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 «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

Разработка программного модуля распознавания номеров железнодорожных вагонов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.406 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____Ю.Г. Плаксина
«___»_____2024 г.

Автор работы,
студент группы КЭ-406
_____Д.А. Дорофеев
«___»_____2024 г.

Нормоконтролёр,
ст. преподаватель каф. ЭВМ
_____ С.В.Сяськов
«___»_____2024 г.

Челябинск-2024

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 «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-406
Дорофеев Данил Александрович
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка программного модуля распознавания номеров железнодорожных вагонов» утверждена приказом по университету от 22 апреля 2024 г. №764/13-12

2. **Срок сдачи студентом законченной работы:** 1 июня 2024 г.

3. **Исходные данные к работе:**

3.1. Среда разработки Visual Studio Code.

3.2. Язык программирования Python.

3.3. Данные для обучения и тестирования модели: 1000 изображений, предоставленные компанией ООО «СТАТЕРА».

3.4. База данных модуля распознавания - SQLite.

3.5. Функциональные требования:

- **загрузка изображений**: пользователь должен иметь возможность загрузить изображение с для распознавания через API сервиса;
- **распознавание текста**: система должна быть способна распознавать номер вагона на загруженных изображениях, независимо от типа вагона;
- **возврат результатов**: после распознавания текста система должна возвращать результаты в консоли, где запущен сервис;
- **хранение данных**: система должна обеспечивать хранение загруженных изображений и результатов распознавания для последующего доступа пользователей;
- **обработка ошибок**: система должна обрабатывать ошибки и предоставлять сообщения об ошибках пользователю в случае непредвиденных ситуаций.

4. **Перечень подлежащих разработке вопросов:**
   - провести анализ существующих решений в области распознавания объектов на изображениях;
   - выявить требования к программному модулю распознавания номеров вагонов;
   - разработать архитектуру предлагаемого решения и выбор нейросетевых алгоритмов для решения задачи;
   - реализовать программный модуль и провести его тестирование.
5. **Дата выдачи задания:** 1 декабря 2023 г

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*Д.А. Дорофеев*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Анализ существующих решений - Поиск решений в открытых источниках - Сравнение преимуществ и недостатков	10.03.2024	
Выявление требований - Определение функциональных требований - Определение нефункциональных требований	21.03.2024	
Разработка архитектуры решения и выбор нейросетевых алгоритмов - Подготовка изображений для обучения нейросетевых моделей - Написание кода обучения нейросетевых моделей - Обучение моделей компьютерного зрения - Сравнение моделей компьютерного зрения - Создание диаграмм архитектуры модуля - Проектирование базы данных	04.04.2024	
Реализация программного модуля и тестирование - Настройка инфраструктуры для модуля - Написание кода бэкенд-части модуля - Написание кода работы моделей в соответствии с парадигмой ООП	25.04.2024	

Этап	Срок сдачи	Подпись руководителя
- Тестирование работоспособности предлагаемого решения - Тестирование качества и скорости распознавания модуля		
Компоновка текста работы и сдача на нормоконтроль	13.05.2024	
Подготовка презентации и доклада	24.05.2024	

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*Д.А. Дорофеев*/

Аннотация

Дорофеев Д.А. Разработка программного модуля распознавания номеров железнодорожных вагонов. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 63 с., библиогр. список – 29 наим.

Данная работа посвящена разработке программного модуля для распознавания номеров железнодорожных вагонов при помощи компьютерного зрения. Современные разработки в области обработки изображений открывают новые возможности для повышения эффективности и безопасности железнодорожных операций. Применение компьютерного зрения может значительно улучшить процессы мониторинга и управления составами, а также обеспечить более высокий уровень безопасности на железнодорожных путях.

Целью является разработать программный модуль распознавания номеров вагонов для автоматизации процессов мониторинга. Для достижения этой цели проведен анализ существующих методов компьютерного зрения и их потенциальное применение в сфере распознавания железнодорожного транспорта.

В ходе работы были проанализированы существующие современные решения, и выбраны модели компьютерного зрения для реализации модуля, который впоследствие может быть использован для автоматизации процессов мониторинга и идентификации составов.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире технологий, автоматизированные системы играют ключевую роль в улучшении эффективности и безопасности различных отраслей. Задача идентификации вагонов рано или поздно встает перед каждым предприятием, которое интенсивно использует железнодорожный транспорт: металлургические, химические, нефтеперерабатывающие производства, предприятия деревообработки, производители строительных материалов. В данной работе рассматривается тема разработки программного модуля для распознавания номеров железнодорожных вагонов с использованием методов машинного обучения.

В связи с постоянным ростом объемов грузоперевозок, эффективное управление и контроль железнодорожного транспорта становится все более значимым. На российских железных дорогах и промышленных предприятиях в настоящее время для идентификации грузовых вагонов и цистерн используется, как правило, ручное визуальное списывание регистрационных номеров транспортных средств, которые представляют собой метки в виде восьмизначной цифровой последовательности. При таком способе списывания оператор на контрольном участке железной дороги просматривает номера вагонов проходящего состава и при необходимости сравнивает их с номерами в определенной передаточной ведомости, называемой также натур-листом. При обнаружении несоответствия производится корректировка натур-листа, формируются управляющие решения. Недостатки рассмотренного способа идентификации: необходимость в постоянном внимании оператора, высокая трудоемкость процесса контроля, недостаточная достоверность информации.

Данная работа направлена на разработку решения, для распознавания номеров железнодорожных вагонов на основе современных методов машинного обучения и компьютерного зрения. Для достижения этой цели необходимо решить следующие задачи:

- провести анализ существующих решений в области распознавания объектов на изображениях;
- выявить требования к программному модулю распознавания номеров железнодорожных вагонов;
- разработать архитектуру предлагаемого решения и выбрать нейросетевые алгоритмы для решения задачи;
- разработать программный модуль и провести его тестирование.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ РАСПОЗНАВАНИЯ ОБЪЕКТОВ

1.1. Анализ литературы и подходов к решению задачи

Одним из наиболее эффективных методов распознавания номеров железнодорожных вагонов является анализ изображений с применением компьютерного зрения [1]. Этот метод предполагает установку камер, зафиксированных на определенном участке железнодорожного пути, для наблюдения за движением вагонов. Однако, данный подход также имеет свои ограничения и технические трудности. Для обеспечения высокой точности распознавания номеров вагонов при анализе изображений необходимо учитывать следующие аспекты:

Фильтрация шума и ложных срабатываний: Для исключения ошибочного распознавания номеров, вызванного шумами на видеоизображениях или изменением условий освещения, применяются методы фильтрации и обработки изображений с использованием алгоритмов машинного обучения.

Оптимизация алгоритмов распознавания: Непрерывное улучшение алгоритмов распознавания номеров вагонов с учетом современных достижений в области компьютерного зрения позволяет повысить эффективность системы и минимизировать количество ошибок.

Существует несколько научных работ, посвященных распознаванию номеров на изображениях. Например, статья 2020 года, написанная студентами из Бразилии [2] в которой описано, как при использования таких архитектур как YOLO и OCR им удалось достичь высокой точности распознавания номеров на вагонах. Несмотря на то, что нумерация железнодорожного состава в Бразилии

отличается от их нумерации в пределах Российской Федерации, можно использовать эти наработки для создания системы, способной с высокой точностью находить и определять номер на вагоне с российской нумерацией. Одна из трудностей, с которой столкнулись авторы этой статьи, это то, что иногда номер на контейнере загрязнен или закрашен. В таком случае помогает тот факт, что зачастую номер дублируется в другом месте вагона, например на шасси, и можно находить сразу несколько областей с номером. В этом случае, если одна область не поддается распознаванию, система работает со второй областью.

В том-же году вышла работа [3], в рамках которой показана возможность использования рекуррентных нейронных сетей для решения задач поиска и распознавания объектов, также в статье описано, как в подобном программном модуле функционально описать взаимодействие между его компонентами.

Добиться очень высокой точности нахождения и распознавания номера можно не только используя модели с большим числом параметров. Например, в статье [4] 2015 года было показано, что на точность могут влиять разные варианты предобработки изображения.

О том, как эффективно обучать модель глубокого обучения и валидировать ее результаты описано в работе Challenges in Deploying Machine Learning: a Survey of Case Studies [5], а как правильно разворачивать программный модуль изложено в статье Studying the Practices of Deploying Machine Learning Projects on Docker [6]

Проведенный анализ литературы позволяет сделать вывод о том, что использование компьютерного зрения в задаче распознавания номеров может быть эффективным инструментом для снижения человеческого фактора при записи номера вагона, поэтому использование алгоритмов компьютерного зрения для определения номера имеет большую актуальность и может стать

важным шагом в цифровизации для каждого предприятия, которое использует железнодорожный транспорт.

1.2. Основные определения нейронных сетей

Нейронная сеть - это компьютерная модель, которая имитирует работу человеческого мозга. Она состоит из множества небольших элементов, называемых нейронами, которые работают вместе, чтобы решать сложные задачи. Каждый нейрон принимает входные данные (рисунок 1), выполняет некоторые вычисления и передает результат следующему нейрону в сети.

Составляющие нейронной сети:

- нейроны: Они являются основными строительными блоками нейронной сети. Каждый нейрон принимает некоторое количество входных данных, умножает их на веса и применяет функцию активации к результату. Это позволяет нейрону делать сложные вычисления;
- слой: Нейроны в сети группируются в слои. Есть входной слой, который принимает данные, скрытые слои, которые выполняют вычисления, и выходной слой, который предоставляет результаты. Каждый слой связан с предыдущим и последующим слоями;
- веса: Каждая связь между нейронами имеет свой вес, который определяет важность входных данных для вычислений нейрона. Эти веса настраиваются во время обучения сети;
- функция активации: Это функция, применяемая к сумме взвешенных входных данных нейрона. Она вводит нелинейность в вычисления, позволяя сети решать более сложные задачи.

Вместе эти компоненты позволяют нейронной сети обучаться на данных и делать прогнозы или принимать решения в зависимости от поставленной задачи.

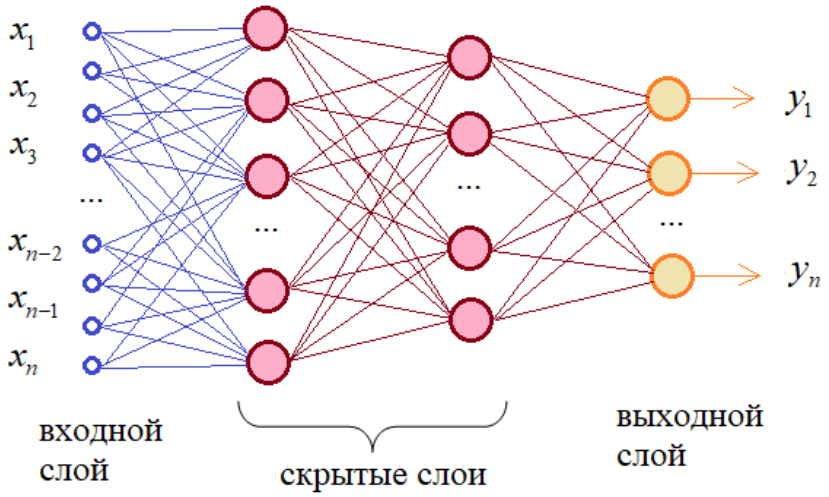

Рисунок 1 – Пример нейронной сети

1.2.1. Описание архитектур детекции объектов

Дальнейшее развитие этой области привело к появлению R-CNN [7], SSD [8] и YOLO [9], которые сочетают в себе детекцию и классификацию объектов в реальном времени. Эти методы стали более эффективными и быстрыми благодаря использованию различных архитектур нейронных сетей, включая сверточные слои, пулинг, и применение различных оптимизаций.

R-CNN (рисунок 2) - это модель для обнаружения объектов на изображениях, которая состоит из нескольких этапов:

1. Выделение пропозиций: Сначала на изображении используется метод выделения пропозиций (например, Selective Search), чтобы найти потенциальные области, где могут находиться объекты. (Эти пропозиции обычно представлены прямоугольными областями, которые содержат объекты.)

2. Извлечение признаков: Области изображения, соответствующие каждой пропозиции, изменяются до фиксированного размера и подаются на вход сверточной сети для извлечения признаков.

3. Классификация объектов: После извлечения признаков для каждой пропозиции используется классификатор (обычно SVM [10] или softmax classifier), чтобы определить, принадлежит ли эта область какому-либо классу объектов или нет.

4. Пост-обработка: Дополнительные шаги могут включать в себя подстройку ограничивающих рамок для более точного позиционирования объектов и устранение дубликатов детекций.

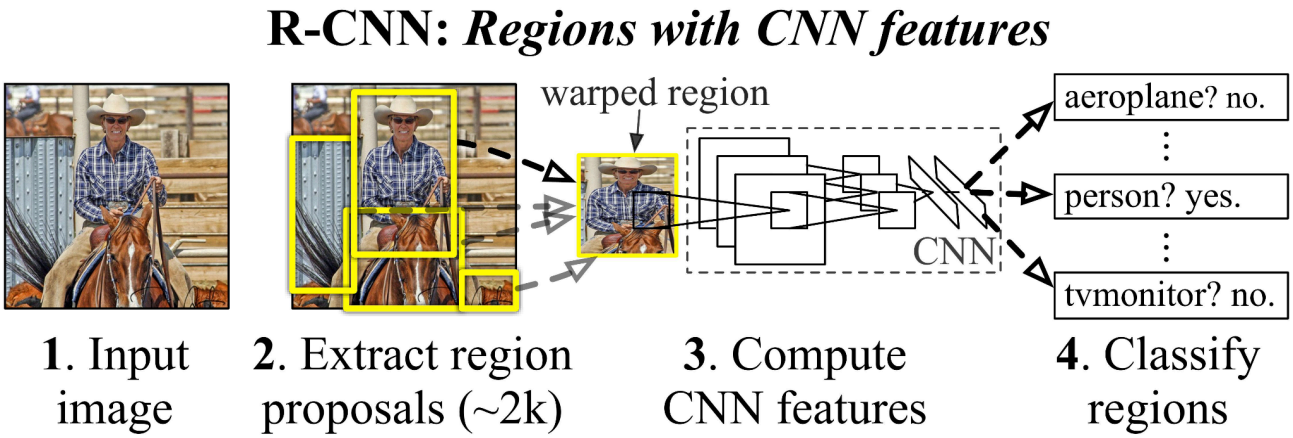

Рисунок 2 – Архитектура R-CNN, включающая этапы приема входного изображения, выделения пропозиции регионов, извлечения признаков сверточной нейронной сетью и классификацию регионов. Источник [7]

Архитектура Faster R-CNN представляет собой улучшенную версию оригинальной R-CNN с дополнительным механизмом для генерации пропозиций. Основное отличие Faster R-CNN от R-CNN заключается в том, что Faster R-CNN внедряет в себя модуль Region Proposal Network, который

позволяет сети генерировать пропозиции автоматически, в отличие от R-CNN, где пропозиции генерируются отдельным алгоритмом (например, Selective Search).

Описание архитектуры Faster R-CNN (рисунок 3):

1. **Region Proposal Network**:
    - основное нововведение Faster R-CNN - это внедрение RPN, который является сверточной сетью, обучаемой находить пропозиции (области), в которых могут содержаться объекты;
    - RPN работает параллельно с общей сверточной сетью и генерирует пропозиции на основе признаков, извлеченных из изображения;
    - RPN предлагает области, которые затем используются для дальнейшей классификации и определения ограничивающих рамок.
2. **Связь с общей сверточной сетью**:
    - Faster R-CNN использует предварительно обученную сверточную сеть (например, VGG, ResNet) для извлечения признаков из изображения, как и в случае с R-CNN;
    - однако в Faster R-CNN RPN и общая сверточная сеть объединены в единую модель, что позволяет значительно ускорить процесс обнаружения объектов.
3. **Интеграция с общей архитектурой**:
    - RPN и сверточная сеть обучаются совместно, что позволяет оптимизировать всю архитектуру Faster R-CNN end-to-end. Это значит, что как RPN, так и общая сверточная сеть, могут быть обучены одновременно на задачу обнаружения объектов.

4. **Увеличение скорости обнаружения**:
    - благодаря внедрению RPN, Faster R-CNN стал значительно быстрее в сравнении с оригинальным R-CNN, так как процесс генерации пропозиций и извлечения признаков объединен и оптимизирован.

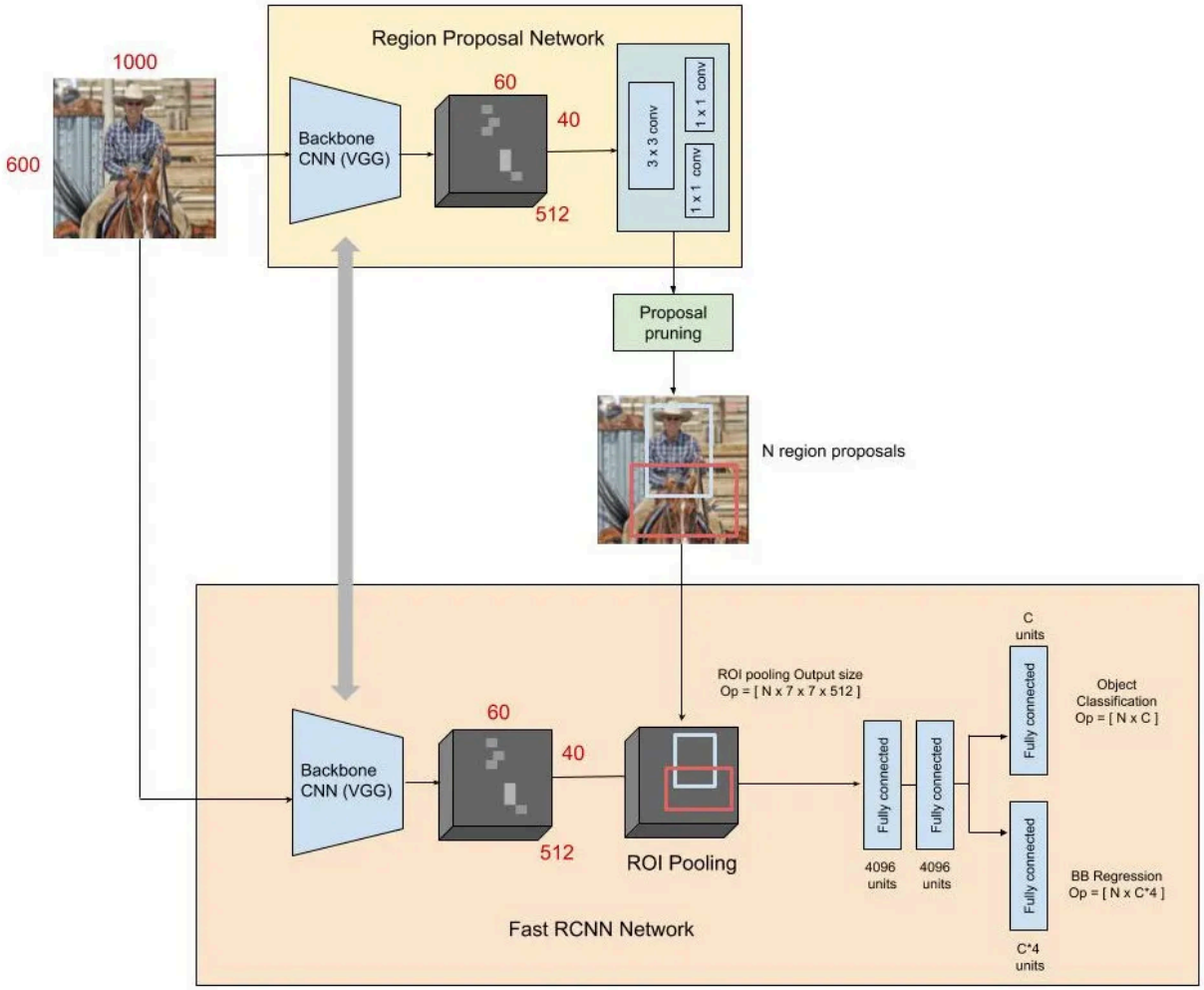

Рисунок 3 – Архитектура нейросети Faster R-CNN

SSD - это инновационная архитектура для обнаружения объектов на изображениях, которая позволяет одновременно обнаруживать объекты различных размеров и пропорций в одном проходе (single shot).

Описание архитектуры SSD (рисунок 4):

1. **Базовая сверточная сеть**:
    - SSD использует предварительно обученную сверточную нейронную сеть (например, VGG, ResNet) для извлечения признаков из входного изображения;
    - эта базовая сверточная сеть применяется к изображению, чтобы извлечь признаки на разных уровнях абстракции.
2. **Формирование признаков на разных уровнях**:
    - SSD использует несколько слоев с разным масштабом, чтобы детектировать объекты различных размеров. Каждый слой представляет собой часть базовой сверточной сети;
    - каждый слой генерирует набор прямоугольных областей разного размера и соотношения сторон в каждом пикселе изображения.
3. **Прогнозирование класса и ограничивающей рамки**:
    - для каждой области SSD прогнозирует ограничивающую рамку вокруг объекта;
    - прогнозы выполняются с помощью сверточных слоев и дополнительных слоев, включающих сверточные фильтры и полносвязные слои.
4. **Многомасштабные признаки**:
    - SSD объединяет признаки с разных уровней масштаба для улучшения обнаружения объектов разных размеров;

- это позволяет SSD успешно обнаруживать как маленькие, так и большие объекты на изображении.

5. **Non-Maximum Suppression**:
   - после прогнозирования ограничивающих рамок применяется процедура Non-Maximum Suppression для удаления дублирующихся детекций и оставления только наиболее уверенных.

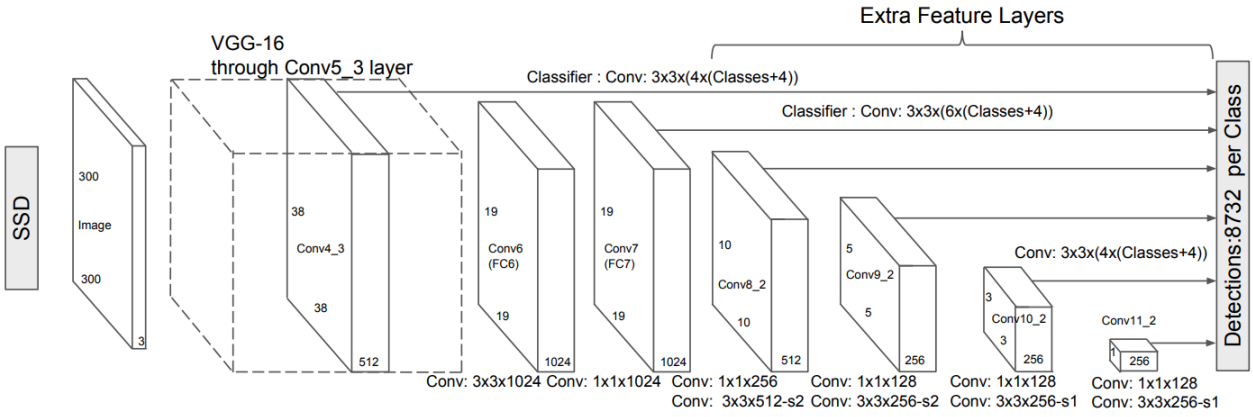

Рисунок 4 – Архитектура нейросети Single Shot Detector (SSD)

Архитектура YOLO представляет собой инновационный подход к обнаружению объектов на изображениях, который позволяет быстро и эффективно определять объекты и их положение в одном проходе через нейронную сеть.

Описание архитектуры YOLO (рисунок 5):

1. **Сверточная база (Backbone)**:
   - YOLO использует сверточную нейронную сеть, такую как Darknet, в качестве backbone для извлечения признаков из входного изображения;

- эта базовая сеть содержит несколько сверточных слоев, а также слои пулинга, которые позволяют извлекать абстрактные признаки изображения на различных уровнях абстракции.

2. **Сеть обнаружения объектов**:
    - после извлечения признаков изображения YOLO использует сверточные слои для предсказания ограничивающих рамок и классов объектов;
    - каждая ограничивающая рамка определяется с использованием координат (центр, ширина, высота) и соответствующей вероятности принадлежности к определенным классам.
3. **Деление изображения на сетку**:
    - изображение делится на сетку фиксированного размера. Каждая ячейка сетки ответственна за обнаружение объектов в определенной области изображения;
    - для каждой ячейки сетки предсказываются ограничивающие рамки и вероятности классов объектов, которые находятся в этой ячейке.
4. **Анкоры**:
    - YOLO использует анкоры - фиксированные ограничивающие рамки разных размеров и соотношений сторон. Эти анкоры помогают предсказывать ограничивающие рамки для объектов различных размеров;
    - для каждой ячейки сетки предсказываются относительные смещения и размеры анкоров, чтобы адаптировать ограничивающие рамки к объектам в этой ячейке.

5. **Непрерывные прогнозы**:
    - основное преимущество YOLO заключается в том, что он работает быстро и эффективно благодаря одновременному предсказанию ограничивающих рамок и классов в одном проходе через сеть;
    - это позволяет YOLO быть очень быстрым и применимым для реального времени обнаружения объектов на изображениях и видео.

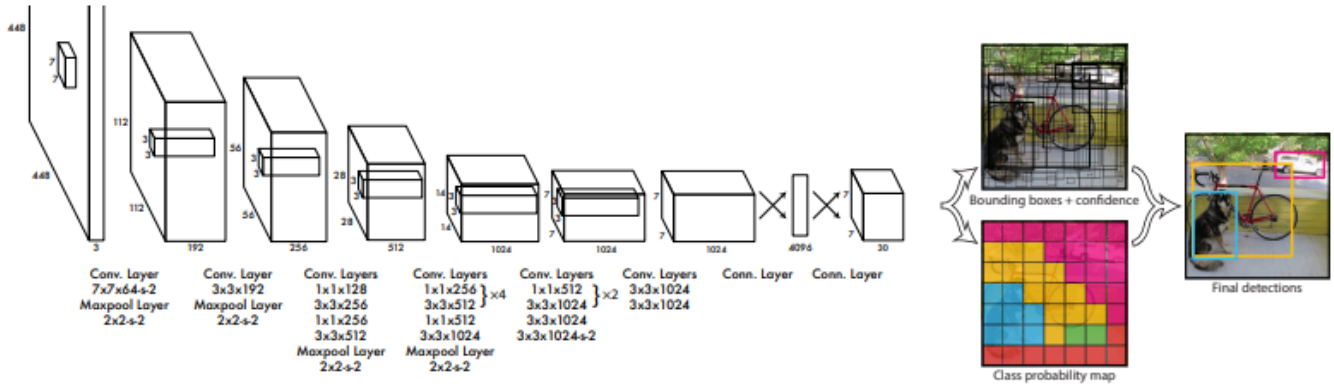

Рисунок 5 – Архитектура нейросети YOLO

Сегодня задача детекции объектов на изображении достигла высокого уровня точности и скорости благодаря использованию современных архитектур нейронных сетей, а также развитию вычислительных ресурсов и алгоритмов обучения. Это позволяет применять детекцию объектов во многих областях, включая железнодорожную отрасль, где она может быть применена для автоматического обнаружения и распознавания номеров на вагонах.

1.2.2. Описание архитектур классификации чисел

Рекуррентные нейронные сети (рисунок 6) - это класс нейронных сетей, специально разработанный для работы с последовательными данными, такими

как тексты, временные ряды, аудио и другие. Они обладают способностью запоминать информацию о предыдущих состояниях и использовать ее для обработки последующих входных данных.

Ключевые характеристики и принципы работы рекуррентных нейронных сетей:

1. **Рекуррентность**:
    - основная особенность RNN - это наличие обратных связей, благодаря которым информация может циркулировать через сеть и передаваться от одного временного шага к другому;
    - на каждом временном шаге RNN принимает входные данные и предыдущее скрытое состояние и выдает новое скрытое состояние и выход.
2. **Скрытое состояние (Hidden State)**:
    - Hidden State RNN - это внутреннее представление сети, которое хранит информацию о предыдущих входах;
    - на каждом временном шаге скрытое состояние обновляется и передается на следующий временной шаг.
3. **Обучение и параметры**:
    - рекуррентные нейронные сети обучаются с использованием алгоритма обратного распространения ошибки, который рассчитывает градиенты по параметрам сети;
    - временная структура RNN создает зависимости между различными временными шагами, что делает обучение более сложным и подверженным проблеме затухания градиента.

4. **Типы RNN**:
    - существует несколько различных архитектур рекуррентных нейронных сетей, таких как простые RNN, LSTM и GRU;
    - LSTM и GRU - это расширения базовой RNN, предназначенные для решения проблемы затухания градиента и сохранения долгосрочных зависимостей.

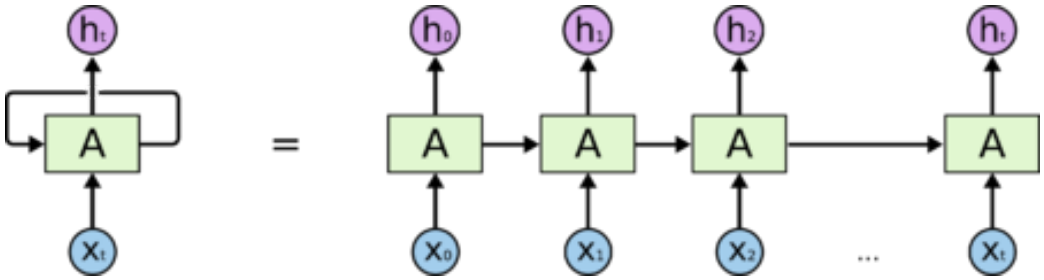

Рисунок 6 – RNN и ее развернутое представление

Рекуррентные нейронные сети играют важную роль в классификации последовательностей чисел. Они хорошо подходят для работы с данными, у которых есть последовательная структура, такими как временные ряды, тексты или звуковые сигналы. Основная идея RNN заключается в том, что они способны учитывать предыдущие состояния сети при обработке каждого нового входного элемента последовательности. Это делает их эффективными в задачах, где важно учитывать контекст или зависимости между элементами последовательности.

Одним из применений RNN в классификации чисел является оптическое распознавание символов (OCR). В OCR подходах последовательность чисел представляется в виде изображений, и задача заключается в распознавании символов на этих изображениях. Для этого часто используются сверточные

нейронные сети для извлечения признаков из изображений, а затем полученные признаки подаются на вход рекуррентным слоям RNN для классификации. Такой подход позволяет учитывать контекст и зависимости между символами в строке при распознавании.

EasyOCR [11] - это инструмент оптического распознавания символов с открытым исходным кодом, который позволяет извлекать текст из изображений. Архитектура EasyOCR основана на моделях глубокого обучения и использует совокупность различных методов для обработки изображений и извлечения текста.

Архитектура модели представляет собой следующее:

1. **Подготовка изображения:**

Модель предварительно обрабатывает изображение, чтобы улучшить его качество и сделать его более подходящим для распознавания текста. Предварительная обработка может включать в себя такие операции, как:

- удаление шума: удалите шум с изображения, применяя различные фильтры, такие как размытие по Гауссу и медианное размытие;
- бинаризация: преобразует изображение в оттенках серого в двоичное изображение с использованием различных методов определения порога, таких как метод Оцу и адаптивное определение порога;
- выравнивание: определяет и исправляет угол перекоса изображения, чтобы обеспечить правильное выравнивание текста.

2. **Распознавание текста (CRNN)**

Архитектура CRNN состоит из трех компонентов, включая сверточные слои, рекуррентные слои и декодирование (CTC).

В CRNN модели компонент сверточных слоев создается путем взятия сверточных слоев и слоев max-пулинга из стандартной модели сверточной сети, такой как ResNet/VGG (полносвязные слои удаляются).

Такой компонент используется для извлечения последовательного представления объекта из входного изображения. Проще говоря, это означает обработку изображения таким образом, чтобы сохранить пространственные отношения между пикселями, а также уменьшить размерность изображения. Эти признаковые представления затем передаются в качестве входных данных рекуррентной нейронной сети.

3. **RNN сеть модели (LSTM)**

Компонент RNN EasyOCR отвечает за обработку последовательного представления признаков входного изображения, сгенерированного компонентом CNN.

Он состоит из нескольких слоев ячеек долговременной краткосрочной памяти (LSTM), которые предназначены для фиксации долговременных зависимостей в последовательных данных путем выборочного запоминания и забывания информации.

Ячейки LSTM в компоненте RNN EasyOCR складываются вместе, при этом каждый слой передает выходные данные следующему слою в качестве входных. Последний слой LSTM выводит последовательность вероятностей для каждого символа входного текста.

Эта выходная последовательность затем передается через компонент (CTC) для генерации окончательно распознанного текста.

4. **CTC компонент**.

После распределения вероятностей, созданного по символам каждой области входного изображения, обычно используется декодирования, в частности, в декодере на основе CTC, чтобы найти наиболее вероятную последовательность символов с учетом выходных вероятностей.

«Жадный» декодер работает путем выбора наиболее вероятного символа на каждом этапе последовательности. Для каждого символа последовательности модель распознавания выводит распределение вероятностей по всем возможным символам. «Жадный» декодер просто выбирает на каждом шаге символ с наибольшей вероятностью и добавляет его к окончательно распознанному тексту.

Beam search decoder алгоритм декодирования, который можно использовать для повышения точности распознавания текста путем рассмотрения нескольких возможностей на каждом этапе последовательности, а не просто выбора наиболее вероятного символа, как это делает жадный декодер.

Преимущество beam search decoder по сравнению с «жадным» декодером заключается в том, что он может учитывать несколько возможностей на каждом этапе, что может помочь уменьшить ошибки из-за неоднозначности между похожими символами или символами.

5. Пост-процессинг распознанного текста

Постобработка включает применение дополнительных методов или алгоритмов для уточнения результатов системы оптического распознавания символов. Основная цель постобработки — исправить любые ошибки или несоответствия в распознанном тексте и повысить его общее качество.

Некоторые распространенные методы постобработки включают в себя:

- проверка орфографии: исправление слов с ошибками путем сравнения их со словарем;
- языковое моделирование: использование специфичных для языка правил или статистических моделей для улучшения связности и грамматики распознаваемого текста;
- нормализация текста: стандартизация формата распознанного текста путем преобразования сокращений, сокращений или специальных символов в их полные формы или стандартные представления.

EasyOCR (рисунок 7) обеспечивает этап постобработки как часть своего алгоритма для повышения точности и читаемости распознанного текста, но точные используемые методы и алгоритмы могут отличаться.

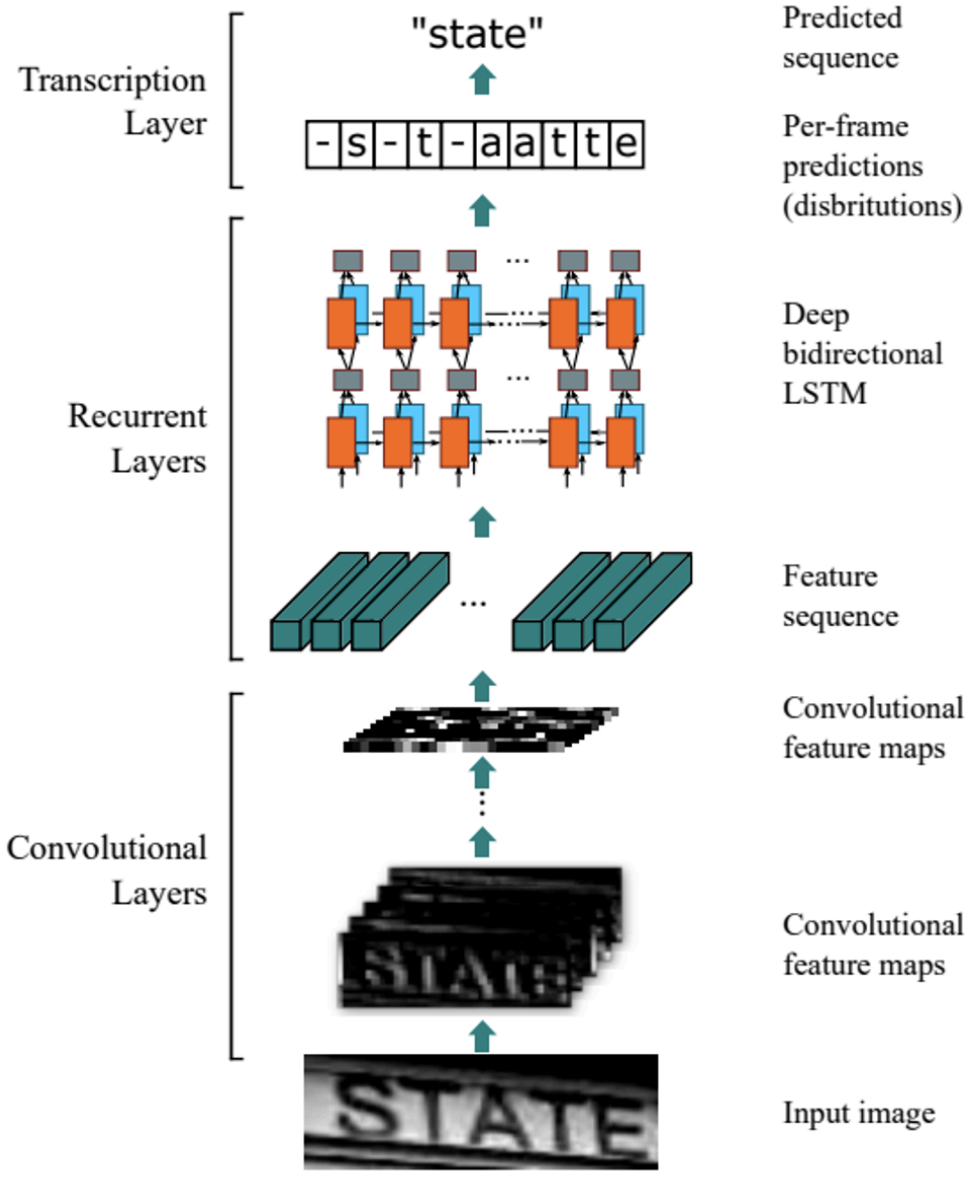

Рисунок 7 – Архитектура EasyOCR

PaddleOCR - основан на фреймворке PaddlePaddle и также предоставляет богатый набор инструментов для работы с текстом на изображениях. PaddleOCR имеет более широкий выбор моделей и поддерживает большее количество языков, включая китайский, японский, корейский и многие другие. Библиотека также предоставляет возможность использовать дополнительные функции, такие как обнаружение текста на изображениях, выравнивание текста и дополнительные оптимизации для работы с текстом на сложных изображениях.[12]

Attention OCR - это архитектура нейронной сети для оптического распознавания символов (OCR), которая использует механизм внимания для сосредоточения на наиболее важных частях изображения при выполнении распознавания текста. Эта архитектура представляет собой комбинацию сверточной нейронной сети (CNN) для извлечения признаков изображения и рекуррентной нейронной сети (RNN) с механизмом внимания для последовательного распознавания символов. [13]

Основная идея Attention OCR заключается в том, чтобы дать модели возможность динамически выбирать, на какие части изображения она должна сосредоточить свое внимание в процессе распознавания текста. Это позволяет модели сосредоточиться на ключевых областях изображения, где находится текст, и игнорировать фоновые элементы или помехи.

Архитектура Attention OCR состоит из следующих основных компонентов:

1. Сверточная нейронная сеть (CNN), которая используется для извлечения признаков изображения. CNN принимает на вход изображение и выдает набор признаков, которые представляют собой высокоуровневые представления изображения.

2. Рекуррентная нейронная сеть (RNN) с механизмом внимания, которая используется для последовательного распознавания текста. RNN принимает на вход признаки изображения, а затем последовательно предсказывает символы текста. В каждом шаге RNN с механизмом внимания сосредотачивается на различных частях изображения, чтобы сделать предсказание на основе текущего контекста.

Преимуществом Attention OCR является его способность работать с изображениями разных размеров и разных стилей текста. Механизм внимания позволяет модели адаптироваться к различным условиям и делает ее более гибкой и точной в сравнении с другими подходами к OCR.

1.3. Анализ существующих современных разработок

Для проведения анализа существующих решений в области разработки программных модулей распознавания номеров железнодорожных вагонов были проанализированы существующие на рынке решения.

В качестве критериев для сравнения были выбраны следующие:

1. Адаптивность - у модуля должна быть предусмотрена функциональная возможность распознавания как по изображению, так и по видеопотоку.
2. Интегрируемость - модуль должен быть интегрируемым с вагонными весами (независимо от производителя).
3. Независимость - возможность использования модуля без интеграции с весовым механизмом.
4. Кроссплатформенность - Возможность запуска решения на ОС Windows 10-11, Linux Desktop/Server 18.04-22.04 и MacOS 12-14.

5. Универсальность - Модуль должен распознавать номер на всех вагонах, контейнерах, цистернах и платформах которые могут быть.

Одним из актуальных решений является программное обеспечение от компании Тенсиб [14], которое предоставляет круглосуточное распознавание номеров на бортах или шасси вагонов, железнодорожных платформ и цистерн в режиме реального времени. Однако их решение не является кроссплатформенным и не работает без интеграции с весовым оборудованием.

Их конкурентом является решение от компании Domination[15], которое распознает вагоны только на видеопотоке, что уже накладывает ограничение на использование этого ПО, вдобавок у него нет возможности интеграции с весовым оборудованием и оно не кроссплатформенно.

Другие решения – от фирмы METRA [16], и SatVision [17], похожи тем, что оба решения не работают со всеми типами вагонов. Вдобавок, решение от METRA не работает без интеграции с весовым оборудованием и не является кроссплатформенным, а решение от SatVision в свою очередь может распознавать вагоны только в видеопотоке.

Таблица 1 – Сравнение современных решений

	Тенсиб	**Domination**	**Metra**	**SatVision**
Адаптивность	+	-	+	-
Интегрируемость	+	-	+	+
Независимость	-	+	-	+
Кроссплатформенность	-	-	-	-
Универсальность	+	+	-	-

После анализа конкурентов можно сформулировать недостатки существующих систем:

- некоторые решения распознают номера только на видеопотоке;
- не все конкуренты поддерживают кроссплатформенность;
- не универсальные. Некоторые решения могут распознавать только номера на вагонах определенного типа, что очень ограничивает их.

Таким образом, на рынке существует несколько решений в области распознавания номеров вагонов, однако данные решения работают только с большим количеством камер конкретного типа и требуют установки дополнительного оборудования. Часть решений может работать исключительно в интеграции с весовым механизмом этой-же компании-производителя, а часть не имеет функциональной возможности интеграции с весовым механизмом вовсе.

1.4. Постановка цели и задач разработки системы

Применение технологий компьютерного зрения позволит автоматизировать процесс контроля подвижного состава на станции обработки, что сократит время финальных операций с грузовыми железнодорожными вагонами; создаст условия для оптимизации затрат времени на обработку вагонов, а также возникновения материальной экономии за счет оптимизации производственных процессов на станции.

Цель: Разработка программного модуля распознавания номеров железнодорожных вагонов на основе современных методов машинного обучения и компьютерного зрения.

Задачи:

1. Провести анализ существующих решений в области распознавания объектов на изображениях.
2. Выявить требования к программному модулю распознавания номеров железнодорожных вагонов.

3. Разработать архитектуру сервиса предлагаемого решения и выбрать нейросетевые алгоритмы для решения задачи.
4. Реализовать программный модуль и провести его тестирование.

Выводы по разделу один

В данном разделе на основе анализа литературы показана актуальность и необходимость применения компьютерного зрения при распознавании номеров. Также был проведен анализ существующих современных разработок и наглядно показано, что имеющиеся на рынке инструменты имеют недостатки.

Для реализации программного модуля распознавания можно использовать следующие технологии:

В качестве инструмента обработки полученного изображения будем использовать OpenCV [18] – это библиотека, которая предоставляет широкий спектр инструментов и алгоритмов для разработки приложений в области распознавания объектов, анализа изображений и видео, например, аффинные преобразования [19]. Средой разработки будет выступать Visual Studio Code (интегрированная среда разработки для языка программирования Python [20]).

Docker [21] для контейнеризации приложения и его компонентов. Это позволит упростить управление приложением и обеспечить его независимость от операционной системы, на которой оно работает.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1. Функциональные требования

1. **Загрузка изображений**: пользователь должен иметь возможность загрузить изображение с для распознавания через API сервиса.
2. **Распознавание текста**: система должна быть способна распознавать номер вагона на загруженных изображениях, независимо от типа вагона.
3. **Возврат результатов**: после распознавания текста система должна возвращать результаты в консоли, где запущен сервис.
4. **Хранение данных**: система должна обеспечивать хранение загруженных изображений и результатов распознавания для последующего доступа пользователей.
5. **Обработка ошибок**: система должна обрабатывать ошибки и предоставлять сообщения об ошибках пользователю в случае непредвиденных ситуаций.

Для реализации данной системы необходим следующий набор подсистем:

1. Бэкенд-сервер, который принимает загруженные пользователем изображения, передает их в сервис распознавания, обрабатывает ответ модели распознавания и возвращает пользователю номер вагона на изображении.
2. Модуль распознавания - использует алгоритмы компьютерного зрения для определения номера на вагоне.
3. СУБД - база данных, обеспечивающая хранение загруженных изображений и данных о распознавании.

2.2. Нефункциональные требования

1. **Производительность**: сервис должен обеспечивать высокую скорость распознавания номера на изображениях (до 1 секунды), чтобы минимизировать время ожидания ответа пользователем.

2. **Надежность**: должна быть эффективная обработка ошибок, чтобы предотвратить сбои и недоступность сервиса.

3. **Требования к хранению данных**:

- решение должно обеспечивать хранение данных, включая изображения и результаты распознавания;
- необходима возможность резервного копирования данных и восстановления в случае сбоев.

4. **Совместимость**:

- сервис должен быть совместим с различными операционными системами (Windows 10, Ubuntu Desktop и Server 18.04-22.04, MacOS 12-14);
- решение должно иметь возможность работать как в интеграции с весовым механизмом, так и без него.

2.3. Обоснование проектных решений

2.3.1. Выбор модели детекции

Для оценки эффективности детекции была проведена серия экспериментов на наборе снимков вагонов. Целью этих экспериментов было оценить точность обнаружения области с номером на изображении.

Для обучения и проведения экспериментов мы использовали набор данных (код 1), состоящий из разнообразных изображений вагонов. Какие-то снимки были приближены к идеальным, а на некоторых была плохая погода, либо номер был не слишком хорошо виден по каким-то другим причинам. Эти

снимки были тщательно размечены для определения точных границ и местоположения области с номером вагона. Примеры изображений приведены на рисунках 8 и 9. Код, использовавшийся для подготовки данных и обучения модели представлен в листинге 1.

Рисунок 8 – Пример изображения из набора данных для обучения

Рисунок 9 – Пример изображения из набора данных для обучения

Листинг 1 – Подготовка модели детекции:

```
# Определяем файл конфигурации для модели. Задаем папки с изображениями, количество классов указываем 1, потому что в предлагаемом решении будет реализован двухэтапный подход, и задачу классификации номера решает другая модель.

data = {
    'names': ['0'],
    'nc': 1,
    'test': '../test/images',
    'train': '../train/images',
    'val': '../val/images'
}

with open('/working/prepared_dataset/data.yaml', 'w') as yaml_file:
    yaml.dump(data, yaml_file, default_flow_style=False)

# После подготовки данных мы обучили модель глубокого обучения YOLOv8s (код 2)
# Создание экземпляра модели YOLOv8s
model = YOLO('yolov8s.pt')

# Обучение модели на данных из тренировочной выборки в течение 200 эпох
model.train(data='/working/prepared_dataset/data.yaml', epochs=200)
```

Затем мы применили модель детекции YOLOv8s, описанную в предыдущем разделе, к каждому изображению из тестового набора данных. Для каждой обнаруженной области номера вагона были вычислены метрики точности. Для оценки точности, мы использовали метрику локализации объекта (IoU) [22]. IoU (Intersection over Union) или Jaccard Index (Индекс Жаккара) на рисунке 10 изображена работа. Данная метрика измеряет степень перекрытия между предсказанными и истинными границами объектов. Чем ближе значение IoU к 1, тем больше пересечение двух рамок. Вычисляется по формуле 1:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|}, \quad (1)$$

где |A∩B| - площадь (в пикселях) перекрытия предсказанной моделью области и площади, указанной при разметке;

|A ∪ B| - площадь (в пикселях) объединения предсказанной моделью области и площади, указанной при разметке.

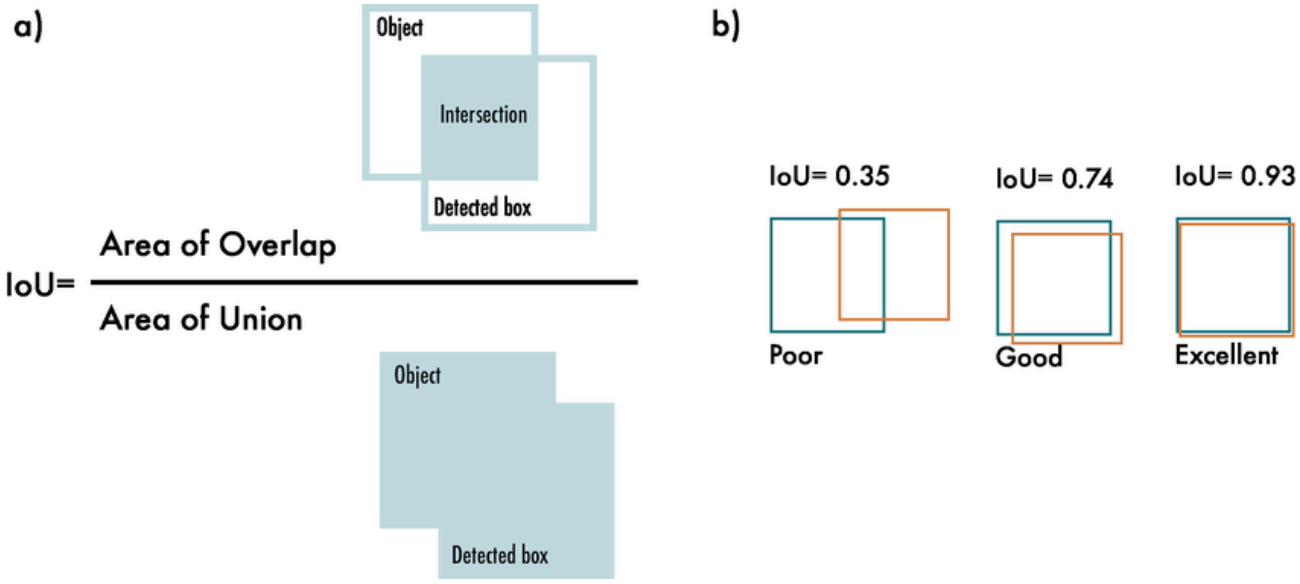

Рисунок 10 – Вычисление коэффициента объединения и пример

Далее, мы сравнили результаты работы каждой модели детекции. В качестве моделей для сравнения были выбраны нейросетевые архитектуры, показывающие наилучшие результаты в различных задачах на текущий момент. Это вариации YOLOv5, YOLOv8 (n - Nano и s - Small версии), а также Faster R-CNN и SSD. Все модели брались из открытых источников: Для YOLO есть специальное API [23] на языке Python, позволяющее легко обучать и использовать любую из версий данной архитектуры. Faster R-CNN и SSD были взяты с официального сайта фреймворка PyTorch. [24]

Был проведен сравнительный анализ точности и скорости работы моделей. Результаты экспериментов показали, что YOLOv8s обладает наибольшей предсказательной способностью для решения нашей задачи, и в тоже время является очень быстрой (таблица 2).

Таким образом, по результатам экспериментов в качестве модели детекции для модуля распознавания была выбрана YOLOv8s.

Таблица 2 – Сравнительный анализ алгоритмов детекции

Модель	**IoU**	**Время обработки одного изображения, мс**
YOLOv8s	0,93	45
YOLOv8n	0,89	39
YOLOv5s	0,91	48
YOLOv5n	0,86	40
Faster R-CNN	0,87	82
SSD	0,86	64

2.3.2. Выбор модели классификации

В данном разделе были проведены эксперименты с различными алгоритмами классификации для предсказания номера на снимках вагонов. Целью экспериментов было сравнение производительности и точности различных моделей классификации.

Перед проведением экспериментов были выбраны следующие модели классификации:

- EasyOCR;
- PaddleOCR;
- AttentionOCR (AOCR).

Каждый из выбранных алгоритмов имеет свои особенности и предполагается, что они могут быть эффективными для предсказания номеров на изображениях вагонов.

Перед проведением экспериментов данные были подготовлены. Исходные изображения были обрезаны моделью-детектором чтобы на изображении находилась только область с номером вагона. Изображения были преобразованы в формат, совместимый с выбранными моделями классификации, в нашем случае в тензоры. Данные были разделены на обучающую и тестовую выборки в соотношении 80:20 и переданы в модели для обучения (листинг 2).

Листинг 2 – Подготовка изображений и обучение модели EasyOCR

```
import os
import shutil
import random

# Путь к папке с данными
data_folder = 'path/to/data_folder'

# Создание папок для train и test
```

```
train_folder = os.path.join(data_folder, 'train')
test_folder = os.path.join(data_folder, 'test')
os.makedirs(train_folder, exist_ok=True)
os.makedirs(test_folder, exist_ok=True)


# Получение списка всех файлов в папке с данными
file_list = os.listdir(data_folder)
random.shuffle(file_list)  # Перемешиваем файлы

# Вычисление количества файлов для train и test
num_train = int(len(file_list) * 0.8)
num_test = len(file_list) - num_train

# Копирование файлов в папки train и test
for i, file_name in enumerate(file_list):
   if i < num_train:
       shutil.copy(os.path.join(data_folder, file_name), os.path.join(train_folder,
file_name))
   else:
       shutil.copy(os.path.join(data_folder, file_name), os.path.join(test_folder,
file_name))
from easyocr.training import train_detector

# Пути к обучающим данным и файлу конфигурации
train_folder = 'path/to/train_folder'
config_file = 'path/to/config.yaml'

# Обучение модели
model = train_detector(config_file, train_folder)
```

По окончании тренировки моделей было проведено тестирование на соответствующей выборке, которая не подавалась на вход модели во время обучения, чтобы узнать реальное качество ответов модели. Также посчитаны метрики (таблица 3).

Для оценки качества работы OCR используются различные метрики. Наиболее распространенной является Character Error Rate (CER) - количество замен, вставок или удалений символов, необходимых для того, чтобы получить истинный текст (формула 2). Для OCR приемлемым значением CER считается примерно 10%, очень хорошим– менее 1%.

$$CER = \frac{S+D+I}{N}, \quad (2)$$

где S – количество цифр, которые модель ошибочно распознала как другие (пример: цифра 7 определилась как 1);

D – количество цифр, которые модель не распознала, пропустила;

I – количество цифр, которые модель ошибочно нашла, хотя их нет;

N – общее число цифр на входном изображении.

Таблица 3 – Сравнение архитектур моделей OCR

Модель	**Character Error Rate, %**	**Время обработки одного изображения, мс**
EasyOCR	8	39
PaddleOCR	12	51
AttentionOCR	11	44

По результатам сравнения было принято решение использовать в модуле распознавания модель EasyOCR как наиболее быструю и эффективную.

3. ОПИСАНИЕ МОДУЛЯ КОМПЬЮТЕРНОГО ЗРЕНИЯ

3.1. Архитектура предлагаемого решения

На рисунке 11 представлена диаграмма работы программного модуля.

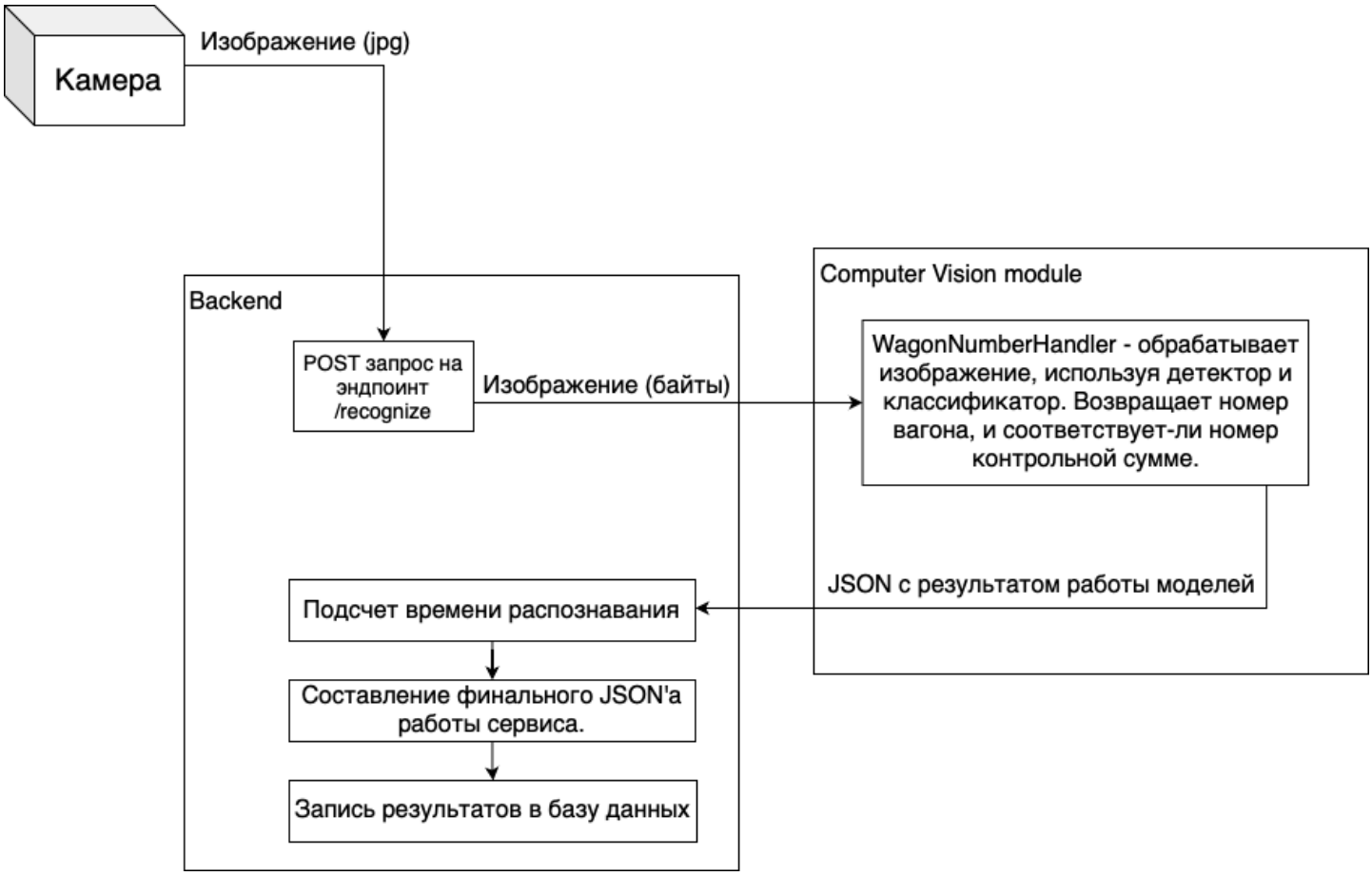

Рисунок 11 – Диаграмма работы программного модуля

На рисунке 12 представлена графическая схема алгоритма модуля распознавания

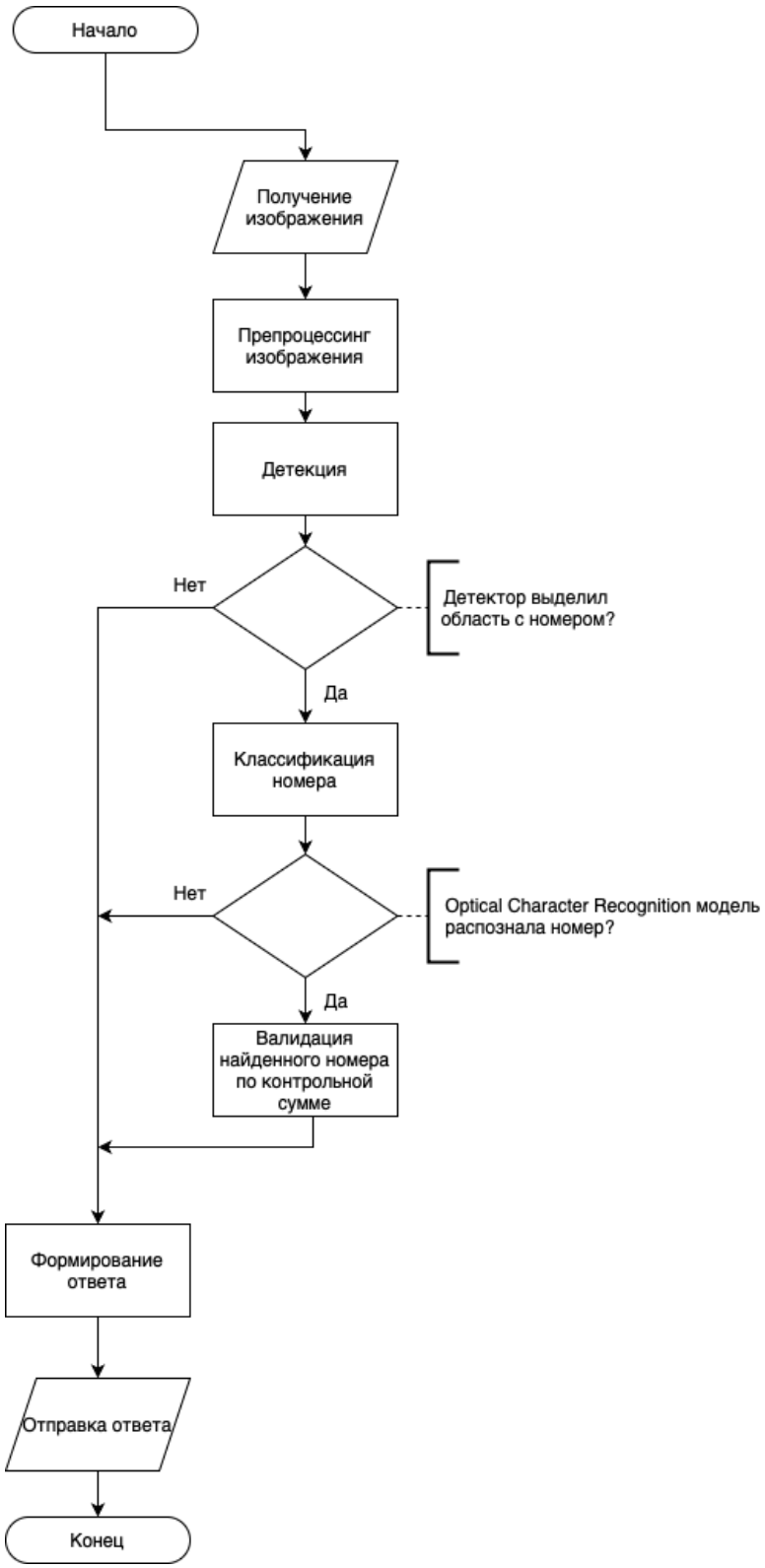

Рисунок 12 – Графическая схема алгоритма модуля распознавания

Реализация функционала работы модуля представлена в листинге А.1 приложения А.

Предлагаемое решение имеет двухэтапный подход:

Первый этап – полученное сервисом изображение отправляется на вход модели-детектору, который определяет область с номером и вырезает ее.

Второй этап – область с номером получает на вход OCR модель, которая распознает и классифицирует номер на фотографии и возвращает строку с номером как ответ.

3.1.1. Описание выбранной модели детекции

Изображение проходит через предварительно обученную нейронную сеть, которая состоит из сверточных и полносвязных слоев. В YOLOv8 часто используются сверточные архитектуры, такие как ResNet [25] или Darknet [26].

Получение выходных данных:

На выходе нейронная сеть выдает прогнозы о наличии объектов на изображении. Для каждого обнаруженного объекта модель возвращает координаты ограничивающего прямоугольника (bounding box), класс объекта и уверенность (confidence) в правильности классификации.

Пороговая фильтрация:

После получения выходных данных производится фильтрация предсказаний, исключая объекты с низкой уверенностью. Обычно устанавливается пороговое значение уверенности, ниже которого объекты не рассматриваются как обнаруженные.

Подавление нескольких областей (Non-maximum suppression) [27]:

Для уменьшения дублирования объектов, близких к друг другу, применяется алгоритм подавления нескольких областей. Он удаляет лишние ограничивающие прямоугольники на основе их перекрытия и уверенности предсказания.

Реализация модели в модуле распознавания представлена в листинге А.3 приложения А.

3.1.2. Описание выбранной модели классификации

Сама модель распознавания состоит из 3 основных компонентов:

– извлечение признаков (ResNet);
– маркировка последовательности (LSTM) [28];

– декодирование (CTC) [29];

Для распознавания последовательностей символов применяется свёрточно-рекуррентная нейронная сеть CRNN (Convolutional Recurrent Neural Network, комбинация DCNN и RNN) и алгоритм CTC (Connectionist Temporal Classification) для декодирования выходных данных нейронной сети в текстовое представление.

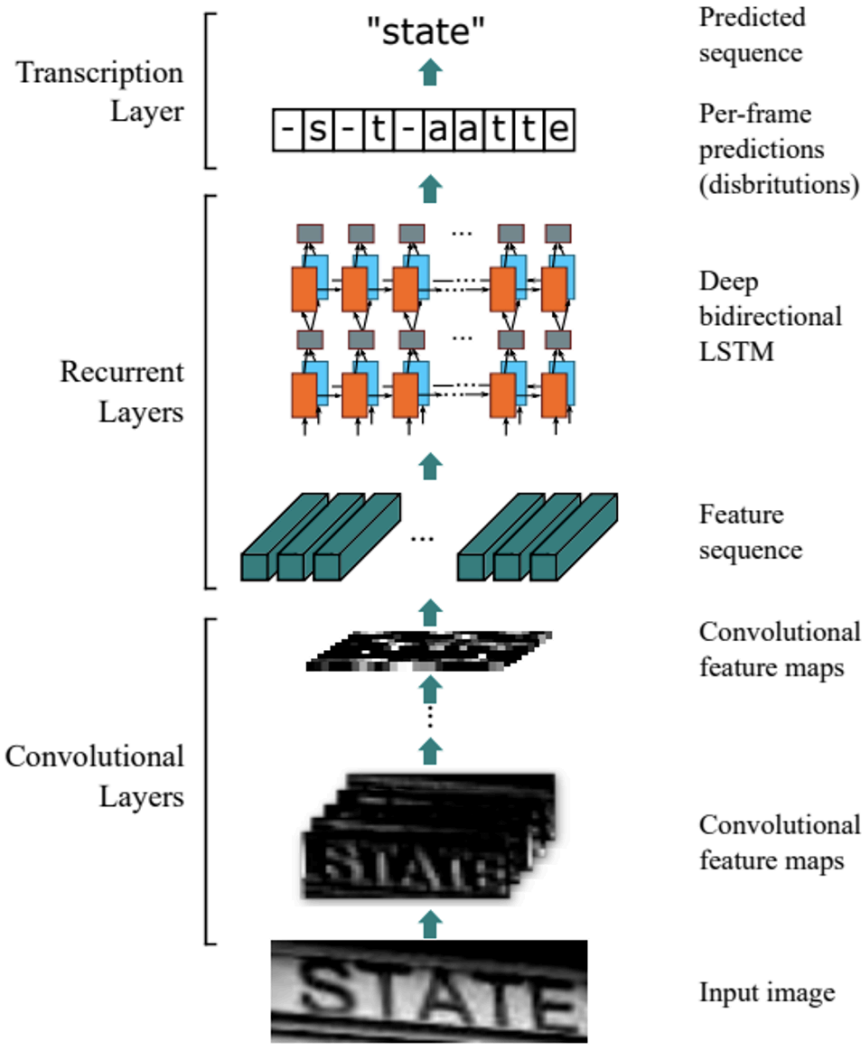

Рисунок 13 – Архитектура модели EasyOCR. Источник [11]

Реализация модели в коде представлена в листинге А.2 приложения А.

3.1.3 Описание данных

На вход сервиса подается изображение в формате jpg/jpeg/png, которое далее преобразуется в байты и подается на вход модулю распознавания. Там оно трансформируется в формат тензора numpy array, которое затем передается в модели компьютерного зрения. На выходе имеем словарь, содержащий ключи wagon_number (номер вагона) и checksum (чексумма – прошел номер проверку на соответствие стандарту записи номеров или нет). После того, как модуль распознавания отработал, сервис создает финальный словарь который включает:

- recognition_result (словарь, ответ модуля распознавания);
- time_of_request (время получение изображения сервером);
- time_of_response (время завершения обработки изображения модулем распознавания);
- recognition_time (время, затраченное на распознавание изображения в секундах).

Словарь отправляется в консоль, где был запущен сервис распознавания. Каждое поле заносится в базу данных (рисунок 14) вместе с изображением в байтах.

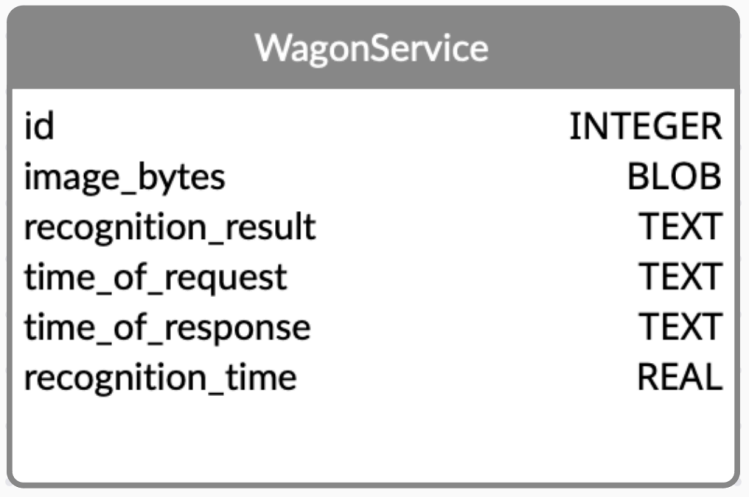

Рисунок 14 – Схема разработанной базы данных истории запусков

В качестве СУБД была выбрана SQLite по нескольким причинам:

- хорошая производительность для небольших и средних нагрузок;
- прост в установке и использовании, не требует дополнительных;
- настроек и администрирования. Она работает как часть приложения;
- надежность: Транзакции и ACID-свойства обеспечивают надежность и целостность данных;
- кроссплатформенность: SQLite поддерживается на множестве платформ, включая Windows, macOS и Linux;
- эффективность: Использует минимальные ресурсы систем.

4. РЕАЛИЗАЦИЯ МОДУЛЯ РАСПОЗНАВАНИЯ И БЕКЕНД-СЕРВЕРА

4.1. Тестирование работоспособности предлагаемого решения

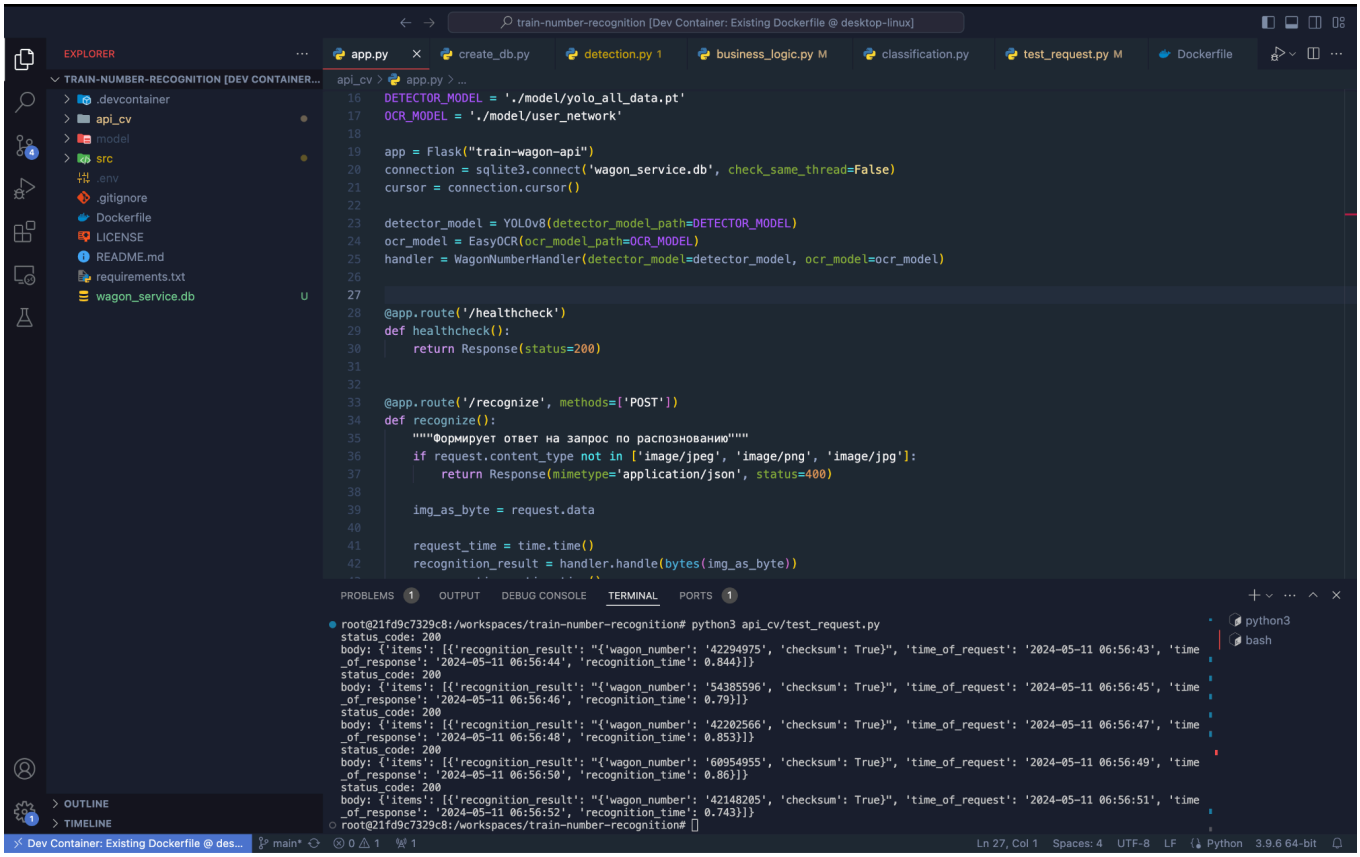

Рисунок 15 – Экранная форма программы

В результате разработки получилось запустить бекенд-серевер, и отправить несколько случайным образом выбранных изображений на распознавание при помощи POST запроса.

Пример изображений, использованных при тестировании (рисунок 16, 17)

Рисунок 16 – Изображение из выборки для тестирования

Рисунок 17 – Изображение из выборки для тестирования

4.2. Тестирование точности и скорости предлагаемого решения

Было проведено тестирование точности распознавания (рисунок 18) и скорости получения ответа от модуля (рисунок 19).

Для тестирования точности распознавания были взяты изображения, не участвовавшие в тренировке модели распознавания, что позволяет говорить о том, что метрики качества, полученные при таком тестировании можно экстраполировать на работу модуля в реальных условиях.

Для тестирования скорости работы модуля 500 раз отправили одно и то же изображение и усреднили полученный результат.

В результате было определено, что точность распознавания (по метрике accuracy) равняется 94%, а среднее время получения ответа равняется 0,22 секунды, или 22 000 мс.

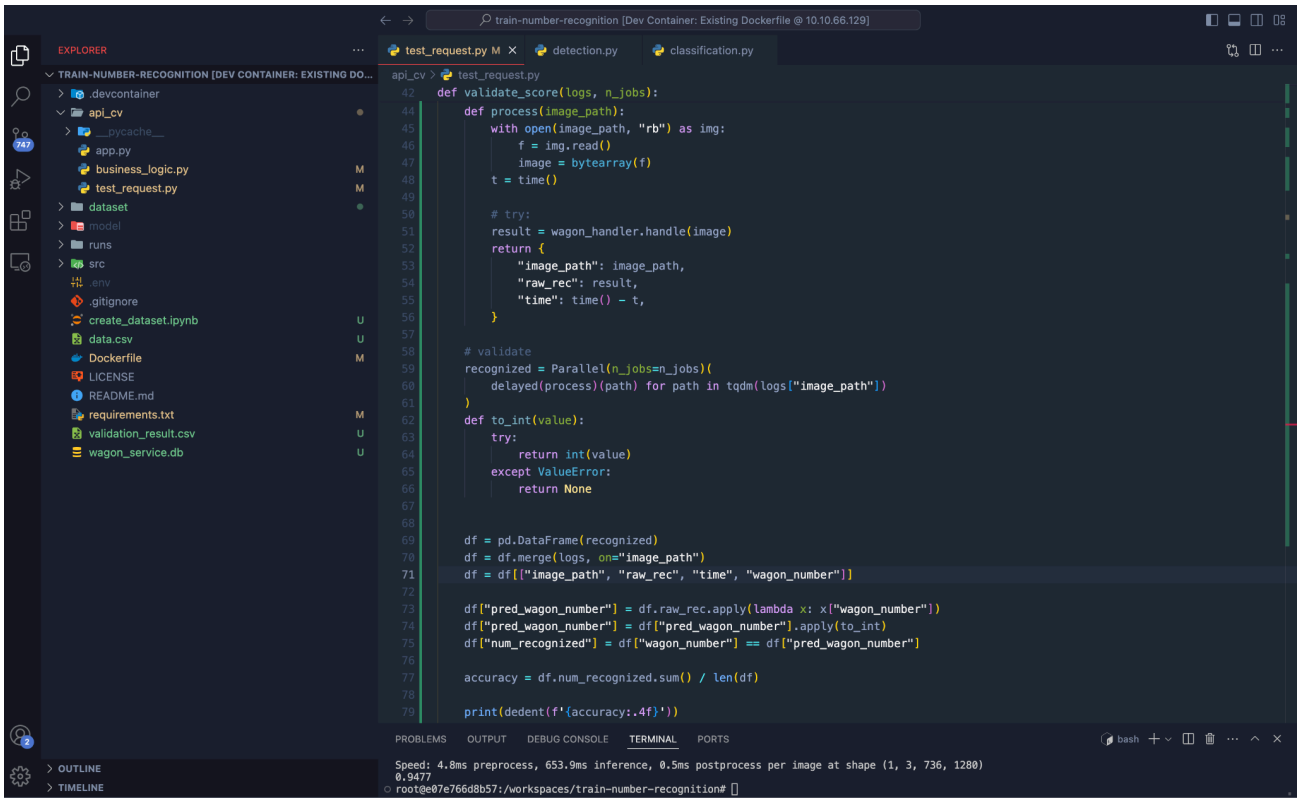

Рисунок 18 – Тестирование точности распознавания

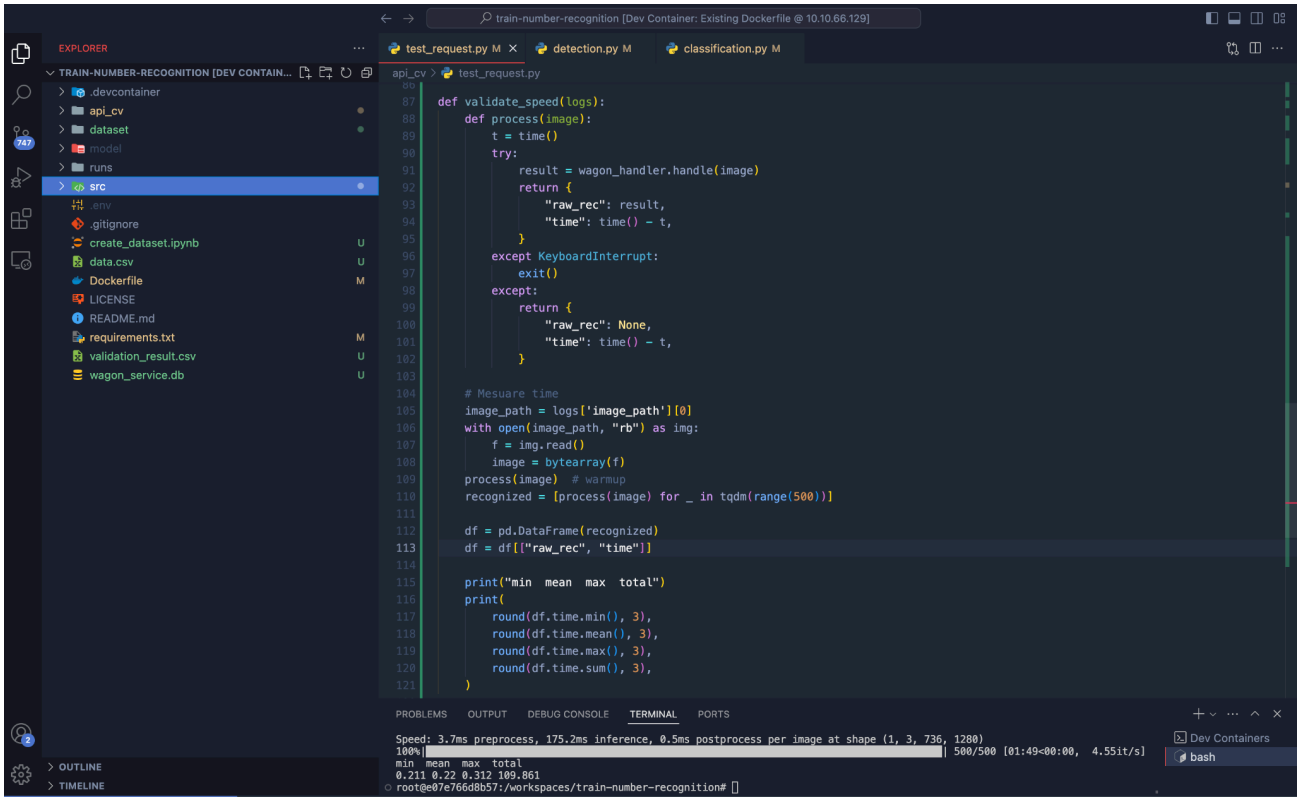

Рисунок 19 – Тестирование скорости получения ответа

ЗАКЛЮЧЕНИЕ

В ходе разработки был проведен анализ существующих современных решений в области распознавания объектов на изображениях, который показал, что решения имеют конкретные недостатки, такие как отсутствие кроссплатформенности, невозможность запуска без весовых систем, или наоборот только в интеграции с ними, также невозможность распознавания номеров железнодорожных вагонов на изображениях и ограниченность в способности распознавания исключительно на вагонах определенного типа.

Были выявлены следующие требования к модулю распознавания:

Функциональные:

1. **Загрузка изображений**: пользователь должен иметь возможность загрузить изображение с для распознавания через API сервиса.
2. **Распознавание текста**: система должна быть способна распознавать номер вагона на загруженных изображениях, независимо от типа вагона.
3. **Возврат результатов**: после распознавания текста система должна возвращать результаты в консоли, где запущен сервис.
4. **Хранение данных**: система должна обеспечивать хранение загруженных изображений и результатов распознавания для последующего доступа пользователей.
5. **Обработка ошибок**: система должна обрабатывать ошибки и предоставлять сообщения об ошибках пользователю в случае непредвиденных ситуаций.

Нефункциональные:

1. **Производительность**: сервис должен обеспечивать высокую скорость распознавания номера на изображениях (до 1 секунды), чтобы минимизировать время ожидания ответа пользователем.
2. **Надежность**: должна быть эффективная обработка ошибок, чтобы предотвратить сбои и недоступность сервиса.
3. **Требования к хранению данных**:
    - решение должно обеспечивать хранение данных, включая изображения и результаты распознавания;
    - необходима возможность резервного копирования данных и восстановления в случае сбоев.

4. **Совместимость**:
    - сервис должен быть совместим с различными операционными системами (Windows 10, Ubuntu Desktop и Server 18.04 и 20.04, MacOS 12-14);
    - решение должно иметь возможность работать как в интеграции с весовым механизмом, так и без него.

Была выполнена разработка собственной системы распознавания номеров. Предлагаемое решение представляет собой модульную систему, которая решает задачи автоматического обнаружения номеров железнодорожных вагонов, их распознавания, а также предоставляет функционал хранения изображений, прошедших распознавание. На вход системе подается изображение в формате jpg/jpeg/png, которое далее подается на вход модулю распознавания. Там его обрабатывают модели компьютерного зрения. На выходе получается словарь, содержащий номер вагона и чексумму – прошел номер проверку на соответствие стандарту записи номеров или нет. После того, как модуль распознавания отработал, сервис создает финальный словарь который включает:

- recognition_result (словарь, ответ модуля распознавания);
- time_of_request (время получение изображения сервером);
- time_of_response (время завершения обработки изображения модулем распознавания);
- recognition_time (время, затраченное на распознавание изображения в секундах).

Словарь отправляется в консоль, где был запущен сервис распознавания. Каждое поле заносится в базу данных вместе с изображением в байтах.

В результате тестирования разработанная система обладает высокой точностью распознавания номеров железнодорожных вагонов, превышая 90% при скорости

предсказания не более 1 секунды. Обработанные запросы сохраняются в базу данных. Модуль также может успешно интегрироваться как в весовые системы, так и работать автономно.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Компьютерное зрение [Электронный ресурс] – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Компьютерное_зрение (дата обращения 05.02.2024)
2. Automatic Counting and Identification of Train Wagons Based on Computer Vision and Deep Learning [Электронный ресурс] - Режим доступа: https://arxiv.org/pdf/2010.16307.pdf (датаобращения 05.02.2024)
3. Development of an automated system for recognizing the parameters of a railway carriage (railway tanks) [Электронный ресурс] - Режим доступа: https://ceur-ws.org/Vol-2803/paper25.pdf (дата обращения 05.02.2024)
4. Ocr accuracy improvement on document images through a novel pre-processing approach [Электронный ресурс] - Режим доступа: https://arxiv.org/pdf/1509.03456.pdf (дата обращения13.02.2024)
5. Challenges in Deploying Machine Learning: a Survey of Case Studies [Электронный ресурс] - Режим доступа: https://arxiv.org/pdf/2011.09926.pdf (дата обращения 13.02.2024)
6. Studying the Practices of Deploying Machine Learning Projects on Docker [Электронныйресурс] - Режим доступа: https://arxiv.org/pdf/2206.00699.pdf (дата обращения 13.02.2024)
7. Rich feature hierarchies for accurate object detection and semantic segmentation [Электронныйресурс] - Режим доступа: https://arxiv.org/pdf/1311.2524 (дата обращения 05.02.2024)

8. SSD: Single Shot MultiBox Detector [Электронный ресурс] - Режим доступа: https://arxiv.org/pdf/1512.02325 (дата обращения 05.02.2024)
9. You Only Look Once: Unified, Real-Time Object Detection [Электронный ресурс] - Режимдоступа: https://arxiv.org/pdf/1506.02640.pdf (дата обращения 05.02.2024)
10. Метод опорных векторов (SVM) [Электронный ресурс] - Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Метод_опорных_векторов_(SVM) (дата обращения 05.02.2024)
11. EasyOCR [Электронный ресурс] - Режим доступа: https://github.com/JaidedAI/EasyOCR(дата обращения 05.02.2024)
12. PP-OCR: A Practical Ultra Lightweight OCR System [Электронный ресурс] - Режим доступа: https://arxiv.org/pdf/2009.09941 (дата обращения 05.02.2024)
13. Attention-based Extraction of Structured Information from Street View Imagery [Электронный ресурс] - Режим доступа: https://arxiv.org/pdf/1704.03549 (дата обращения 05.02.2024)
14. Tensib [Электронный ресурс] - Режим доступа: https://tensib.ru/katalog/avtomatizatsiya-asutp/sistema-videofiksatsii-i-raspoznovaniya-nomerov/sistema-raspoznavaniya-nomerov-vagonov/ (дата обращения 13.02.2024)
15. Domination [Электронный ресурс] - Режим доступа: https://domination.one/products/analitycs/litsenziya-modul-videoanalitiki-raspoznavanie-nomerov-zhd-vagonov-bazovyy-kanal/ (дата обращения 13.02.2024)
16. Metra [Электронный ресурс] - Режим доступа: https://www.metra.ru/services/raspoznavanie-nomerov-zh-d-vagonov/ (дата обращения 13.03.2024)

17. Satvision [Электронный ресурс] - Режим доступа: https://satvision-cctv.ru/catalog/programmnoe_obespechenie/moduli_analitiki/3234/ (дата обращения 17.02.2024)
18. Mastering OpenCV with Python [Электронный ресурс] - Режим доступа: https://opencv.org/university/course/mastering-opencv-with-python/ (дата обращения 17.02.2024)
19. Аффинные преобразования на плоскости [Электронный ресурс] - Режим доступа: https://edu.mmcs.sfedu.ru/pluginfile.php/12359/mod_resource/content/10/Лекция%204.%20Аффинные%20преобразования.pdf (дата обращения 17.02.2024)
20. Лучано, Р. Python. К вершинам мастерства. [Электронный ресурс] — Режим доступа: http://e.lanbook.com/book/93273 (дата обращения 17.02.2024)
21. Docker Docks [Электронный ресурс] — Режим доступа: https://docs.docker.com/get-started/overview/ (дата обращения 17.02.2024)
22. Intersection over Union (IoU) for object detection [Электронный ресурс] — Режим доступа: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (дата обращения 25.03.2024)
23. Ultralytics YOLOv8 Docs [Электронный ресурс] —Режим доступа: https://docs.ultralytics.com/modes/train/ (дата обращения 17.02.2024)
24. Pytorch Docs [Электронный ресурс] — Режим доступа: https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html (дата обращения 17.02.2024)

25. Deep Residual Learning for Image Recognition [Электронный ресурс] —Режим доступа: https://arxiv.org/pdf/1512.03385 (дата обращения 24.03.2024)
26. YOLOv3: An Incremental Improvement [Электронный ресурс] —Режим доступа: https://arxiv.org/pdf/1804.02767v1 (дата обращения 25.03.2024)
27. Non Maximum Suppression [Электронный ресурс] —Режим доступа: https://paperswithcode.com/method/non-maximum-suppression (дата обращения 25.03.2024)
28. A tutorial into Long Short-Term Memory Recurrent Neural Networks [Электронный ресурс] —Режим доступа: https://arxiv.org/pdf/1909.09586 (дата обращения 25.03.2024)
29. Connectionist Temporal Classification Loss [Электронный ресурс] —Режим доступа: https://paperswithcode.com/method/ctc-loss (дата обращения 25.03.2024)

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД МОДУЛЯ РАСПОЗНАВАНИЯ

Листинг А.1 – Класс-обработчик изображений

```
import numpy as np
import cv2

class WagonNumberHandler:
    def __init__(self, detector_model, ocr_model):
        self.detector_model = detector_model
        self.ocr_model = ocr_model

    def check_railcar_number(self, number_str):
        # Проверяем номер на то, состоит-ли он из 8 цифр
        if not number_str.isdigit() or len(number_str) != 8:
            return False

        # Определяем множители для подсчета чексуммы
        multipliers = [2, 1, 2, 1, 2, 1, 2, 1]

        # Считаем сумму частных
        total = sum(int(digit) * multiplier for digit, multiplier in zip(number_str,
multipliers))

        # Проверяем, делится-ли результат на 10
        return total % 10 == 0

    def process_image(self, image_bytes: bytes):
        # получение изображения
        image = np.asarray(bytearray(image_bytes), dtype="uint8")
        image = cv2.imdecode(image, cv2.IMREAD_COLOR)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        # вызов детектора
        result = self.detector_model.predict(image)
        bboxes = result[0].boxes.xyxy.cpu().numpy()
        if len(bboxes) != 0:
            contours = bboxes[0]
        else:
            print('Детектор не нашел область с номером')
            return 'detector error'

        cropped_image = image[int(contours[1]):int(contours[3]),
                              int(contours[0]):int(contours[2])]
```

Окончание листинга А.1

```
        # вызов ocr модели для распознавания номера на выделенной области
        ocr_result = self.ocr_model.predict(cropped_image)
        if ocr_result is not None:
            wagon_number = ocr_result

        return wagon_number

    def handle(self, image):
        wagon_number = self.process_image(image)
        checksum = self.check_railcar_number(wagon_number)

        result = {"wagon_number": wagon_number, "checksum": checksum}
        return result
```

Листинг А.2 – класс модели-классификатора номера

```
from abc import ABC, abstractmethod
import easyocr

class OpticalCharacterRecognizer(ABC):
    @abstractmethod
    def predict(self, cropped_image):
        pass

class EasyOCR(OpticalCharacterRecognizer):
    def __init__(self, ocr_model_path):
        self.ocr_model_path = ocr_model_path
        self.reader = easyocr.Reader(['ru'],
                        model_storage_directory=self.ocr_model_path,
                        user_network_directory=self.ocr_model_path,
                        recog_network='custom_ocr',
                        detector=False,
                        gpu=False,
                        )

    def predict(self, cropped_image) -> str:
        # Получает обрезанное детектором изображение, отдает номер
        ocr_result = self.reader.recognize(cropped_image)
        number = ocr_result[0][-2]
        return number
```

Листинг А.3 – Класс модели-детектора области с номером

```
from abc import ABC, abstractmethod
import numpy as np
from ultralytics import YOLO


class Detector(ABC):
   @abstractmethod
   def predict(self, image: np.ndarray):
       pass

class YOLOv8(Detector):
   def __init__(self, detector_model_path):
       self.detector_model_path = detector_model_path
       self.model = YOLO(self.detector_model_path)

   def predict(self, image):
      # Модель получает на вход исходное изображение, находит область с номером и
возвращает ее координаты
       result = self.model.predict(image, save=True, imgsz=1280, conf=0.6)
       return result
```

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД БЭКЕНД-СЕРВЕРА

Листинг В.1 – Скрипт с серверной логикой работы модуля распознавания

```
# Импоритируем нужные для работы сервера модули и библиотеки
import json
import logging
from datetime import datetime
import time
from flask import Flask
import sqlite3

from flask import Response, request
from src.classification import EasyOCR
from src.detection import YOLOv8
from api_cv.business_logic import WagonNumberHandler
# Определяем пути до файлов с моделями
DETECTOR_MODEL = os.getenv('DETECTOR_MODEL')
OCR_MODEL = os.getenv('OCR_MODEL')
# Создаем само приложение, которое будет получать запросы, и подключаемся к БД
app = Flask("train-wagon-api")
connection = sqlite3.connect('wagon_service.db', check_same_thread=False)
cursor = connection.cursor()
# Создаем экземпляры классов моделей и обработчика.
detector_model = YOLOv8(detector_model_path=DETECTOR_MODEL)
ocr_model = EasyOCR(ocr_model_path=OCR_MODEL)
handler = WagonNumberHandler(detector_model=detector_model, ocr_model=ocr_model)

# Определяем эндпоинты
@app.route('/healthcheck')
def healthcheck():
   return Response(status=200)

@app.route('/recognize', methods=['POST'])
def recognize():
   """Формирует ответ на запрос по распознаванию"""
   if request.content_type not in ['image/jpeg', 'image/png', 'image/jpg']:
       return Response(mimetype='application/json', status=400)
   img_as_byte = request.data
   # Отправляем изображение в обработчик и замеряем время ответа
   request_time = time.time()
   recognition_result = handler.handle(bytes(img_as_byte))
   response_time = time.time()
   elapsed_time = response_time - request_time
```

Окончание листинга В.1

```
     logging.info({'message': f'Image processing on CV completed in {elapsed_time} seconds',
                   'elapsed_time': elapsed_time,
                   'state': 'processing_completed'})

   final_result = {
       "items": [
           {
               "recognition_result": str(recognition_result),
                                                                    "time_of_request": datetime.fromtimestamp(request_time).strftime('%Y-%m-%d %H:%M:%S'),
                                                                   "time_of_response": datetime.fromtimestamp(response_time).strftime('%Y-%m-%d %H:%M:%S'),
               "recognition_time": round(elapsed_time, 3)
           }
       ]
   }
    sql = 'INSERT INTO WagonService (image_bytes, recognition_result, time_of_request, time_of_response, recognition_time) VALUES (?,?,?,?,?)'
   val = (img_as_byte,
          final_result['items'][0]['recognition_result'],
          final_result['items'][0]['time_of_request'],
          final_result['items'][0]['time_of_response'],
          final_result['items'][0]['recognition_time'])
  # Отправляем результаты в базу данных
   cursor.execute(sql, val)
   connection.commit()

               return     Response(response=json.dumps(final_result),     status=200, mimetype="application/json")

if __name__ == "__main__":
   # Запуск модуля распознавания
   app.run(host="0.0.0.0", port=5000)
```