Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет социально-экономических и компьютерных наук

Белова Александра Владимировна

РАЗРАБОТКА МОДЕЛЕЙ ДЛЯ ПОИСКА ГЕОНОВ И ИСПОЛЬЗОВАНИЯ ИХ В ЗАДАЧАХ ИДЕНТИФИКАЦИИ ОБЪЕКТОВ ВНЕШНЕГО МИРА И ОКРУЖАЮЩЕГО ПРОСТРАНСТВА ПО ФОТОГРАФИЧЕСКИМ ИЗОБРАЖЕНИЯМ

Выпускная квалификационная работа

студента образовательной программы «Программная инженерия»
по направлению подготовки *09.03.04 Программная инженерия*

Руководитель
Доцент кафедры
информационных технологий
в бизнесе.

_____

Л.А. Мыльников

Пермь, 2025 год

Аннотация

Работа посвящена разработке алгоритма компьютерного зрения для классификации типов сцен изображений по набору примитивных объектов на них. Описана реализация программной системы для тестирования алгоритма.

Во «Введении» представлено обоснование актуальности исследования алгоритма классификации изображений на основе примитивных составляющих объектов на них. Определены объект и предмет исследования, поставлены цель и задачи.

В первой главе приведен анализ предметной области компьютерного зрения в робототехнике, анализ существующих методов детекции объектов и классификации изображений. На основе анализа выявлены функциональные и пользовательские требования к системе.

Вторая глава описывает процесс проектирования ансамбля моделей детекции и классификации для реализации алгоритма. Приведен процесс формирования набора данных для обучения моделей на основе реальных и синтетических изображений. Приведены результаты обучения моделей детекции, процесс разработки и тестирования моделей классификации на основе механизма внимания GRU с использованием библиотеки PyTorch. Проведено объединение моделей нейросетей в алгоритм классификации изображений.

В третьей главе представлен процесс проектирования и разработки системы на языке Python для тестирования алгоритма. На основе требований, выявленных на этапе анализа выявлены прецеденты использования и сформированы диаграммы последовательности в нотации UML.

Четвертая глава содержит результаты тестирования алгоритма на фотографиях реального мира. Описан процесс тестирования программной системы с помощью HTTP запросов инструментом Postman.

Работа содержит 87 страниц основного текста, 50 страниц приложений, 53 рисунков и 12 таблиц. Библиографический список включает 19 наименований.

«Разработка моделей для поиска геонов и использования их в задачах идентификации объектов внешнего мира и окружающего пространства по фотографическим изображениям», Белова А.В.

Оглавление

Введение

В настоящее время область робототехники активно развивается и востребованность роботизированных систем, использующих компьютерное зрение, растет. Примером таких систем могут быть роботы-манипуляторы, роботизированные конвейеры на производстве и роботы со свободой передвижения в сфере услуг и рекламы.

Компьютерное зрение позволяет роботам выполнять задачи навигации в пространстве, распознавания объектов и манипуляции ими для выполнения поставленных задач. В основе систем компьютерного зрения роботов могут лежать методы машинного обучения или искусственного интеллекта для задач детекции и классификации. Однако многие роботы и манипуляторы ориентируются в пространстве с помощью заранее заданных алгоритмов и схем локаций. Основной проблемой таких решений являются столкновения с объектами, вызываемые изменениями окружения или незапланированными смещениями. Дополнительно, недостаток разнообразных аннотированных данных влияет на качество обучения систем зрения.

Было решено проанализировать существующие методы компьютерного зрения для идентификации объектов и классификации изображений и исследовать возможность создания нового метода на основе рассмотренных идей. Также требуется разработать систему, позволяющую применить данный метод для получения информации с изображений, полезной для задач навигации в пространстве и манипуляции объектами.

На этапе анализа был выявлен перспективный подход к задачам детекции и классификации, использующий разложение сложных объектов на примитивные составляющие. Использование данной идеи в различных сценариях и задачах показало хорошие результаты и позволило повысить генерализацию обучаемых моделей нейронных сетей. Было принято решение изучить возможность применения метода получения контекстной информации из примитивных объектов на изображении для классификации сцен.

Поставлена **гипотеза** - с помощью нейронной сети возможно определить класс локации на изображении только по набору примитивных объектов и их положению на нем.

Таким образом, **предметом исследования** являются модели и методы компьютерного зрения для идентификации объектов и классификации типов окружения с помощью примитивных объектов.

Объектом исследования являются фотографические изображения объектов и локаций окружающего мира.

Цель исследования - разработка программной системы с моделями нейронных сетей для поиска примитивных объектов и использования их в задачах идентификации объектов и окружающего пространства по фотографическим изображениям.

Для достижения цели исследования были поставлены следующие **задачи**:

1. Провести обзор и анализ существующих методов и моделей, способов детекции и классификации в компьютерном зрении и определить требования к разрабатываемой системе.
2. Подготовить данные для обучения и тестирования моделей.
3. Осуществить отбор моделей для идентификации объектов на изображении, разработать модель классификации сцен изображений, обучить и ансамблировать модели.
4. Провести оценку качества отдельных моделей и ансамбля с использованием метрик F1-score и mAP, определить статус гипотезы.
5. Разработать архитектуру системы для решения задач идентификации объектов и классификации окружающего пространства на изображениях, реализовать и протестировать систему.

Глава 1 Анализ существующих способов решения задач детекции и классификации

В данной главе описан анализ существующих методов детекции объектов и классификации изображений. Рассмотрены идеи использования примитивных составляющих объектов для улучшения качества классификации моделей.

1.1 Введение в область компьютерного зрения в робототехнике

В последнее время область робототехники активно развивается и роботизированные устройства находят свое применение в различных областях, например в производстве, сельском хозяйстве, рекламе.

Зачастую роботы используются для выполнения физических задач, таких как перемещение, сортировка объектов, сборка деталей или передвижение в пространстве. Для выполнения таких задач используются специальные алгоритмы или системы искусственного интеллекта, анализирующие окружение для расчета движений и выполняющие их. Важной частью данных систем являются системы компьютерного зрения, позволяющие в реальном времени анализировать окружение и выделять объекты интереса для взаимодействия или обхода препятствий, для построения пути движения.

Основным подходом к решению задач детекции и классификации в системах компьютерного зрения являются модели нейронных сетей, способные обрабатывать как двухмерные изображения, так и трехмерные сцены, полученные с помощью лидаров (LiDAR) и аналогичных устройств. Второй метод может показывать более точные результаты, но высокая стоимость оборудования делает его менее доступным.

Модели нейронных сетей, работающие с двухмерными изображениями и не требующие специального оборудования, активно развиваются и используются в робототехнике для определения точек захвата объектов, отслеживания перемещения объектов и классификации [2].

1.2 Анализ существующих способов детекции объектов и классификации изображений в компьютерном зрении

Классическим подходом к решению задач детекции и классификации являются сверточные нейронные сети (Convolutional Neural Networks). Их основной идеей является использование сверточных слоев для эффективного выделения паттернов и последовательностей пикселей, которые в дальнейшем сравниваются с данными обучения и используются для определения класса изображения или объекта. Данный подход демонстрирует высокое качество распознавания классов, обученные на одних данных сверточные модели разных архитектур, таких как Alexnet, VGG-16 и Inception-V3, демонстрируют метрики качества в 79-90% [3].

Модели сверточных нейронных сетей также успешно применяются в прикладных исследованиях и проектах. Возможности сверточных нейросетей были продемонстрированы в задачах идентификации и определения расположения объектов в неструктурированных сценариях [4]. Метод позволил роботу-манипулятору в реальном времени распознавать объекты и оптимальные способы их захвата со средней точностью в 99.19%.

Описанные выше методы показывают довольно высокое качество распознавания и локализации объектов, однако модели зачастую обучаются распознавать определенные объекты и не могут адаптироваться к новым вариациям объектов без дополнительного обучения. Для создания более обобщенной модели детекции был предложен новый подход, использующий методы сегментации и упрощения трехмерных объектов до их примитивных составляющих (см. рисунок 1) [5].

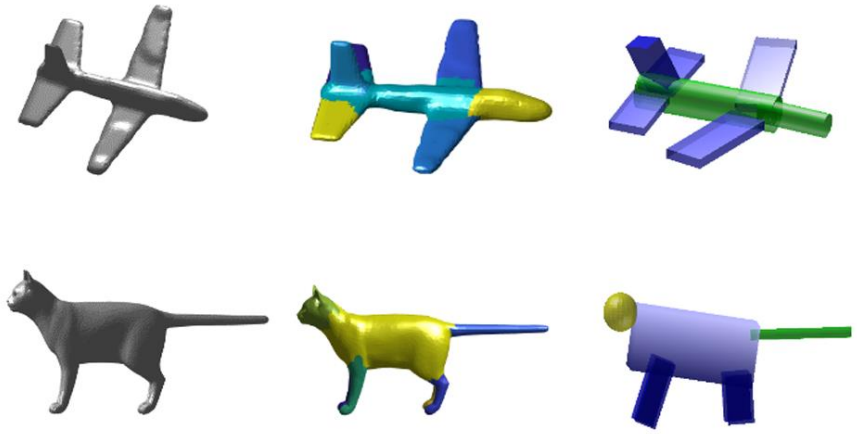

Рисунок 1 - Пример сегментации и упрощения сложного объекта до примитивных составляющих [5]

Из облака точек сложного объекта, полученного с помощью камеры с восприятием глубины, обученной моделью выделялись примитивные объекты из заранее определенного списка – сфера, эллипсоид, цилиндр и параллелепипед. Далее наборы примитивных объектов передавались на вход модели детекции для определения точек захвата объекта. Данный подход позволил с высокой точностью предсказывать возможные точки захвата для объектов, на которых модель не была обучена.

Метод сегментации комплексных объектов на примитивные части может быть успешно использован для классификации не только трехмерных объектов, но и двухмерных изображений. Исследователями была предложена идея рекурсивного поиска вложенных иерархичных структур в изображениях [6]. На рисунке 2 представлен пример рекурсивного разложения – на изображении выделяется группа людей, дерево и здание, которое далее может быть разделено на крышу и основную часть. Процесс может продолжаться до выделения самых базовых частей объектов. Информация о компонентах изображения и их внутренних иерархичных связях позволяет модели классификации более точно определить контекст изображения и повысить качество распознавания.

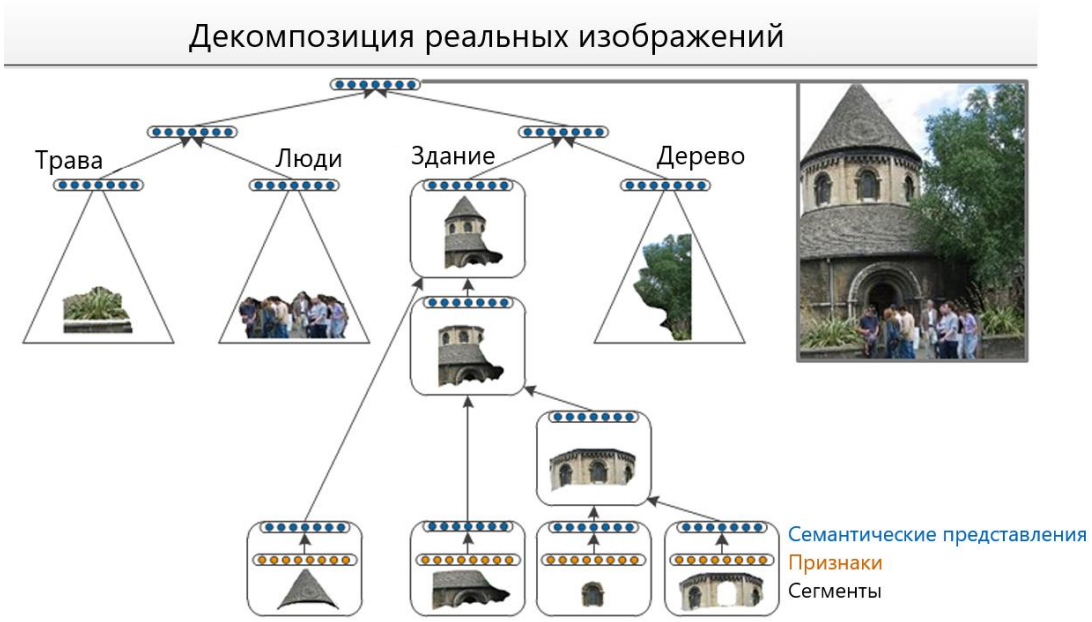

Рисунок 2 - Пример рекурсивного разложения изображения на составляющие Адаптировано из [6]

Идея использования примитивных компонентов объектов была также успешно применена в задаче идентификации двухмерных объектов [1]. Алгоритм использует сверточную нейронную сеть для определения составных примитивных частей объекта и их отношений, например «находится над», «стоит на» или «больше чем», создавая описание объекта для определения его класса. Данный алгоритм позволил создать модель, довольно точно определяющую классы объектов, но при этом не требующую для обучения широкой выборки изображений разнообразных объектов окружающего мира. Для расширения списка распознаваемых объектов достаточно добавить описание предмета на уровне его примитивных частей.

Методы получения дополнительной контекстной информации об объектах или изображениях на основе их примитивных составляющих показали хорошие результаты в задачах детекции объектов и классификации изображений. Идентификация примитивных объектов, их расположения и способов взаимодействия с ними может помочь роботам строить более точные модели окружающего пространства и повысить качество выполняемых задач.

Возникает вопрос, можно ли с помощью нейронной сети определить класс локации на изображении только по набору примитивных объектов и их положению на нем.

1.3 Требования к разрабатываемой системе классификации сцен

Для исследования и тестирования нового метода классификации изображений на основе примитивных объектов необходимо разработать программную систему с моделями нейронных сетей, удовлетворяющую следующим **бизнес-требованиям**:

1. Идентифицировать на изображении примитивные объекты из заранее определенного списка классов, определить их расположение и размер.

2. Классифицировать тип локации на изображении по набору примитивных объектов и их расположению на нем, используя заранее определенный набор классов.

Основными пользователями системы будут являться лица, создающие роботов, которые используют системы компьютерного зрения для навигации в пространстве и для манипуляции окружающими объектами.

Разрабатываемая система будет использована для определения статуса поставленной гипотезы, а также может быть использована системами компьютерного зрения роботов на этапе обработки изображений, для предоставления роботу необходимой информации об окружении.

Процесс анализа роботом окружения для выполнения задач манипуляции объектами или навигации в пространстве представлен в формате BPMN диаграммы на рисунке 3. Для обработки изображения и выделения объектов интереса робот передает изображение с камер системам зрения и получает результаты обработки для дальнейшего использования.

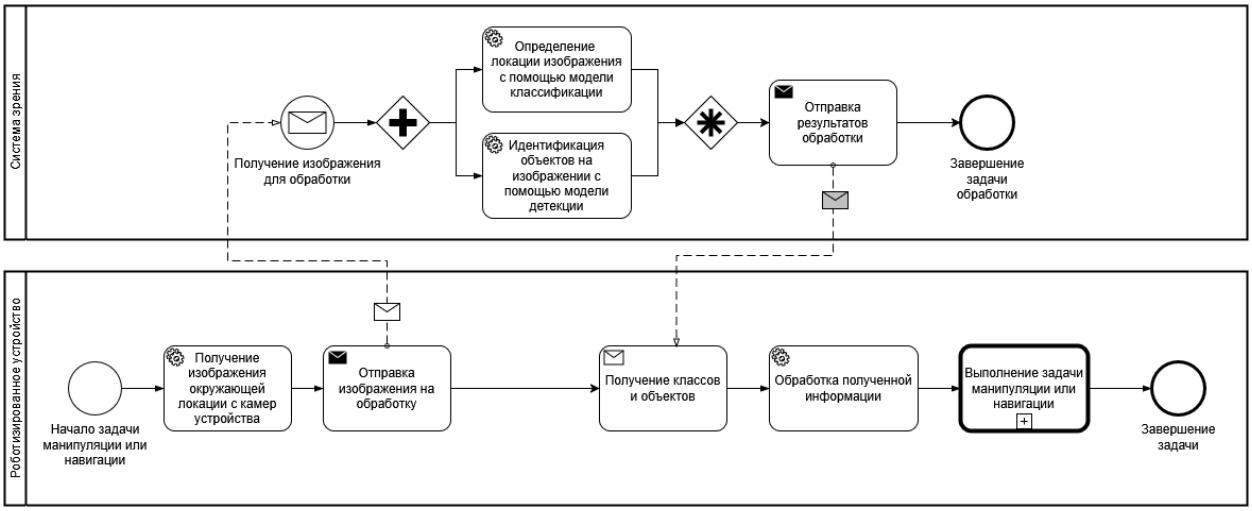

Рисунок 3 - Диаграмма процесса анализа окружения с системами зрения AS IS

На основе информации, полученной на этапе анализа, могут быть выделены следующие **пользовательские требования** к разрабатываемой системе:

- Пользователи и внешние системы могут загружать изображение в систему для обработки и получать результат предсказаний примитивов и вероятностей классов сцены.
- Пользователи и внешние системы могут загружать в систему изображения форматах PNG, JPEG, JPG для обработки.

Системы компьютерного зрения зачастую являются одним из модулей роботизированного устройства и используют технологии передачи данных по сети для обмена информацией, одной системой зрения может единовременно пользоваться несколько устройств. Для удобства обработки предсказаний системами робота или человеком требуется разработать формат хранения данных. Наиболее гибким и популярным является формат JSON, отлично подходящий для передачи последовательностей данных предсказаний различной длины.

Для обеспечения работы разрабатываемой системы с возможными внешними системами необходимо реализовать следующие **нефункциональные требования**:

1. Система должна реализовывать возможность параллельной обработки нескольких запросов.
2. Система должна реализовывать передачу изображений и результатов обработки по сети с использованием протокола HTTP.
3. Для передачи результатов предсказания (ограничивающие рамки объектов детекции и вероятности классов локации) система должна использовать формат JSON.

Системы компьютерного зрения должны быть способны быстро обрабатывать изображения для работы в режиме реального времени. Основным способом ускорения выполнения расчетов является технология параллельных вычислений CUDA, поддерживаемая графическими процессорами Nvidia. Устройства без поддержки технологии пусть и медленнее, но должны быть также способны использовать систему для получения предсказаний моделей нейросетей.

Система моделей нейронных сетей должна соответствовать следующим требованиям для эффективного выполнения задач:

- Обработка одного изображения системой должна занимать не более одной секунды на тестовом стенде (процессор Intel Xeon CPU E5-2666 v3, видеокарта Nvidia GeForce RTX 3060 с поддержкой CUDA, оперативная память 32ГБ).
- Система поддерживает использование технологии CUDA на графических ускорителях (производитель Nvidia).
- Система поддерживает работу на устройствах с графическими ускорителями без поддержки технологии CUDA.

Выводы по главе 1

В результате проведенного анализа существующих решений для задач детекции и классификации в системах компьютерного зрения сделан вывод о том, что использование примитивных составляющих объекта или изображения является перспективным методом для повышения качества их классификации нейросетевыми моделями.

Использование поэлементного распознавания позволяет разработать новый метод классификации изображений.

Для системы, разрабатываемой в целях проверки возможностей алгоритма распознавания примитивов и сцен на изображениях, сформулированы пользовательские, нефункциональные и бизнес-требования, на их основе составлено техническое задание на разработку системы (см. ПРИЛОЖЕНИЕ А).

Глава 2 Разработка алгоритма классификации сцен на изображении по набору примитивных объектов

В данной главе описан процесс разработки алгоритма классификации сцен на изображении с использованием набора примитивных объектов. Рассмотрены модели нейросетей для выполнения задач детекции объектов и классификации изображений. Собраны и аннотированы данные для обучения моделей.

2.1 Проектирование алгоритма идентификации примитивов и классификации сцен

Для определения статуса поставленной гипотезы необходимо разработать алгоритм выделения примитивных объектов на изображении и обработки полученной информации для определения класса локации изображения.

2.1.1 Проектирование ансамбля моделей

Комплексные задачи классификации можно решать, разрабатывая модели нейросетей со сложной архитектурой, множеством слоев и параметров, однако разделение задач на более простые подзадачи может показать лучшие результаты при меньших затратах вычислительных мощностей и времени на обучение моделей.

Одним из подходящих методов является ансамблирование нескольких моделей нейросетей. Узкоспециализированные модели, способные извлекать дополнительную контекстную информацию для использования другими моделями, показывают заметное улучшение в качестве результатов классификации. Успешным примером использования данной стратегии является ансамбль моделей для классификации эмоций человека на изображении – модель детекции микро-выражений лица используется для получения дополнительной информации, используемой моделью классификации, анализирующей общее изображение и результаты работы первой модели [7].

В целях проверки гипотезы будет реализован ансамбль моделей детекции и классификации, однако модель классификации локации не будет иметь доступа к изначальному изображению, только к набору примитивных объектов, полученных с помощью модели детекции.

В результате разработки системы для классификации сцен ожидается изменить процесс анализа окружения – работа моделей детекции объектов и классификации сцены изображения будет проводиться не параллельно независимыми моделями, а последовательно системой связанных друг с другом моделей (см. рисунок 4)

Рисунок 4 - Диаграмма процесса анализа окружения с системами зрения TO BE

2.1.2 Выбор модели детекции примитивных объектов

Основной фокус данного исследования заключается в проверке поставленной гипотезы и разработке метода классификации изображений – для выполнения задачи детекции примитивных объектов будет использована готовая модель нейросети, дообученная на необходимые классы.

Наиболее популярным методом детекции объектов на изображениях являются сверточные нейронные сети семейств YOLO, R-CNN и SSD. Модели двухэтапной детекции, например R-CNN, отличаются повышенной точностью, но более низкой скоростью вычислений. Для задачи детекции объектов в реальном времени более приоритетным параметром модели будет скорость обработки изображения, которая в среднем выше у моделей детекции с одним этапом – YOLO и SSD.

Для описания размера и положения найденных на изображении объектов модели детекции используют ограничивающие рамки (bounding box) следующих вариантов:

- Горизонтальные рамки, параллельные осям изображения.
- Ориентированные рамки, содержащие информацию об угле поворота объекта на изображении.

Первый подход является классическим и наиболее часто применяемым в прикладных задачах в разных областях робототехники. Одним из применений

горизонтальных ограничительных рамок роботами в области сельского хозяйства является использование моделей семейства YOLO для детекции томатов и определения их уровня спелости перед сбором урожая [8].

Однако горизонтальные рамки не всегда являются оптимальным решением для задач реального мира. При детекции удлиненных объектов, расположенных под углом, рамки зачастую включают слишком большое количество фона вокруг объекта, приводя к снижению точности детекции (см. рисунок 5) [9]. Использование ориентированных рамок для определения положения объекта в условиях реального окружения показало заметное улучшение точности локализации объектов [9, 10].

***Рисунок 5 - Сравнение горизонтальных и ориентированных ограничительных рамок объектов** [9]*

Данный подход является более перспективным для решения задач детекции объектов в робототехнике, предоставляя более точную информацию о положении и направлении объекта для манипуляций или навигации, и будет использован в исследовании.

Модели детекции, использующие ориентированные ограничительные рамки будут предоставлять следующую информацию о примитивных объектах для задачи классификации:

- класс объекта;
- координаты объекта по осям X и Y изображения;
- ширина и высота объекта;
- угол поворота объекта.

Для выбора итоговой модели детекции алгоритма будет проведено сравнение ряда моделей разных версий и размеров, способных работать с ориентированными

рамками – YOLOv8, YOLOv11. Модели, предобученные распознавать широкий ряд повседневных объектов, будут модифицированы под задачу исследования и дообучены распознавать примитивные фигуры.

Реализации моделей семейств R2CNN и R-SSD, работающих с ориентированными рамками объектов, не находятся в открытом доступе и не будут тестироваться в рамках проекта.

2.1.3 Выбор архитектуры модели классификации сцен

Изображения реального мира могут содержать различное количество объектов – модель классификации должна быть способна обрабатывать векторы примитивных объектов разного размера для определения класса сцены.

Для решения такого типа задач используются архитектуры моделей с механизмами внимания:

- LSTM (Long Short-Term Memory);
- GRU (Gated Recurrent Unit).

Архитектура рекуррентных нейронных сетей с LSTM позволяет запоминать элементы длинной последовательности и использовать их для принятия решений. Модели с LSTM могут замечать сложные зависимости между объектами и признаками с помощью долгосрочной памяти, что увеличивает качество классификации.

Примером успешного применения данного метода является разработка модели классификации кожных повреждений для идентификации рака кожи по набору изображений [11]. Использование комбинации модели детекции, извлекающей набор признаков из изображения, и модели с LSTM для обработки вектора признаков позволило повысить качество классификации с 87-94% до 90-95% по сравнению с классическим подходом с единственной моделью. Метод также позволил снизить количество ложно негативных предсказаний и показал свою эффективность в задачах классификации с векторами данных разных размеров.

Другим механизмом внимания, способным извлекать важную информацию из последовательностей входных данных разных длин, является механизм управляемых рекуррентных блоков GRU. Идейно похожий на LSTM, механизм ворот обновления позволяет при последовательной обработке входных данных

«запоминать» нужную информацию, а ворота сброса «забывать» ранее сохраненную, но уже не релевантную информацию.

Задача определения погодных изменений является очень важной в области сельского хозяйства. Классические рекуррентные нейронные сети склонны быстро забывать информацию при обработке длинных последовательностей входных данных, а также менее точно улавливают долгосрочные зависимости. Внедрение моделей с GRU в проекте предсказания погоды на основе списка параметров, таких как температура, влажность воздуха и свойства почвы, позволило повысить точность предсказаний с 86% до 94% по сравнению с простой рекуррентной нейросетью, обученной на том же наборе данных [12].

Несмотря на то, что механизмы GRU и LSTM реализуют похожую идею внимания и механизмов памяти, они показывают более хорошие результаты при выполнении определенных типов задач. Согласно результатам исследования, сравнивающего показатели моделей с механизмами GRU и LSTM в задачах классификации на основе временных рядов, при обучении на одном наборе данных уровень средней ошибки моделей на этапе тестирования был близок, однако среднее время обучения моделей с GRU меньше, как и разброс ошибок [13].

Результаты исследования показывают, что в задачах классификации без необходимости обработки очень длинных векторов входных данных модели с механизмом GRU требуют меньше времени на обучение и показывают более хорошие результаты, чем LSTM. Для задач с длинными последовательностями данных модели с LSTM могут показать более высокую точность классификации, но потребуют большего объема обучающей выборки для получения результата.

Так как проект направлен на проверку гипотезы и исследование нового алгоритма классификации сцен на изображениях, механизм GRU является более подходящим для выполнения задачи – последовательности примитивных объектов не будут длинными, а меньшие затраты времени на обучение позволят быстро вносить изменения при разработке модели.

2.1.4 Определение метрик для оценки качества разработанного алгоритма

При решении задачи детекции объектов важна как точность определения положения и размера объекта, так и корректность назначенного объекту класса.

Для определения качества локализации объекта будет использована метрика **IoU** (Intersection over Union) – отношение площади пересечения предсказанной моделью области и действительной области объекта к площади объединения областей. Чем ближе значение метрики к единице, тем более точно модель определяет размер и положение объекта на изображении. Данная метрика также будет одним из настраиваемых параметров при обучении модели, позволяющей установить границу, при которой модель будет отбрасывать предсказания, в которых она недостаточно уверена.

Качество предсказания классов примитивных объектов будет определяться метрикой точности, усредненной по классам – **mAP** (mean Average Precision). При нормализации метрики по количеству объектов в классе можно снизить влияние дисбаланса классов на переобучение модели.

Метрика **F1-score** позволяет учитывать точность (Precision) и полноту (Recall) предсказаний модели. Точность показывает, сколько значений предсказанного класса действительно относились к этому классу. Полнота – сколько значений класса модель определила правильно из общего количества объектов данного класса.

Качество модели классификации и итоговое качество всего разработанного алгоритма также будет оцениваться с помощью метрик F1-score и mAP. Средняя точность по классам классификатора, выбирающего случайные значения, будет равно 1/n, где n – количество классов. Таким образом, алгоритм, точность предсказаний которого больше данного параметра, может считаться перспективным и подтверждающим поставленную гипотезу.

2.2 Подготовка данных для обучения и тестирования моделей

Для обучения и тестирования моделей нейросетей необходимо подготовить наборы данных, содержащие изображения с размеченными примитивными объектами и классами сцен.

2.2.1 Определение списков классов для задач детекции и классификации

Для обучения моделей детекции был сформирован **список базовых классов примитивов**: сфера, куб, цилиндр, конус, торус.

Основанием для выбора данных примитивных фигур послужила теория «Распознавания по компонентам» И. Бидермана [14]. Ее идея заключается в том, что объекты повседневной жизни могут быть упрощены до набора двух-трех геонов (объемных примитивных фигур) и быть все еще узнаваемы человеком по расположению и характеристикам примитивных составляющих. В целях тестирования разрабатываемого алгоритма выбран наименьший набор классов примитивов, максимально отличных друг от друга.

При формировании списка классов локаций для тестирования алгоритма учитывалась потенциальная релевантность локаций для роботизированных устройств и непохожесть классов друг на друга. В целях тестирования концепта классификации локаций, не специфичных для определенных задач или областей робототехники, выбирались более обобщенные классы, например жилое помещение, а не спальня, кухня и ванная комната.

Для обучения и тестирования модели классификации был сформирован следующий **список классов локаций**: теплица, офис, склад, жилое помещение.

2.2.2 Обзор методов создания искусственных изображений

Количество и качество данных для обучения моделей нейросетей напрямую влияет на их точность и качество выполнения задач. При недостатке данных модель даже с самой передовой и сложной архитектурой может показать не самые лучшие результаты. Одним из способов решения данной проблемы является генерация искусственных данных.

Были рассмотрены две идеи генерации синтетических изображений:

- Генерация двухмерного объекта на двухмерный фон.
- Помещение трехмерного объекта в сцену с двухмерным фоном.

Первый способ заключается в использовании генеративных нейронных сетей для создания изображения объекта интереса, который далее помещается на фоновое изображение с размытием границ [15].

Несмотря на хорошие результаты при генерации данных для прикладных задач, серьезной проблемой данного метода является зависимость качества получаемых данных от качества обучения генеративной нейросети. В задачах с сильной проблемой недостатка данных может быть применен второй метод генерации, заключающийся в использовании трехмерных моделей объектов интереса для помещения на двухмерное изображение с подходящим фоном. Применение данного метода для расширения обучающей выборки в задаче детекции строительной техники на изображениях позволило повысить точность обученных моделей на 2% [16].

В настоящее время в открытом доступе отсутствуют готовые наборы данных с размеченными примитивными объектами. Вместе с ручной разметкой данных для формирования обучающих датасетов было решено использовать второй метод генерации искусственных изображений, так как качественные трехмерные модели примитивных объектов создать не сложно.

2.2.3 Программная генерация искусственных изображений

Для генерации искусственных изображений с примитивными объектами выбранным методом была использована среда моделирования Blender, позволяющая использовать скрипты на Python для автоматического создания рандомизированных сцен с объектами.

Создание сцен и преобразование их в изображение проводилось с использованием библиотеки BlenderProc, предназначенной для автоматической генерации и аннотации фотореалистичных данных в среде Blender [17].

С использованием библиотеки был создан скрипт на Python, позволяющий создавать набор примитивных трехмерных моделей со случайными текстурами и положением, определять положение камеры, рендерить двухмерное изображение и сохранять его в формат hdf5, полный скрипт представлен в ПРИЛОЖЕНИИ Б.

Перед запуском генерации сцен в скрипте определяются параметры количества кадров для одной сцены и количество генерируемых сцен.

При создании наполнения трехмерных сцен случайным образом настраиваются следующие параметры:

– Количество объектов примитивов на сцене (от 6 до 12).

– Размер, положение и поворот примитивных объектов.
– Текстуры объектов.
– Угол и интенсивность освещения сцены.
– Положение камеры для создания двухмерного изображения.
– Точка фокуса камеры для повышения реалистичности изображений.

Библиотека «BlenderProc» поддерживает автоматическую аннотацию сгенерированных данных с помощью горизонтальных ограничивающих рамок, но не поддерживает ориентированные. На основе метода библиотеки для создания карт сегментации был разработан алгоритм получения ориентированных ограничивающих рамок для сцены, содержащий следующие шаги:

– Назначение идентификатора примитивному объекту при его создании.
– Генерация карты сегментации сцены на основе идентификаторов объектов (см. рисунок 6).
– Объединение пикселей каждого отдельного объекта карты сегментации в структуру «shapely.MultyPoint».
– Определение ориентированной ограничивающей рамки каждого объекта с помощью метода «shapely.oriented_envelope».
– Сохранение класса примитива и координат рамки в текстовый файл для каждого объекта.

Рисунок 6 - Карта сегментации сгенерированной сцены

Готовые сцены примитивных объектов помещаются на случайный фон из подготовленного списка, состоящего из 12 изображений без идентифицируемых примитивов. Фоновые изображения модифицируются для повышения разнообразия создаваемых данных – выбирается случайный фрагмент изображения, к которому применяется размытие со случайной мощностью.

В результате выполнения скрипта создаются двухмерные изображения с наборами примитивных объектов и текстовые файлы аннотаций для каждого изображения (см. рисунок 7).

Рисунок 7 - Пример размеченного синтетического изображения с примитивными объектами

На созданных изображениях наблюдается наложение фигур друг на друга и выход фигур за границы кадра, что не является проблемой. На изображениях реального мира примитивные составляющие объектов будут формировать сложные объекты похожим образом – обучение моделей детекции на подобных данных способно улучшить качество их распознавания.

2.2.4 Результаты формирования наборов данных

В результате создания синтетических изображений с примитивными объектами было получено 305 изображений. Во время генерации примитивов создание объекта каждого из классов было равновероятным, в результате разметки данных сильный дисбаланс классов не обнаружен (см. таблицу 1).

Таблица 1 - Распределение классов примитивов на синтетических данных

Класс примитива	Количество	Доля, %
Куб	525	20.7
Конус	476	18.8
Цилиндр	506	19.9
Сфера	512	20.2
Торус	516	20.4
Всего	**2535**	**100**

Для обучения модели классификации было вручную размечено 104 фотографии реального мира для каждого из четырех выбранных классов локаций. Среди размеченных примитивных объектов на изображениях офисов, складов и жилых помещений преобладали кубы и цилиндры, а на изображениях теплиц – сферы, что изменило баланс классов. В результате разметки реальных изображений были получено новое распределение классов примитивов (см. таблицу 2).

Таблица 2 - Распределение классов примитивов на синтетических и реальных данных

Класс примитива	Количество	Доля, %
Куб	2571	41.8
Конус	481	7.8
Цилиндр	1186	19.4
Сфера	1385	22.5
Торус	522	8.5
Всего	**6145**	**100**

После разметки изображений для задачи классификации сцены был получен следующий набор классов – 21 офис, 23 склада, 31 жилое помещение и 29 теплиц. Пример разметки примитивов на реальных изображениях всех классов сцен представлен на рисунке 8.

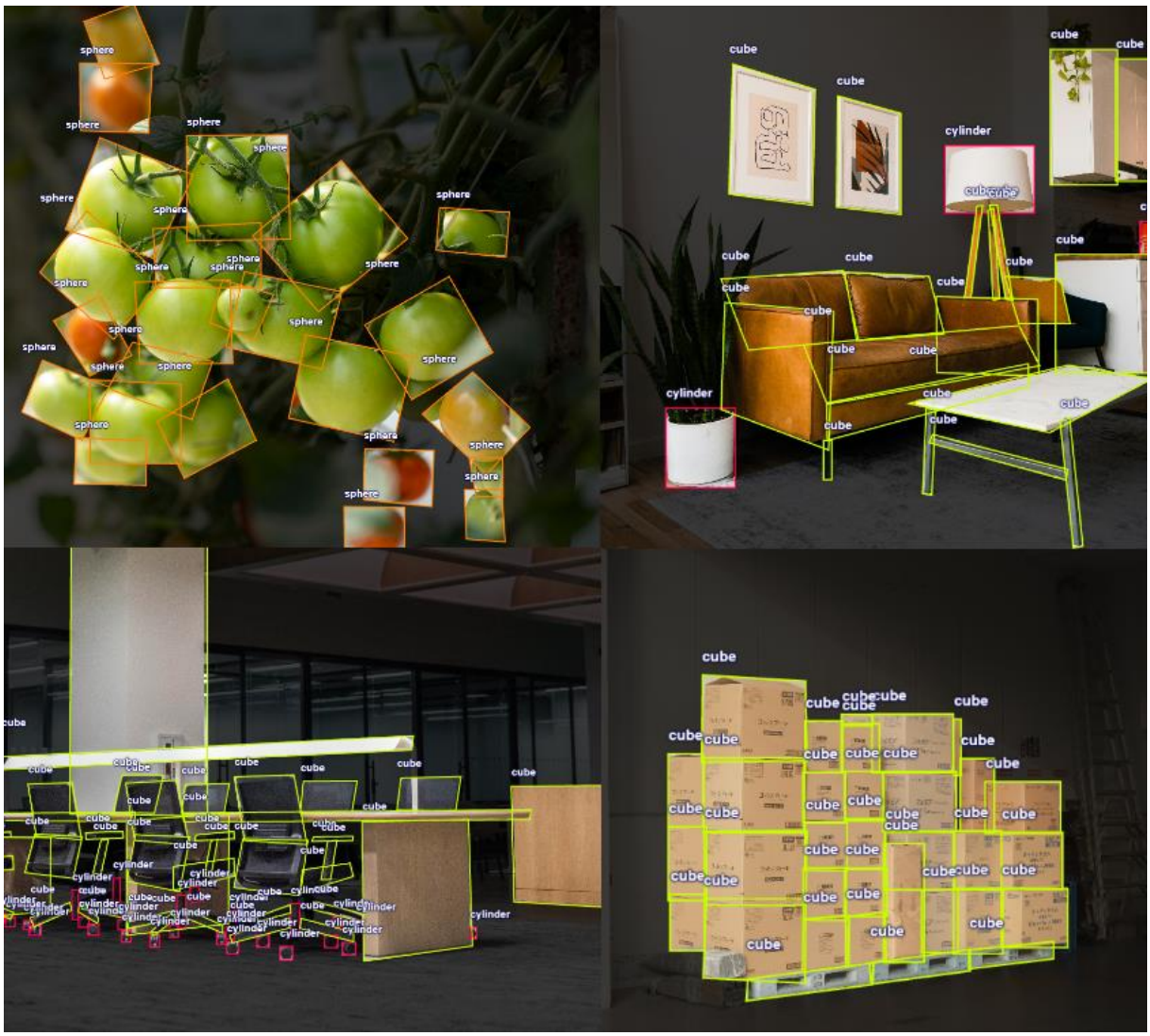

Рисунок 8 - Пример разметки примитивов на изображениях классов сцен

Собранные изображения реального мира распространяются под лицензией, позволяющей свободное личное и коммерческое использование изображений без необходимости указывать авторство или запрашивать разрешение на использование. В описании изображений и в пользовательском соглашении сайта не был указан запрет на использование изображений в качестве данных для обучения нейронных сетей, однако во время разметки некоторых изображений было заметно добавление сильного шума после скачивания качественного изображения. Предположительно, данный шум направлен на «отравление» обучаемых моделей компьютерного зрения и ухудшение их качества. Зашумленные изображения были удалены из набора данных.

Для расширения обучающей выборки и улучшения качества распознавания модели детекции к тренировочным изображениям будут применены методы аугментации:

– случайная обрезка фрагмента;
– случайные повороты;
– изменение насыщенности изображения;
– коллаж нескольких фрагментов изображения со смещением.

2.3 Реализация и обучение моделей нейросетей

После принятия решения о создании ансамбля моделей детекции примитивных объектов с ориентированными ограничивающими рамками и модели классификации сцен с механизмом внимания GRU, была составлена схема взаимодействия моделей, представленная на рисунке 9.

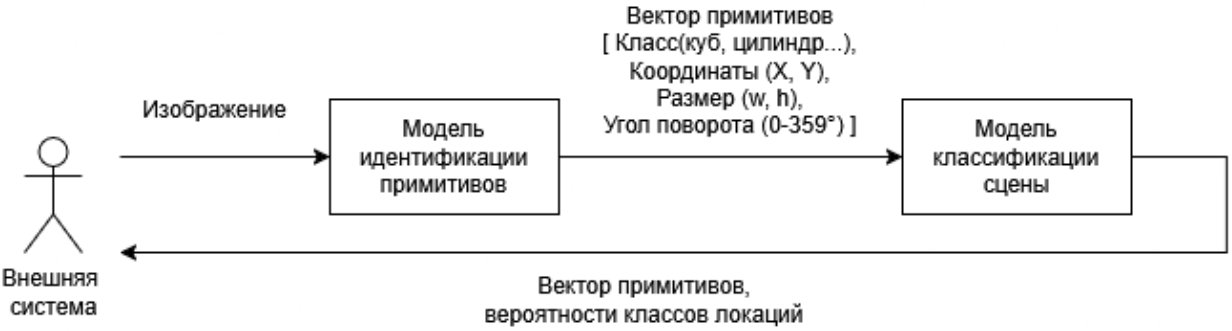

Рисунок 9 - Схема взаимодействия моделей детекции и классификации системы

При обработке изображения моделью детекции примитивов формируется вектор, содержащий информацию о классе, положении и угле поворота объектов на изображении, далее передаваемый модели классификации сцены. В результате работы алгоритма возвращается вектор примитивных объектов и вероятности классов сцен для обработанного изображения.

Наборы размеченных данных для обучения моделей нейросетей разделяются на тренировочные, валидационные и тестовые данные в пропорциях 7:2:1.

Обучение и тестирование моделей проводится на тестовом стенде со следующими характеристиками:

– процессор Intel Xeon CPU E5-2666 v3;
– видеокарта NVIDIA GeForce RTX 3060 с поддержкой CUDA;
– оперативная память 32ГБ.

2.3.1 Обучение моделей детекции примитивов

Для выбора модели детекции было проведено обучение ряда предобученных моделей YOLO на собранном датасете примитивов. Сравнение качества и точности моделей проводилось по параметрам, определенным ранее. Для корректного сравнения результатов перед обучением каждой модели устанавливается одинаковый сид генераторов случайных чисел для параметров обучения и окружения.

Первая итерация обучения моделей проводилась только на синтетических данных для проверки возможностей моделей. В результате обучения моделей разных версий и размеров получены результаты, представленные в таблице 3.

Таблица 3 – Результаты обучения моделей YOLO на синтетических данных

Модель	**Количество параметров, млн**	**Количество эпох обучения**	**Метрика F1-score**	**Метрика PR-curve**
YOLOv8n-obb	3.1	300	0.8	0.845
YOLOv8m-obb	26.4	300	0.83	0.889
YOLOv11n-obb	2.7	300	0.8	0.851
YOLOv11m-obb	20.9	300	0.85	0.898
YOLOv11l-obb	26.2	300	**0.86**	0.897
YOLOv11x-obb	**58.8**	300	**0.86**	**0.905**

При увеличении размера моделей заметно увеличение времени обучения модели и улучшение качества распознавания примитивов при одинаковом наборе данных и количестве эпох обучения. В среднем, модели YOLO версии 11 показывают результаты лучше, чем модели версии 8 схожего размера.

Наилучший результат в текущих условиях показала модель YOLOv11x-obb, графики ее метрик PR-curve и F1-score для отдельных классов примитивов и усредненные по классам представлены на рисунках 10 и 11.

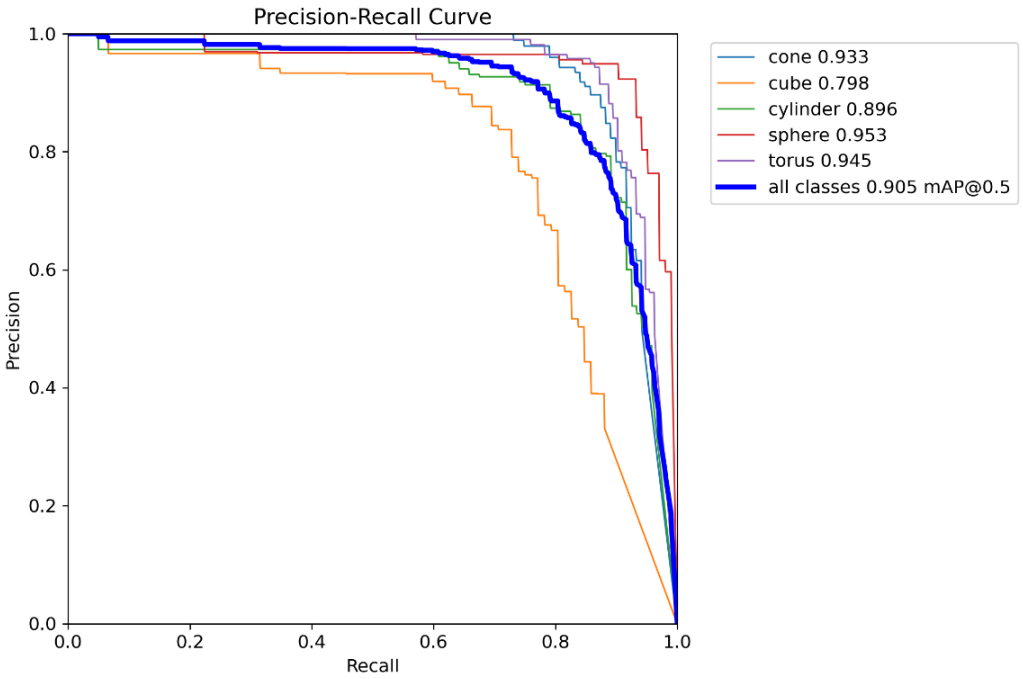

Рисунок 10 - График метрики PR-curve модели YOLOv11x-obb

График метрики PR-curve на рисунке 10 показывает соотношение точности (Precision) и полноты (Recall) предсказаний модели. Кривая для усредненной по всем классам точности, «all classes» в легенде графика, показывает наилучшее значение метрики при указанной границе уверенности IoU. Модель на графике достигает метрики mAP = 0.905 при границе IoU = 0.5.

Площадь под кривой показывает качество определения классов найденных объектов. Чем ближе кривая модели к правому верхнему углу, тем идеальнее она классифицирует.

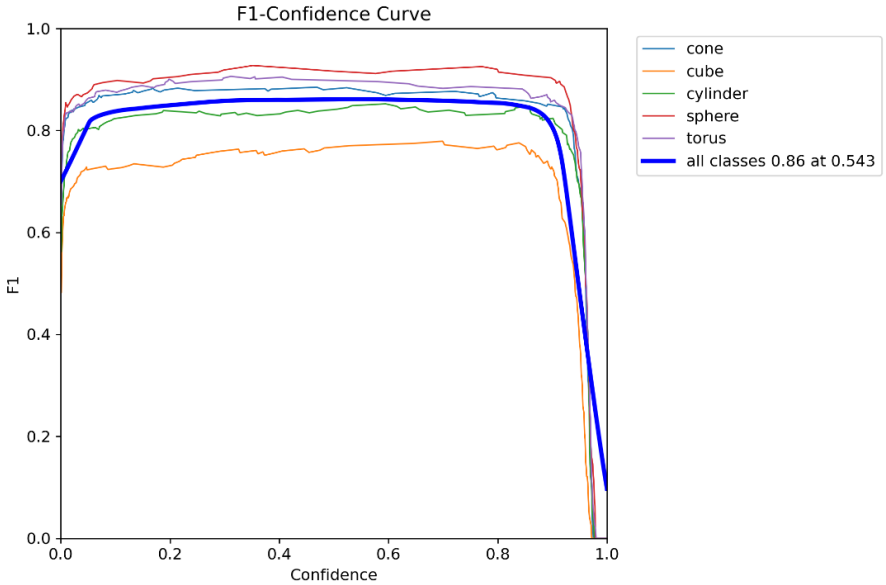

Рисунок 11 - График метрики F1-score модели YOLOv11x-obb

График F1-Confidence curve на рисунке 11 показывает значение метрики F1-score при разных порогах уверенности модели. Усредненная по классам кривая, «all classes» в легенде графика, показывает наилучшее значение метрики и границу уверенности, при которой оно достигается. Данная модель показывает наилучший результат 0.86 при границе уверенности в 0.543.

После добавления реальных фотографий с размеченными примитивами была проведена вторая итерация обучения наиболее перспективных моделей маленького и большого размера (см. таблицу 4). В столбце «количество эпох обучения» указана сумма эпох первой и второй итерации.

Таблица 4 - Результаты обучения моделей YOLO на синтетических и комбинированных данных

Модель	**Количество параметров, млн**	**Количество эпох обучения**	**Метрика F1-score**	**Метрика PR-curve**
YOLOv11n-obb	2.7	300+200	0.66	0.680
YOLOv11m-obb	20.9	300+200	**0.73**	0.710
		300+500	0.7	0.724
YOLOv11x-obb	58.8	300+200	**0.73**	**0.753**
		300+500	0.72	0.750

Добавление реальных изображений в выборку для обучения и тестирования снизило качество распознавания моделей. Несмотря на то, что модели были обучены хорошо распознавать синтетические фигуры, объекты реального мира могут сильно отличаться от них, что вызывает трудности у моделей. Графики новых метрик модели YOLOv11x-obb, обученной на 300 и 200 эпохах, представлены на рисунках 12-13.

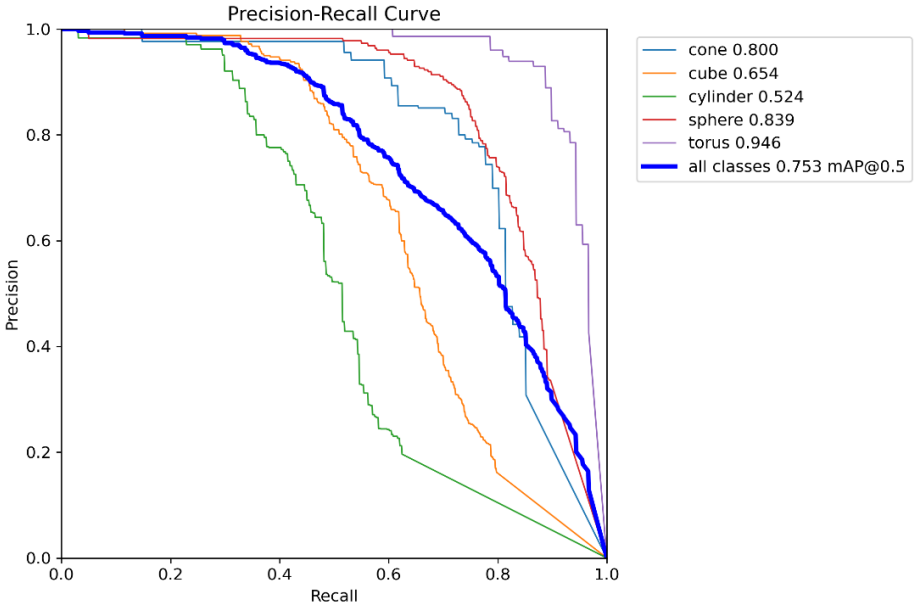

Рисунок 12 - График метрики PR-curve модели YOLOv11x-obb после добавления реальных данных

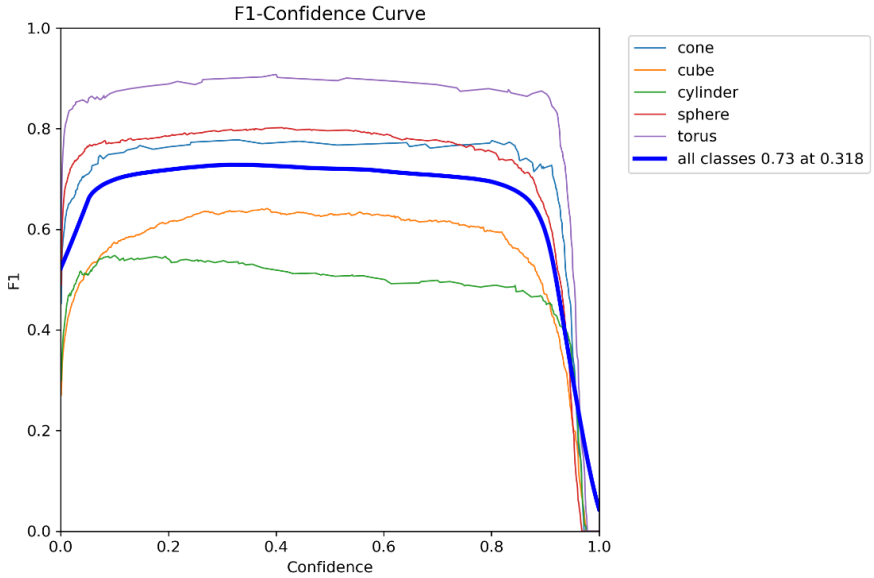

Рисунок 13 - График метрики F1-score модели YOLOv11x-obb после добавления реальных данных

Заметно сильное снижение метрик точности и полноты (Precision-Recall) для объектов классов «цилиндр» и «куб». Причиной этого может быть преобладание данных объектов классов на реальных изображениях, ошибка в детекции которых приводит к более сильному снижению метрик.

Во время обучения параметры допустимой площади пересечения предсказания с реальностью и уверенности модели в предсказании были установлены по умолчанию – IoU = 0.7, conf = null. В результате визуализации предсказания на одном из тестовых изображений модель выделяет объекты, в

которых она уверена на 28-46% (см. рисунок 14), что может снижать точность последующей классификации изображений. Для дальнейшего обучения и использования моделей детекции будет установлен порог минимальной уверенности для модели conf = 0.6.

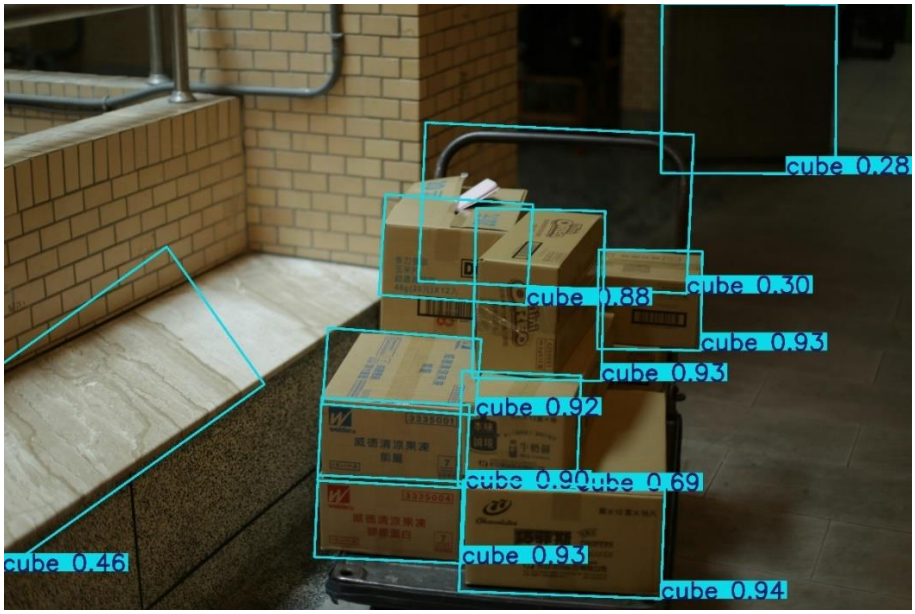

Рисунок 14 - Результат предсказания модели детекции на реальном изображении с низким порогом уверенности

По результатам второй итерации обучения моделей детекции было решено провести эксперимент – дополнительно обучить и протестировать модели только на реальных размеченных изображениях. При сравнении качества моделей, обучавшихся с добавлением синтетическими данными и полностью без них, на реальных изображениях можно оценить выбранный метод генерации изображений примитивов. Результаты третей итерации обучения представлены в таблице 5.

Таблица 5 – Результаты обучения моделей YOLO на реальных данных

Модель	Модель предобучена на синтетических данных	Количество эпох обучения	Метрика F1-score (conf=0.6)	Метрика PR-curve
YOLOv11n-obb	-	500	0.25	0.320
	+	300+200	0.26	0.297
YOLOv11m-obb	-	500	0.27	0.357
	+	300+200	0.30	0.315
	+	300+500	0.30	0.313
YOLOv11x-obb	-	500	**0.39**	0.359
	+	300+200	0.27	**0.483**
	+	300+500	0.28	0.480

Сильное снижение метрик качества всех моделей может быть следствием переобучения. Модели, обученные только на реальных данных, показывают чуть более высокое значение F1-score, но значение PR ниже, чем у моделей, знакомых с синтетическими данными. Модели размеров «n» (nano) и «m» (medium) с увеличением количества эпох обучения показывают увеличение F1-score и уменьшение PR, но метрики модели размера «x» (extra large) показывают противоположные тренды. Вероятно, большее количество параметров модели позволяет ей замечать больше сложных зависимостей в изображениях и показывать более высокую точность и полноту при детекции примитивов.

Более длительное обучение моделей на небольшом количестве размеченных изображений приводит к их переобучению на тренировочных данных и к ухудшению качества на тестовых данных.

При ручной разметке данных ограничительные рамки могли быть указаны недостаточно точно для некоторых объектов, что при обучении влияет на качество локализации модели даже при ее корректном распознавании классов моделей. Вычисленные метрики моделей могут быть несколько ниже действительных.

Для выбора итоговой модели детекции веса лучших моделей были сохранены и протестированы на наборе реальных размеченных фотографий каждого класса для определения итоговых метрик качества моделей. Выбранные данные не использовались при обучении моделей, что исключает искажение результатов

тестирования. Для итогового тестирования выбраны модели, показавшие лучшие результаты на итерациях обучения с разными наборами данных (см. таблицу 6).

Таблица 6 - Результаты тестирования моделей YOLO на реальных данных

Модель	**Данные и эпохи обучения**	**Метрика F1-score (conf=0.6)**	**Метрика PR-curve**	**Время работы, мс**
YOLOv11n-obb	Синтетические (300) Комбинированные (200)	0.67	0.776	61.1
YOLOv11m-obb	**Синтетические (300)** **Комбинированные (200)**	**0.85**	**0.876**	**61.4**
YOLOv11x-obb	Реальные (500)	0.25	0.320	73
	Комбинированные (300) Реальные (200)	0.26	0.319	66
	Синтетические (300) Комбинированные (200)	0.81	0.854	62.5

При сравнении моделей одного размера, можно заметить более высокое качество детекции у моделей, тренировочные выборки которых содержали синтетические данные. Модели, обучавшиеся только на небольшом наборе реальных данных, показывают худшие результаты при тестировании.

На рисунке 15 представлена информация о количестве объектов примитивов на изображениях и метрики точности (P), полноты (R), средней точности при пороге уверенности 50% (mAP50) и средней точности при уверенности модели от 50% до 95% (mAP50-95) для каждого класса.

yolov11n-obb
Синтетические+
Комбинированные
данные

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	11	477	0.966	0.582	0.776	0.657
cone	3	5	1	0.2	0.6	0.48
cube	8	208	0.976	0.596	0.791	0.686
cylinder	5	192	0.885	0.24	0.571	0.427
sphere	6	71	0.969	0.873	0.924	0.797
torus	1	1	1	1	0.995	0.895

yolov11m-obb
Синтетические+
Комбинированные
данные

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	11	477	0.99	0.762	0.876	0.799
cone	3	5	1	0.6	0.8	0.645
cube	8	208	0.98	0.721	0.857	0.818
cylinder	5	192	0.969	0.49	0.733	0.587
sphere	6	71	1	1	0.995	0.949
torus	1	1	1	1	0.995	0.995

yolov11x-obb
Реальные
данные

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	11	477	0.392	0.23	0.32	0.245
cone	3	5	0	0	0	0
cube	8	208	0.508	0.303	0.393	0.287
cylinder	5	192	0.667	0.0312	0.348	0.226
sphere	6	71	0.784	0.817	0.859	0.714
torus	1	1	0	0	0	0

yolov11x-obb
Комбинированные+
Реальные
данные

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	11	477	0.413	0.219	0.319	0.254
cone	3	5	0	0	0	0
cube	8	208	0.579	0.298	0.446	0.343
cylinder	5	192	0.583	0.0365	0.309	0.247
sphere	6	71	0.9	0.761	0.84	0.678
torus	1	1	0	0	0	0

yolov11x-obb
Синтетические+
Комбинированные
данные

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	11	477	0.986	0.719	0.854	0.772
cone	3	5	1	0.4	0.7	0.64
cube	8	208	0.962	0.726	0.857	0.812
cylinder	5	192	0.968	0.469	0.722	0.564
sphere	6	71	1	1	0.995	0.95
torus	1	1	1	1	0.995	0.895

Рисунок 15 - Сравнение метрик моделей по классам примитивов

Модели, обучавшиеся на реальных данных с сильным недостатком размеченных конусов и торусов не смогли корректно предсказать ни одного объекта данных классов. Равные пропорции объектов классов в синтетических данных позволили моделям различать примитивы в реальных данных с довольно неплохой точностью – 60-80%. Точность распознавания остальных классов у моделей, обучавшихся на сгенерированных изображениях примитивов также в среднем выше. Выбранный метод генерации синтетических данных доказал свою эффективность в расширении обучающей выборки для моделей детекции примитивных объектов.

Отличие времени обработки одного изображения среди моделей разных размеров незначительно – повышенная точность модели большего размера оправдывает увеличенное время ее расчетов. Для реализации алгоритма будет

использована модель YOLOv11m-obb, обученная на 300 эпохах синтетических данных и 200 эпохах комбинированных данных.

На рисунке 16 и 17 представлены графики метрик F1-score и PR-curve выбранной модели детекции. Граница минимальной уверенности модели в предсказании, установленная на 60%, приводит к отсутствию показателей метрики F1-score на первом графике при уверенностях ниже установленного порога – значение метрики на пороге растягивается до левой границы графика. Значение метрики F1-score модели достигает 85%.

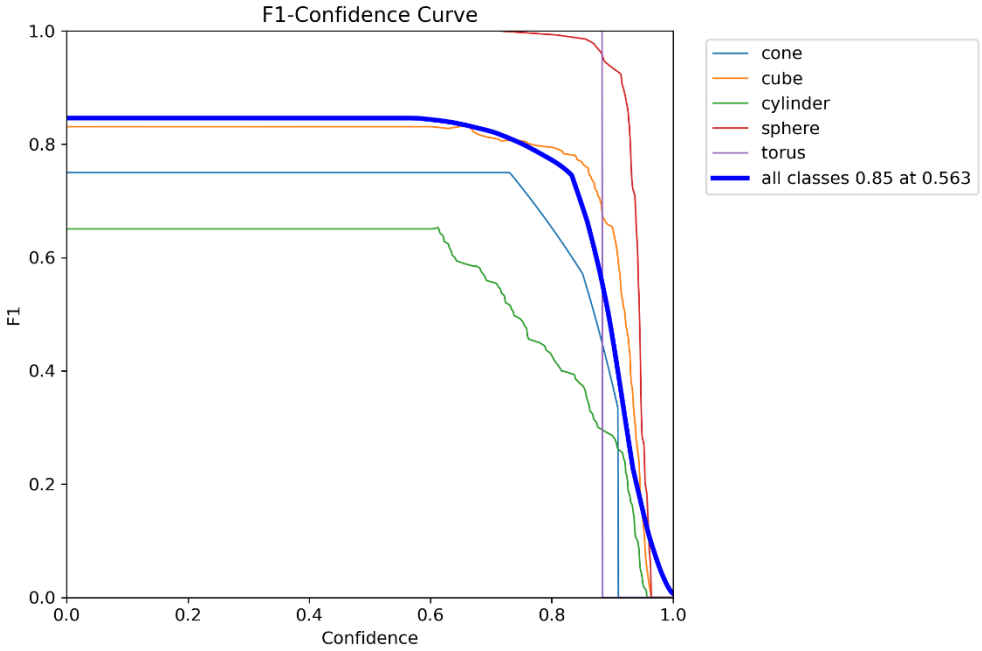

Рисунок 16 - График метрики F1-Confidence curve модели YOLOv11m-obb

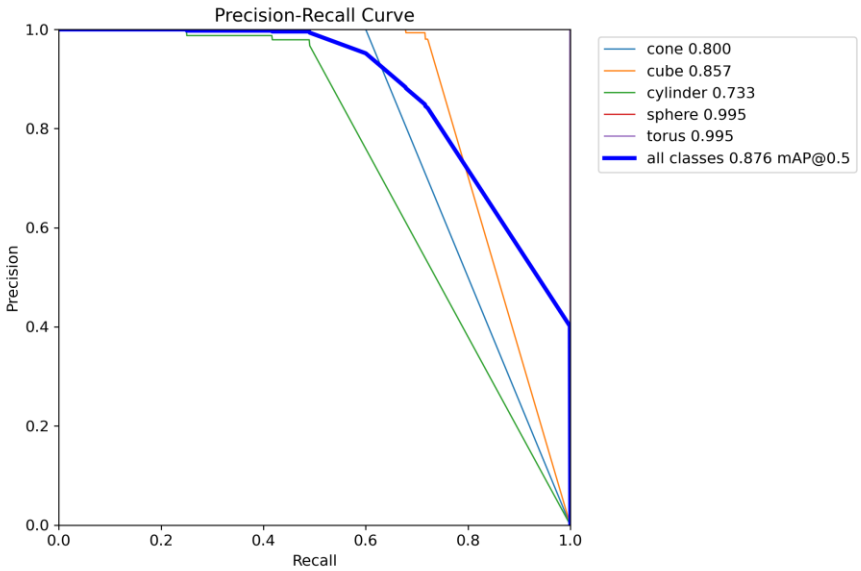

Рисунок 17 - График метрики PR-curve curve модели YOLOv11m-obb

На тестовых данных модель показывает среднюю точность предсказания по всем классам в 0.876 при границе уверенности 50%. Модель почти идеально различает примитивы классов «сфера» и «торус» со значением точности в 99.5%, чуть лучше остальных классов.

Рисунок 18 представляет нормализованную матрицу ошибок. Диагональ матрицы показывает долю верно предсказанных объектов на тестовых данных, а остальные клетки матрицы – количество ложных срабатываний. Можно заметить, что модель достаточно хорошо научилась распознавать кубы, сферы и торусы, чуть хуже конусы и в половине случаев – цилиндры. Согласно правому столбцу матрицы – модель не смогла распознать 40% конусов, 28% кубов и половину цилиндров. Однако среди зон тестовых данных, отмеченных как «фон» или отсутствие объекта, модель предположила наличие кубов или цилиндров в равных пропорциях.

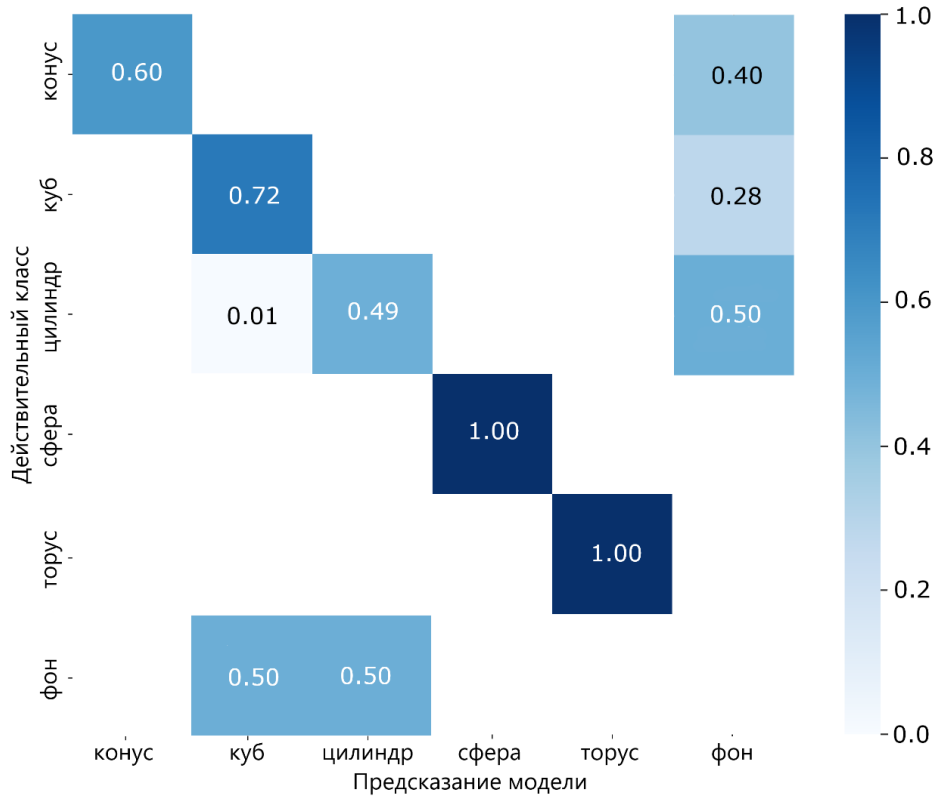

Рисунок 18 - Нормализованная матрица ошибок модели YOLOv11m-obb

Пример предсказаний выбранной модели на тестовых данных можно увидеть на рисунке 19, где присутствуют все классы сцен и примитивных объектов.

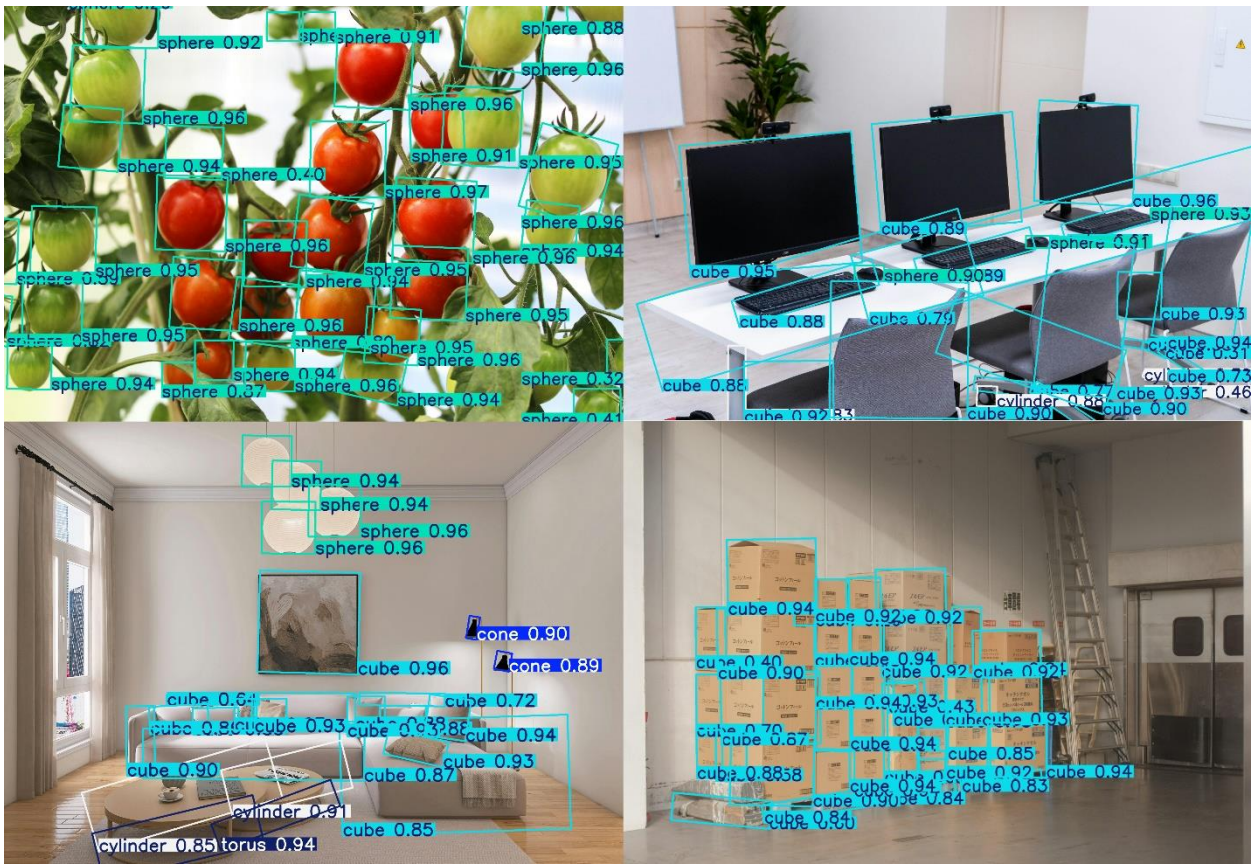

Рисунок 19 - Пример детекции примитивных объектов моделью YOLOv11m-obb на изображениях всех классов сцен

2.3.2 Подготовка данных для модели классификации

Для работы с данными о примитивных объектах на изображении использована библиотека Pandas, позволяющая эффективно обрабатывать большие наборы данных. Размеченные примитивные объекты на изображениях необходимо привести к формату последовательности, который способна обработать модель классификации с механизмом GRU.

Выбранная модель детекции YOLOv11 может предоставлять информацию о примитивных объектах на изображении в нескольких форматах:

– «хухухуху», класс и координаты углов ограничительных рамок объектов.

– «xywrh», класс, центр, ширина и высота ограничительной рамки объекта.

Диаграмма вариантов представления ограничивающих рамок моделями YOLO представлена на рисунке 20 [18].

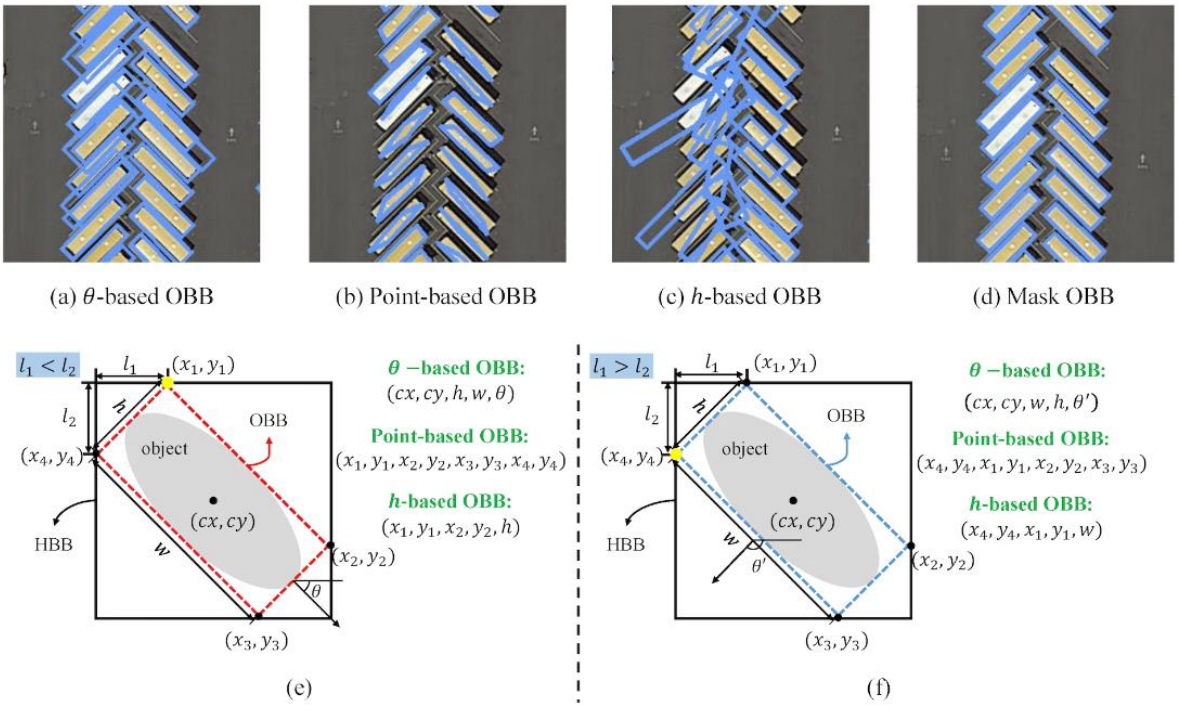

***Рисунок 20 - Варианты представления ограничивающих рамок моделями YOLO** [18]*

Данные во втором формате могут быть легче восприняты человеком, но для модели нейронной сети более явные признаки объектов (размеры, положение, поворот) могут как положительно повлиять на качество классификации, так и не оказать эффекта. Формат описания примитивов не сильно влияет на общую архитектуру модели классификации – остается неизменным факт векторного представления входных данных с динамической длиной последовательности между разными изображениями.

Для проверки зависимости обучаемой модели от формата входных данных требуется изменить только первый слой архитектуры модели, поэтому оба метода будут протестированы.

Данные примитивов необходимо нормализировать для корректного обучения и эксплуатации модели, иначе различия размеров обрабатываемых изображений будут приводить к совершенно разным результатам предсказаний. Формат данных «xyxyxyxy» имеет вариацию с нормализацией координат относительно осей X и Y изображения – «xyxyxyxyn». Однако для формата «xywhr» не предусмотрена нормализованная версия из-за неоднозначности способа нормализации ширины и высоты наклоненной ограничивающей рамки объекта.

На рисунке 21 представлены два рассмотренных варианта нормализации наклонных линий на прямоугольном изображении – (а) проекция наклонной линии

на оси изображения и нормализация ее как гипотенузы по формуле Пифагора, (б) нормализация наклонной линии относительно отрезка, параллельного ей и выходящего из угла изображения.

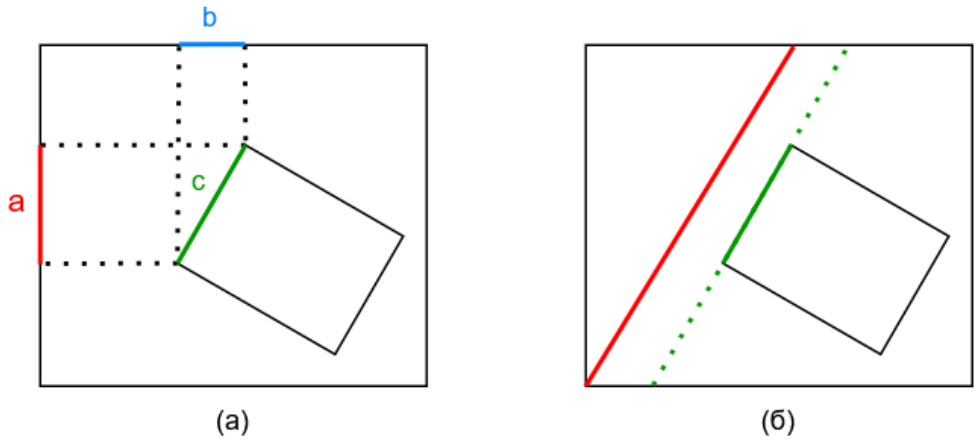

Рисунок 21 - Варианты нормализации наклонных осей ограничивающих рамок объектов

Нормализованные значения ширины или длины рамки, вычисленные с помощью первого способа, можно будет корректно сравнивать между собой, в отличие от значений, полученных вторым вариантом. Несмотря на использование параллельных прямых, исходящих из угла изображения для получения самого длинного возможного отрезка – длины отрезков будут отличаться в зависимости от угла наклона рамки относительно осей изображения. Таким образом, будет использован первый вариант нормализации для предобработки значений рамок примитивов.

Для обучения моделей классификации ранее собранный и размеченный набор реальных изображений был преобразован в две последовательности данных. Первая последовательность хранит список классов изображений, вторая – список объектов класса «pandas.Dataframe», хранящих информацию о примитивах и их признаках, содержащихся в соответствующем по порядку изображении. На рисунках 22-23 представлены примеры данных одного изображения в форматах «xywhr» и «xyxyxyxyn» соответственно.

	primitive_class_0	primitive_class_1	primitive_class_2	primitive_class_3	x	y	width	height	rotation
0	0	1	0	0	0.531226	0.623692	0.085081	0.255103	0.010068
1	0	1	0	0	0.809977	0.843863	0.311893	0.379792	1.570796
2	0	1	0	0	0.304045	0.720149	0.248525	0.457252	1.098199
3	0	1	0	0	0.427103	0.691146	0.083601	0.117527	0.052016
4	0	0	0	1	0.210096	0.775455	0.156811	0.098213	1.561035
5	0	0	0	1	0.288356	0.835432	0.012969	0.019205	0.020211
6	0	0	0	1	0.322662	0.861432	0.020000	0.014866	1.570796
7	0	0	0	1	0.374348	0.822829	0.012925	0.018971	0.041661
8	0	0	0	1	0.369355	0.783718	0.014286	0.010673	1.570796
9	0	0	1	0	0.333335	0.770003	0.063948	0.012007	1.508536
10	0	0	1	0	0.306462	0.813718	0.016504	0.053572	1.103749
11	0	0	1	0	0.326474	0.826861	0.010524	0.052845	0.208799
12	0	0	1	0	0.357340	0.806164	0.041941	0.015124	0.277491
13	0	1	0	0	0.339504	0.680355	0.136331	0.096323	1.423241
14	0	1	0	0	0.121022	0.929433	0.060739	0.008592	1.511132

Рисунок 22 – Пример датафрейма примитивов одного изображения для задачи классификации в формате «xywhr»

	primitive_class_0	primitive_class_1	primitive_class_2	primitive_class_3	x1	y1	x2	y2	x3	y3	x4	y4
0	0	1	0	0	0.575038	0.497643	0.489972	0.495718	0.487414	0.749740	0.572480	0.751665
1	0	1	0	0	0.999873	0.687916	0.620081	0.687916	0.620081	0.999810	0.999873	0.999810
2	0	1	0	0	0.507925	0.755477	0.451047	0.505437	0.100164	0.684821	0.157043	0.934861
3	0	1	0	0	0.471622	0.640024	0.388414	0.630288	0.382584	0.742267	0.465792	0.752003
4	0	0	0	1	0.259953	0.852182	0.258434	0.696575	0.160238	0.698729	0.161757	0.854337
5	0	0	0	1	0.295027	0.826289	0.282067	0.825701	0.281686	0.844575	0.294645	0.845164
6	0	0	0	1	0.330095	0.851432	0.315229	0.851432	0.315229	0.871432	0.330095	0.871432
7	0	0	0	1	0.381172	0.814289	0.368286	0.813082	0.367524	0.831368	0.380409	0.832575
8	0	0	0	1	0.374692	0.776575	0.364019	0.776575	0.364019	0.790861	0.374692	0.790861
9	0	0	1	0	0.341258	0.800618	0.337337	0.737718	0.325411	0.739389	0.329332	0.802289
10	0	0	1	0	0.330390	0.810894	0.326664	0.794289	0.282534	0.816542	0.286260	0.833147
11	0	0	1	0	0.336575	0.804575	0.326803	0.799922	0.316372	0.849147	0.326145	0.853800
12	0	0	1	0	0.377360	0.804004	0.339243	0.793146	0.337319	0.808324	0.375436	0.819181
13	0	1	0	0	0.394663	0.723869	0.377121	0.605850	0.284346	0.636841	0.301888	0.754861
14	0	1	0	0	0.127084	0.958858	0.123500	0.898861	0.114961	0.900007	0.118544	0.960004

Рисунок 23 - Пример датафрейма примитивов одного изображения для задачи классификации в формате «xyxyxyxyn»

Категориальный признак «класс примитива» был закодирован в четыре числовых признака «primitive_class_{n}», где n – номер класса от 0 до 3. Столбцы содержат значение «1», если объект является данным классом или «0» в обратном случае. Для предотвращения проблемы корреляции данных, пятый столбец для последнего класса примитива был удален.

Разделение данных на тренировочные, валидационные и тестовые проводится с помощью разработанного метода «split_train_val_test», его код приведен далее:

```
def split_train_val_test(dfs_dict, train_prop=0.7, val_prop=0.2,
test_prop=0.1, seed=42):

    random.seed(seed)
    train = []
    val = []
    test = []

    # add all classes to train/val/test in proportions
    for key, value in dfs_dict.items():
        print(key)
        classes_list = [scene_classes_inv[key]] * len(value)
        data = list(zip(classes_list, value))
        random.shuffle(data)

        t_threshold = int(train_prop * len(value))
        v_threshold = int((train_prop + val_prop) * len(value))
        train += data[0:t_threshold]
        val += data[t_threshold:v_threshold]
        test += data[v_threshold:]
        print([x[0] for x in data[0:t_threshold]], [x[0] for x in
data[t_threshold:v_threshold]],
              [x[0] for x in data[v_threshold:]])

    random.shuffle(train)
    random.shuffle(val)
    random.shuffle(test)

    tr_targ, tr_feat = zip(*train)
    v_targ, v_feat = zip(*val)
    te_targ, te_feat = zip(*test)
    return tr_targ, tr_feat, v_targ, v_feat, te_targ, te_feat
```

Передаваемые последовательности классов изображений и соответствующих им наборов примитивов разделяются на отдельные списки, при этом все данные одного класса сцены распределяются в каждый набор данных согласно указанным пропорциям, что позволяет обеспечить баланс классов и нахождение экземпляров каждого класса во всех выборках. Данные полученных наборов перемешиваются в случайном порядке для предотвращения получения моделью последовательно большого количества данных одного класса и быстрого переобучения модели на него.

При обучении и тестировании модель для предсказания должна получать только признаки изображений (набор примитивов в случае задачи), а список действительных классов изображений используется для вычисления функции

потерь или метрик точности модели. Для удобства работы с данными создан класс «CustomDataset», реализующий интерфейс последовательности:

```
class CustomDataset:
    def __init__(self, features, labels):
        self.features = [*features]
        self.labels = [*labels]
        self.len = len(features)

    def __getitem__(self, index):
        items =
torch.tensor(self.features[index].astype(float).values,
dtype=torch.float32)
        label = torch.tensor(self.labels[index], dtype=torch.float32)

        # Sort bboxes to go from up left to bottom down with x/y bbox
centers
        sorted_indices = items[:, 4].sort()[1]
        new_items = items[sorted_indices]
        sorted_indices = new_items[:, 5].sort()[1]
        new_items = new_items[sorted_indices]

        return new_items, label

    def __len__(self):
        return self.len
```

Последовательности классов изображений и их примитивных объектов оборачиваются в объект класса для дальнейшего поэлементного получения при необходимости. Переданные примитивы сортируются по четвертому и пятому столбцу признаков для их расположения по порядку следования от левого верхнего угла изображения до правого нижнего. Для аннотаций в формате «xywhr» сортировка проводится по координатам центра, а для аннотаций в формате «xuxuxuxup» по координатам левого верхнего угла ограничивающей рамки примитива.

2.3.3 Реализация модели классификации сцен

Рекуррентные нейронные сети, изначально разработанные для обработки двухмерных изображений, также могут анализировать временные ряды или последовательности данных. Механизм управляемых рекуррентных блоков GRU позволяет еще более успешно извлекать зависимости длинных последовательностей. Модель нейросети для классификации изображений будет разрабатывается с использованием библиотеки PyTorch, предоставляющей готовые архитектурные слои для создания нейронных сетей.

Для создания модели классификации был определен класс «GRUClassifier», наследующий базовый класс «torch.nn.Module», который реализует механизмы обучения, тестирования и эксплуатации модели. Код класса приведен далее:

```
class GRUClassifier(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers,
num_classes, bidirectional=False, dropout=0):
        super().__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.h_layers_multiplier = 1
        if bidirectional:
            self.h_layers_multiplier = 2

        # layers
        self.gru = nn.GRU(input_size=input_size,
                          hidden_size=hidden_size,
                          num_layers=num_layers,
                          dropout=dropout,
                          bidirectional=bidirectional)
        self.fc = nn.Linear(hidden_size * self.h_layers_multiplier,
num_classes)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, x: PackedSequence, *args):
        # Set initial hidden and cell states - GRU
        h0 = torch.zeros(self.num_layers * self.h_layers_multiplier,
                         x.batch_sizes.max(),
                         self.hidden_size).to(device)

        # Forward propagate RNN
        out, _ = self.gru(x, h0)
        out, sizes = pad_packed_sequence(out)
        out = torch.stack([out[size - 1][i] for i, size in
enumerate(sizes)])
        # Decode the hidden state of the last time step
        out = self.fc(out)

        out = self.softmax(out)
        return out
```

Архитектура модели создана с помощью определения следующих слоев в методе инициализации класса с помощью модуля «torch.nn»:

– «GRU», слой с механизмом управляемых рекуррентных блоков для предсказания класса изображения;
– «Linear», полносвязный слой для преобразования множества выходов GRU к четырем выходам для каждого класса задачи;
– «LogSoftmax», логарифмическая функция активации для преобразования выходов слоев нейросети к значению от 0 до 1.

Для выбора итоговой архитектуры модели классификации необходимо обучить и сравнить несколько моделей с разными параметрами слоя GRU, настройка архитектуры модели происходит с помощью передаваемых при инициализации класса параметров:

- «input_size», размер входного слоя, соответствует количеству параметров объекта ограничивающей рамки примитива.
- «hidden_size», количество параметров скрытого состояния GRU.
- «num_layers», количество рекуррентных слоев GRU.
- «dropout», вероятность выключения нейрона выходного слоя GRU при обучении модели.
- «bidirectional», отвечает за вариант слоя GRU – односторонний или двухсторонний проход по последовательности входных данных.

Метод «forward» реализует процесс вычисления предсказания модели, передавая входные данные через обозначенные слои. Во время обучения модели вызов метода сопровождается вычислением функции потерь и обновлением весов модели с помощью оптимизатора. В качестве оптимизатора обучения модели выбран «torch.optim.Adam», реализующий метод адаптивной оценки момента.

Модель классификации с механизмом GRU может обрабатывать последовательности объектов любой длины, например изображение с 10 или 40 примитивами. Информация о каждом объекте обработки проходит через слои GRU и изменяет веса скрытого состояния памяти для дальнейшего выполнения предсказания класса. Для возможности последовательной обработки данных нескольких изображений, модель «GRUClassifier» ожидает получить на вход объект класса «pack_sequence» модуля «torch.nn.utils.rnn» - данная структура позволяет объединить несколько тензоров примитивов разной длины в один объект, заполняя недостающие элементы пустыми значениями.

При длительном обучении на тренировочном наборе данных модель может переобучиться, и при тестировании на новых данных показать точность сильно ниже, чем ожидалось. Для контроля переобучения моделей создан класс

«EarlyStopper», отвечающий за остановку процесса обучения модели при отсутствии уменьшения функции потерь на этапе валидации:

```
class EarlyStopper:
    def __init__(self, patience=1, min_delta=0):
        self.patience = patience
        self.min_delta = min_delta
        self.counter = 0
        self.min_validation_loss = float('inf')

    def early_stop(self, validation_loss):
        if validation_loss < self.min_validation_loss:
            self.min_validation_loss = validation_loss
            self.counter = 0
        elif validation_loss > (self.min_validation_loss +
self.min_delta):
            self.counter += 1
            if self.counter >= self.patience:
                return True
        return False
```

На каждом этапе обучения объект класса «EarlyStopper» получает новое значение функции потерь, если на протяжении указанного в параметре «patience» количества этапов значение не становилось меньше, чем минимальное зафиксированное значение с допустимым отклонением «min_delta», подается сигнал об остановке обучения.

Для выбора итоговой модели классификации было проведено обучение шести моделей с различными наборами параметрами слоя GRU. Результаты обучения на 150 эпохах с форматом входных данных «xywhr», включающим координаты центра, размеры ограничивающей рамки примитива и угол ее поворота представлены в таблице 7.

Таблица 7 - Результаты обучения моделей классификации на нормализованных данных формата «xywhr»

Модель	model_0	model_1	model_2	model_3	model_4	model_5
Количество слоев GRU	10	20	20	10	20	20
Параметры скрытого состояния GRU	32	32	32	32	32	32
Двусторонняя модель GRU	-	-	+	-	-	+
Вероятности Dropout слоя	0	0	0	0.2	0.2	0.2
Мин. значение функции потерь на обучении	**0.419**	0.895	0.519	0.438	0.927	0.508
Макс. точность на обучении	0.808	0.597	0.761	**0.831**	0.544	0.775
Мин. значение функции потерь на валидации	0.891	1.108	0.915	**0.885**	1.116	0.944
Макс. точность на валидации	**0.650**	0.400	**0.650**	**0.650**	0.450	**0.650**
Оптимальная эпоха обучения	109	47	88	85	43	82

В таблице отражены настройки параметров архитектуры моделей, минимальные зафиксированные значения функции потерь на тренировочных и валидационных данных, максимальные значения точности и эпоха обучения, на которой была достигнута максимальная точность предсказаний при валидации.

В среднем, при вероятности отключения выходных нейронов слоя GRU на этапе обучения отличной от нуля, достижение более высокой точности на валидационных данных происходит за меньшее количество эпох обучения.

На рисунке 24 представлены графики функций потерь (ошибок) моделей на протяжении эпох обучений, точками отмечены моменты достижения минимального значения функции потерь.

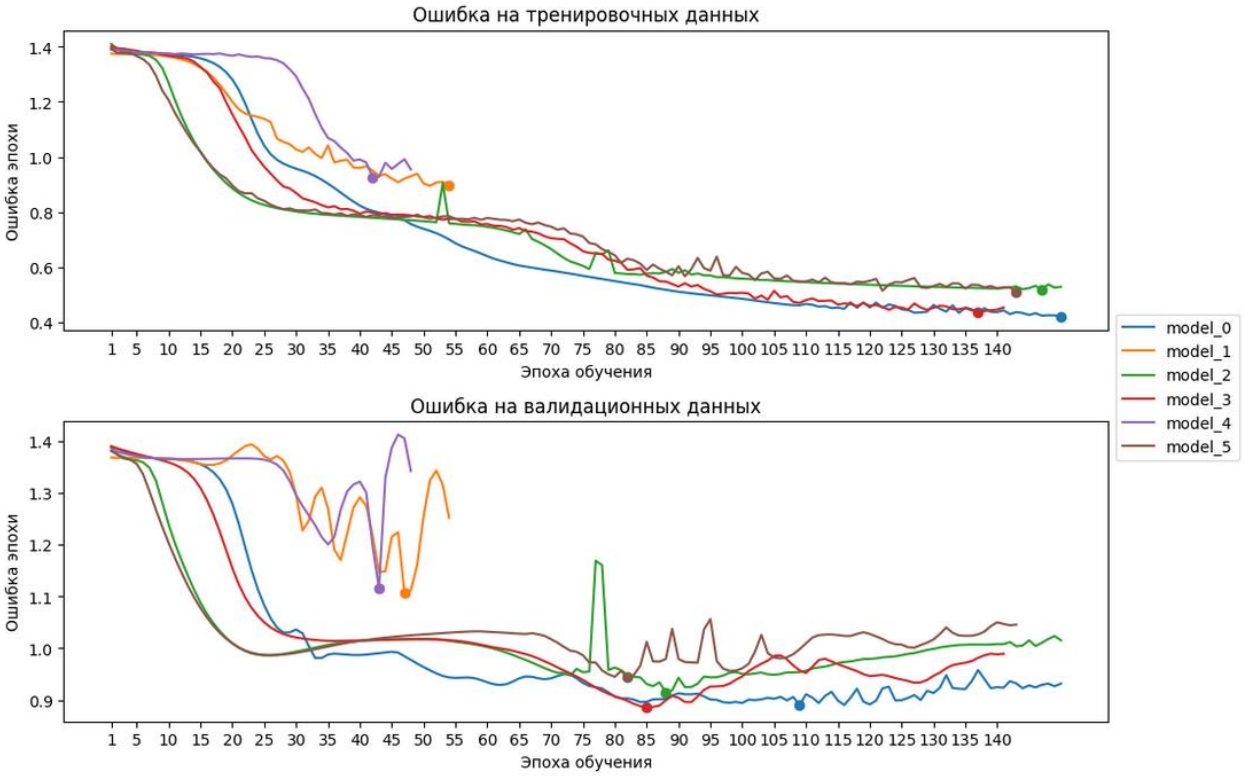

Рисунок 24 - Графики значений функций потерь моделей классификации при формате данных «xywhr»

Можно заметить, что все тестируемые модели показывают способность к обучению и снижению значений функции потерь с течением эпох. Все модели при длительном обучении на небольшом объеме данных склонны переобучаться – после снижения ошибки на обучении и валидации до некоторого момента, ошибка на тренировочных данных продолжает падать, а на данных валидации начинает расти.

Линии функций потерь моделей «model_1» и «model_4» на тестовых и валидационных данных имеют самые медленные темпы снижения, обучение моделей останавливается на 50-55 эпохе из-за резкого повышения ошибки на этапе валидации.

При достижении точности предсказаний на тестовых данных в 60-80%, точность моделей при валидации не превышает 65%. Для определения влияния формата входных данных на качество получаемых моделей, на данных в формате «хухухухуп» был обучен набор моделей с аналогичными наборами параметров и данных (см. таблицу 8).

Таблица 8 - Результаты обучения моделей классификации на нормализованных данных формата «хухухухуп»

Модель	model_0	model_1	model_2	model_3	model_4	model_5
Количество слоев GRU	10	20	20	10	20	20
Параметры скрытого состояния GRU	32	32	32	32	32	32
Двусторонняя модель GRU	-	-	+	-	-	+
Вероятности Dropout слоя	0	0	0	0.2	0.2	0.2
Мин. значение функции потерь на обучении	0.476	0.972	0.510	**0.461**	0.934	0.533
Макс. точность на обучении	0.775	0.499	0.775	**0.789**	0.513	0.749
Мин. значение функции потерь на валидации	**0.712**	1.242	0.979	0.947	1.184	0.948
Макс. точность на валидации	**0.700**	0.450	0.650	0.650	0.400	0.650
Оптимальная эпоха обучения	144	34	81	70	53	107

Информация о положении и наклонах объектов в формате координат углов ограничивающих рамок примитивов оказалась более полезной для модели – точность классификации сцен на валидационных данных модели «model_0» достигла 70% несмотря на то, что после 145 эпох обучения ее качество также начало снижаться.

На рисунке 25 представлен график функций потерь обученных моделей. Общие тренды обучения моделей одной архитектуры схожи при различных форматах данных – обучение моделей «model_1» и «model_4» заканчивается раньше, чем остальных, а ошибка всех моделей на валидации начинает расти к концу обучения.

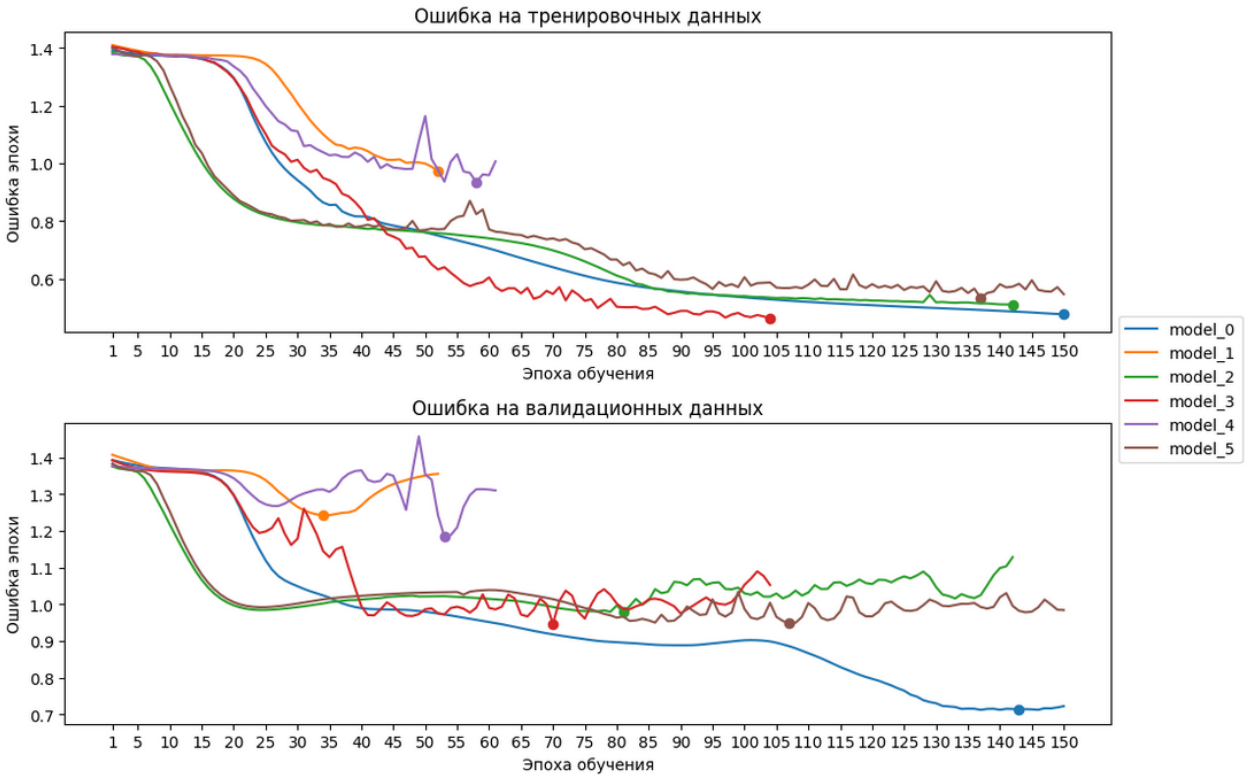

Рисунок 25 - Графики значений функций потерь моделей классификации при формате данных «хухухухуп»

Несмотря на то, что модель «model_3» показывает более быстрое снижение ошибки на тренировочных данных, ее итоговое качество оказывается не самым лучшим. Модель «model_0» отличается более плавным снижением значения ошибок без резких скачков и достигает значения функции потерь на валидационных данных в 0.712.

Среди всех двенадцати обученных моделей «model_0» с данными формата «хухухухуп» показала самое высокое качество классификации, на рисунке 26 представлен отчет о качестве модели на тестовых данных.

	precision	recall	f1-score	support
office	1.00	0.25	0.40	4
livingroom	0.50	1.00	0.67	6
warehouse	1.00	0.50	0.67	4
greenhouse	1.00	0.83	0.91	6
accuracy			0.70	20
macro avg	0.88	0.65	0.66	20
weighted avg	0.85	0.70	0.69	20

Рисунок 26 - Результаты тестирования модели model_0

В результате обучения модель лучше всего научилась распознавать сцены теплиц - значение метрики F1-score достигает 0.91 при точности 1. Сцены жилых

комнат и складов модель корректно распознает в двух из трех случаев, а сцены офисов меньше чем в половине.

Нормализованная матрица ошибок на рисунке 27 позволяет понять, какие классы сцен модель склонна путать между собой. В обучающем наборе данных объектов каждого из классов сцен были примерно одинаковое количество, но модель обучилась распознавать сцены теплиц лучше всего с точностью = 1 и полнотой = 0.83. При этом распознавание сцен офиса и склада для модели дается сложнее всего, правильный класс был назначен только 25% изображений офиса и 50% изображений склада. Наибольшей части изображений модель ошибочно назначает класс жилой комнаты, также как и всем действительным изображениям данного класса, полнота = 1.

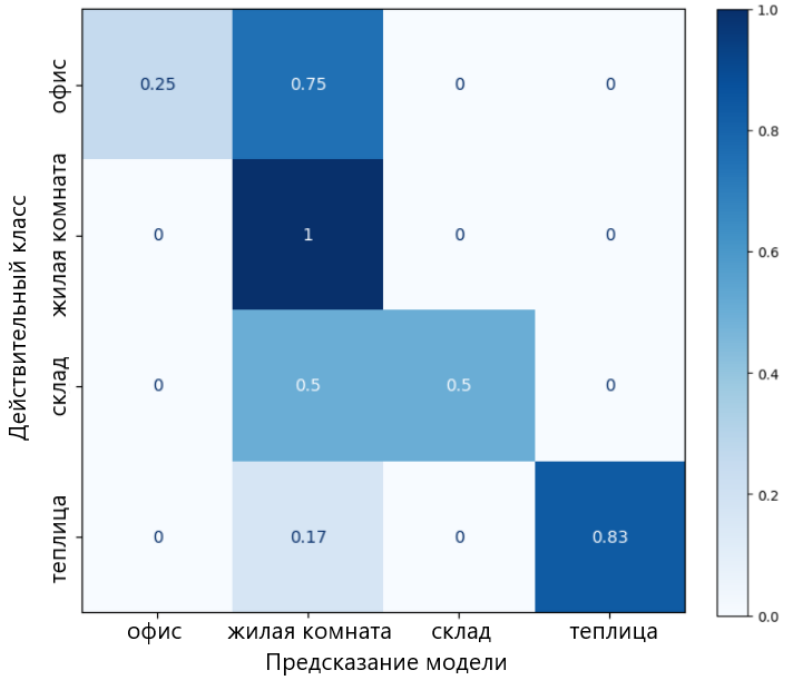

Рисунок 27 - Нормализованная матрица ошибок модели классификации «model_0»

По результатам обучения модель классификации «model_0» показала наилучший результат с метрикой F1-score = 0.7 и была выбрана для реализации алгоритма классификации сцен.

2.3.4 Ансамблирование моделей бинарной классификации

При анализе результатов предсказаний разработанной модели в задаче классификации с четырьмя классами сцен можно заметить, что распознавание некоторых классов получается у модели хуже, чем других. При увеличении

количества классов без изменения архитектуры модели будет происходить ухудшение ее качества, так как для более сложных задач требуется более сложная структура модели.

Одним из подходов, способным улучшить качество моделей классификации в подобных задачах, является ансамблирование моделей, решающих задачи бинарной классификации – «one vs all» [19]. Каждая модель обучена определять принадлежность изображения к одному единственному классу, например – «Офис» или «Не офис». Преимуществом данного подхода, по сравнению с единственной моделью классификации, является возможность добавления новых классов изображений в задачу без необходимости переобучения всех ранее обученных моделей.

Было решено провести эксперимент и обучить для выбранных классов сцен четыре модели классификации с одинаковой архитектурой. Ранее размеченные изображения были сформированы в наборы данных для каждой модели и размечены на два класса – изображение принадлежит к определенному классу сцены или нет.

Модели бинарной классификации были обучены аналогично мультиклассовым моделям, графики их функций потерь во время обучения представлены на рисунке 28. В таблице 9 описаны итоги обучения моделей.

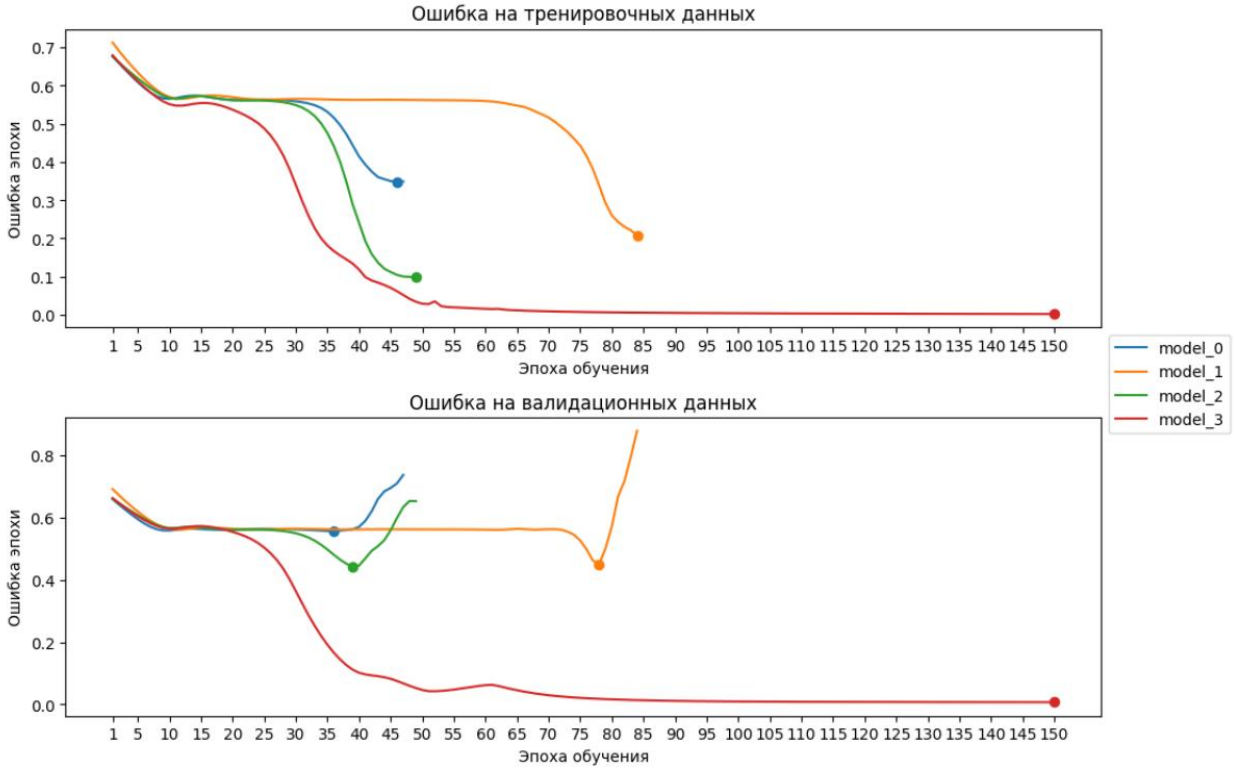

Рисунок 28 – Графики значений функций потерь моделей бинарной классификации

Таблица 9 - Результаты обучения бинарных моделей классификации

Модель	model_0	model_1	model_2	model_3
Класс сцены	Офис	Жилая комната	Склад	Теплица
Количество слоев GRU	10			
Параметры скрытого состояния GRU	32			
Двусторонняя модель GRU	-			
Вероятности Dropout слоя	0			
Мин. значение функции потерь на обучении	0.348	0.207	0.098	0.002
Макс. точность на обучении	0.879	0.821	0.750	0.982
Мин. значение функции потерь на валидации	0.557	0.450	0.442	0.007
Макс. точность на валидации	**0.750**	**0.750**	**0.750**	**0.938**
Оптимальная эпоха обучения	36	79	39	150

Можно заметить, что модели бинарной классификации в среднем заканчивают обучение раньше и имеют более высокую точность на этапе валидации, чем мультиклассовые модели различных архитектур. Точность распознавания одного определенного класса моделей равна 75% для Офисов, Жилых комнат и Складов и 93.8% для Теплиц.

Веса обученных моделей с наивысшим показателем точности на валидации были сохранены для дальнейшего тестирования и ансамблирования. Тестовый набор данных состоит из восьми изображений, по два каждого из классов сцен, результаты тестирования моделей представлены на рисунке 29.

Модель распознавания Офисов показывает точность в 75%, модели Жилых комнат и Складов 87.5%, а модель Теплиц идеально отделяет изображения своего класса от других с точностью и полнотой равными единице.

```
Evaluating network on 8 images in test set -- Test loss: 0.073 -- Test accuracy: 0.750,
              precision    recall  f1-score   support

      office       0.50      0.50      0.50         2
  not_office       0.83      0.83      0.83         6

    accuracy                           0.75         8
   macro avg       0.67      0.67      0.67         8
weighted avg       0.75      0.75      0.75         8

Evaluating network on 8 images in test set -- Test loss: 0.025 -- Test accuracy: 0.875,
                precision    recall  f1-score   support

    livingroom       0.67      1.00      0.80         2
not_livingroom       1.00      0.83      0.91         6

      accuracy                           0.88         8
     macro avg       0.83      0.92      0.85         8
  weighted avg       0.92      0.88      0.88         8

Evaluating network on 8 images in test set -- Test loss: 0.074 -- Test accuracy: 0.875,
               precision    recall  f1-score   support

    warehouse       1.00      0.50      0.67         2
not_warehouse       0.86      1.00      0.92         6

     accuracy                           0.88         8
    macro avg       0.93      0.75      0.79         8
 weighted avg       0.89      0.88      0.86         8

Evaluating network on 8 images in test set -- Test loss: 0.000 -- Test accuracy: 1.000,
                precision    recall  f1-score   support

    greenhouse       1.00      1.00      1.00         2
not_greenhouse       1.00      1.00      1.00         6

      accuracy                           1.00         8
     macro avg       1.00      1.00      1.00         8
  weighted avg       1.00      1.00      1.00         8
```

Рисунок 29 - Отчеты классификации бинарных моделей на тестовых данных

На нормализованных матрицах ошибок (см. рисунок 30) можно сравнить метрики полноты моделей. Модель с классами Теплиц идеально выделяет изображения своего класса, в то время как остальные модели лучше определяют какие изображения не принадлежат к их классу.

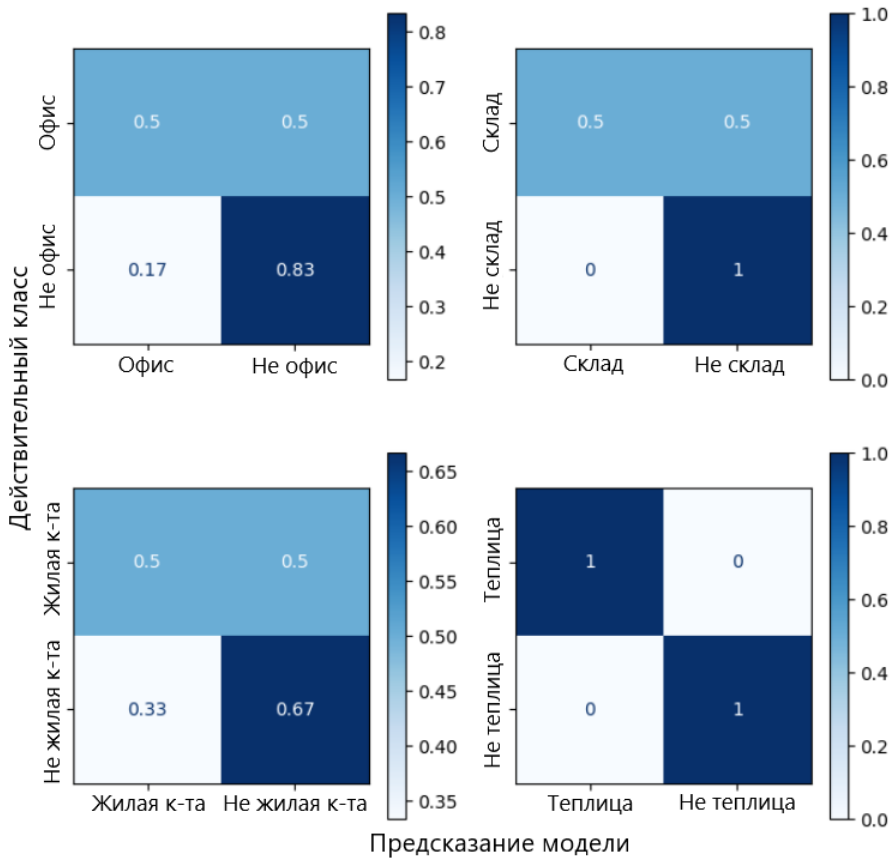

Рисунок 30 - Нормализованные матрицы ошибок бинарных моделей классификации

Принцип работы ансамбля моделей бинарной классификации заключается в обработке изображения каждой моделью для получения уверенностей в принадлежности изображения к их классам. Далее на основе уверенностей всех моделей определяется итоговый класс изображения, наиболее популярным вариантом определения является простой выбор класса «наиболее уверенной модели» без назначения голосам моделей весов.

Модели бинарной классификации были объединены в ансамбль, при обработке изображения уверенности каждой модели в принадлежности к ее классу формируют список, далее обрабатываемый функцией активации «nnf.Softmax» для получения вероятностей класса сцены изображения. Результаты тестирования ансамбля моделей представлены на рисунках 31-32.

	precision	recall	f1-score	support
office	0.14	0.25	0.18	4
livingroom	0.62	0.83	0.71	6
warehouse	1.00	0.25	0.40	4
greenhouse	0.50	0.33	0.40	6
accuracy			0.45	20
macro avg	0.57	0.42	0.42	20
weighted avg	0.57	0.45	0.45	20

Рисунок 31 - Отчет классификации ансамбля бинарных моделей

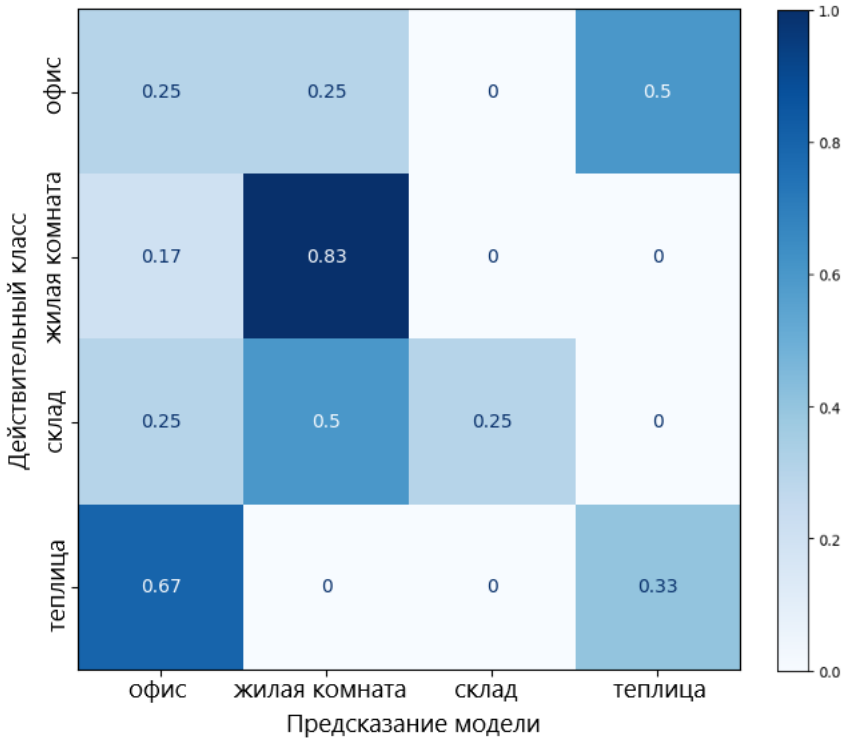

Рисунок 32 - Нормализованная матрица ошибок ансамбля бинарных моделей

Несмотря на высокую точность отдельных моделей, точность ансамбля достигла всего 45%, что все равно выше точности случайного классификатора с метрикой в 25%. По сравнению с результатами мультиклассовой модели на рисунке 31, данная модель имеет больший разброс предсказаний по классам и чаще путает их. Наибольшая точность предсказаний получена для Жилых комнат, а с хорошей моделью распознавания Теплиц ансамбль смог определить только 33% Теплиц корректно.

2.3.5 Объединение системы моделей нейронных сетей

В результате обучения моделей детекции YOLO была выбрана модель YOLOv11m-obb, показавшая среднюю точность предсказания примитивов 87.6%.

Разработанная модель классификации с механизмом GRU достигла точности предсказаний на подготовленных данных в 70%, ансамбль бинарных моделей классификации показал точность распознавания в 45%. Для реализации алгоритма модели инициализируются с помощью предварительно сохраненных весов моделей.

Объединение моделей детекции примитивов и классификации сцен для алгоритма требует преобразования выходных данных модели детекции в формат «xyxyxyxyn» и формирование последовательности примитивов в объект класса «torch.nn.utils.rnn.pack_sequence» для выполнения задачи классификации моделью с механизмом GRU. Для выполнения необходимых преобразований будут использованы ранее разработанные функции и классы.

Был проведен тестовый запуск алгоритма с моделью классификации для предсказания примитивов и класса сцены на изображении реального мира из тестового набора данных. Изображение формата PNG передано на вход модели детекции, полученный список ограничивающих рамок примитивов приведен к необходимому формату и передан модели классификации.

Предсказания модели классификации приведены в формат вероятностей классов сцен, выбран наиболее вероятный класс (см. рисунок 33). В результате работы алгоритма на изображении были идентифицированы примитивные объекты и корректно определен класс жилой комнаты с уверенностью модели в 73% (см. рисунок 34). Более подробное тестирование метода с разными моделями классификации представлено в Главе 4.

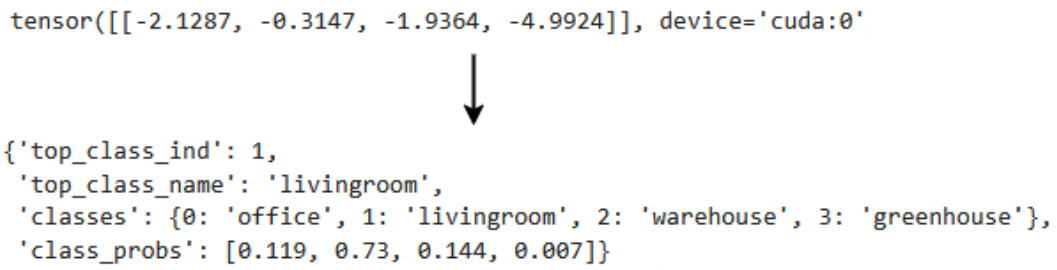

Рисунок 33 - Форматирование предсказаний алгоритма к вероятностям классов сцен

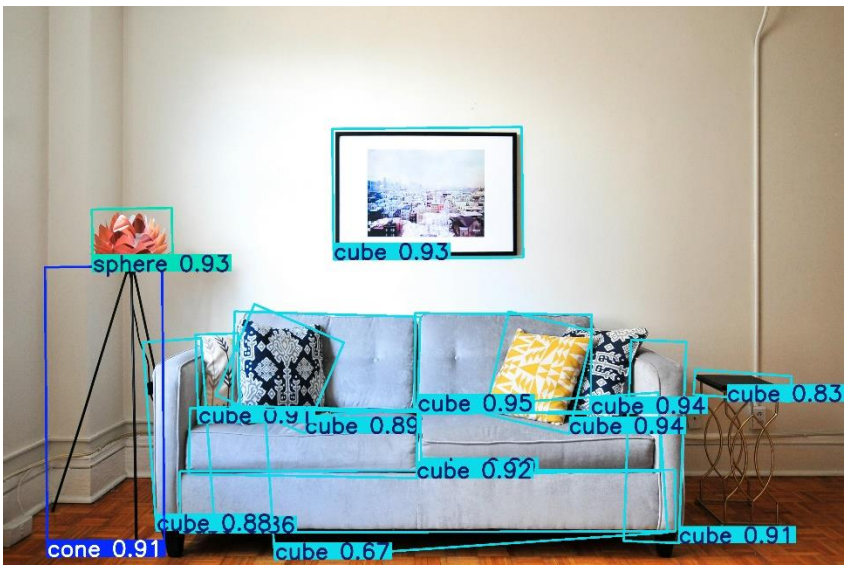

Рисунок 34 - Результаты идентификации примитивов на изображении с помощью разработанного алгоритма

Выводы по главе 2

На основе методов, выбранных на этапе анализа, спроектирован ансамбль моделей нейросетей для реализации алгоритма идентификации примитивных объектов на изображении и определения с их помощью класса сцены. Определены классы примитивных объектов и сцен локаций, для распознавания которых будут разрабатываться и обучаться модели. Для оценки качества моделей и алгоритма выбраны метрики F1-score, mAP и IoU.

Сформирован набор данных для обучения и тестирования моделей, состоящий из 104 фотографий офисов, складов, теплиц и жилых комнат. В целях расширения обучающей выборки рассмотрены методы создания синтетических данных. Как наиболее подходящий, выбран метод формирования сцен с трехмерными моделями примитивных объектов на фоне реального изображения. Реализован скрипт генерации рандомизированных синтетических данных в среде моделирования Blender с помощью библиотеки BlenderProc. В результате работы скрипта создан набор размеченных данных, содержащий около 500 объектов примитивов каждого класса.

Для задачи детекции примитивов проанализировано два метода идентификации объектов – с помощью прямоугольных и ориентированных ограничительных рамок, последний способ показал свою эффективность при решении задач на изображениях реального мира. Сформирован список

существующих предобученных моделей нейросетей семейства YOLO, способных работать с ориентированными ограничительными рамками. По результатам обучения моделей версий YOLOv8-obb и YOLOv11-obb разных размеров на синтетических, реальных и комбинированных данных модель YOLOv11m-obb показала лучшие результаты с точностью предсказания примитивов на изображениях реального мира в 87.6%.

Среди моделей, способных обрабатывать последовательности примитивов разной длины наиболее подходящими оказались модели с механизмами внимания GRU, требующие меньший объем данных для обучения, чем с механизмами LSTM. С помощью библиотеки PyTorch реализована модель нейросети со слоем GRU, способная предсказывать вероятности классов локаций на изображениях на основе последовательности ограничивающих рамок примитивов. В результате обучения моделей с различными параметрами архитектуры получена модель классификации, показывающая точность предсказания в 70% на реальных изображениях. Также протестирован метод ансамблирования моделей бинарной классификации, полученный ансамбль достиг точности в 45% на тестовых данных.

Полученные модели детекции и классификации объединены в алгоритм, способный идентифицировать на изображении примитивные объекты и на основе их размеров, положения и угла поворота определять класс сцены.

Глава 3 Разработка системы для идентификации примитивов и классификации сцен

В данной главе описан процесс проектирования и реализации программной системы для тестирования разработанного алгоритма идентификации примитивов и классификации сцен на изображениях.

3.1 Сценарии использования системы для классификации сцен изображений на основе примитивных объектов

На основе требований, выявленных на этапе анализа, определены варианты использования системы, представленные на рисунке 35. Наиболее приоритетным является «Запрос на классификацию сцены на изображении», включающий в себя два основных требования к алгоритму – произвести «Предсказание границ и расположения примитивных объектов» и «Предсказание класса сцены».

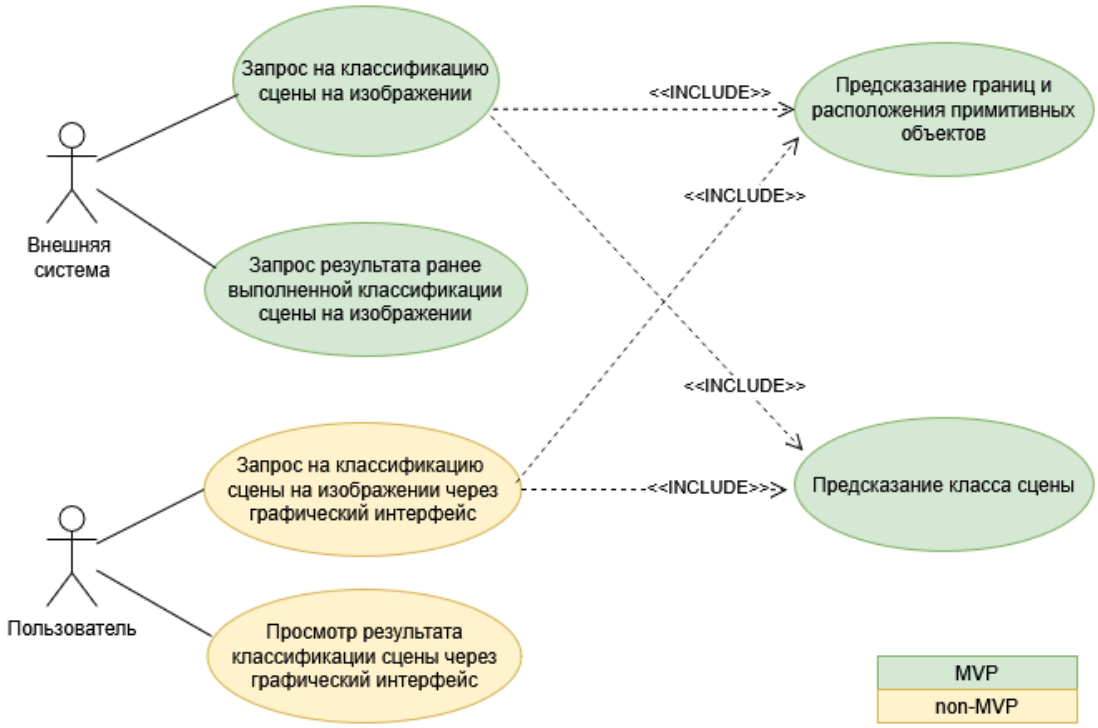

Рисунок 35 – Диаграмма прецедентов системы идентификации примитивов и классификации сцен изображений

Сценарий основного прецедента «Запрос на классификацию сцены на изображении» описан в таблице 10, сценарии прецедентов «Предсказание границ и расположения примитивных объектов» и «Предсказание класса сцены» включены в представленный сценарий как шаги 3 и 4 основного потока.

Сценарий «Запрос результата ранее выполненной классификации сцены на изображении» представлен в таблице 11.

Сценарии, связанные с использованием Пользователем графического интерфейса для взаимодействия с системой представлены в ПРИЛОЖЕНИИ В.

Таблица 10 - Сценарий прецедента «Запрос на классификацию сцены на изображении»

Идентификатор и название	UC-1. Запрос на классификацию сцены на изображении
Автор	Белова А.В., 02.03.2025
Акторы	**Основной:** Внешняя система
Описание	Прецедент позволяет Внешней системе отправить изображение на обработку Системой для дальнейшего получения предсказания примитивных объектов на изображении и вероятностей классов сцены изображения.
Триггер	Внешняя система отправляет запрос на обработку изображения к серверу Системы.
Предварительные условия	PRE-1. Внешняя система зарегистрирована разработчиком в Системе и получила токен аутентификации. PRE-2. Внешняя система авторизована в Системе, в запросе присутствует токен аутентификации.
Выходные условия	POST-1. Отправленное на обработку изображение и полученные к нему предсказания сохранены Системой и доступны для дальнейшего просмотра.
Основной поток	
Внешняя система	**Система**
1. Внешняя система отправляет запрос с изображением на обработку.	2. Система получает изображение и создает задачу со статусом «В очереди», сохраняет полученное изображение. Система возвращает идентификатор созданной задачи в ответе на запрос. При появлении свободной модели идентификации примитивов, система отправляет изображение на обработку и изменяет статус задачи на «В обработке».
3. Внешняя система получает идентификатор созданной задачи.	4. Система получает вектор предсказаний примитивов на изображении (класс, расположение, размер, поворот) и отправляет его свободной модели классификации на обработку. Система добавляет к соответствующей задаче информацию о полученных примитивах.

	5. Система получает вероятности классов изображения от модели классификации и добавляет к соответствующей задаче информацию о полученных вероятностях классов и о наиболее вероятном классе. Статус задачи изменяется на «Выполнена».
Альтернативные потоки	
	4.1 Система получает пустой вектор примитивов от модели идентификации (модель не смогла определить ни один объект). Система отправляет в ответе на запрос пустой вектор примитивов и класс изображения «unknown». Статус задачи изменяется на «Выполнена», к ней добавляется информация о **нулевых** вероятностях классов, наиболее вероятный класс записывается как «unknown».
	5.1 Система получает вероятности классов изображения, но несколько классов имеют наибольшую равную вероятность. Система отправляет полученный список примитивов, список вероятностей классов и выбранный наиболее вероятным класс «uncertain» в ответе на запрос. Статус задачи изменяется на «Выполнена», к ней добавляется информация о вероятностях классов, наиболее вероятный класс записывается как «uncertain».

Таблица 11 – Сценарий прецедента «Запрос результата ранее выполненной классификации сцены на изображении»

Идентификатор и название	UC-2. Запрос результата ранее выполненной классификации сцены на изображении
Автор	Белова А.В., 02.03.2025
Акторы	**Основной:** Внешняя система
Описание	Прецедент позволяет Внешней системе запросить результаты обработки изображения по идентификатору задачи.
Триггер	Внешняя система отправляет запрос на получение результатов обработки изображения к Серверу системы.
Предварительные условия	PRE-1. Внешняя система зарегистрирована разработчиком в Системе и получила токен аутентификации. PRE-2. Внешняя система авторизована в Системе, в запросе присутствует токен аутентификации.

	PRE-3. Внешняя система создала задачу на обработку и использует ее идентификатор в запросе.
Выходные условия	-
Основной поток	
Внешняя система	**Система**
1. Внешняя система отправляет запрос на получение результатов предсказания.	2. Система получает идентификатор задачи и проверяет наличие и валидность токена аутентификации Внешней системы.
	3. Внешняя система аутентифицирована, Система собирает информацию о результатах задачи по ее идентификатору.
5. Внешняя система получает запрошенную информацию о статусе задачи и результатах предсказания.	4. Система отправляет Внешней системе статус задачи, информацию о идентифицированных объектах и вероятностях классов локаций в ответе на запрос.
Альтернативные потоки	
	3.1 Внешняя система не прошла аутентификацию, Система возвращает информацию об ошибке.
	4.1 В Системе нет информации о задаче с указанным идентификатором, Система возвращает информацию об ошибке.

3.2 Проектирование архитектуры системы для моделей нейросетей

Данный исследовательский направлен на тестирование разработанного алгоритма и поставленной гипотезы, поэтому при разработке программной системы для алгоритма фокус будет направлен на приоритетные прецеденты, отмеченные на диаграмме. Пользовательский интерфейс будет разработан с целью демонстрации реализованного метода и будет представлять собой прототип, не ориентированный на качественный пользовательский опыт.

3.2.1 Проектирование архитектуры и выбор инструментов реализации

Модели нейронных сетей исследуемого алгоритма разработаны на языке Python – для системы идентификации примитивов и классификации сцен также будет использован Python.

При проектировании архитектуры системы был сделан выбор в пользу микросервисной архитектуры, позволяющей изолировать модели нейросетей в отдельные сервисы. Преимуществом изолирования моделей является простота их замены на модели других версий или размеров, а также возможность быстрого добавления микросервисов одинаковых моделей для распределения нагрузки и повышения эффективности обработки запросов при потенциальном расширении системы. Дополнительно, процесс вычислений модели в отдельном микросервисе не будет блокировать интерпретатор Python и останавливать работу основного сервиса системы.

Разработанные модели детекции и классификации могут работать независимо друг от друга при наличии готовых данных для обработки. Для реализации параллельной обработки нескольких запросов к системе и для распределения нагрузки между микросервисами принято решение использовать брокер сообщений Apache Kafka. Он также позволит реализовать передачу данных между моделями нейросетей, свободная модель при появлении задачи сразу приступит к ее обработке.

Для хранения предсказаний моделей и реализации последовательной обработки изображения сначала моделью детекции, а потом классификации необходимо создать базу данных и реализовать структуру «задач». Задача хранит в себе всю информацию о текущем этапе предсказания и статус для корректной обработки брокером сообщений. В качестве хранилища данных выбрана реляционная СУБД PostgreSQL.

При создании запроса на предсказание пользователь или внешняя система передают изображение, информация о примитивах и классах которого будет сохранена в базу данных для дальнейшего доступа. В целях обеспечения безопасности данных, пользователь может иметь доступ только к задачам и предсказаниям, которые он создал сам с помощью механизмов и токенов авторизации.

Было решено разработать четыре основных компонента системы – главный API сервер системы для принятия запросов и предобработки данных, сервисы моделей детекции и классификации, пользовательский интерфейс. Диаграмма компонентов системы представлена на рисунке 36.

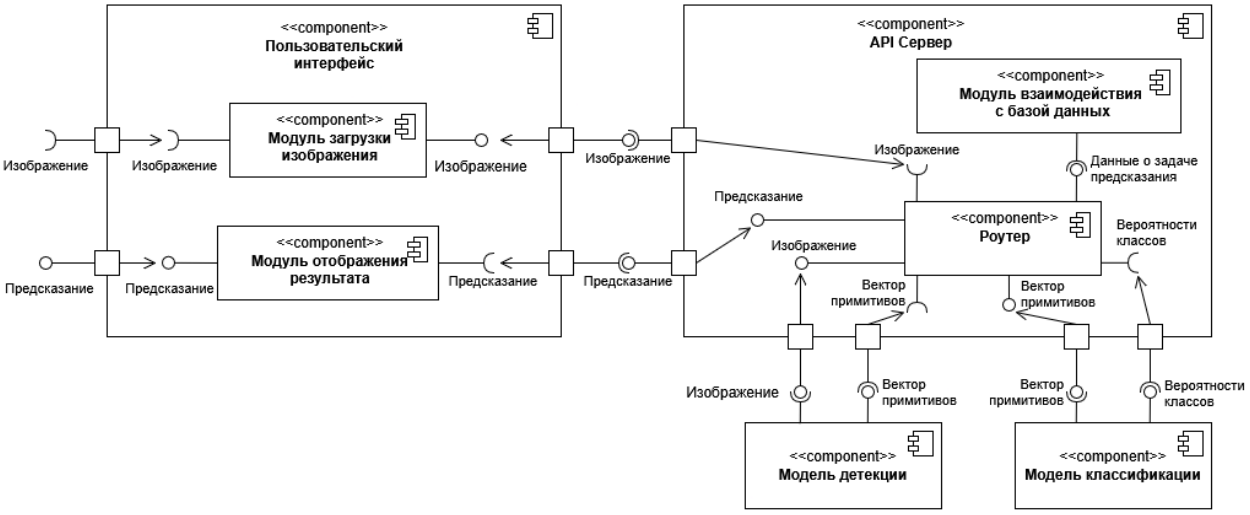

Рисунок 36 - UML диаграмма компонентов системы

Компонент «Роутер» основного «API сервера» отвечает за получение изображений для обработки, взаимодействие с базой данных для создания задач и обновления их информации, а также за вызов предсказаний моделей, размещенных в сервисах. При получении предсказаний моделей детекции и классификации сервер форматирует данные в JSON структуру и возвращает результат запроса. Взаимодействие с «Роутером» возможно напрямую или через компонент «Пользовательского интерфейса».

Для реализации API основного сервера и сервисов моделей выбран фреймворк Python FastAPI, также предоставляющий инструменты OAuth2 для авторизации. Для взаимодействия пользователя с системой определен набор эндпоинтов, следующих принципам REST API (см. таблицу 12).

Таблица 12 - Эндпоинты основного сервера системы

№	Адрес	Описание метода	Метод
1	api/oauth/login	Получение токена авторизации	POST
2	api/tasks	Создание задачи на предсказание	POST
3	api/tasks	Получение списка задач	GET
4	api/tasks/{id}/status	Получение статуса задачи	GET
5	api/tasks/{id}/input	Получение входных данных задачи (изображения)	GET

6	api/tasks/{id}/result	Получение результата выполнения задачи	GET
7	api/models	Получение списка моделей нейросетей	GET
8	api/tasks/{id}	Удаление задачи	DELETE

Полная документация API, включающая описание эндпоинтов, требуемые параметры HTTP запроса, варианты кодов и тел ответа представлена в ПРИЛОЖЕНИИ Г.

Фреймворк FastAPI также позволяет реализовать пользовательский интерфейс в формате веб-страниц с помощью инструмента HTML-шаблонизации Jinja2.

3.2.2 Проектирование базы данных системы

Для отслеживания процесса предсказания класса сцены на изображении и для передачи данных между моделями предлагается идея «Задач», хранящих в себе статус выполнения и элементы предсказания – список идентифицированных примитивов и вероятностей классов.

На рисунке 37 представлена схема базы данных, включающая две основные сущности - «Задача» и «Предсказания». На одном изображении сцены может быть идентифицировано неопределенное количество объектов примитивов, таблица «Предсказания» отвечает за хранение отдельных примитивов и содержит поля класса примитива, координат ограничивающей рамки и уверенность модели.

Таблицы «Класс примитива» и «Класс сцены» являются словарями, содержащим возможные значения классов примитивов и сцен соответственно. Аналогично, таблицы «Модели классификации», «Модели детекции» содержат названия и версии моделей для сохранения в «Задаче» информации о том, кто должен обработать или уже обработал изображение. Таблица «Статус» содержит возможные значения статусов задачи - «В очереди», «В обработке» и «Выполнена».

Таблица «Пользователи» хранит информацию о логине и хэшированном пароле для проверки авторизации при попытке доступа к задаче.

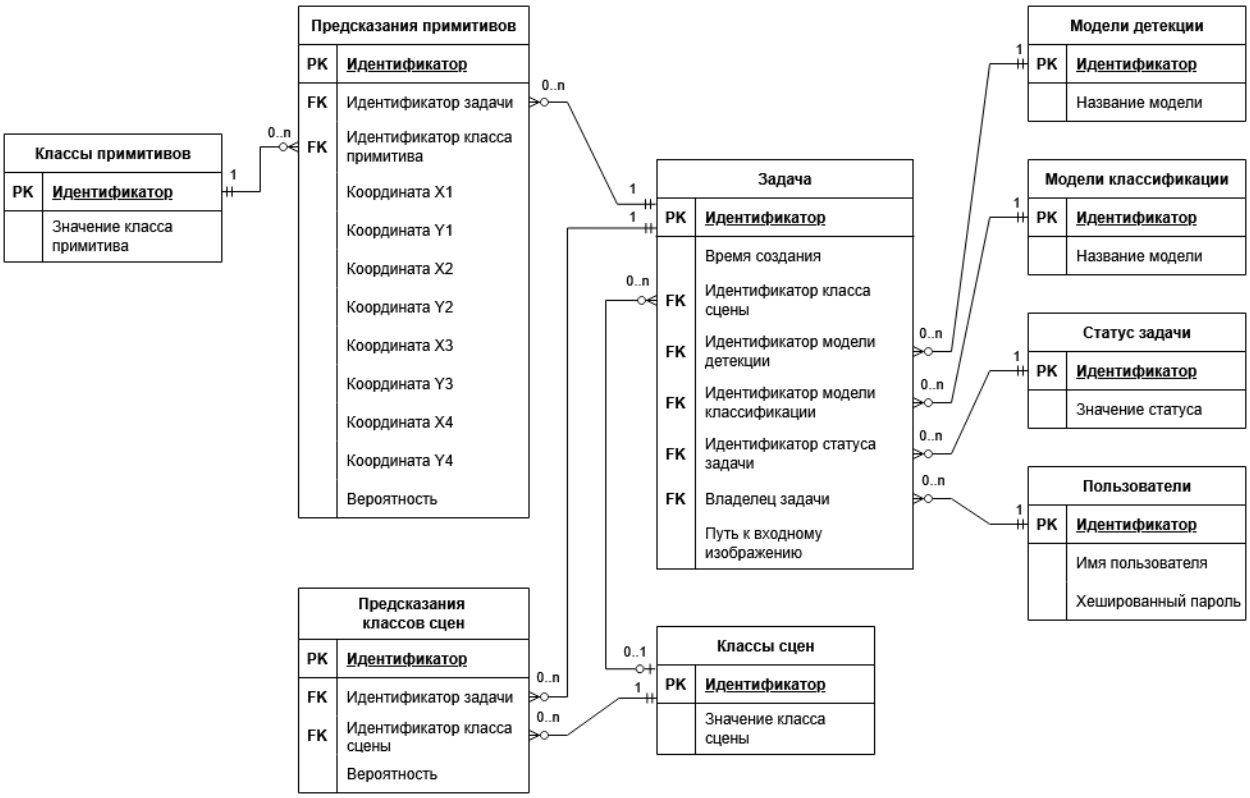

Рисунок 37 - Схема базы данных системы

Отношения базы данных находится в первой нормальной форме, так как все атрибуты сущностей являются атомарными и неделимыми. Справочные данные вынесены в отдельные таблицы классов примитивов, сцен и названий моделей.

Вторая нормальная форма отношений реализована отсутствием частичных функциональных зависимостей – каждый атрибут задачи и предсказания полностью зависит только от первичного ключа таблицы.

Транзитивные функциональные зависимости между атрибутами отсутствуют, данные классов и названий моделей не повторяются, а ссылаются на соответствующие справочные таблицы – база данных находится в третьей нормальной форме.

3.2.3 Проектирование взаимодействий компонентов системы

Для основного прецедента системы «Запрос на предсказание сцены на изображении» составлена диаграмма последовательности, представленная на рисунке 38.

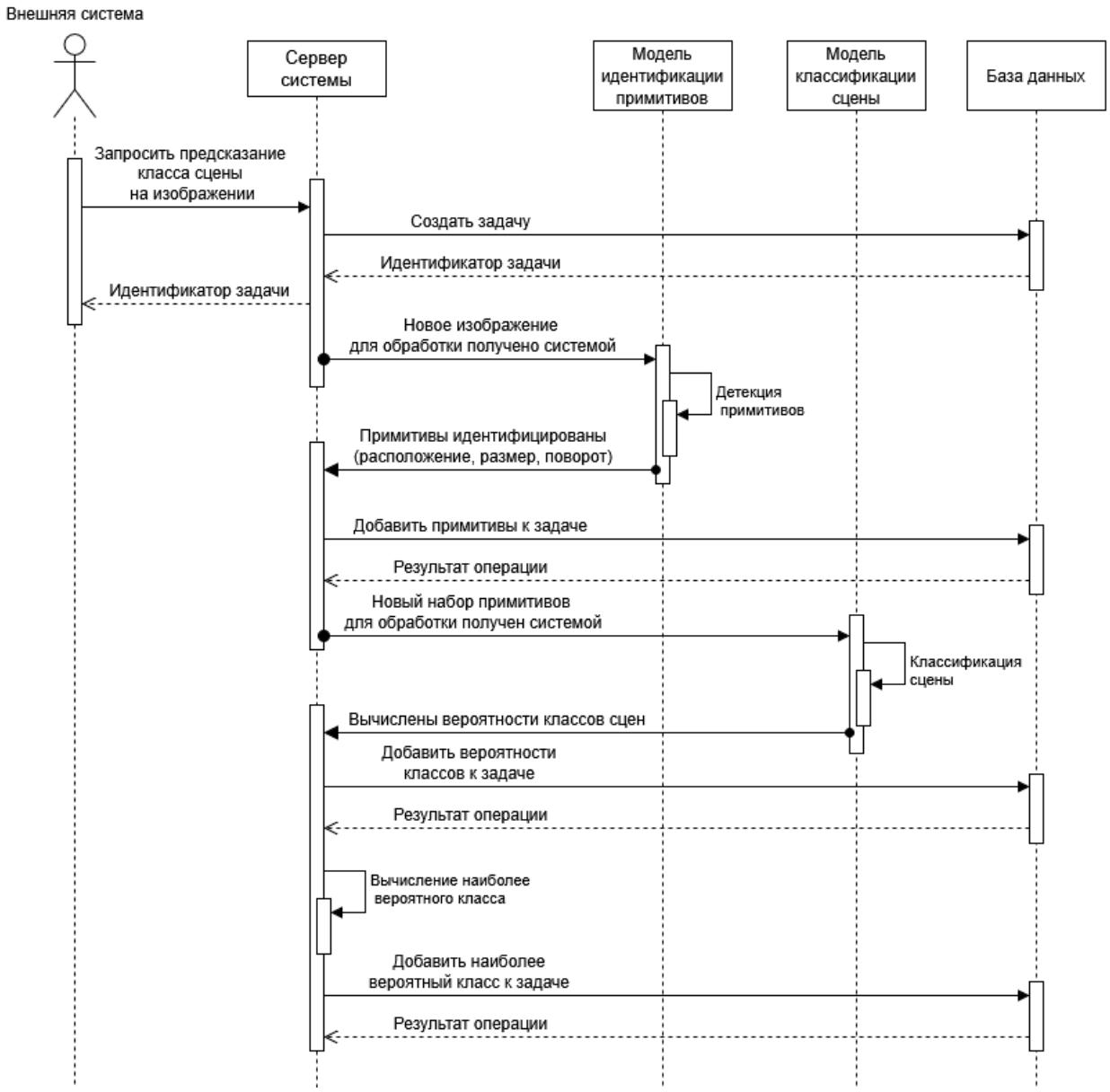

Рисунок 38 - Диаграмма последовательности для прецедента «Запрос на предсказание сцены на изображении»

Диаграмма включает в себя последовательности взаимодействий между основным сервером системы и сервисами моделей детекции и классификации для прецедентов «Предсказание границ и расположения примитивных объектов» и «Предсказание классов сцены» соответственно.

Выполнение предсказания производится в асинхронном режиме, поэтому для получения результатов необходимо сделать запрос с полученным идентификатором задачи. Для прецедента «Запрос результата ранее выполненной классификации сцены на изображении» диаграмма последовательности представлена на рисунке 39.

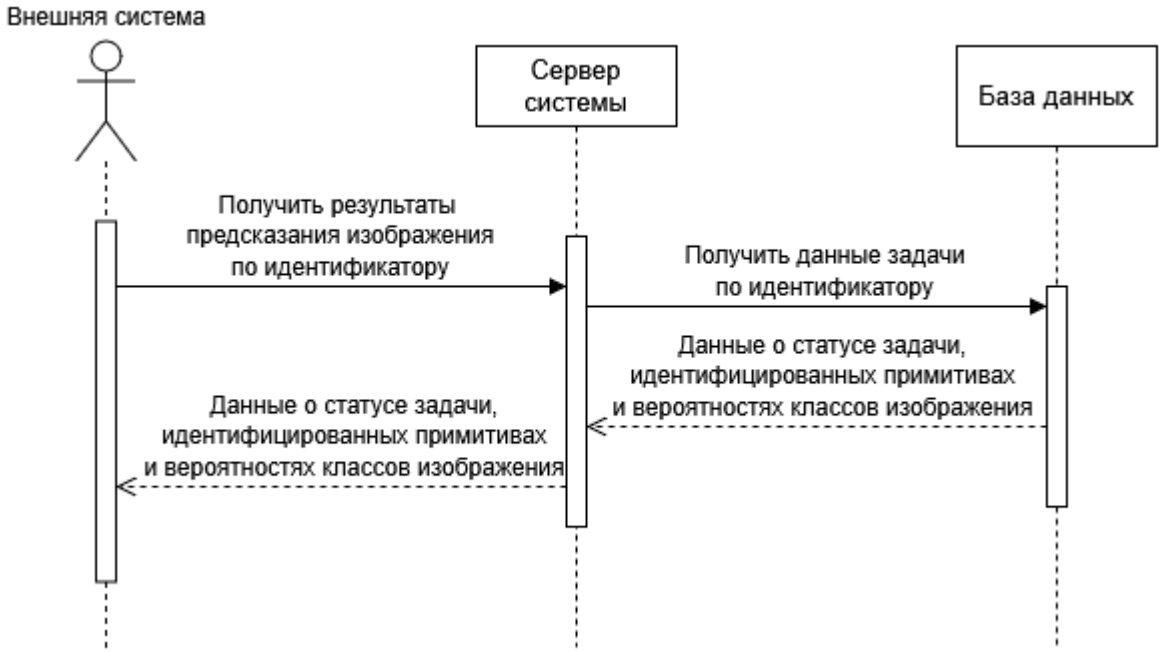

Рисунок 39 - Диаграмма последовательности для прецедента «Запрос результата ранее выполненной классификации сцены на изображении»

Взаимодействием с базой данных занимается только основной сервер, предобрабатывая и направляя информацию необходимым моделям. Пользователь или внешняя система при создании запроса на обработку изображения и при получении предсказаний взаимодействует только с основным сервером системы.

Диаграммы последовательности для прецедентов Пользователя с графическим интерфейсом представлены в ПРИЛОЖЕНИИ Д.

3.3 Реализация системы

Реализация микросервисной архитектуры системы включает в себя разработку основного FastAPI сервера системы и сервисов моделей детекции и классификации для разработанного алгоритма классификации сцен. Для взаимодействия сервера с базой данных необходимо разработать и реализовать запросы к данным, подключение к серверу PostgreSQL, методы инициализации таблиц.

3.3.1 Разработка ORM моделей данных и запросов к данным

Для работы с данными из PostgreSQL внутри сервера FastAPI решено реализовать ORM модели данных с помощью библиотеки SqlAlchemy. Библиотека

Pydantic используется для создания схем, описывающих необходимые поля данных для их получения и загрузки в базу данных.

Модели данных таблиц реализованы классами, наследующими класс базовой модели SQLAlchemy «declarative_base». В модели определены типы данных, ограничения для значений полей, первичный и внешние ключи для реализации отношений сущностей.

Далее представлен пример класса ORM модели таблицы «Задачи»:

```
class Tasks(Base):
    __tablename__ = "tasks"

    id = Column(Integer, primary_key=True, nullable=False)
    owner_id = Column(Integer, ForeignKey("users.id"), nullable=False)
    created_at = Column(DateTime, default=func.now(), nullable=False)
    scene_class_id = Column(Integer, ForeignKey("scene_class.id"))
    detection_model_id = Column(Integer,
ForeignKey("detection_models.id"), nullable=False)
    classification_model_id = Column(Integer,
ForeignKey("classification_models.id"), nullable=False)
    status_id = Column(Integer, ForeignKey("status.id"),
nullable=False)
    input_path = Column(String, nullable=False)
```

Схема Pydantic позволяет проводить валидацию данных при их загрузке в базу данных или при получении. Для каждой сущности определено три класса:

- «{имя модели}_Create», схема для создания объекта и загрузки в базу данных, определяет только передаваемые при создании поля.
- «{имя модели}», схема для получения объекта из базы данных, включает поля первичных ключей для идентификации объекта.
- «{имя модели}_Base», схема, содержащая общие поля предыдущих схем.

Для быстрого форматирования объектов схем в формат JSON объекта предусмотрен класс «Config», определяющий способ создания JSON объекта. В случае моделей системы формируется словарь с полями в качестве ключей и значениями полей в качестве значений.

Пример Pydantic схемы «Задачи» приведен далее:

```
class TaskBase(BaseModel):
    owner_id: int
    detection_model_id: int
    classification_model_id: int
    status_id: int
    input_path: str

class TaskCreate(TaskBase):
    pass

class Task(TaskBase):
    id: int
    created_at: datetime.datetime
    scene_class_id: int | None

    class Config:
        from_attributes = True
```

Хранение ORM моделей, схем и запросов к данным реализовано через структуру папок, отдельных для каждой сущности. В файлах «dao.py» с помощью ORM SQLAlchemy реализованы SQL запросы к данным: создание объекта и его загрузка в базу данных, получение всех объектов с определенным параметром, удаление объектов.

Далее представлен пример асинхронных запросов для получения информации о задаче и для создания новой задачи с помощью схемы и модели объекта:

```
async def get_task(db: AsyncSession, owner_id: int, task_id: int) ->
Union[models.Tasks, None]:
    q = select(models.Tasks).filter(models.Tasks.owner_id == owner_id,
models.Tasks.id == task_id)

    return (await db.execute(q)).scalar()

async def create_task(db: AsyncSession, task: task_schema.TaskCreate):
    task = models.Tasks(owner_id=task.owner_id,
                        detection_model_id=task.detection_model_id,

classification_model_id=task.classification_model_id,
                        status_id=task.status_id,
                        input_path=task.input_path)

    db.add(task)
    await db.commit()
    await db.refresh(task)

    return task
```

Для инициализации базы данных и ее обновлений используется библиотека Alembic с системой миграций. Инициализирующая миграция содержит код создания таблиц с помощью моделей ORM SQLAlchemy, заполнение справочных таблиц данными о классах примитивов, сцен, моделях детекции и классификации, а также добавление тестовых данных задач для проверки работы сервера.

3.3.2 Реализация проверки авторизации пользователя

Механизм авторизации пользователя реализован с помощью модуля OAuth2 библиотеки FastAPI. В рамках исследовательского проекта не предусмотрена возможность регистрации, информация о логинах и паролях вносится в базу данных разработчиком.

При выполнении запроса к базе данных пользователю необходимо предоставить валидный токен аутентификации JWT, срок жизни которого составляет 30 дней, после чего пользователю потребуется получить новый для продолжения работы с приложением.

Далее представлен код метода авторизации и получения нового токена на основе логина и пароля пользователя:

```
@router.post("/login")
async def login_for_access_token(
        form_data: Annotated[OAuth2PasswordRequestForm, Depends()],
        db: AsyncSession = Depends(database.get_db)
) -> Union[user_schemas.Token, None]:
    user = await authenticate_user(db=db, username=form_data.username,
password=form_data.password)
    if not user:
        raise HTTPException(
            status_code=status.HTTP_422_UNPROCESSABLE_ENTITY,
            detail="Incorrect username or password",
            headers={"WWW-Authenticate": "Bearer"},
        )
    access_token_expires = timedelta(days=ACCESS_TOKEN_EXPIRE_DAYS)
    access_token = create_access_token(
        data={"sub": user.username},
expires_delta=access_token_expires
    )
    return user_schemas.Token(access_token=access_token,
token_type="bearer")
```

Предоставленный логин сверяется с существующими в базе данных в чистом виде, а пароль для сравнения хешируется алгоритмом «bcrypt» для защиты информации в случае ее утечки из базы данных.

При успешной проверке существования пользователя происходит генерация нового JWT токена на основе секретного ключа и алгоритма «HS256» с установкой времени жизни. Полученный токен пользователю необходимо передавать в теле запроса для доступа к методам системы.

3.3.3 Разработка роутера основного сервера системы

В основе сервера FastAPI лежит роутер класса «APIRouter» с методами эндпоинтов, позволяющий пользователю с помощью HTTP запросов взаимодействовать с системой и получать предсказания моделей нейросетей.

В основном исполняемом файле сервера объявлен общий роутер, к которому при инициализации сервера подключаются роутеры задач, моделей и авторизации.

На примере создания задачи рассмотрим структуру запросов, код метода представлен далее:

```
@router.get("/{task_id}", response_model=task_schemas.Task)
async def get_task_by_id(current_user: Annotated[User,
Depends(get_current_user)],
                         task_id: int,
                         db: AsyncSession = Depends(database.get_db)
                         ) -> Union[task_schemas.Task, None]:
    # will get task only if auth
    task = await task_dao.get_task(db=db, owner_id=current_user.id,
task_id=task_id)

    if task is None:
        raise HTTPException(status_code=404, detail=f"Task with id
{task_id} does not exist.")
    return task
```

Метод эндпоинта реализован с помощью декоратора с определением метода «GET» и адреса для запроса. В параметре «response_model» определен класс объекта, который должен быть возвращен при успешном выполнении запроса – используются ранее разработанные Pydantic схемы.

Перед выполнением кода запроса будет выполнено получение значений переменных, определенных как зависимости с помощью структуры «Depends()». Для всех методов взаимодействующих с базой данных данным способом открывается подключение к базе данных «db». Функции, требующие авторизацию, проверяют доступ текущего пользователя «current_user» к запрашиваемым данным проверкой совпадения с автором задачи.

Ошибки в подключении к базе данных, отсутствии прав доступа или связанные с отсутствием запрашиваемых данных обрабатываются с помощью отправки исключений с соответствующим кодом ответа и комментарием о сути проблемы.

Успешно реализовано 8 методов, определенных на этапе проектирования, которые позволяют пользователю:

- получать токен аутентификации;
- создавать задачи для предсказания примитивов и класса сцены на изображении;
- просматривать статус, входные изображения задач и информацию о результатах предсказаний;
- просматривать названия моделей классификации и детекции для выбора при создании задачи;
- просматривать существующие личные задачи
- удалять ранее созданные задачи.

3.3.4 Разработка сервисов моделей детекции и классификации

Модели нейросетей для решения задач детекции и классификации реализуются в формате микросервисов выполняющих одну функцию – обработка входящих изображений или последовательностей примитивов и возвращение результатов предсказаний.

Классы моделей детекции инициализируются из сохраненных весов обученных моделей, как представлено далее:

```
d_weights_path = r"...\weights\best.pt"
detection_model = YOLO(d_weights_path)
```

При инициализации класса модели классификации требуется также передать параметры слоев модели:

```
c_weights_path = r"... \ClassifierModels\best_gru_model.pt"
input_size = 12
hidden_size = 32
num_layers=10
num_classes = 4

classification_model = GRUClassifier(input_size=input_size,
                                     hidden_size=hidden_size,
                                     num_layers=num_layers,

num_classes=num_classes).to(device)
classification_model.load_state_dict(torch.load(c_weights_path))
```

Входные данные передаются модели для выполнения расчета, а получаемые предсказания возвращаются серверу. Основным способом взаимодействия моделей с сервером FastAPI является брокер сообщений, описанный далее.

3.3.5 Настройка брокера сообщений Apache Kafka

Передача входных данных для моделей и результатов предсказания между сервисами системы осуществляется с помощью брокера сообщений Apache Kafka.

В основном продюсером сообщений о задачах будет основной сервер FastAPI, а консьюмерами – модели детекции и классификации, принимающие задачи на обработку. Но модели детекции и классификации также будут продюсерами сообщений с результатами обработки, а основной сервер будет консьюмером для принятия результатов и записи данных в БД.

Для передачи сообщений разных типов созданы следующие топики:

- «detect-primitives», для задач на обработку моделью детекции;
- «classify-image», для задач на обработку моделью классификации;
- «classify-image-binary», для задач на обработку ансамблем бинарных моделей классификации;
- «detected-primitives», для результатов модели детекции;
- «classified-image», для результатов модели классификации.

Для создания сообщений от сервера для сервисов моделей в асинхронном режиме создан класс «AsyncProducer», его код представлен далее:

```
class AsyncProducer:
    def __init__(self, configs):
        self._producer = Producer(configs)
        self._cancelled = False
        self._poll_thread = Thread(target=self._poll_loop)
        self._poll_thread.start()

    def _poll_loop(self):
        while not self._cancelled:
            self._producer.poll(2.0)

    def close(self):
        self._cancelled = True
        self._poll_thread.join()

    def produce(self, topic, value, loop):
        """
        An awaitable produce method.
        """
        result = loop.create_future()

        def delivery_callback(err, msg):
            if err:
                loop.call_soon_threadsafe(result.set_exception,
KafkaException(err))
            elif msg.error():
                loop.call_soon_threadsafe(result.set_exception,
KafkaException(msg.error()))
            else:
                loop.call_soon_threadsafe(result.set_result, msg)

        self._producer.produce(topic, value,
on_delivery=delivery_callback)
        return result
```

Экземпляр данного класса создается при инициализации сервера системы и при необходимости отправки сообщения выполняет метод «dispatch_task_detect_primitives» или «dispatch_task_classify_image», создающий сообщение с данными для модели в соответствующем топике с помощью метода «produce» класса. На основе модели классификации, указанной в задаче, происходит выбор топика для выполнения задачи классификации.

Консьюмер сервера подписывается на чтение сообщений в соответствующих топиках. Так как модель возвращает результат с идентификатором задачи, при обработке сообщения можно определить для какой задачи в базе данных предназначены объекты примитивов или вероятности классов и загрузить

соответствующую информацию в базу данных. Так как основной поток сервера FastAPI занят сервером uvicorn, консьюмер сообщений разворачивается в отдельном потоке в функции «lifespan», передаваемой при инициализации сервера:

```
@asynccontextmanager
async def lifespan(_app: FastAPI):
    current_loop = asyncio.get_running_loop()
    consumer_thread = Thread(target=lambda:
asyncio.run(server_consumer(current_loop)))
    consumer_thread.start()

    yield
    sp.close_producer()


app = FastAPI(lifespan=lifespan)
```

Выражение «yield» отвечает за выполнение основного жизненного цикла сервера – взаимодействие с пользователями и моделями. Перед началом работы сервера происходит инициализация консьюмера и подписка на топики сообщений, а после завершения - отключение консьюмера.

Основная выполняемая функция сервисов моделей детекции и классификации содержит цикл, выполняющий чтение и обработку сообщений во время всего времени работы сервиса, ее код представлен далее:

```
async def main():
    consumer = get_consumer()

    try:
        while True:
            msg = consumer.poll(10.0)
            if msg is None:
                print("Waiting...")
            elif msg.error():
                print("ERROR: %s".format(msg.error()))
            else:
                # Extract the (optional) key and value, and print.
                print("Consumed event from topic
{topic}.".format(topic=msg.topic()))

                # Get task id to send result back and get image to
predict
                task_id, image_bytes =
msg.value()[0:TASK_ID_BYTE_SIZE], msg.value()[TASK_ID_BYTE_SIZE:]

                # Process image
                image = Image.open(io.BytesIO(image_bytes))
                result = await ai_model.process(image)
                print(f"DETECTION MODEL RESULT for
task_id({int.from_bytes(task_id)}):", result)
```

```
                # Send resulting primitives to server
                answ = await
dispatch_task_detected_primitives(task_id=task_id,

result=json.dumps(result).encode("utf-8"),

loop=asyncio.get_event_loop())
    except KeyboardInterrupt:
        pass
    finally:
        # Leave group and commit final offsets
        consumer.close()

asyncio.run(main())
```

При обработке сообщения модель примитивов, как показано на рисунке, выделяет из сообщения идентификатор задачи для возвращения результата и изображение в формате байт-строки для идентификации примитивов. После выполнения расчета модель формирует JSON файл с описанием объектов и отправляет его в топик для результатов соответствующего типа с идентификатором задачи.

Модели классификации действуют аналогичным образом – принимают JSON файл примитивов в формате строки и преобразовывают к необходимому формату для дальнейшего предсказания вероятностей классов и отправки сообщения серверу.

3.3.6 Разработка пользовательского графического интерфейса системы

Для реализации прецедентов «Запрос на классификацию сцены на изображении через графический интерфейс» и «Просмотр результата классификации сцены через графический интерфейс» созданы две HTML страницы для загрузки изображения и для получения результатов выполнения задачи соответственно. Для доступа к страницам необходимо передавать JWT токен аутентификации в заголовке запроса, аналогично доступу к системе без интерфейса.

В рамках API сервера созданы следующие эндпоинты для доступа к веб-страницам интерфейса:

- api/ui/tasks (POST)
- api/ui/tasks/{task_id} (GET)

Первая страница предназначена для создания запроса на обработку изображения, ее разметка представлена на рисунке 40.

Рисунок 40 - Веб-страница создания запросов на предсказание

При создании задачи необходимо указать модели детекции и классификации, которые будут использованы при обработке изображения – для этого на странице реализованы выпадающие списки с вариантами моделей, получаемыми запросом к серверу системы. Выбор изображения производится с помощью HTML формы, принимающей файлы форматов JPG, JPEG и PNG. Ответы сервера об успешном создании задачи с присвоенным ей идентификатором или об ошибке отображаются после отправки запроса.

Страница просмотра результата выполнения задачи должна предоставлять информацию об идентифицированных примитивах, вероятностях классов сцен и о наиболее вероятном классе сцены на изображении. Для этого была реализована страница, пример которой представлен на рисунке 41.

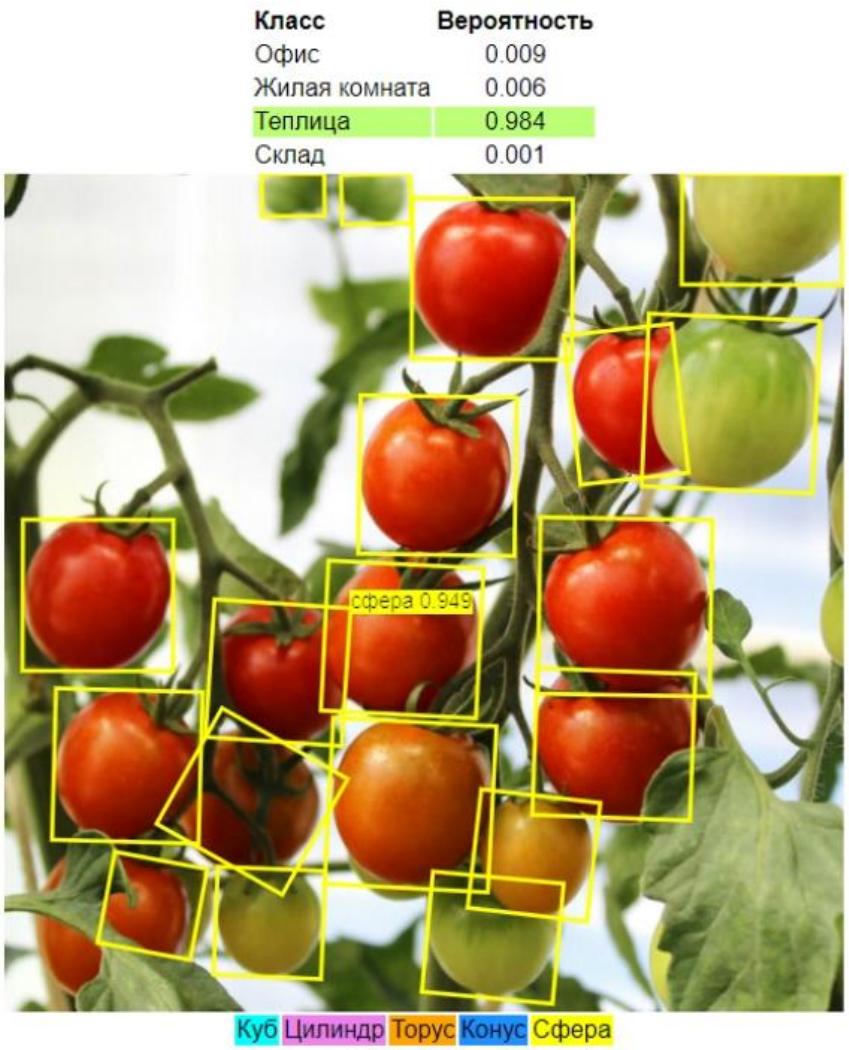

Рисунок 41 – Веб-страница визуализации результатов предсказаний моделей детекции и классификации системы

Шаблонизатор Jinja2 позволяет передавать результаты предсказаний примитивов и классов в формате JSON объекта для генерации разметки страницы, возвращаемой в запросе. Существующие классы сцен изображений и их вероятности для обработанного изображения формируют таблицу, в которой наиболее вероятный класс сцены помечается зеленым цветом. На входном изображении задачи ориентированными рамками помечаются все идентифицированные примитивы, каждому классу примитива назначен определенный цвет в соответствии с легендой, помещенной ниже изображения.

При большом количестве рамок объектов на изображении, подписи класса примитива и уверенности модели детекции в нем будут зачастую накладываться друг на друга и быть не читаемыми. Для решения данной проблемы был реализован JavaScript скрипт страницы, позволяющий отобразить класс и вероятность примитива нажатием на его рамку.

3.3.7 Контейнеризация сервера системы и сервисов моделей

Развертывание приложения производится с помощью контейнеризации с помощью платформы Docker. Настроены файлы docker-compose и Dockerfile для поднятия следующих контейнеров приложения:

– основной сервер FastAPI «task-service-app»;
– сервис модели детекции «detection-model-service»;
– сервис модели классификации «classification-model-service»;
– сервис ансамбля бинарных моделей классификации «classification-binary-model-service»;
– сервер базы данных PostgreSQL «task-service-postgres»;
– сервис с брокером сообщений Apache Kafka «kafka»;
– сервис инициализации топиков брокера сообщений «init-kafka».

Во избежание ошибок, между контейнерами настроены зависимости – контейнер инициализации топиков брокера сообщений запустится только после подтверждения статуса «healthy» сервиса самого брокера сообщений. Сервисы моделей детекции и классификации ожидают запуска основного сервера сообщений, а он – запуск сервера базы данных и окончание процесса инициализации топиков сервиса «init-kafka».

Результат успешного запуска контейнеров, инициализации базы данных и топиков брокера сообщений на тестовом стенде представлен на рисунке 42.

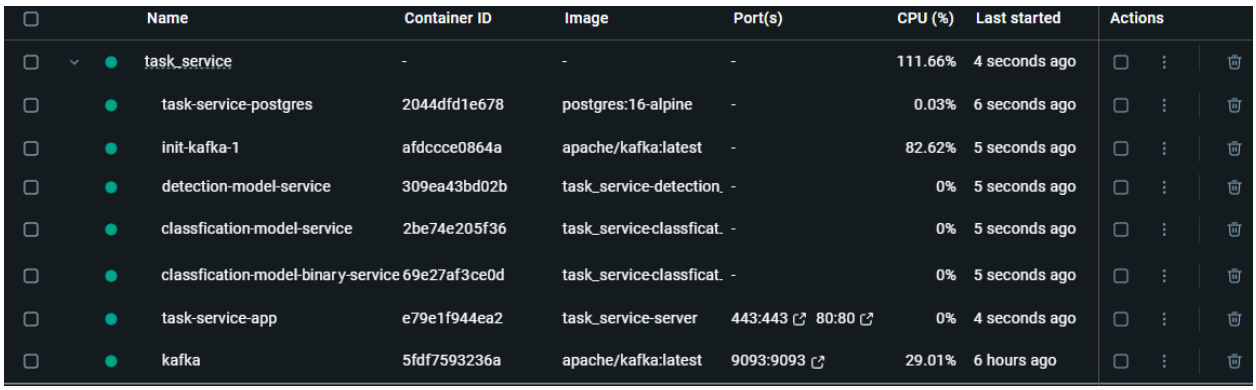

Рисунок 42 - Результат размертывания контейнеров приложения в Docker

Выводы по главе 3

На основе требований, выделенных на этапе анализа, составлена диаграмма прецедентов для взаимодействия пользователей и внешних систем с системой

идентификации примитивов и классификации сцен локаций на изображениях. Для прецедентов, приоритетных для тестирования разработанного алгоритма, составлены сценарии использования. С учетом архитектурных решений составлена диаграмма последовательности для основных прецедентов.

Для реализации приложения выбрана микросервисная архитектура, позволяющая выделить модели нейросетей в отдельные сервисы, что в будущем позволит удобно заменять модели на другие версии или размеры, а также расширять систему, добавляя несколько контейнеров моделей одного типа.

Для сохранения данных о предсказаниях примитивов и классов сцен изображения предложена структура «задач», также хранящая статус выполнения запроса на предсказание. Спроектирована база данных, находящаяся в третьей нормальной форме и содержащая три основных и пять справочных сущностей. Для работы с данными выбрана библиотека ORM SQLALchemy, созданы модели данных и схемы данных Pydantic.

Реализована система, состоящая из основного сервера, разработанного на Python FastAPI, сервисов моделей и сервера СУБД PostgreSQL. Разработаны REST API методы роутера сервера системы, позволяющие взаимодействовать с системой посредством HTTP запросов, определенных на этапе проектирования. Сформирована документация по API системы. Система аутентификации пользователя для доступа к данным реализована с помощью библиотеки OAuth2 и генерации временных JWT токенов.

Сервисы содержат классы обученных моделей детекции и классификации, загружающихся из сохраненных весов. Основным способом передачи информации для обработки моделями и результатов предсказания является брокер сообщений Apache Kafka, настроены топики для обмена сообщениями между сервисами.

Развертывание приложения произведено с помощью платформы Docker на локальном устройстве – созданы контейнеры основного сервера приложения, сервисы моделей детекции и классификации, сервер базы данных, сервисы брокера сообщений и инициализатора топиков брокера. Контейнеры успешно подняты и их работоспособность проверена.

Исходный код программы доступен в публичном репозитории GitHub по ссылке https://github.com/Miraellax/HSE-CSSD-Python. Сформировано руководство пользователя, представленное в ПРИЛОЖЕНИИ Е.

Глава 4 Тестирование системы и метода классификации сцен

В данной главе описаны процессы тестирования разработанного алгоритма классификации сцен изображений на наборе размеченных изображений реального мира и тестирования программной микросервисной системы с помощью HTTP запросов.

4.1 Тестирование алгоритма классификации сцен изображений по набору примитивных объектов

Для определения качества разработанного алгоритма классификации при тестировании было рассмотрено два варианта ансамбля моделей алгоритма – модель детекции и мультиклассовая модель классификации, модель детекции и ансамбль бинарных моделей классификации. По результатам тестирования будет выбран лучший набор моделей.

4.1.1 Тестирование алгоритма с мультиклассовой моделью классификации

По отдельности модели детекции примитивов и классификации сцен достигли качества в 87.6% и 70% соответственно при тестировании на размеченных вручную изображениях. На этапе тестирования будет проверена точность алгоритма при последовательной работе моделей, что может повлиять на точность классификации алгоритма.

Для тестирования алгоритма сформирован набор из 32 изображений, участвовавших в тестовых выборках моделей детекции и классификации, но не использовавшихся для обучения моделей. Набор содержит по 8 изображений классов сцен – офисы, теплицы, склады и жилые комнаты.

По результатам последовательного запуска моделей детекции и классификации получен отчет о классификации, представленный на рисунке 43.

	precision	recall	f1-score	support
office	1.00	0.12	0.22	8
livingroom	0.42	1.00	0.59	8
warehouse	1.00	0.50	0.67	8
greenhouse	1.00	1.00	1.00	8
accuracy			0.66	32
macro avg	0.86	0.66	0.62	32
weighted avg	0.86	0.66	0.62	32

Рисунок 43 - Отчет классификации алгоритма

Наиболее точные результаты алгоритм показывает при классификации сцен Теплиц – все изображения Теплиц были идентифицированы, и каждая классификация изображения как теплицы была корректной (precision = 1, recall = 1). Наименьшая точность алгоритма наблюдается для класса Офисов – F1-score = 0.22. Метрики классификации F1-score Жилых комнат и Складов выше среднего. Точность классификации алгоритма, усредненная по классам сцен, достигла 66%.

На рисунке 44 представлена нормализованная матрица ошибок, при сравнении с матрицей ошибок модели классификации можно заметить ряд изменений. На основе предсказаний модели детекции точность классификации Офисов упала на 13%, алгоритм ошибочно классифицировал данные изображения как Жилые комнаты. При этом точность предсказания классов Жилых комнат не изменилась и осталась равна 1, а 17% Теплиц, ранее идентифицированных как Жилые комнаты теперь определяются корректно. Точность классификации Складов не изменилась, алгоритм также путает половину изображений с классом Жилых комнат.

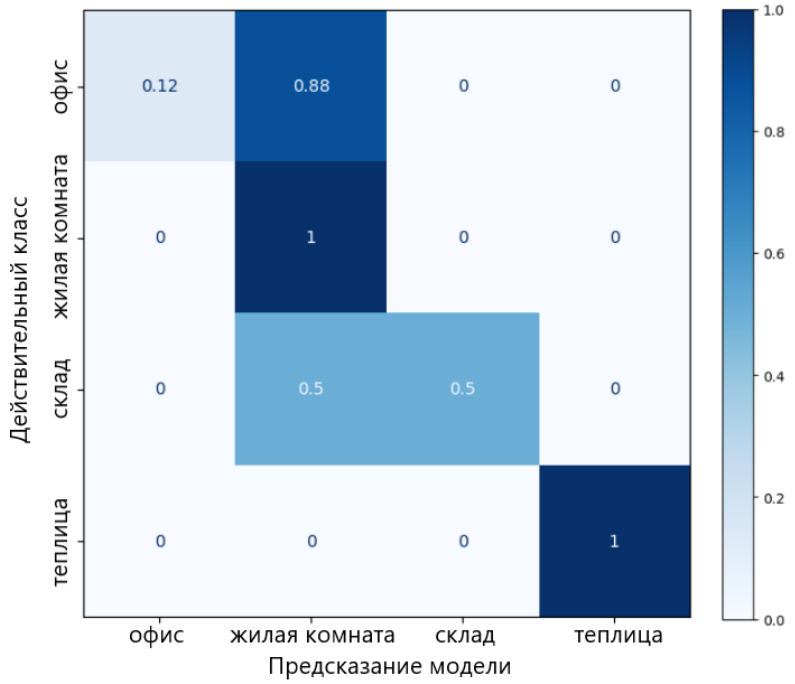

Рисунок 44 - Нормализованная матрица ошибок алгоритма классификации сцен локаций

Рассмотрим изображения, сцены которых алгоритм смог определить корректно и в которых допустил ошибку. На рисунке 45 представлено изображение Теплицы, которое алгоритм корректно классифицировал. Ложных предсказаний для данного класса изображений нет, точность и полнота классификации алгоритма составляет 100%.

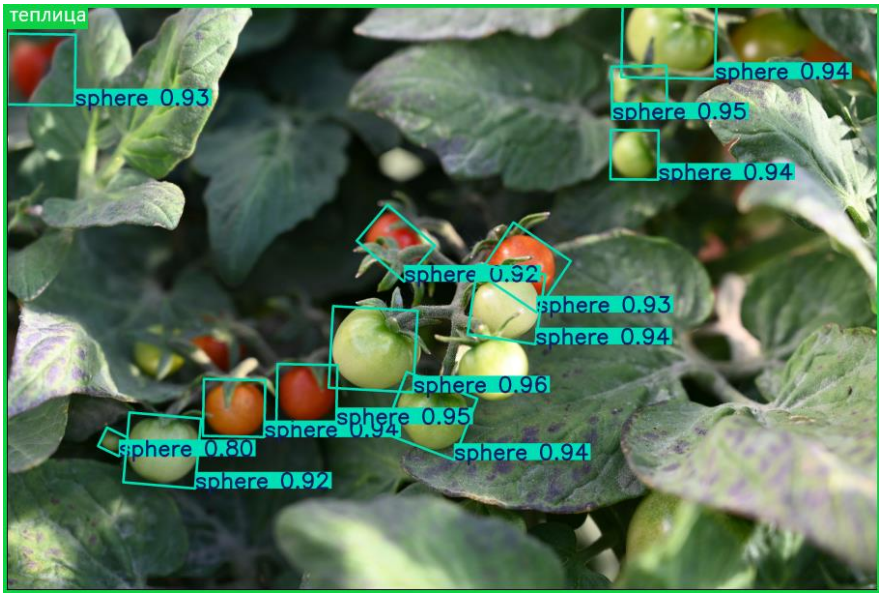

Рисунок 45 - Предсказание класса «Теплица»

На рисунке 46 изображены предсказания алгоритма «Жилые комнаты», но только левое изображение действительно является Жилой комнатой, правое

изображение является Офисом. Около 88% Офисов классифицировано как Жилая комната, согласно матрице ошибок алгоритма.

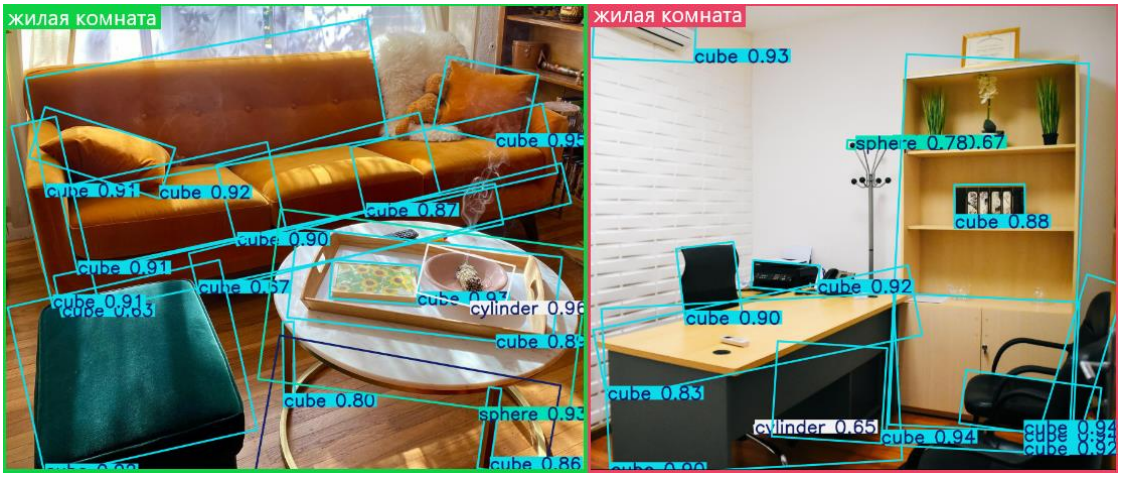

Рисунок 46 - Сравнение предсказаний класса «Жилая комната»

Рисунок 47 представляет изображения Складов, из которых левое корректно классифицировано как Склад, а правое спутано с Жилой комнатой. На ошибку в классификации могло повлиять недостаточное количество идентифицированных объектов кубов – многие изображения Складов имеют большое количество рядом расположенных примитивов «Куб», как на изображении слева.

Рисунок 47 - Сравнение предсказаний класса для изображений «Склада»

На рисунке 48 представлены изображения Офисов, классифицированные как Офис и Жилая комната. Можно заметить, что правильно классифицированное изображение содержит гораздо большее количество Цилиндров и Кубов, расположенных в примерной форме офисных стульев.

Скорее всего, большая часть изображений Офисов классифицирована как Жилая комната из-за недостатка идентифицированных примитивов, отличающих жилую комнату с диваном и столом от офиса со стулом и столом.

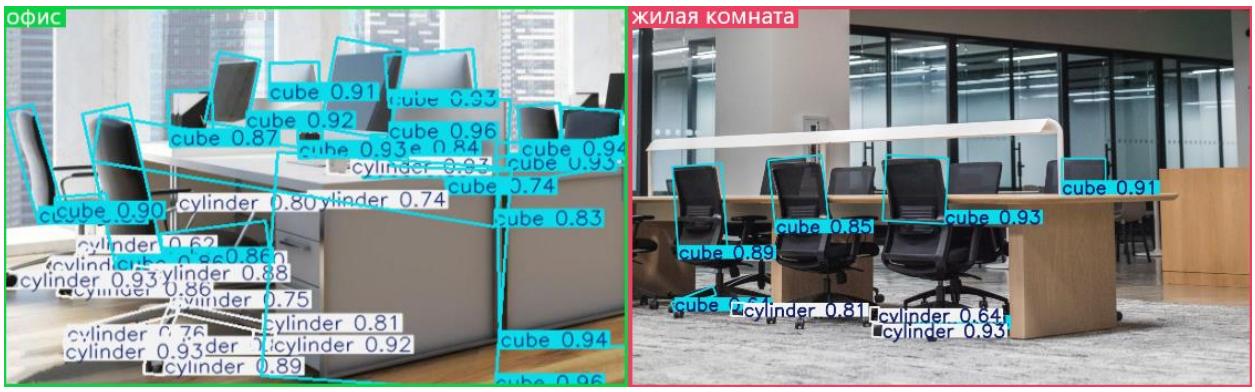

Рисунок 48 - Сравнение предсказаний для изображения «Офис»

После тестирования алгоритма на изображениях реального мира можно заметить снижение точности с 70 до 66%, по сравнению с тестированием модели классификации. Классификатор, случайно предсказывающий классы изображений из четырех доступных, будет иметь среднюю точность в 25%. Точность разработанного алгоритма заметно превышает данный показатель, что говорит о его потенциале и подтверждении поставленной гипотезы.

4.1.2 Тестирование алгоритма с ансамблем бинарных моделей классификации

Для сравнения качества алгоритмов с разными моделями классификации, при тестировании данного варианта алгоритма был использован тот же подготовленный набор реальных данных из 32 изображений. Результаты тестирования алгоритма с ансамблем моделей классификации представлен на рисунках 49 и 50.

	precision	recall	f1-score	support
office	0.50	0.12	0.20	8
livingroom	0.73	1.00	0.84	8
warehouse	1.00	0.50	0.67	8
greenhouse	0.53	1.00	0.70	8
accuracy			0.66	32
macro avg	0.69	0.66	0.60	32
weighted avg	0.69	0.66	0.60	32

Рисунок 49 – Отчет классификации алгоритма классификации сцен с ансамблем бинарных моделей

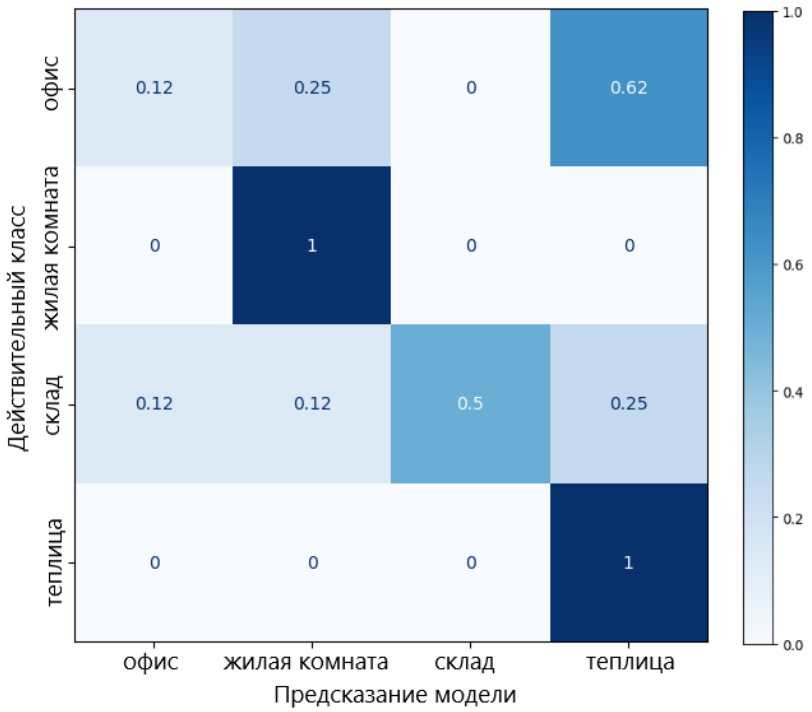

Рисунок 50 – Нормализованная матрица ошибок алгоритма классификации сцен с ансамблем бинарных моделей

Несмотря на точность ансамбля моделей всего в 45%, при его объединении с моделью детекции точность алгоритма достигла показателя в 66% и сравнялась с первым рассмотренным вариантом алгоритма.

При сравнении матриц ошибок можно заметить, что значения полноты для классов сцен одинаковы, однако ложные предсказания данного варианта алгоритма сильнее разбросаны по разным классам сцен. Алгоритм с единственной моделью классификации показывает более высокие метрики точности (precision) определения классов сцен – метрика точности, усредненная по классам, достигает 86%, по сравнению с 69% данного алгоритма.

Подход к решению задачи с помощью ансамбля моделей бинарной классификации также показал себя лучше случайного классификатора и превысил порог точности в 25%. Несмотря на более низкое качество распознавания по сравнению с алгоритмом с единственной моделью классификации, данный алгоритм имеет потенциал к расширению набора целевых классов без переобучения существующих моделей.

4.2 Тестирование системы идентификации примитивов и классификации сцен изображений

Тестирование системы проводилось согласно Программе и методике испытаний (см. ПРИЛОЖЕНИЕ Ж). Результаты сценарного тестирования части системы с графическим интерфейсом представлены в ПРИЛОЖЕНИИ И.

Тестирование сервера системы проводилось с помощью инструмента Postman, позволяющего создавать HTTP запросы с авторизацией и передачей изображений для обработки.

Для каждого из разработанных эндпоинтов сервера создан набор запросов, проверяющих все возможные ситуации и предусмотренные коды ответов.

Каждый запрос имеет скрипт тестирования на языке JavaScript и с использованием фреймворка Chai.js для проверки получаемого кода ответа и содержимого тела ответа. На рисунке 51 представлен пример тестового скрипта для запроса «Получение результатов предсказания» с корректным токеном авторизации для существующей задачи пользователя.

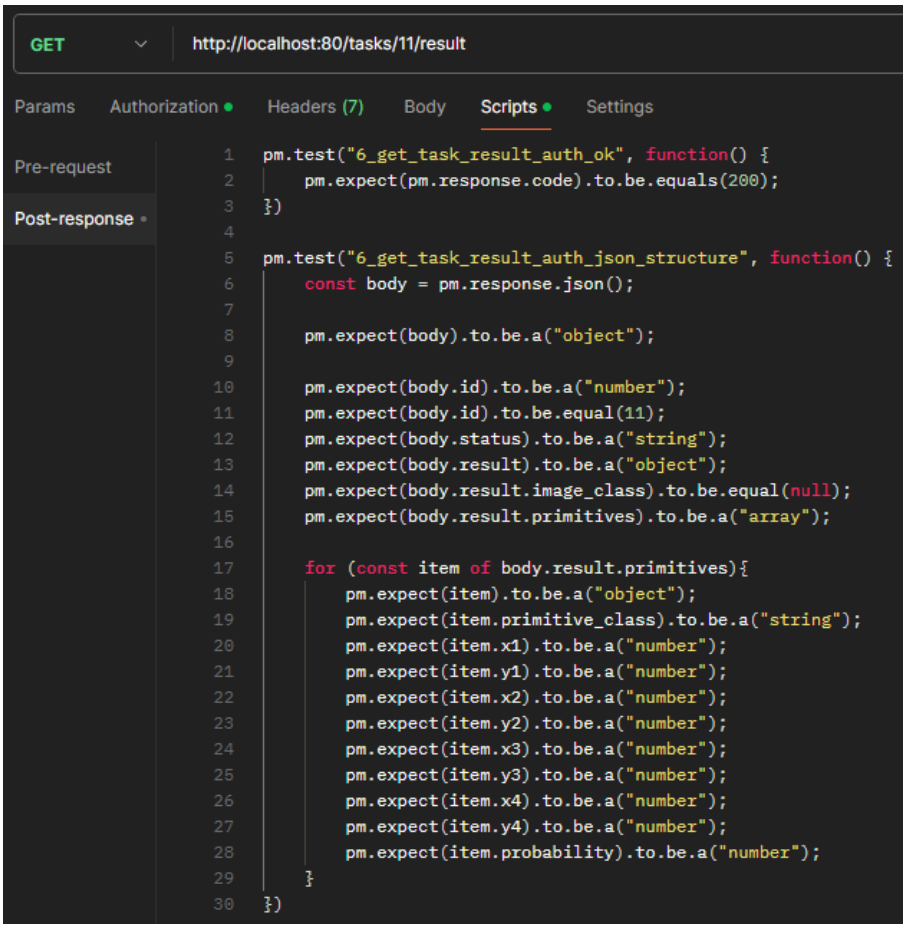

Рисунок 51 - Скрипт тестирования запроса «Получение результатов предсказания»

Ожидается, что данный запрос пройдет проверку аутентификации и получит содержимое предсказаний существующей задачи – первый тест проверяет соответствие кода ответа числу 200, второй тест анализирует структуру JSON файла с данными идентифицированных примитивов. Пример структуры ответа представлен на рисунке 52.

Рисунок 52 - Тело ответа на запрос «Получение результатов предсказания»

По результатам тестирования система корректно обрабатывает все запросы пользователя и возможные ошибки, разработанные тесты завершены успешно (см. рисунок 53).

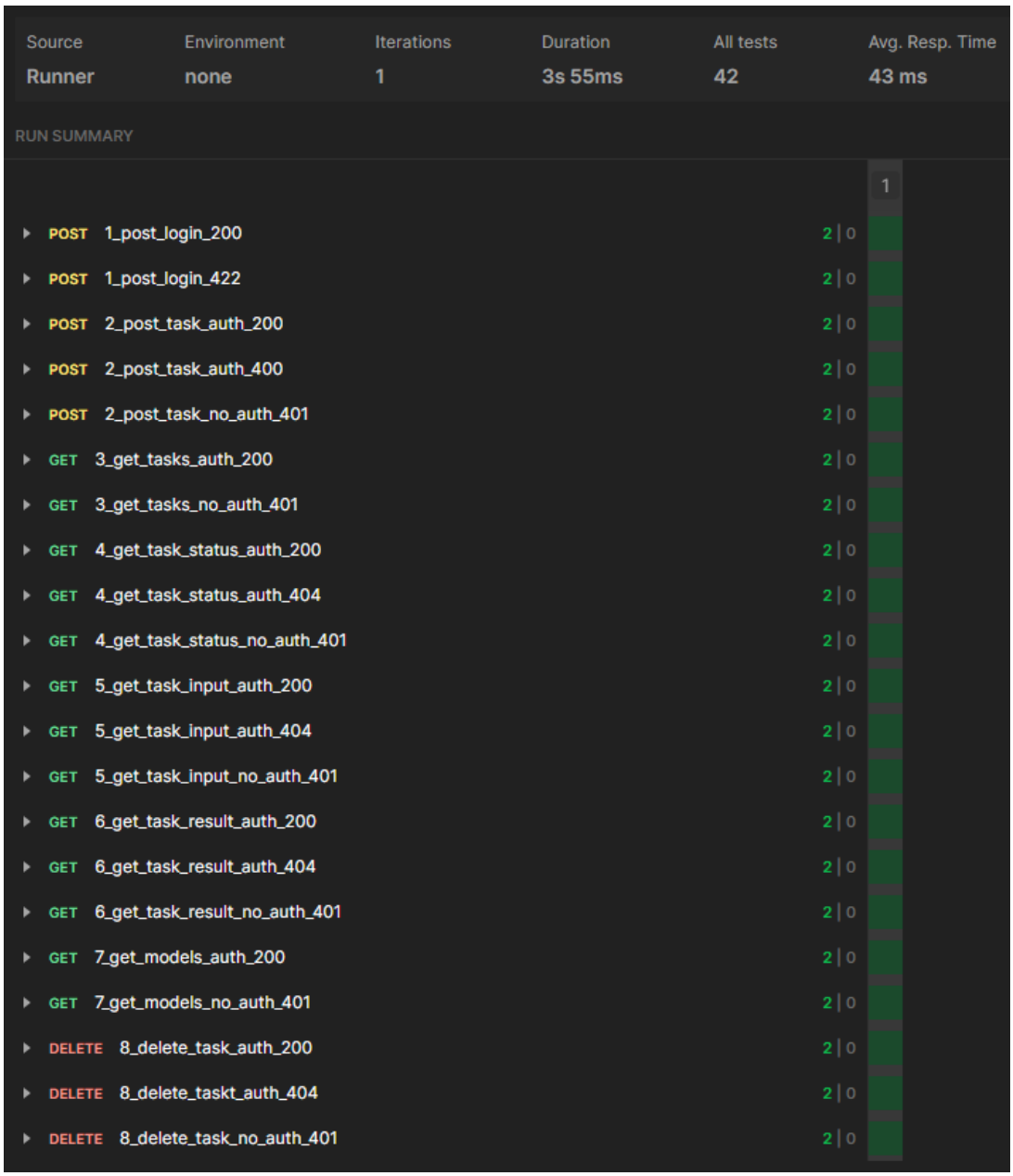

Рисунок 53 - Результаты тестирования API сервера системы

Заключение

В результате выполнения исследовательского проекта был разработан алгоритм идентификации примитивных объектов и классификации сцены локации на изображении на их основе, который может быть использован в системах зрения роботизированных устройств, предоставляя информацию об окружении для выполнения задач манипуляции объектами и навигации в пространстве.

В основе алгоритма лежит ансамбль моделей детекции примитивов и классификации изображения на их основе. Для тестирования нового алгоритма компьютерного зрения определен набор классов примитивов (куб, конус, сфера, цилиндр, торус) и набор классов локаций (офис, склад, жилая комната, теплица).

При выборе модели детекции был обучен ряд моделей семейства YOLO разных версий, способных работать с ориентированными ограничительными рамками объектов – YOLOv8-obb и YOLOv11-obb. Во время формирования набора обучающих данных был использован метод генерации синтетических изображений примитивов для расширения выборки. Метод генерации двухмерных изображений с использованием трехмерных моделей примитивных объектов показал свою эффективность, позволив обучить модели детекции, точность предсказаний которых на реальных данных на 45-55% превысило точность моделей, обученных только на реальных данных. Модель YOLOv11m-obb показала лучшее качество распознавания примитивов с метрикой F1-score = 0.85, mAP = 87.6% и была выбрана для реализации алгоритма.

При разработке модели классификации были рассмотрены механизмы внимания и памяти LSTM и GRU, позволяющие обрабатывать последовательности данных разной длины, что требуется для определения класса сцены по набору примитивных объектов на изображении. В результате сравнения был выбран механизм GRU, как менее требовательный к времени обучения и размеру обучающей выборке, чем LSTM. С помощью библиотеки PyTorch был разработан класс модели с использованием слоя GRU, ряд моделей был обучен с разными наборами параметров для определения наиболее подходящей архитектуры модели для решения поставленной задачи. По результатам обучения выбрана модель классификации, показавшая значение метрики F1-score = 0.7 при тестировании на

наборах примитивов с реальных размеченных изображений. Рассмотрен подход с ансамблированием моделей бинарной классификации для решения задачи.

При ансамблировании моделей детекции и классификации для последовательной работы при обработке изображения получен алгоритм классификации сцен изображений по набору примитивных объектов на них с двумя вариантами моделей классификации. В результате тестирования алгоритмов на реальных изображениях получена точность предсказания 66%. Данный показатель превышает метрику точности 25% алгоритма случайного выбора класса изображения из четырех возможных, что говорит о перспективности развития метода. Поставленная на этапе анализа **гипотеза подтверждена** – классификация сцен изображений только по набору идентифицированных примитивных объектов на нем возможна.

Для тестирования разработанного алгоритма была реализована система с микросервисной архитектурой, в которой модели детекции и классификации выделены в сервисы для возможности потенциального расширения системы и параллельной обработки множества запросов. Передача данных для обработки и результатов предсказаний реализована с помощью брокера сообщений Apache Kafka. Для хранения данных о предсказаниях используется СУБД PostgreSQL, работа с данными внутри системы происходит с помощью ORM моделей SQLAlchemy и схем Pydantic.

Развертывание системы проведено с использованием Docker контейнеров. Система успешно развернута, ее работа протестирована с помощью HTTP запросов инструментом Postman для всех эндпоинтов и вариантов ответов, определенных на этапе проектирования системы.

Реализованный алгоритм классификации сцен на изображении на основе данных о размере, положении и наклоне примитивных объектов на нем может быть использован в области робототехники для выполнения задач манипуляции объектами и навигации в пространстве. Преимуществом метода является возможность идентифицировать сложные объекты, на которых модель детекции не была обучена, с помощью идентификации примитивных составляющих без необходимости дополнительного обучения.

Библиографический список

1. Сливницин П., Мыльников Л. Распознавание объектов по составляющим их примитивам и отношениям между ними // Informatics and Automation. – 2023. Т. 22. № 3. – С. 511–540.
2. Data-driven robotic visual grasping detection for unknown objects: A problem-oriented review / Tian H. [и др.] // Expert Syst Appl – 2023. Т. 211. – С. 118624.
3. Scene Recognition from Image Using Convolutional Neural Network / Masood S. [и др.] // Procedia Comput Sci. – 2020. Т. 167. – С. 1005–1012.
4. Zhang T., Zhang C., Hu T. A robotic grasp detection method based on auto-annotated dataset in disordered manufacturing scenarios // Robot Comput Integr Manuf. – 2022. Т. 76. – С. 102329.
5. Liu H. W., Cao C. Q. Grasp Pose Detection Based on Point Cloud Shape Simplification // IOP Conference Series: Materials Science and Engineering: Institute of Physics Publishing, – 2020. – С. 012007.
6. Parsing Natural Scenes and Natural Language with Recursive Neural Networks / Socher R. [и др.] // Proceedings of the 28th International Conference on Machine Learning: ICML, – 2011. – С. 129–136.
7. Towards facial micro-expression detection and classification using modified multimodal ensemble learning approach / Zhang F. [и др.] // Information Fusion. – 2025. Т. 115. – С. 102735.
8. Tomato recognition and location algorithm based on improved YOLOv5 / Li T. [и др.] // Comput Electron Agric. – 2023. Т. 208. – С. 107759.
9. Arbitrarily Oriented Dense Object Detection Based on Center Point Network in Remote Sensing Images / Wang P. [и др.] // Remote Sens (Basel). – 2022. Т. 14. № 7. – С. 1536.

10. Integrated detection of coconut clusters and oriented leaves using improved YOLOv8n-obb for robotic harvesting / Fu Y. [и др.] // Comput Electron Agric. – 2025. Т. 231. – С. 109979.

11. Temporal integration of ResNet features with LSTM for enhanced skin lesion classification / Padhy S. [и др.] // Results in Engineering. – 2025. Т. 25. – С. 104201.

12. Akilan T., Baalamurugan K. M. Automated weather forecasting and field monitoring using GRU-CNN model along with IoT to support precision agriculture // Expert Syst Appl. – 2024. Т. 249. – С. 123468.

13. Schneider N., Lermer M., Reich C. Investigating the Performance of CNN Feature Extractor-Based LSTM and GRU variants for Time Series Classification // Procedia Comput Sci. – 2024. Т. 246. – С. 1070–1079.

14. Irving Biederman. Recognition-by-components: a theory of human image understanding. // Psychol Rev. – 1987. Т. 94(2). – С. 115–147.

15. Giakoumoglou N., Pechlivani E. M., Tzovaras D. Generate-Paste-Blend-Detect: Synthetic dataset for object detection in the agriculture domain // Smart Agricultural Technology. – 2023. Т. 5. – С. 100258.

16. Generating synthetic images for construction machinery data augmentation utilizing context-aware object placement / Lu Y. [и др.] // Developments in the Built Environment. – 2025. Т. 21. – С. 100610.

17. BlenderProc / Denninger M. [и др.] // – 2019.

18. Mask OBB: A semantic attention-based mask oriented bounding box representation for multi-category object detection in aerial images / Wang J. [и др.] // Remote Sens (Basel). – 2019. Т. 11. № 24. – С. 2930.

19. Ensembles of deep one-class classifiers for multi-class image classification / Novotny A. [и др.] // Machine Learning with Applications. – 2025. Т. 19. – С. 100621.