представлена на рисунке 2.6.

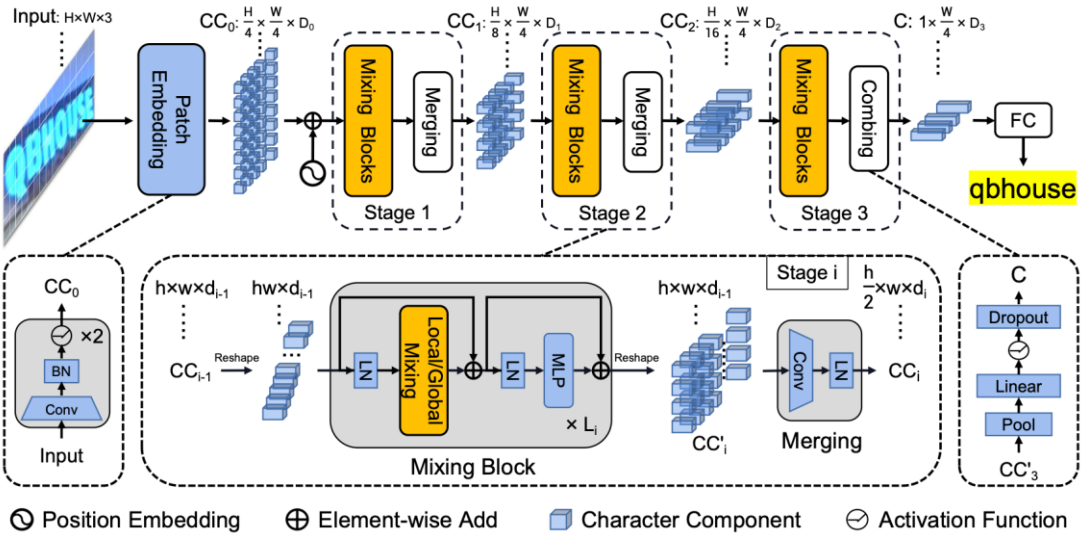

Рисунок 2.6 – Архитектура SVRT

Авторами исследования было предложено следующие архитектуры SVTR представленные на рисунке 2.7.

Models	$[D_0, D_1, D_2]$	$[L_1, L_2, L_3]$	Heads	D_3	Permutation	Params (M)	FLOPs (G)
SVTR-T	[64,128,256]	[3,6,3]	[2,4,8]	192	$[L]_6[G]_6$	4.15	0.29
SVTR-S	[96,192,256]	[3,6,6]	[3,6,8]	192	$[L]_8[G]_7$	8.45	0.63
SVTR-B	[128,256,384]	[3,6,9]	[4,8,12]	256	$[L]_8[G]_{10}$	22.66	3.55
SVTR-L	[192,256,512]	[3,9,9]	[6,8,16]	384	$[L]_{10}[G]_{11}$	38.81	6.07

Рисунок 2.7 – Варианты архитектур SVRT

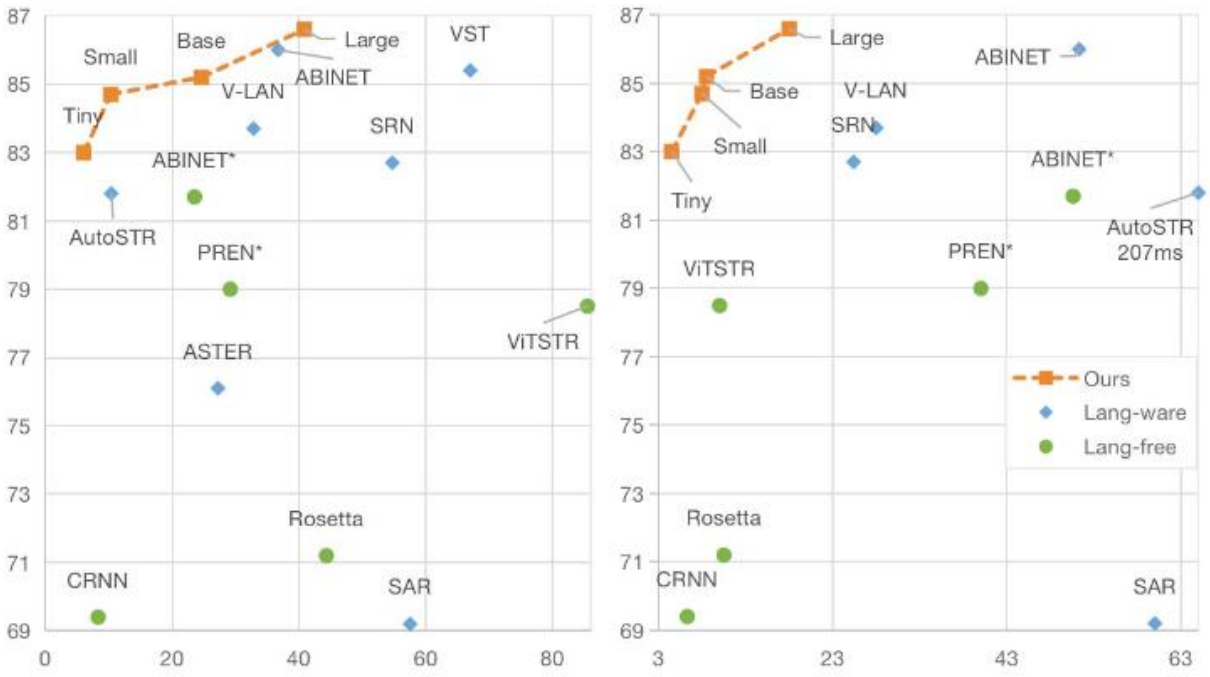

Рисунок 2.8 – Сравнение разных моделей на датасете IC15

В данном параграфе была рассмотрена специализированная визуальная модель для распознавания текстов в сценах – SVTR, которая лежит в основе PaddleOCR. Эта модель извлекает многоуровневые признаки символов, описывающие как локальные паттерны, похожие на штрихи, так и зависимости между компонентами на различных масштабах высоты. Это позволяет выполнять задачу распознавания с использованием одной

визуальной модели, обладающей такими преимуществами, как точность, эффективность и кросс-лингвистическая универсальность.

Предложенные версии SVTR с различной архитектурой можно применять в различных приложениях. Эксперименты на контрольных наборах данных для английского и китайского языков подтвердили высокую эффективность предложенной модели. SVTR демонстрирует конкурентоспособную точность и скорость по сравнению с современными методами.

Использование SVTR в PaddleOCR подчеркивает его важность и эффективность, предлагая передовые возможности для распознавания текстов в различных сценариях.

2.4 Выводы по главе 2

Вторая глава посвящена разработке архитектуры лабораторной информационной системы (ЛИС) и выбору инструментов и стека технологий для ее реализации. Основное внимание уделено обеспечению функциональности, надежности и масштабируемости системы.

Архитектура ЛИС включает клиентскую часть, серверную часть и микросервис для обработки распознавания текста (OCR). Для серверной части был выбран фреймворк Ruby on Rails, который обеспечивает быструю и эффективную разработку веб-приложений. Он поддерживает модульность, масштабируемость и предоставляет множество инструментов для тестирования и отладки.

Микросервисная архитектура позволяет выделить функции OCR в отдельный сервис, что упрощает управление, обновление и масштабирование этой части системы. Для реализации OCR был выбран микросервис на основе библиотеки с открытым исходным кодом PaddleOCR, которая

продемонстрировала высокую производительность и точность распознавания текста по сравнению с аналогичными библиотеками.

Были рассмотрены способы хранения файлов изображений и документов, обрабатываемых микросервисом OCR, такие как локальное хранилище, облачное S3-хранилище, база данных и файловый сервер. Было принято решение на стадии MVP выбрать локальное хранилище как предпочтительное решение из-за своей простоты и быстроты доступа. Также отмечено, что можно перейти на S3-хранилище по мере роста приложения.

Была рассмотрена модель SVTR, лежащая в основе PaddleOCR, и механизм работы архитектуры SVTR. Было отмечено, что модель SVTR, исключающая последовательное моделирование и использующая токенизацию изображения по компонентам, демонстрирует значительные преимущества в точности и скорости по сравнению с традиционными гибридными моделями, такими как CRNN.

Таким образом, был сделан вывод, что разработанная архитектура лабораторной информационной системы, основанная на микросервисной модели и использующая современные технологии, такие как Ruby on Rails и PaddleOCR, обеспечивает высокую производительность, гибкость и масштабируемость. Применение модели SVTR для распознавания текста демонстрирует значительные преимущества по точности и скорости, делая данную систему перспективным решением для автоматизации и оптимизации лабораторных процессов в строительной отрасли.

3. ЭКСПЕРИМЕНТЫ И РЕЗУЛЬТАТЫ

3.1 Описание датасета

Для обучения модели было принято решение подготовить данные. Вначале была исследована возможность использования готовых размеченных данных на кириллице в открытом доступе, однако такие данные в приемлемом качестве найти не удалось. Поэтому было решено сгенерировать синтетические данные на основе списка слов на русском языке.

Учитывая специфику задачи, при генерации были применены разные механизмы для разнообразия данных, такие как наклон текста в изображении под определенным углом, волнообразная деформация текста, применение различных цветов текста и шрифтов, размытие изображения, применение неоднородного фона и т.д. Эти механизмы способствуют улучшению обучения модели, обучая её на разнообразных примерах, которые могут встречаться в реальных условиях. Также было решено, что текст на изображениях должен содержать от одного до трех случайно выбранных из словаря слов.

Для генерации синтетических данных использовался инструмент TextRecognitionDataGenerator (TRDG) [29]. Во время генерации датасета возникла проблема, когда текст на изображениях не отображался, и вместо букв появлялись вопросительные знаки и символы прямоугольника (рисунок 3.1). Проблема была вызвана использованием шрифтов, не поддерживающих кириллицу. Для решения этой проблемы, был выбран пул часто используемых шрифтов, поддерживающих кириллицу, и передан на вход генератора.

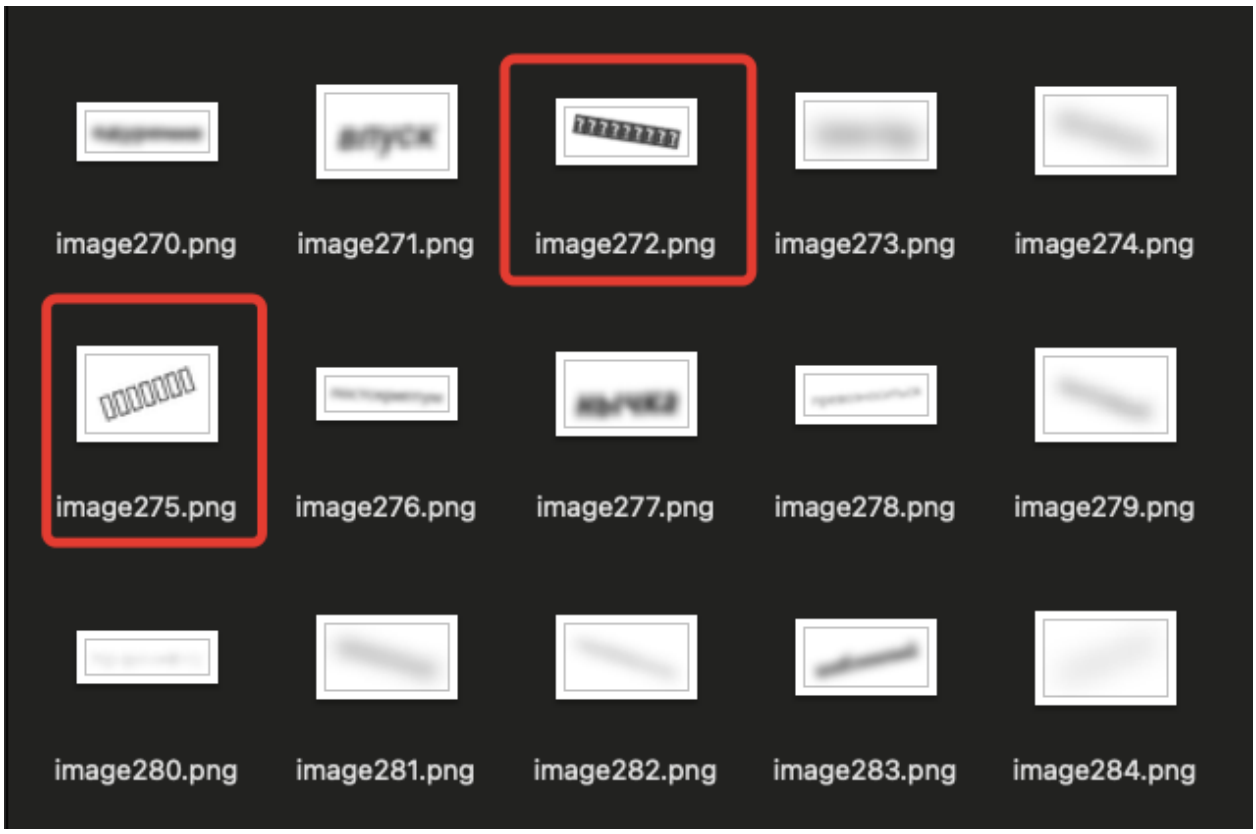

Рисунок 3.1 – Пример набора изображений с неподдерживаемыми шрифтами

Таким образом, было подготовлено около 10 тысяч изображений для обучения, 2000 для валидации и 500 для теста, что составляет 80:16:4 процента соответственно.

Источником данных стал репозиторий на GitHub со списком слов на русском языке [30; 31], включающий более 1,5 млн слов во всех морфологических формах.

TRDG – удобный инструмент, который можно использовать как в консоли, так и импортировать в Python-скрипты. В случае необходимости быстрой генерации тестовых данных можно выполнить генерацию непосредственно из командной строки. Для этого необходимо склонировать репозиторий библиотеки с GitHub, перейти в директорию библиотеки и запустить скрипт с передачей необходимых параметров:

```
git clone
https://github.com/Belval/TextRecognitionDataGenerator.git
```

```
// переходим в директорию trdg
cd TextRecognitionDataGenerator/trdg
// вызываем скрипт, указывая параметры
python3 run.py -c 10 -k 15 \
     -rk -bl 0.5 -rbl \
     -tc '#000000,#888888' -f 64
```

где, c – количество генерируемых изображений, k(rk) – skew_angle, random_skew, угол наклона текста и случайный наклон текста, bl, rbl – blur, random_blur, степень размытия и случайное размытие текста, f – размер шрифта текста, tc – text_color, цвет текста.

В данной работе для удобства генерации данных был разработан скрипт [32], который при запуске создает необходимые директории, генерирует изображения с текстом с заданными параметрами аугментации и создает текстовый файл с названиями файлов изображений и соответствующим текстом, содержащимся в сгенерированных изображениях. Примеры полученных данных и структура директории с данными приведены на рисунках 3.2 и 3.3.

После получения текстового файла от библиотеки TRDG, его необходимо привести в формат, который поддерживает PaddleOCR (рисунок 3.4), чтобы избежать ошибок при обучении. Для этого воспользуемся скриптом из репозитория PaddleOCR, вызвав его в командной строке с указанием исходного и результирующего текстовых файлов:

```
python PaddleOCR/ppocr/utils/gen_label.py
--mode="rec"
--input_path=path/to/train/labels.txt
--output_label=path/to/train_labels.txt
```

Более полный список команд и скрипт будет приведен в git репозитории по ссылке [32].

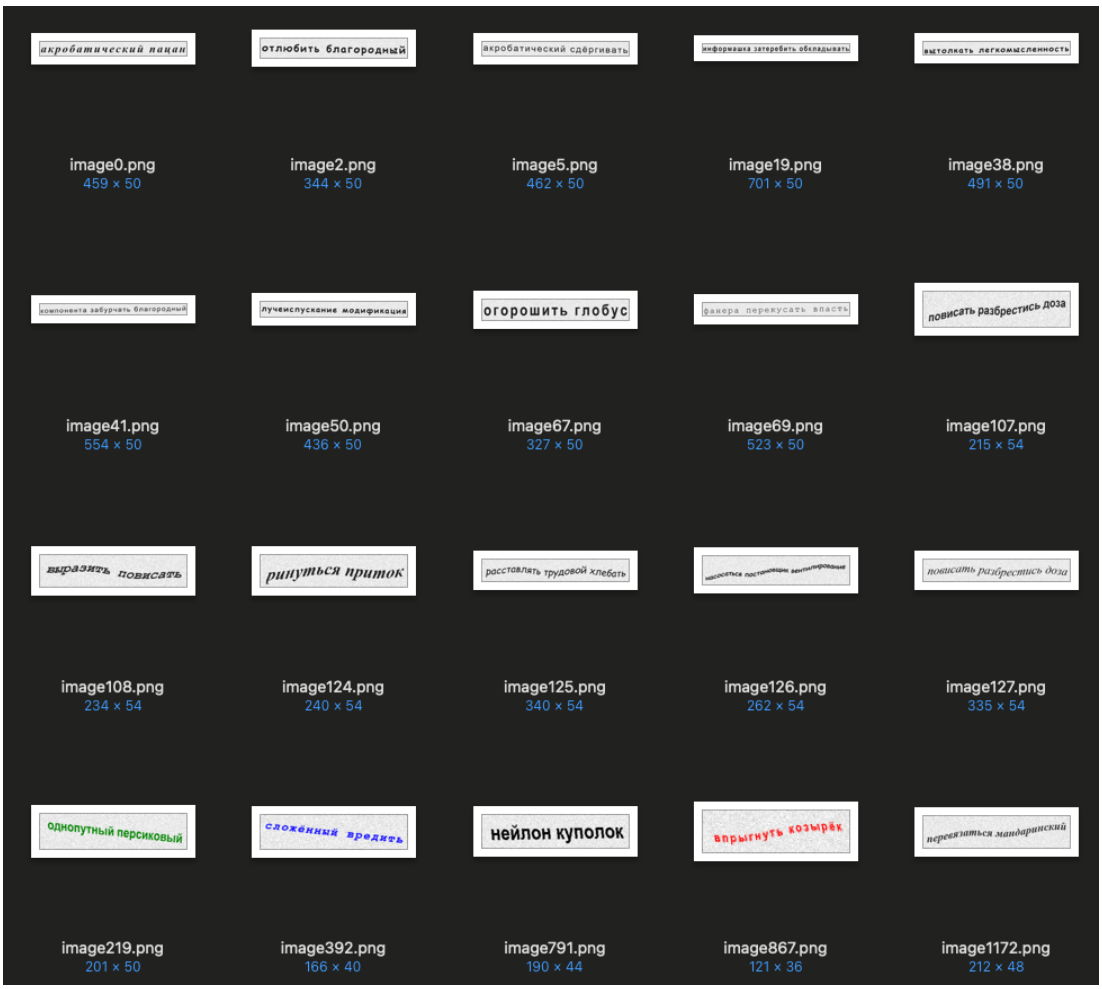

Рисунок 3.2 – Пример набора данных, сгенерированных библиотекой TRDG

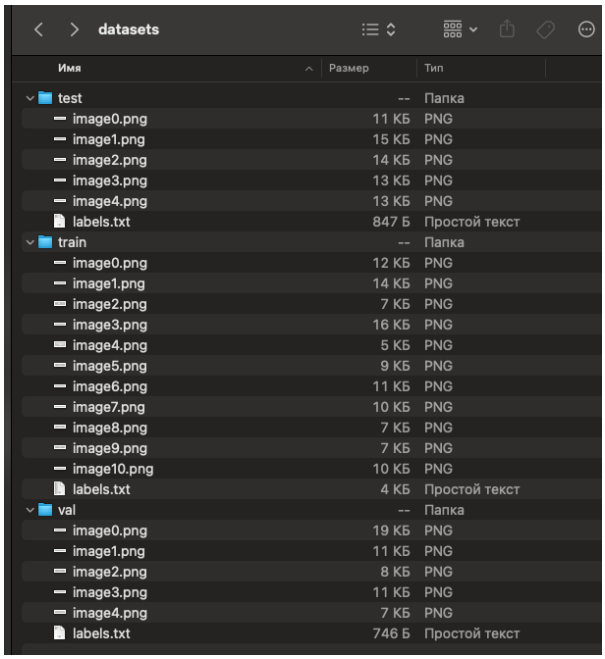

Рисунок 3.3 – Пример структуры данных для обучения

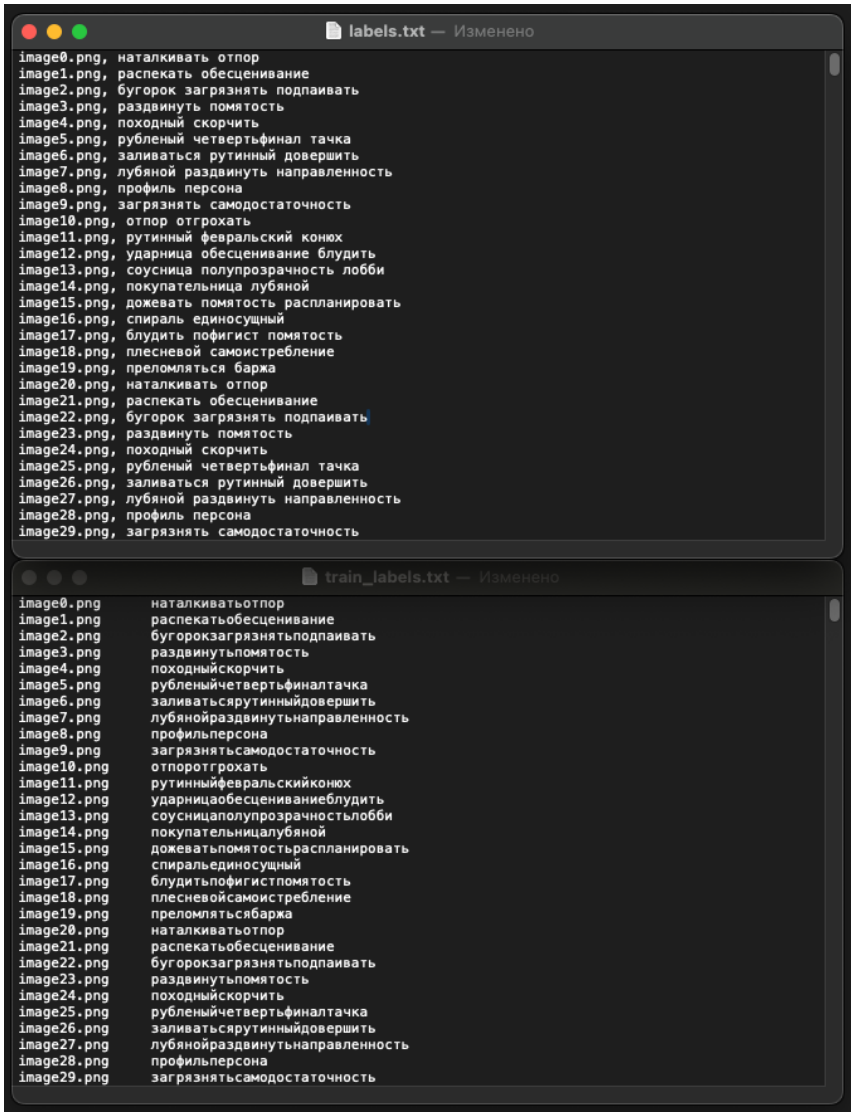

Рисунок 3.4 – Пример исходного и обработанного текстового файла с тегами, верхняя часть до, нижняя после приведения аннотаций в формат PaddleOCR

Таким образом, были получены данные для обучения, валидации и тестирования, соответствующие задачам исследования. Инструмент для генерации синтетических данных для обучения модели распознавания документов оказался вполне приемлемым. Если требуется обучить модель распознавать текст на фотографиях или дорожных знаках, то лучше подготовить набор данных, приближенный к такому сценарию. Например, инструмент SynthText подойдет для такого случая [33; 34]. Этот инструмент

генерирует данные, используя различные фоновые изображения и виды деформаций текста в изображении. Отличительной особенностью этой библиотеки является высокая вариативность генерируемых изображений и изгибов генерируемых слов, разнообразие фоновых изображений и алгоритм размещения слов в специальных областях изображения, что создает эффект наложения текста на реальный объект. Это особенно полезно для задач распознавания текста с рекламных вывесок, дорожных знаков и т.д. Пример данных, созданных с помощью SynthText, приведен на рисунке 3.4.

Рисунок 3.5 – Пример набора данных, сгенерированных SynthText

Список возможных инструментов для генерации данных приведен в документации к PaddleOCR. В него входят такие инструменты, как

Text_renderer, SynthText, SynthText (Chinese_version), TextRecognitionDataGenerator, SynthText3D, UnrealText, SynthTIGER [17].

Для специальных задач, требующих уникальных и специфических данных для обучения, существуют несколько способов подготовки данных. Один из них - разметка данных вручную с помощью инструмента от разработчика PaddleOCR - PPOCRLabelv2, либо с использованием стороннего программного обеспечения, такого как Labelme [35]. Также можно воспользоваться услугами сервисов, таких как Toloka [36], где за определенную сумму специалисты произведут разметку датасета. Однако использование ручного труда влечет за собой временные и финансовые затраты.

Важность разнообразия и качества набора данных неоспорима. Необходимо учитывать различные случаи в зависимости от задачи. Для некоторых задач подойдут синтетические данные, для других потребуется ручная разметка и сбор данных. Например, в рамках данного проекта, при его развитии, можно добавить распознавание рукописного текста [37; 38] или проверку подлинности подписей и печатей. Хотя существуют генераторы рукописных данных, более качественным решением будет подготовка набора данных, размеченного на реальных примерах.

3.2 Обучение модели

В выбранном движке для распознавания текста PaddleOCR используются различные архитектуры, такие как CRNN, представляющая собой сочетание CNN и RNN, а также SVRT, применяемая в более новых версиях PaddleOCR.

В репозитории разработчики PaddleOCR представляют несколько типов моделей (таблица 3.1). Взаимосвязь этих типов моделей показана на рисунке 3.6.

Таблица 3.1 – типы моделей для PaddleOCR

Тип модели	Формат модели	Описание
inference model	inference.pdmodel inference.pdiparams	Используется для вывода на основе движка вывода Paddle
trained model, pre-trained model	*.pdparams *.pdopt *.states	Контрольные точки модели, сохраненные в процессе обучения, которые хранят параметры модели, в основном используются для оценки модели и непрерывного обучения.
nb model	*.nb	Модель, оптимизированная с помощью Paddle-Lite, которая подходит для сценариев развертывания на мобильных устройствах (для развертывания модели nb требуется Paddle-Lite).

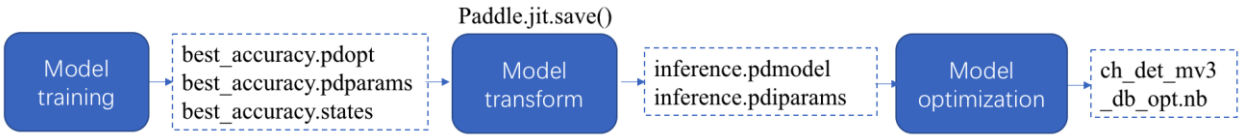

Рисунок 3.6 – Взаимосвязь типов моделей

Также PaddleOCR предоставляет набор предобученных моделей для детекции, распознавания и классификации текста. В основном эти модели хорошо зарекомендовали себя для китайского и английского языков, но также разработчики представляют и мультиязычные модели.

Так как наша задача – разработать сервис, который будет преимущественно использоваться в РФ для работы с документами, содержащими кириллицу, для экспериментов по дообучению модели была

выбрана предобученная модель cyrillic_PP-OCRv3_rec из предлагаемых разработчиками в репозитории PaddleOCR.

Для этого необходимо скачать модель cyrillic_PP-OCRv3_rec по ссылке [39] и файл конфигурации обучения этой модели cyrillic_PP-OCRv3_rec.yml [40].

Далее создаем директорию проекта project. Затем в директории project создаем директорию pretrained_models и размещаем скачанную модель и файл конфигурации.

Отредактируйте файл конфигурации, указав путь к директории, где находятся данные для обучения и валидации, а также к файлам аннотаций, содержащим информацию о именах файлов изображений и тексте, находящемся на этих изображениях. Также укажите количество эпох для обучения (рисунок 3.7, рисунок 3.8).

Убедитесь, что ключ use_gpu оставлен со значением false, если обучение будет происходить в среде без GPU. Также следует учитывать, что при использовании графического процессора необходимо установить версию paddlepaddle-gpu с поддержкой графического процессора.

pretrain_models > ! cyrillic_PP-OCRv3_rec_.yml

```yaml
Global:
  debug: false
  use_gpu: false
  epoch_num: 25
  log_smooth_window: 20
  print_batch_step: 2
  save_model_dir: ./output/v3_cyrillic_mobile
  save_epoch_step: 3
  eval_batch_step: [0, 20]
  cal_metric_during_train: true
  pretrained_model:
  checkpoints:
  save_inference_dir:
  use_visualdl: false
  infer_img: doc/imgs_words/ch/word_1.jpg
  character_dict_path: /PaddleOCR/ppocr/utils/dict/cyrillic_dict.txt
  max_text_length: &max_text_length 25
```

Рисунок 3.7 – Настройка конфигурации для обучения

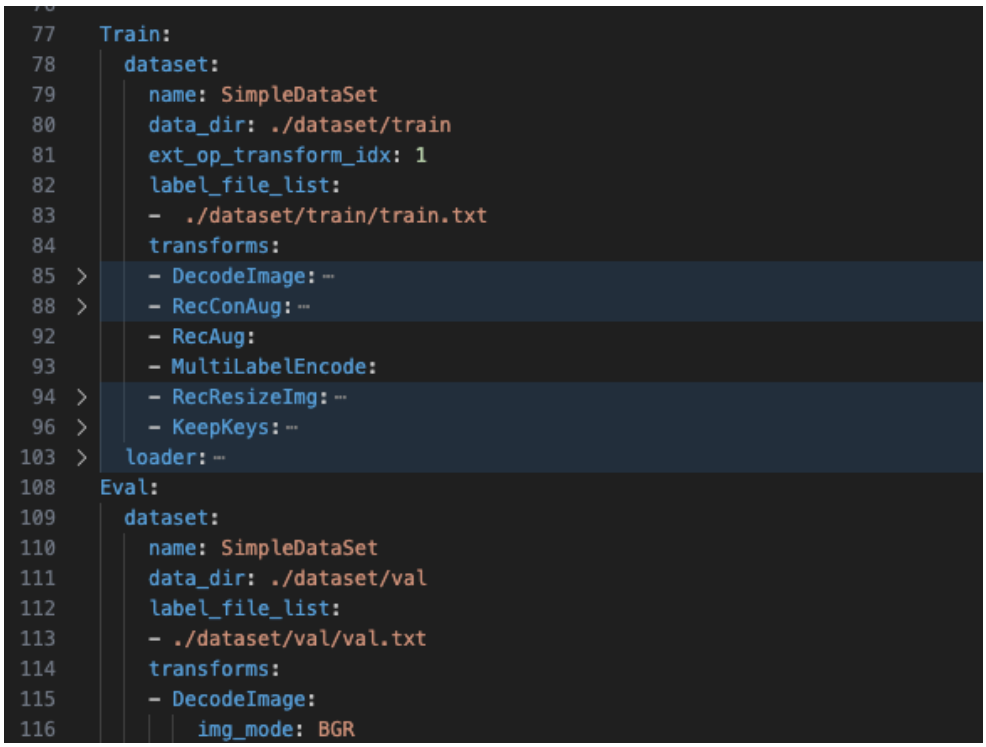

Рисунок 3.8 – Настройка конфигурации для обучения

Клонируем репозиторий PaddleOCR в директорию проекта и переходим в директорию PaddleOCR.

```
git clone https://github.com/PaddlePaddle/PaddleOCR.git
cd PaddleOCR
```

Запуск команды обучения выглядит так:

```
python3 tools/train.py
-c ./pretrain_models/cyrillic_PP-OCRv3_rec.yml
-o Global.pretrained_model=
     ./pretrain_models/ cyrillic_PP-OCRv3_rec/best_accuracy
```

После окончания обучения, экспортируем инференс модель

```
python3 tools/export_model.py
-c ./output/cyrillic_PP-OCRv3_rec/config.yml
-o Global.pretrained_model=
     ./output/cyrillic_PP-OCRv3_rec/latest
Global.save_inference_dir=
```

```
./output/cyrillic_PP-OCRv3_rec/model_inference
```

Для проверки работы модели, запускаем скрипт и передаем тестовый документ в формате png для распознавания текста:

```
python3 test.py
```

Результатом будет изображение с примером распознанного текста, которое можно использовать для наглядности и оценки результата. Пример изображения представлен на рисунке 3.9.

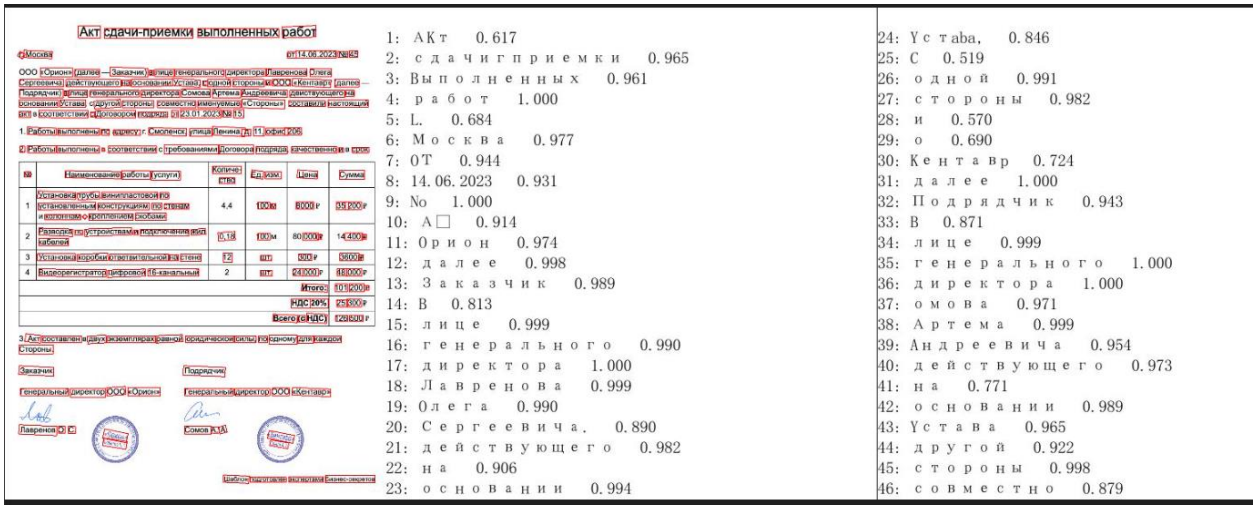

Рисунок 3.9 – Пример распознавания документа (SVRT)

Пробное обучение проводилось в течение 10 эпох с использованием небольшого количества данных для проверки работоспособности скриптов. В связи с ограниченными ресурсами на локальном ПК было принято решение проводить обучение в сервисе Google Colaboratory.

Из-за высокой вычислительной сложности обучение занимает более 12 часов, а сессия Google Colaboratory отключается при длительном отсутствии активности. Поэтому было решено обучать модель на меньшем количестве данных и сохранять промежуточные модели в Google Drive.

Основное обучение проводилось в Google Colab в течение 25 эпох. В качестве входных данных использовались подготовленные тренировочные данные в количестве 10 тысяч изображений с текстом и около 2000 валидационных изображений.

Последние версии PaddleOCR используют модели, основанные на архитектуре SVTR. По сравнению с предыдущими версиями на основе CRNN (Resnet34), модели на базе SVTR демонстрируют более высокую точность распознавания [41].

Используемые в новых версиях PaddleOCR архитектура SVTR, показала, что по результатам распознавания выигрывают версии на основе CRNN (Resnet34).

По результатам эксперимента, архитектура SVTR, используемая в новых версиях PaddleOCR, показала более высокие результаты распознавания по сравнению с версиями на основе CRNN (Resnet34) (таблица 3.2).

Таблица 3.2 – сравнение моделей по Левентштейну

Название модели	Mean Levenshtein case sensitive	Mean Levenshtein case insensitive	Inference CPU
CRNN	85,2%	89,4%	7,5 it/s (1 it - 133 ms)
SVTR-Tiny	88,7%	92,9%	32,7 it/s (1 it - 30 ms)

Результаты проведенных экспериментов показали, что архитектура SVTR, используемая в новых версиях PaddleOCR, демонстрирует более высокие результаты распознавания текста по сравнению с предыдущими версиями на основе CRNN (Resnet34). Это подтверждает целесообразность использования модели cyrillic_PP-OCRv3_rec для разработки микросервиса OCR, что позволит обеспечить высокую точность и производительность при работе с документами, содержащими кириллицу. Обучение на дополнительных данных позволили значительно повысить качество модели.

ЗАКЛЮЧЕНИЕ

Целью данной выпускной квалификационной работы было проектирование лабораторной информационной системы (ЛИС) с использованием алгоритмов машинного обучения. Одной из ключевых задач стало создание микросервиса для оптического распознавания текста (OCR).

Для достижения поставленных целей были выполнены следующие задачи:

1. Проектирование архитектуры ЛИС;

Была разработана архитектура ЛИС, основанная на микросервисной модели, обеспечивающей высокую производительность, гибкость и масштабируемость системы. В качестве основного стека технологий были выбраны Ruby on Rails для основного приложения и PaddleOCR для микросервиса OCR. Архитектура системы была спроектирована таким образом, чтобы обеспечить эффективное взаимодействие между клиентской частью, серверной частью и микросервисом для обработки задач.

2. Выбор алгоритмов машинного обучения для ЛИС;

Для задачи распознавания текста была выбрана библиотека PaddleOCR, которая использует современные алгоритмы машинного обучения, такие как CRNN и SVTR. Архитектура SVTR показала более высокую точность и производительность по сравнению с предыдущими версиями на основе CRNN. Модель cyrillic_PP-OCRv3_rec была выбрана для дообучения на дополнительных данных, что позволило достичь высокой точности распознавания текстов на кириллице.

3. Сбор данных и обучение модели для распознавания текста.

Для обучения модели были подготовлены синтетические данные на основе списка слов на русском языке. Использование инструментов для генерации данных, таких как TextRecognitionDataGenerator, позволило создать разнообразный и качественный набор данных. Процесс обучения модели был

проведен в сервисе Google Colaboratory, что обеспечило необходимую вычислительную мощность для обработки большого объема данных. Модель была обучена и протестирована, показав ожидаемо высокую точность распознавания текста.

Разработанный микросервис для распознавания текста OCR для лабораторной информационной системы демонстрирует высокую эффективность распознавания и производительность. Применение модели SVTR в PaddleOCR позволило значительно улучшить результаты распознавания текста, что делает данную систему перспективным решением для автоматизации и оптимизации лабораторных процессов в строительной отрасли. Точность распознавания составляет более 90%.

Таким образом, выполненные в рамках данной работы исследования и разработки показывают, что использование современных технологий и алгоритмов машинного обучения в лабораторных информационных системах может значительно улучшить процессы анализа и интерпретации данных, повысив их точность и эффективность.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Молчанова, Е. И. Тенденции развития и технологии программного обеспечения отечественных аналитических лабораторий [Текст] / Е.И. Молчанова // Заводская лаборатория. Диагностика материалов. – 2015. – № 4. – С. 59-68.

2. Гребенюк, Е. А. Построение совершенных систем автоматизации (ЛИМС) производственных лабораторий предприятий технологических отраслей [Текст] / Е.А. Гребенюк, Э.Л. Ицкович // Управление развитием крупномасштабных систем MLSD'2018. – 2018. – № 1. – С. 74-85.

3. ГОСТ Р 54360-2011. Лабораторные информационные менеджмент-системы (ЛИМС). Стандартное руководство по валидации ЛИМС [Текст]: дата введения 2011-10-01. – М: Стандартинформ, 2012. – 40 с.

4. Панфилов, И. А. Моделирование процессов лабораторной информационной системы на производственном предприятии [Текст] / И.А. Панфилов // Перспективы науки. – 2023. – № 1. – С. 40-45.

5. You, Z. Integration of Industry 4.0 Related Technologies in Construction Industry: A Framework of Cyber-Physical System [Электронный ресурс] / Z. You, L. Feng // IEEE Access. 2020. Т. 8. Integration of Industry 4.0 Related Technologies in Construction Industry: https://ieeexplore.ieee.org/document/9133409 (дата обращения: 22.03.2024).

6. Бутылин, Е. В. Разработка лабораторной информационной системы контроля качества нефтепродуктов [Электронный ресурс] / Е. В. Бутылин, П. Г. Михайлова // Успехи в химии и химической технологии: https://cyberleninka.ru/article/n/razrabotka-laboratornoy-informatsionnoy-sistemy-kontrolya-kachestva-nefteproduktov/viewer (дата обращения: 20.03.2024)

7. Ангапов, В. Д. Использование технологий машинного обучения в защите информационных систем [Текст] / В.Д. Ангапов // Наука, техника и образование. – 2023. – № 4 (92). – С. 20-26.

8. Аббасов, М. Ш. Применение искусственного интеллекта в системе поддержки принятия управленческих решений [Текст] / М.Ш. Аббасов // Развитие теории и практики управления социальными и экономическими системами. – 2022. – № 11. – С. 85-89.

9. Что такое интеллектуальная обработка документов (IDP)? [Электронный ресурс]: https://powerautomate.microsoft.com/ru-ru/intelligent-document-processing/ – (дата обращения: 29.04.2024)

10. OCR vs. IDP: What’s The Difference? [Электронный ресурс]: https://www.abbyy.com/blog/ocr-vs-idp/ – (дата обращения: 29.04.2024)

11. Tesseract Open Source OCR Engine [Электронный ресурс]: https://github.com/tesseract-ocr/tesseract – (дата обращения: 18.05.2024)

12. Качалин, В. С. Сравнительный анализ различных систем оптического распознавания символов при работе с текстом, написанным с помощью кириллического алфавита [Текст] / В.С. Качалин, Ю.Н. Панов, Н.Л.Э. Попов // Современные наукоемкие технологии. – 2022. – №8. – С. 65.

13. ContentReader PDF (ранее ABBYY FineReader PDF) [Электронный ресурс]: https://www.tadviser.ru/index.php/Продукт:ContentReader_PDF_(ранее_ABBYY_FineReader_PDF) – (дата обращения: 18.05.2024)

14. Y. Baek. Character Region Awareness for Text Detection [Электронный ресурс] / Y. Baek, B. Lee, D. Han // arXiv: http://arxiv.org/abs/1904.01941 (дата обращения: 18.05.2024)

15. Коночкин, К. Н. Задача распознавания реквизитов документов с использованием систем оптического распознавания символов [Текст] / К.Н. Коночкин // Электронные информационные системы. – 2023. – №4(39). – С. 98-113.

16. JaidedAI/EasyOCR – EasyOCR [Электронный ресурс]: https://github.com/JaidedAI/EasyOCR – (дата обращения: 18.05.2024)

17. PaddleOCR - data synthesis tools [Электронный ресурс]: https://github.com/PaddlePaddle/PaddleOCR/blob/main/doc/doc_en/data_synthesi

s_en.md – (дата обращения: 13.05.2024)

18. Кочеткова, А. А. Автоматизация паспортного контроля с помощью технологий оптического распознавания символов [Электронный ресурс] / А.А. Кочеткова, М.Б. Кравец, Е.В. Данилова // Современные проблемы и технологии в сфере туризма, сервиса и предпринимательства: российский и зарубежный опыт: https://www.elibrary.ru/item.asp?id=54141391 (дата обращения: 18.05.2024)

19. Документация Yandex Vision OCR [Электронный ресурс]: https://yandex.cloud/ru/docs/vision/concepts/ocr/ – (дата обращения: 13.05.2024)

20. Кислицын, Е. В. Современные технологии разработки программного обеспечения [Электронный ресурс]: [электронное учебное пособие] / Е. В. Кислицын, М. А. Панов // Научная электронная библиотека elibrary.ru: https://www.elibrary.ru/item.asp?id=47434129 (дата обращения: 30.04.2024)

21. Умаров, М. А. У. Исследование и разработка системы оплаты парковок и платных дорог с помощью мобильного сервиса и систем видеонаблюдения [Электронный ресурс] / М.А.У. Умаров, Й.Г. Крастев // Альманах научных работ молодых ученых университета ИТМО: https://www.elibrary.ru/item.asp?id=44558175 (дата обращения: 18.05.2024)

22. 37signals [Электронный ресурс]: https://37signals.com – (дата обращения: 01.05.2024)

23. Is Ruby Dead? [Электронный ресурс]: https://isrubydead.com/ – (дата обращения: 16.05.2024)

24. TIOBE Index [Электронный ресурс]: https://www.tiobe.com/tiobe-index/ – (дата обращения: 16.05.2024)

25. SVTR – state-of-the-art нейросеть для задачи OCR [Электронный ресурс]: https://habr.com/ru/articles/686884/ – (дата обращения: 19.05.2024)

26. Y. Du. SVTR: Scene Text Recognition with a Single Visual Model [Электронный ресурс] / Y. Du, Z. Chen, C. Jia // arXiv: http://arxiv.org /abs/2205.00159 (дата обращения: 19.05.2024)

27. Baoguang Shi. ASTER: An Attentional Scene Text Recognizer with Flexible Rectification [Электронный ресурс] / Baoguang Shi, Mingkun Yang // IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/8395027 (дата обращения: 19.05.2024)

28. Dosovitskiy, A. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale [Электронный ресурс] / A. Dosovitskiy // arXiv: http://arxiv.org/abs/2010.11929 (дата обращения: 20.05.2024)

29. Belval/TextRecognitionDataGenerator – A synthetic data generator for text recognition [Электронный ресурс]: https://github.com/Belval /TextRecognitionDataGenerator – (дата обращения: 10.05.2024)

30. dkulagin/kartaslov – Открытые данные на Карте слов [Электронный ресурс]: https://github.com/dkulagin/kartaslov – (дата обращения: 20.05.2024)

31. danakt/russian-words – List of Russian words [Электронный ресурс]: https://github.com/danakt/russian-words – (дата обращения: 18.05.2024)

32. Lemeri02/trdg_script – Скрипт для генерации данных [Электронный ресурс]: https://github.com/Lemeri02/trdg_script – (дата обращения: 24.05.2024)

33. ankush-me/SynthText – SynthText [Электронный ресурс]: https://github.com/ankush-me/SynthText – (дата обращения: 18.05.2024)

34. datanomica/SynthText-Russian – SynthText Russian [Электронный ресурс]: https://github.com/datanomica/SynthText-Russian – (дата обращения: 18.05.2024)

35. HumanSignal/label-studio: Label Studio [Электронный ресурс]: https://github.com/HumanSignal/label-studio – (дата обращения: 13.05.2024)

36. Toloka [Электронный ресурс]: https://toloka.ai/ru/ – (дата обращения: 13.05.2024)

37. Handwritten Signature - Feature Extraction [Электронный ресурс]: https://kaggle.com/code/eryash15/handwritten-signature-feature-extraction – (дата обращения: 14.03.2024)

38. Cyrillic Handwriting Dataset – dataset for handwriting character

recognition [Электронный ресурс]: https://www.kaggle.com/datasets /constantinwerner/cyrillic-handwriting-dataset – (дата обращения: 14.03.2024)

39. Cyrillic Recognition Trained Model for PaddleOCR [Электронный ресурс]: https://paddleocr.bj.bcebos.com/PP-OCRv3/multilingual/cyrillic_PP-OCRv3_rec_train.tar – (дата обращения: 18.05.2024)

40. Cyrillic PP-OCRv3 recognition config [Электронный ресурс]: https://github.com/PaddlePaddle/PaddleOCR/blob/main/configs/rec/PP-OCRv3/multi_language/cyrillic_PP-OCRv3_rec.yml – (дата обращения: 18.05.2024)

41. Обучаем SVTR-Tiny для распознавания текста сцены [Электронный ресурс]: https://habr.com/ru/companies/datanomica/articles/705696/ – (дата обращения: 18.05.2024)