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

РАБОТА ПРОВЕРЕНА

Рецензент
Начальник отдела
суперкомпьютерного моделирования
НИУ ВШЭ, к.ф.-м.н. доцент

_____ П.С. Костенецкий

«___»_____ 2024 г.

ДОПУСТИТЬ К ЗАЩИТЕ

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

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка системы для фиксации перемещения рабочих на производстве на основе алгоритмов компьютерного зрения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.04.02.2024.308-1401.ВКР

Научный руководитель,
доцент кафедры СП, к.ф-м.н.
_____ Г.И. Радченко

Автор работы,
студент группы КЭ-220
_____ А.А. Ясновский

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

Челябинск, 2024 г.

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

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

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистранта
студенту группы КЭ-220
Ясновскому Андрею Александровичу,
обучающемуся по направлению
02.04.02 «Фундаментальная информатика и информационные технологии»
(магистерская программа «Машинное обучение и анализ больших данных»)

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка системы для фиксации перемещения рабочих на производстве на основе алгоритмов компьютерного зрения.
2. **Срок сдачи студентом законченной работы:** 20.05.2024 г.
3. **Исходные данные к работе**

3.1. Rojas R.A Systematic Introduction. // Springer-Verlag, March 1996, Berlin, Germany. – 1996. – 506 p.

3.2. Géron A. Hands On Machine Learning with Scikit-Learn and TensorFlow. // O'Reilly Media March 2017, Sebastopol. – 2017. – 718 p.

3.3. Ripley B. Pattern Recognition and Neural Networks – Cambrige University Press, 1st edition, 2008. – 416 p.

3.4. Lemos-Paiao A.P., Silva C.J. A New Compartmental Epidemio-logical Model with a Case Study of Portugal. // Center for Research and Development in Mathematics and Applications (CIDMA), 2020. – 16 p.

3.5. Breaking down mean average Precision (maP). [Электронный ресурс] URL: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52 (дата обращения: 10.04.2024 г.).

4. Перечень подлежащих разработке вопросов

4.1. Провести анализ предметной области.

4.2. Провести обзор научной литературы.

4.3. Проектирование архитектуры приложения.

4.4. Реализация системы.

4.5. Тестирование системы.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. Г.И. Радченко

Задание принял к исполнению А.А. Ясновский

ГЛОССАРИЙ

1. *Нейронные сети* – это один из ключевых инструментов в области искусственного интеллекта и машинного обучения. Они представляют собой вычислительные модели, вдохновленные биологическими нейронными сетями, которые составляют мозг живых существ [1].
2. *Машинное обучение* – это область искусственного интеллекта (ИИ), которая фокусируется на разработке алгоритмов и моделей, способных автоматически извлекать знания из данных и делать прогнозы или принимать решения без явного программирования на каждую отдельную задачу [2].
3. *Детектирование* – обнаружение объекта на изображении [3].
4. *Метрика* – это количественная мера, используемая для оценки качества модели в задачах машинного обучения и анализа данных. Метрики позволяют понять, насколько хорошо или плохо модель выполняет свои задачи, и служат инструментом для сравнения разных моделей и методов [3].
5. *Обучающая выборка* – часть набора данных, предназначенная для обучения модели [3].
6. *Валидационная выборка* – часть набора данных, предназначенная для промежуточного оценивания модели [3].
7. *Сверточная нейронная сеть (CNN)*– это специальная архитектура искусственных нейронных сетей [3].
8. *Тензор* – многомерная матрица, содержащая элементы одного типа данных [4]
9. *Docker* – это платформа для контейнеризации приложения [5].

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В настоящее время автоматизация производственных процессов является одним из основных трендов в промышленности. Реализация системы слежения за перемещением рабочих на производстве считается одним из ключевых шагов в этом направлении.

Реализация системы слежения за перемещением рабочих на производстве является необходимой мерой для современных предприятий, которые стремятся к оптимизации своей работы и повышению конкурентоспособности на рынке. Круглосуточное слежение за процессами, происходящими на производстве, требует финансовых затрат на обеспечение рабочих, наблюдающих за процессом через видеокамеры, а также временных затрат, связанных с ежедневным просмотром большого количества накопившегося видеоматериала и его хранения.

Решить эту проблему может помочь система, которая позволяет выделять и сохранять необходимые части видеопотока, на которых происходят конкретные действия. Это поможет оптимизировать рабочие процессы, выявить уязвимости или провести оценку полезного времени рабочих в течение рабочего дня. Данная система может помочь в решении следующих задач:

1) сокращение финансовых затрат для найма избыточной рабочей силы для отслеживания процессов на производстве;

2) оптимизация хранения видеоданных, содержащих информацию о рабочих процессах;

3) оптимизация процесса отслеживания рабочего времени сотрудников предприятия;

4) предотвращение воровства и других неправомерных действий на предприятии, а именно в технических помещениях;

5) отслеживание приезда технических служб и других наемных рабочих со стороны.

Далее в работе рассматриваются подходы для разработки подобной системы, а также приводится пример архитектуры системы для определения перемещения рабочих на предприятии.

Постановка задачи

Целью выпускной квалификационной работы является разработка системы для фиксации перемещения рабочих на производстве на основе алгоритмов компьютерного зрения. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области;
2) провести обзор научной литературы;
3) проектирование архитектуры приложения;
4) реализация системы;
5) тестирование приложения.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 54 страниц, объем списка литературы – 28 источников.

В первой главе описывается обзор предметной области. В ней приводятся основные теоретические сведения, которые пригодятся в разработке системы.

Вторая глава посвящена описанию теоретических сведений. В ней описываются архитектуры нейронных сетей в задаче компьютерного зрения.

В третьей главе описывается архитектура приложения. В ней приведены требования к системе, показана основная архитектура приложения и описаны диаграммы.

В четвертой главе описывается реализация приложения. В ней показана разработка инфраструктуры, описана реализация системы.

В пятой главе описывается тестирование приложения. В ней описывается тестирование приложения по функциональным требованиям.

В приложении приведены листинги написанного кода.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор аналогов

В настоящее время машинное обучение играет ключевую роль в обработке и анализе больших объемов данных с целью извлечения полезной информации. Эти алгоритмы находят применение в самых разных областях, включая задачи, которые, казалось бы, просты для человека. Однако, применение машинного обучения в таких задачах позволяет значительно оптимизировать процессы и улучшить точность результатов [6].

Задача распознавания объектов на изображениях является частью области компьютерного зрения [7], которая специализируется на анализе визуальных данных, таких как изображения и видео. Эти системы позволяют анализировать содержание изображений и извлекать ценные данные на основе типовых образцов.

Искусственные нейронные сети являются одним из наиболее широко используемых алгоритмов в машинном обучении. Они моделируют структуру и функциональность биологического мозга, состоящего из нейронов. Несмотря на то, что искусственные нейронные сети представляют собой лишь упрощенную модель биологического аналога, они успешно применяются для решения разнообразных задач, включая классификацию изображений.

Первоначальные модели нейронных сетей представляли собой простые нейронные модели, принимавшие на вход набор значений и выдающие соответствующий выход. В процессе обучения такие модели подбирали оптимальные веса, аналогичные синапсам в биологическом мозге. Этот процесс позволял модели обучаться на примерах и делать предсказания на основе входных данных.

1.2. Задача детекции объектов на видеопотоке

Задача детекции объектов на видеопотоке представляет собой ключевой этап в обработке видеоданных, направленный на автоматическое

обнаружение и идентификацию объектов в реальном времени. Она заключается в анализе последовательности кадров видео с целью обнаружения на них различных объектов, таких как автомобили, люди, животные и другие объекты интереса.

Основная цель детекции объектов заключается в точном определении местоположения и границ каждого объекта на кадре, а также в их классификации по заранее определенным категориям. Для достижения этой цели используются различные алгоритмы и модели машинного обучения, включая глубокие нейронные сети, такие как YOLO (You Only Look Once) [8], SSD (Single Shot MultiBox Detector) [9] и Faster R-CNN (Region-based Convolutional Neural Network) [10].

Процесс детекции объектов на видеопотоке начинается с разбиения видеопотока на отдельные кадры. Затем каждый кадр анализируется с помощью модели детекции, которая выделяет на нем области, содержащие объекты, и определяет их границы и классификацию. Полученная информация может быть дополнительно обработана для улучшения точности детекции и устранения ложных срабатываний.

Задача детекции объектов на видеопотоке имеет широкий спектр применений в различных областях, включая системы видеонаблюдения, автоматизированные системы безопасности, мониторинг транспортных потоков, медицинскую диагностику и другие. Она играет важную роль в повышении безопасности, эффективности и комфорта в различных сферах жизни и деятельности.

1.3. Обзор научной литературы

В статье [11] авторы представляют новую модель, которая основана на механизме внимания. Этот механизм обеспечивает эффективное обнаружение координат и демонстрирует отличную производительность в задаче обнаружения масок на лицах и также подойдет для обнаружения объектов на изображении. Благодаря этой архитектуре модель способна

извлекать низкоуровневые особенности изображения, что повышает точность и надежность ее работы.

На рисунке 1 представлена детализированная архитектура сети ShuffleCANet, предложенной авторами статьи [11]. Используя комбинацию шаффлблоков и механизма внимания, модель достигает баланса между высокой производительностью и эффективностью вычислений. Такая архитектура обещает значительное улучшение в области обнаружения объектов на изображениях, включая задачу детектирования лиц с масками.

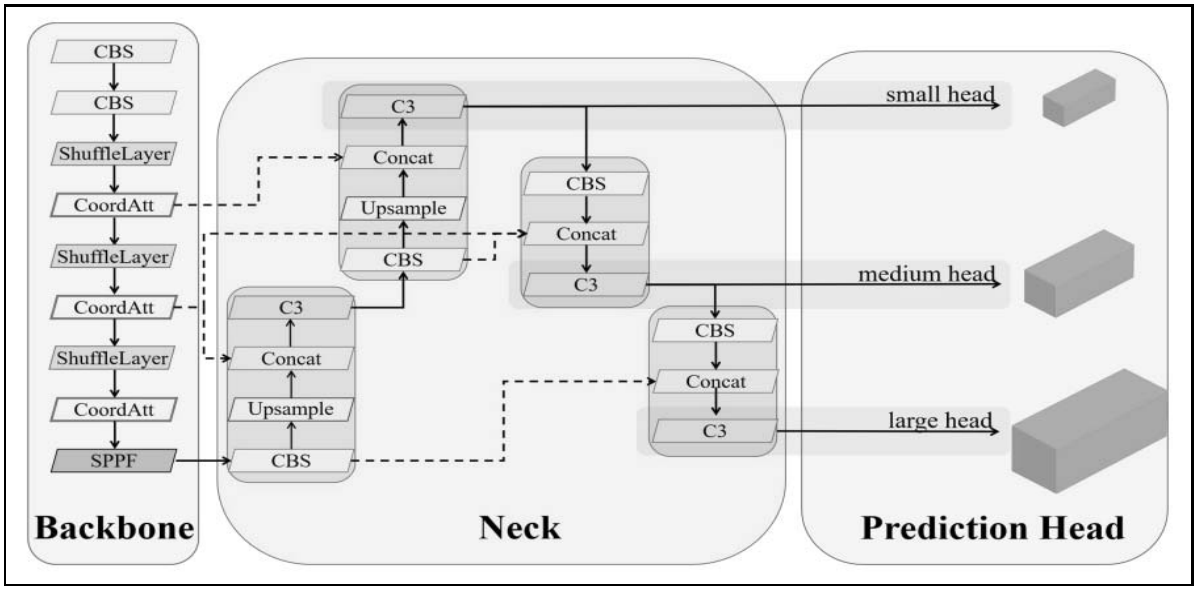

Рисунок 1 – Архитектура модели ShuffleCANet

В качестве исходных данных был взят набор данных AIZOOTech [12], содержащий 7 959 изображений. Для оценивания качества работы модели использовались метрики average precision (AP) и mean average precision (mAP).

В статье [13] авторы представляют новый алгоритм для обнаружения ношения шлемов. Предложенный ими алгоритм, основанный на модели YOLOv5 с механизмом многоканального внимания (MCA), предлагает улучшенную скорость и точность обнаружения.

Авторы отмечают, что использование механизма многоканального внимания позволяет алгоритму эффективно выделять ключевые признаки изображений, что способствует как улучшению точности, так и снижению

вычислительной сложности. Основываясь на этих результатах, алгоритм MCA-YOLOV5-Light обеспечивает баланс между скоростью и точностью обнаружения.

При решении задачи обнаружения мелких объектов, собранная информация о признаках мелких целей постепенно ослабевает по мере увеличения числа слоев сети, поэтому алгоритм легко может ошибиться и пропустить обнаружение мелких целей. В данной работе модуль MCA встроен в опорную сеть, чтобы обогатить сетевой сбор признаков. На рисунке 2 показана структурная схема YOLOV5, в которую добавлен модуль MCA.

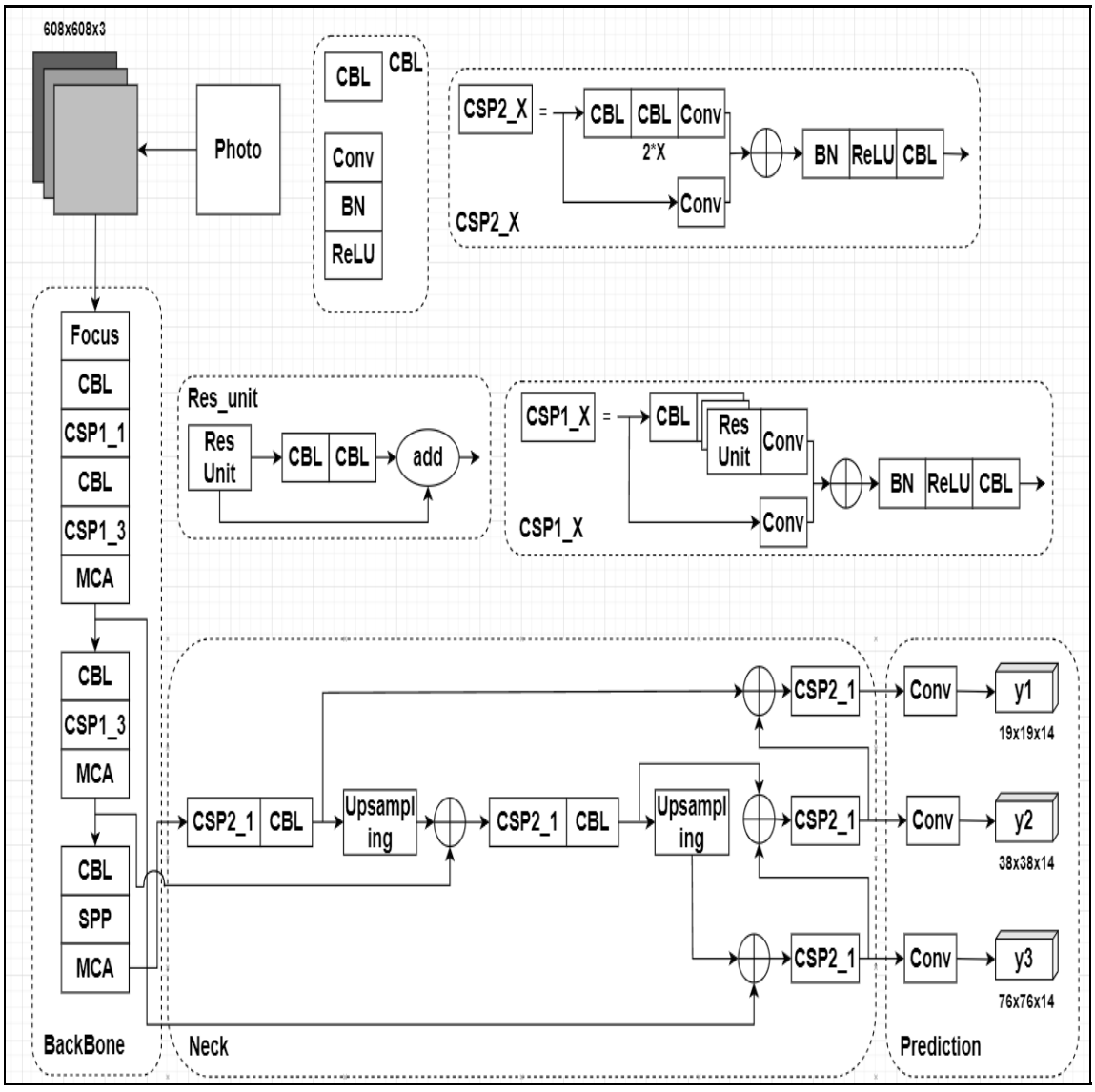

Рисунок 2 – Структура сетевой модели MCA-YOLOv5

Для оценки эффективности алгоритма использовались стандартные метрики, такие как Average Precision (AP) и mean Average Precision (mAP). Эти метрики позволяют не только оценить качество работы алгоритма в целом, но и его способность точно определять объекты с различными характеристиками. Например, при анализе обнаружения шлемов безопасности эти метрики помогают понять, насколько алгоритм правильно распознает объекты разного размера и формы.

Для анализа работы модели использовался набор данных, включающий 7 076 изображений, описанный в исследовании. Оценка эффективности алгоритма проводилась с использованием таких метрик, как точность обнаружения и средняя точность обнаружения, что позволило сравнить производительность MCA-YOLOV5-Light с альтернативными методами.

Результаты частичного обнаружения шлемов безопасности при применении алгоритма MCA-YOLOv5-Light на тестовой выборке изображений представлены на рисунке 3. Красные прямоугольники показывают обнаруженные моделью объекты со шлемами, в то время как синие указывают на головы без защитного снаряжения. По данным рисунка видно, что алгоритм способен более точно определить, надели ли сотрудники шлемы в условиях многолюдной обстановки, что подтверждает эффективность предложенного метода.

Полученные данные показали высокую точность обнаружения шлемов безопасности и других головных уборов в различных ситуациях, подтверждая отличную работу алгоритма MCA-YOLOv5-Light. Следует отметить, что модель способна успешно распознавать объекты даже в сложных условиях, включая плотные скопления людей и переменное освещение, что делает ее перспективной для применения в различных областях, где требуется точное определение объектов.

Рисунок 3 – Результаты частичного обнаружения

Статья [14] представляет собой исследование, основанное на архитектуре YOLOv5 и методах суперразрешающей реконструкции для обнаружения лиц на изображениях.

Авторы представляют инновационный подход, объединяющий YOLOv5 и суперразрешающую реконструкцию для улучшения процесса обнаружения лиц на изображениях. Использование YOLOv5 обеспечивает быструю и точную локализацию лиц, в то время как суперразрешающая реконструкция улучшает качество изображений, что помогает в сложных условиях освещения или размытости.

Архитектура метода позволяет извлекать низкоуровневые особенности изображения, что повышает точность и надежность его

работы. Это подтверждается результатами экспериментов, проведенных авторами, где предложенный метод продемонстрировал значительное улучшение производительности по сравнению с другими существующими подходами. На рисунке 4 показана архитектура модели SR-YOLOv5.

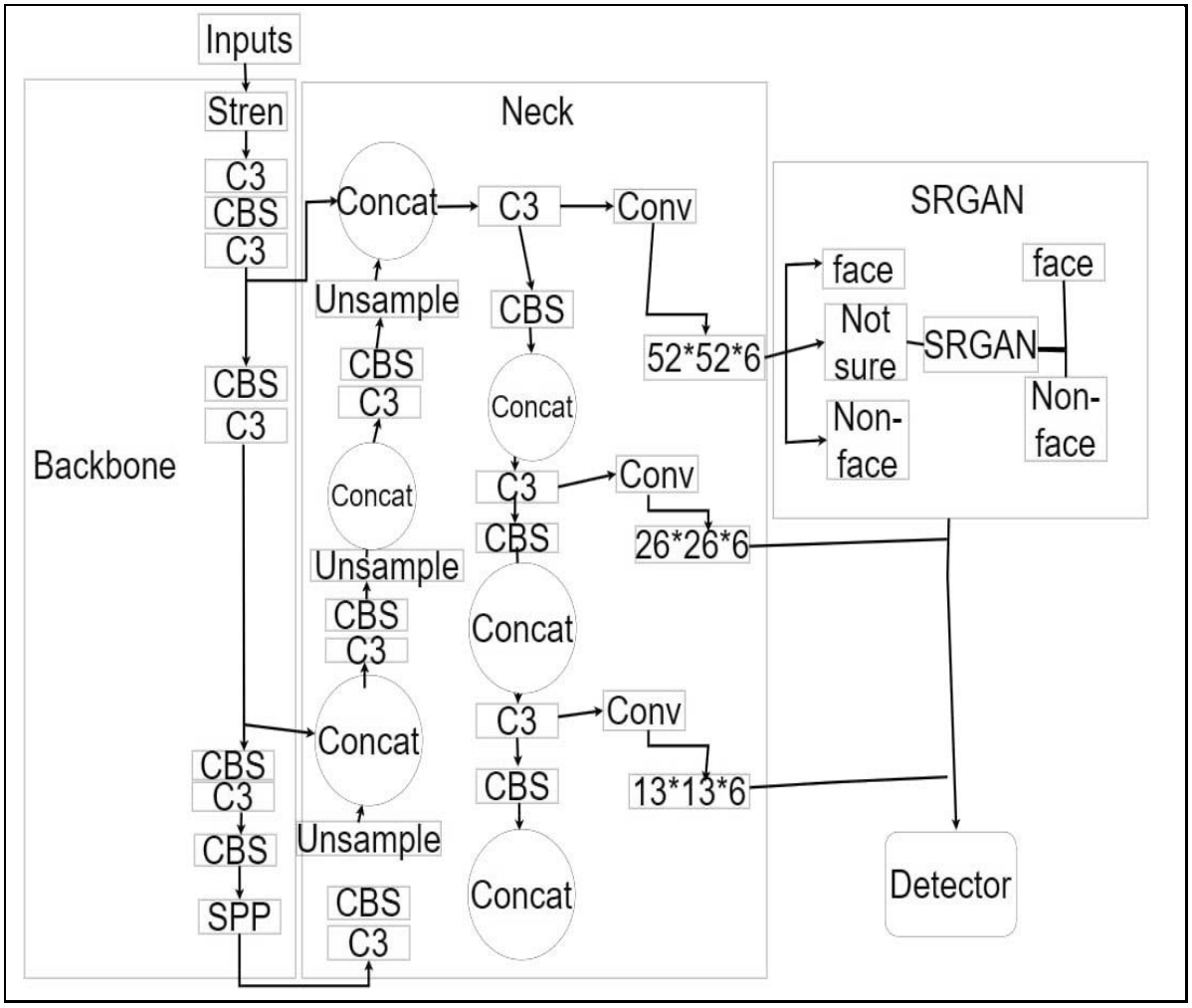

Рисунок 4 – Архитектура модели SR-YOLOv5

Одним из ключевых преимуществ предложенного метода является его способность обрабатывать изображения с высокой скоростью, что делает его пригодным для реального времени приложений, таких как системы безопасности или мониторинга.

На рисунке 5 представлено описание структуры набора данных, использованного авторами статьи для обучения модели SR-YOLOv5.

Datasets	Pictures	Faces
Wider face	32203	393703
AFW	205	473
FDDB	2845	5171
Pascal face	851	1341
IJB-A	24327	49759
MALF	5250	11931

Рисунок 5 – Описание наборов данных для модели SR-YOLOv5

На рисунке 6 представлен результат работы модели SR-YOLOv5, после всех этапов обучения.

Рисунок 6 – Результаты работы модели SR-YOLOv5

Статья [15] представляет собой исследование, направленное на обнаружение малых объектов в реальном времени на производственных площадках с использованием архитектуры ISR-YOLOv4. Методика авторов статьи включает в себя обнаружение малых объектов, что является важным аспектом для обеспечения безопасности на производственных площадках.

В статье подробно описывается модель ISR-YOLOv4, которая является комбинацией YOLOv4 и с рядом других методов, таких как ISR

(Incremental Strategy Refinement). Архитектура ISR-YOLOv4 оптимизирована для работы в условиях ограниченных вычислительных ресурсов и высокой скорости обработки изображений. Это позволяет модели обнаруживать малые объекты с высокой точностью и скоростью, что критически важно для обеспечения безопасности на производственных линиях. На рисунке 7 можно увидеть архитектуру модели ISR-YOLOv4.

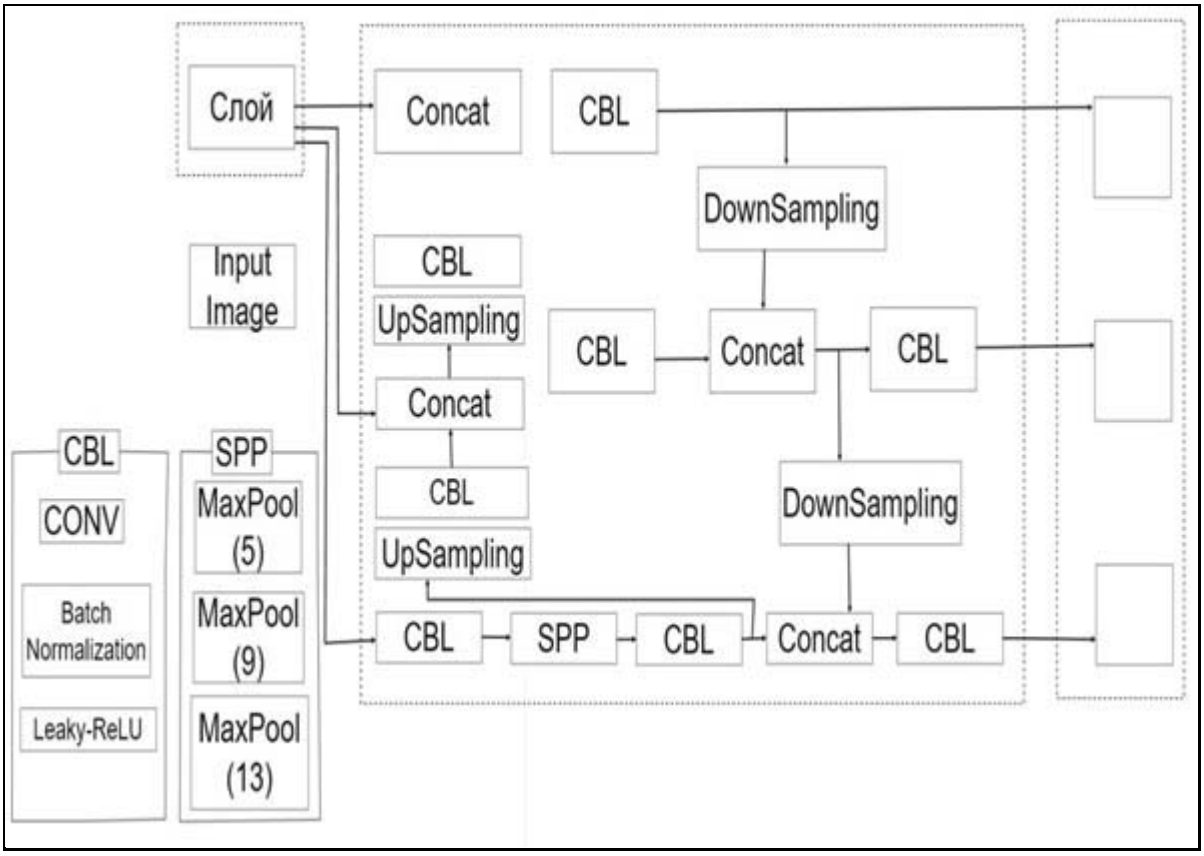

Рисунок 7 – Архитектура модели ISR-YOLOv4

Эксперименты, проведенные авторами статьи, демонстрируют высокую производительность и точность модели ISR-YOLOv4 на реальных данных с производственных площадок. Модель успешно справляется с обнаружением малых объектов в реальном времени, что подтверждает ее эффективность и практическую применимость.

Дополнительно, авторы проводят анализ результатов работы модели на различных наборах данных и представляют сравнение с другими существующими методами обнаружения малых объектов, что дополняет

общее понимание о производительности модели в контексте текущих достижений в области.

На рисунке 8 представлен результат работы модели ISR-YOLOv4, после всех этапов обучения.

Рисунок 8 – Результаты работы модели ISR-YOLOv4

1.4. Обзор инструментов

PyTorch

Библиотека PyTorch [16], предоставляющая широкий спектр инструментов для работы с нейронными сетями, включая область компьютерного зрения, обработки естественного языка и текстов. Встроенные методы обеспечивают распределенное обучение, оптимизацию и повышенную производительность.

Этот фреймворк известен своим высоким уровнем гибкости при проектировании нейронных сетей, что позволяет разработчикам создавать и настраивать модели с учетом специфических требований и задач. Одним из преимуществ PyTorch является возможность взаимодействия с

графическими процессорами (GPU) без необходимости в дополнительных затратах на реализацию соответствующих функций.

Кроме того, PyTorch пользуется широкой популярностью, что обусловлено формированием большого сообщества разработчиков вокруг него. Благодаря этому, в интернете доступно множество информации по применению PyTorch к различным задачам, что делает его востребованным инструментом в области машинного обучения и исследований.

Одной из ключевых концепций в PyTorch являются тензоры, которые являются основными элементами для проведения вычислений. Тензоры позволяют значительно ускорить выполнение различных математических операций и обеспечивают безопасность за счет отсутствия поддержки динамической типизации.

OpenCV

OpenCV (Open Source Computer Vision Library) [17] является мощным инструментарием для работы с изображениями и видео в области компьютерного зрения. Библиотека предоставляет разнообразные функции и алгоритмы для анализа, обработки изображений, распознавания объектов, трекинга движущихся объектов и многого другого.

Основные особенности OpenCV включают в себя широкий выбор инструментов, обеспечивающих фильтрацию, сегментацию, детекцию объектов и многое другое. Библиотека также известна своей высокой производительностью и оптимизацией, что позволяет обрабатывать большие объемы данных эффективно даже при работе с многопоточностью и параллельными вычислениями.

Гибкая архитектура OpenCV позволяет легко интегрировать ее в различные приложения и среды разработки. Простой и понятный интерфейс программирования приложений (API) делает работу с библиотекой доступной для разработчиков, что позволяет создавать сложные системы компьютерного зрения.

Кроме того, OpenCV поддерживает активное сообщество разработчиков, что способствует обмену опытом и знаниями, а также разработке новых алгоритмов и приложений. Это обеспечивает доступ к обширной базе знаний и возможность получить поддержку и помощь от опытных специалистов.

Архитектура OpenCV организована вокруг модульной структуры, которая позволяет легко интегрировать различные компоненты и функциональные блоки. Библиотека предоставляет множество модулей для работы с изображениями и видео, включая функции для чтения и записи файлов, фильтрации, сегментации, а также распознавания и классификации объектов.

OpenCV включает в себя также модули машинного обучения, такие как метод опорных векторов, случайные леса, нейронные сети и другие, которые могут быть использованы для решения различных задач компьютерного зрения. Кроме того, библиотека обеспечивает функции для работы с потоками данных, что позволяет эффективно обрабатывать видеопотоки в реальном времени.

OpenCV широко применяется в различных областях, включая медицину, робототехнику, автоматизацию производства, безопасность, анализ поведения и академические исследования. Ее функциональность используется для решения задач распознавания лиц, детектирования объектов, мониторинга и трекинга движущихся объектов, анализа медицинских изображений и многое другое.

TensorFlow

TensorFlow [18] – это открытая библиотека для машинного обучения и глубокого обучения, разработанная компанией Google. Она предоставляет инструменты и ресурсы для построения и обучения различных моделей машинного обучения, включая нейронные сети, как для исследовательских, так и для коммерческих целей.

Одной из ключевых особенностей TensorFlow является его гибкость и масштабируемость. Библиотека позволяет создавать и обучать модели на различных уровнях абстракции, начиная от низкоуровневого определения графов вычислений и операций до высокоуровневых API, таких как Keras, который упрощает процесс создания и обучения моделей.

Основные преимущества TensorFlow включают гибкость, масштабируемость, кроссплатформенность, богатую экосистему и активное сообщество. TensorFlow поддерживает различные архитектурные решения, включая полностью связанные сети, сверточные нейронные сети (CNN), рекуррентные нейронные сети (RNN) и их комбинации. Это позволяет исследователям и разработчикам выбирать подходящую архитектуру для своих конкретных задач.

Библиотека также предоставляет инструменты для распределенного обучения моделей на множестве процессоров и устройств, включая графические процессоры (GPU) и процессоры сопроцессоров (TPU). Это позволяет обрабатывать большие объемы данных и ускорять обучение моделей.

Богатая экосистема TensorFlow включает в себя библиотеки расширений, инструменты визуализации и множество предварительно обученных моделей, доступных через TensorFlow Hub.

Наконец, TensorFlow является проектом с открытым исходным кодом, что позволяет разработчикам и исследователям вносить свой вклад в развитие библиотеки. Его активное сообщество обеспечивает поддержку, обмен опытом и доступ к различным ресурсам, таким как обучающие материалы и примеры кода.

1.5. Сравнение инструментов

Рассмотрим сравнение трех инструментов: PyTorch, TensorFlow и OpenCV, с учетом требований системы.

PyTorch, OpenCV и TensorFlow – все они являются популярными библиотеками в мире машинного обучения и компьютерного зрения, но у них разные особенности и применения.

Можно выделить следующие преимущества библиотеки PyTorch.

1. Гибкий фреймворк для обучения нейронных сетей, обладает высокой степенью контроля и удобным интерфейсом.

2. Высокая производительность и эффективность, оптимизированные для различных аппаратных платформ.

3. Поддерживает динамические вычисления и широко используется в научных исследованиях и разработке новых алгоритмов машинного обучения.

4. Мощное сообщество и множество ресурсов для обучения и поддержки.

К недостаткам данной библиотеки можно отнести.

Несмотря на широкие возможности, требует больше усилий для интеграции с обработкой изображений и видео по сравнению с специализированными библиотеками, такими как OpenCV.

К преимуществам TensorFlow можно отнести следующее.

1. Крупнейший фреймворк для машинного обучения с широкой экосистемой инструментов для обучения, развертывания и управления моделями.

2. Высокая производительность и эффективность, оптимизированные для различных аппаратных платформ.

3. Простое масштабирование и поддержка для облачных вычислений.

К недостаткам данной библиотеки можно отнести.

Более сложный синтаксис и настройка, что может затруднить начальную настройку и эксплуатацию.

OpenCV представляет собой инструмент со следующими преимуществами.

1. Специализированный инструмент для обработки изображений и видео с богатым набором алгоритмов и функций.

2. Простота использования и интеграции, особенно для задач компьютерного зрения.

3. Высокая производительность и эффективность при обработке видеопотоков в реальном времени.

К недостаткам данной библиотеки можно отнести.

1. Ограниченные возможности в области глубокого обучения по сравнению с PyTorch и TensorFlow.

2. В некоторых случаях может потребоваться дополнительная интеграция с другими библиотеками для использования специализированных моделей глубокого обучения, таких как YOLOv5.

Исходя из требований системы, особенно учитывая обработку видеопотоков и обнаружение объектов, OpenCV представляется наиболее подходящим инструментом. Его специализация на обработке изображений и видео, в сочетании с высокой производительностью и легкой интеграцией, делает его идеальным выбором.

Выводы по первой главе

В первой главе квалификационной работы был проведен обзор научной литературы, в ходе которого были рассмотрены основные аспекты проблематики распознавания объектов на изображениях и видеопотоках.

Проведен анализ различных методов и алгоритмов, таких как YOLOv5, YOLOv4, а также рассмотрение применения инструментов, таких как PyTorch, TensorFlow и OpenCV, позволяет сделать вывод о том, что существует широкий спектр подходов и инструментов для решения задачи детекции объектов на видеопотоках.

Исходя из проведенного обзора, можно заключить, что дальнейшее исследование и разработка в данной области имеют важное значение и могут привести к созданию более точных, эффективных и масштабируемых систем для обнаружения и классификации объектов в реальном времени.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. Сверточные нейронные сети

Сверточные нейронные сети применяются для обработки данных с сеточной топологией, включая временные ряды и изображения. Этот вид сетей широко используется на практике и демонстрирует устойчивость к сдвигам изображений, что делает их привлекательным выбором для данной задачи [19].

В сверточных нейронных сетях применяется математическая операция свертки, которая включает в себя матричное перемножение входных данных и ядра свертки, что в конечном итоге порождает карту признаков. Стандартная архитектура сверточного слоя включает три основные компоненты, их структура изображена на рисунке 9. При обработке изображений и выявлении объектов выходы из сверточных слоев проходят через нелинейную функцию активации, что позволяет сети лучше выявлять и адаптироваться к различным признакам и структурам в данных.

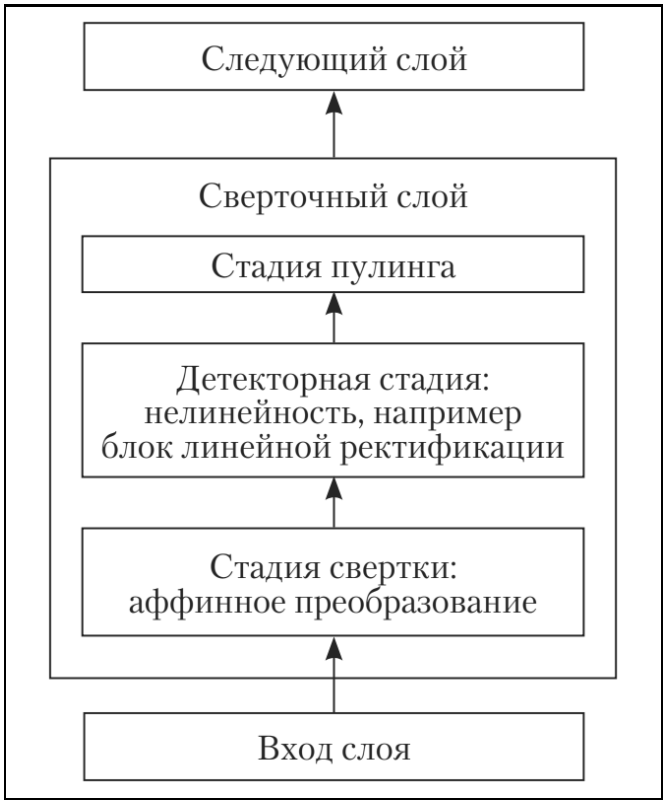

Рисунок 9 – Структура сверточного слоя

На этапе пулинга происходит замена выходных данных сети агрегированной статистикой близлежащих выходов. Например, операция максимального пулинга возвращает максимальное значение в заданной окрестности, что способствует созданию приблизительной инвариантности к небольшим смещениям входных данных и обеспечивает устойчивость к небольшим сдвигам входа.

Эта характеристика позволяет сетям лучше адаптироваться к различным условиям и разнообразным данным, что делает их востребованными в широком спектре задач компьютерного зрения. В частности, она обеспечивает эффективное распознавание объектов на изображениях в условиях изменяющегося освещения, различных углов обзора или частичных перекрытий, что является ключевым для успешного применения систем компьютерного зрения в реальных условиях.

Описанная структура сверточных нейронных сетей является эффективным инструментом для решения различных задач в области компьютерного зрения, что объясняет их широкое применение в практике.

2.2. Топология YOLOv5

Детектирование в архитектуре YOLOv5 построено по принципу одностадийного детектора, который состоит из backbone, neck и head. Приведенные ключевые части следуют друг за другом, представляя некий конвеер, решают определенные задачи на каждом уровне. По итогу работы нейронной сети с подобной архитектурой формируется полная информация об объектах на изображении, включая детектирование границ объектов и определения их типов.

На рисунке 10 представлена архитектура модели YOLOv5, содержащая описание всех ключевых слоев нейронной сети.

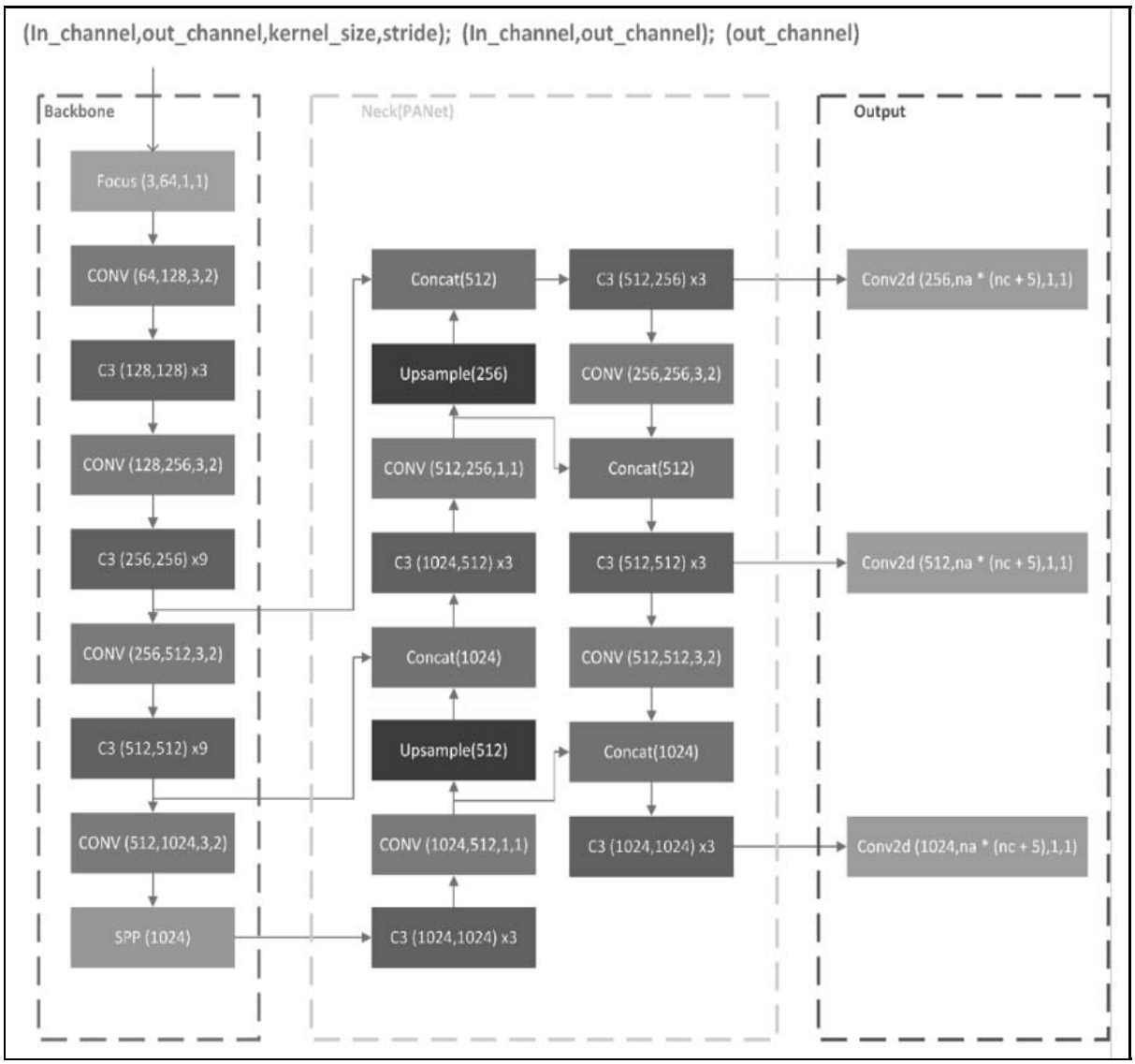

Рисунок 10 – Архитектура YOLOv5

Backbone

Backbone является фундаментальным элементом в архитектуре YOLOv5, который выполняет роль основного анализатора изображения. По сути, это первый шаг в процессе детектирования, где изображение проходит через серию специализированных слоев для подготовки данных к дальнейшему анализу. Backbone может быть представлен как система фильтров, которая сканирует изображение для выявления основных признаков и структур, которые могут указывать на наличие объектов.

После первичной подготовки изображения backbone разбивает его на ряд первичных карт признаков. Это можно сравнить с разложением

сложного пазла на отдельные элементы, каждый из которых отображает определенный аспект или деталь изображения. Для этого используются сверточные слои, которые применяют различные фильтры для выявления границ, текстур и других важных характеристик. Эти первичные карты признаков служат важным исходным материалом для последующего этапа обработки и анализа внутри нейронной сети.

После создания первичных карт backbone использует сверточные слои для выявления более сложных и высокоуровневых признаков. Это процесс абстракции, где на основе базовых признаков формируются более сложные структуры и паттерны, такие как формы объектов или их контекст в сцене. Эта стадия крайне важна для последующего этапа детектирования, так как именно на основе этих высокоуровневых признаков модель будет делать выводы о наличии и расположении объектов на изображении.

Этот процесс можно сравнить с анализом картинки пазла: backbone разбирает изображение на части и определяет, какие из них соответствуют кусочкам пазла, а какие нет. Затем он объединяет эти кусочки, чтобы сформировать полную картину, которую модель может интерпретировать для определения объектов на изображении. Таким образом, backbone является ключевым этапом в процессе детектирования, обеспечивая модели достоверную информацию для последующего анализа.

Neck

Это часть нейронной сети, которая получает признаки высокого уровня информации от backbone, предшествующего слоя или модуля нейронной сети, обычно отвечающего за извлечение более общих характеристик из входных данных. Neck играет роль посредника, который принимает эти сложные и абстрактные признаки и готовит их для дальнейшей обработки. Используя полученные признаки, Neck применяет нейронную сеть с архитектурой, такой как PAN (Personal Area Network), для формирования трех карт признаков разных размерностей. Карты признаков представляют собой различные слои данных, каждый из которых содержит

информацию разной степени детализации или абстракции. Это позволяет сети анализировать входные данные на нескольких уровнях абстракции одновременно.

Таким образом, Neck выполняет важную функцию в архитектуре нейронной сети, помогая эффективно адаптировать и представить входные данные для более глубоких слоев сети. Формирование трех карт признаков разных размерностей обогащает информацию, извлеченную из backbone, делая представление данных более многоуровневым и способствуя более точному анализу и распознаванию образов.

Кроме того, Neck является ключевым компонентом для обеспечения масштабируемости и гибкости архитектуры нейронной сети. Его способность адаптировать признаки разного уровня абстракции позволяет сети эффективно работать с различными типами входных данных и различными задачами, такими как обнаружение объектов, сегментация изображений или классификация объектов. Это делает Neck важным элементом в разработке и оптимизации современных алгоритмов глубокого обучения для компьютерного зрения.

Архитектура PAN показана на рисунке 11.

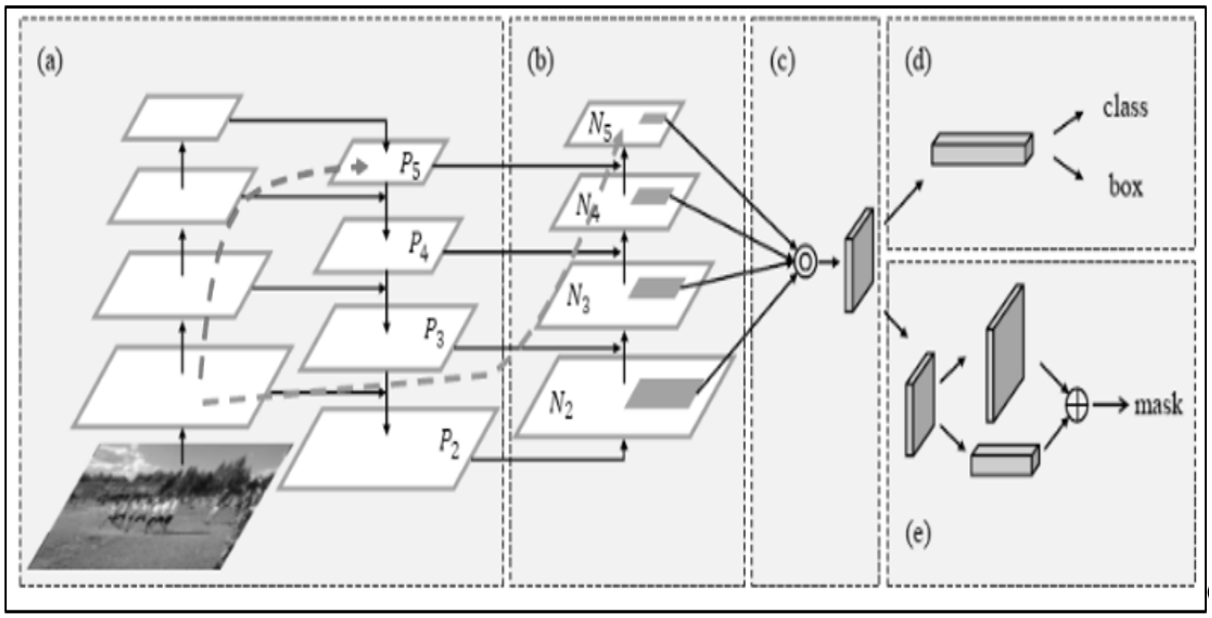

Рисунок 11 – Архитектура PAN

Head

Head в архитектуре YOLOv5 играет ключевую роль в финальной части процесса детектирования объектов. После того как изображение проходит через backbone и neck, где извлекаются различные признаки и обрабатываются, эти признаки передаются в head для конечного анализа. Head работает как детектив, который на основе этих признаков пытается точно определить, какие объекты находятся на изображении и где они расположены.

Head имеет три выхода, каждый из которых представляет собой своего рода окно в процессе анализа. Каждый из этих выходов обрабатывает информацию о разных аспектах объектов на изображении. На конце каждого выхода расположены как сверточные слои, так и полносвязные слои, которые обрабатывают признаки для генерации конечных предсказаний. Эти слои смотрят на изображение под разными углами и масштабами, что позволяет модели быть более точной и устойчивой к различным условиям и типам объектов. Это обеспечивает более высокую точность и надежность в определении и распознавании объектов на изображениях, что делает YOLOv5 востребованным инструментом в области компьютерного зрения.

Выводы по второй главе

Во второй главе были обобщены ключевые теоретические аспекты, касающиеся сверточных нейронных сетей (CNN) и особенностей их топологии. В частности, были рассмотрены принципы работы CNN, их способность к извлечению иерархических признаков из изображений, а также основные архитектурные элементы, такие как сверточные слои, слои подвыборки, слои активации и полносвязные слои. Более того, в этой главе была подробно описана топология нейронной сети YOLOv5. А также были описаны архитектурные особенности и особенности в производительности нейронной сети.

3. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

3.1. Требования к системе «Окошко»

Проектируемая система (система «Окошко») предназначена для контроля перемещения рабочих на производстве. Владелец предприятий нуждался в эффективном мониторинге рабочих процессов. В частности, ему было важно обеспечить выполнение работы в определенное время, а также предотвратить кражи в помещении.

Разрабатываемая система «Окошко» должна обеспечить непрерывный мониторинг производственных процессов, анализируя видеопоток с помощью компьютерного зрения. С ее помощью пользователь должен точно определять время начала и окончания работ, а также автоматически реагировать на подозрительные ситуации, такие как несанкционированный доступ к хранилищам.

Для достижения эффективной работы с приложением и обеспечения удобства пользования необходимо определить функциональные и нефункциональные требования.

Функциональные требования

1. Пользователи должны иметь возможность создавать отдельные участки для удобного управления своими предприятиями.

2. Пользователь должен иметь возможность добавлять неограниченное количество камер для каждого предприятия, при условии доступности физической камеры видеонаблюдения на объекте.

3. Пользователи должны иметь возможность настройки параметров слежения. Эта функция позволит адаптировать систему под конкретные нужды и требования пользователей, обеспечивая более точное и целевое слежение.

4. Пользовательская функциональность должна предоставлять возможность просмотра видео, на которых зафиксирован момент срабатывания системы компьютерного зрения. Это позволит пользователям

наблюдать за работой системы в реальных условиях и контролировать ее эффективность.

Нефункциональные требования

1. Система должна быть написана на языках программирования Python и JavaScript. Это обеспечит использование современных и популярных технологий, что облегчит разработку и поддержку системы, а также позволит привлекать к проекту более широкий круг специалистов.

2. Система должна быть реализована с использованием модульной архитектуры. Модульный подход к построению системы обеспечит ее гибкость и удобство в поддержке, а также упростит процесс внесения изменений и добавления новых функций. Такая архитектура позволяет легко заменять или обновлять отдельные компоненты системы без необходимости полной переработки всего приложения.

3. Система должна обеспечивать графический пользовательский интерфейс. Наличие интуитивно понятного интерфейса повысит удобство использования системы для конечных пользователей, сделает ее доступной для людей с различным уровнем технической подготовки и поможет минимизировать количество ошибок при работе с системой.

4. Необходима поддержка обработки видео в различных форматах, включая MP4, AVI и другие.

На основе выделенных функциональных требований была составлена диаграмма вариантов использования системы «Окошко» (рисунок 12) с использованием UML (Unified Modeling Language) – универсального языка моделирования, предназначенного для описания, визуализации и документирования различных аспектов программных систем. UML позволяет разработчикам и пользователям получать четкое представление о функциональности и структуре системы, облегчая процесс ее проектирования и дальнейшего развития. Диаграмма вариантов использования служит важным инструментом для коммуникации между

техническими специалистами и бизнес-пользователями, обеспечивая единое понимание целей и возможностей системы.

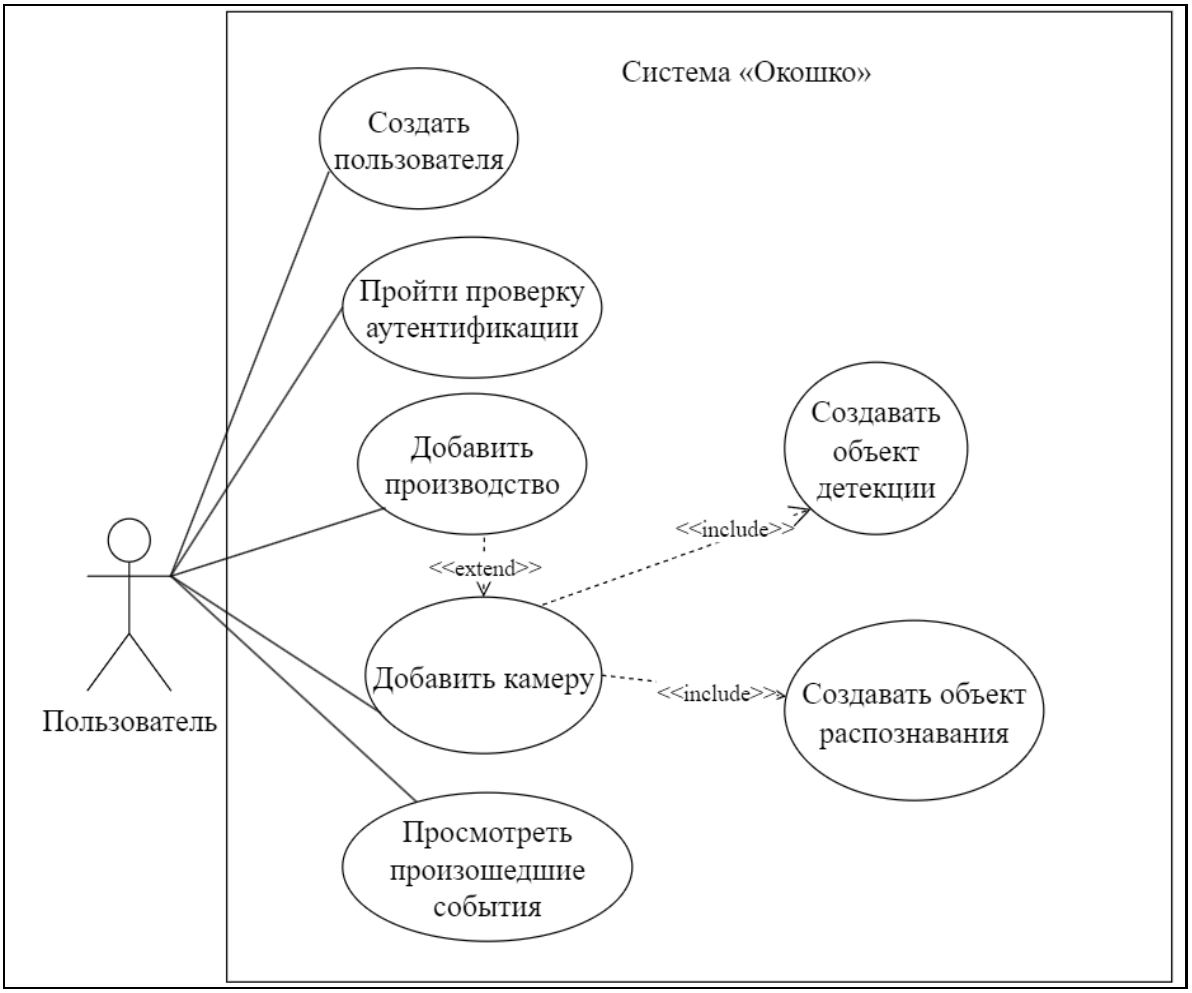

Рисунок 12 – Диаграмма вариантов использования системы «Окошко»

Актер системы – «Пользователь», который может быть как обычным сотрудником, так и администратором производства, взаимодействует с системой через ряд функциональных возможностей.

1. Вариант использования «Создать пользователя». Пользователь может создать нового пользователя, которому будет выдан доступ к системе по логину и паролю, который был указан при создании.

2. Вариант использования «Пройти аутентификацию». Пользователь перед тем, как начать пользоваться системой, должен вести свой логин и пароль, чтобы система провести аутентификацию пользователя.

3. Вариант использования «Пройти аутентификацию». Пользователь перед тем, как начать пользоваться системой, должен вести свой логин и пароль, чтобы система провести аутентификацию пользователя.

4. Вариант использования «Добавить производство». Пользователь имеет возможность зарегистрировать новое производство в системе. Это действие приводит к созданию новой сущности в базе данных, которая может быть связана с неограниченным количеством видео наблюдательных камер. Вся информация о добавленном производстве включается в специализированный реестр и сохраняется через API.

5. Вариант использования «Добавить камеру». Пользователь может установить новую видеокамеру, которая будет автоматически интегрирована в систему управления производством. Вся необходимая информация о видеокамере, включая технические параметры и местоположение, заносится в базу данных, обеспечивая легкость доступа и управления. Далее на стороне приложения открывается видео поток, где ожидается событие по типу, который выбрал пользователь.

6. Вариант использования «Просмотреть произошедшие события». Пользователь может получить доступ к полной истории событий и инцидентов на производстве, что позволяет проводить анализ происшествий и мониторинг текущей ситуации. Эта функция предоставляет не только информацию о времени и месте каждого события, но и видео отчет.

7. Вариант использования «Создать объект детекции». Пользователь при добавлении камеры, может выбрать функционал камеры. Будет выбран алгоритм, который будет следить за движением на камере.

8. Вариант использования «Создать объект распознавания». Пользователь при добавлении камеры, может выбрать функционал камеры. Будет выбран алгоритм, который будет распознавать человека в спецодежде на камере.

3.2. Описание архитектуры системы «Окошко»

Система спроектирована на базе трехуровневой клиент-серверной архитектуры. Основной целью архитектуры является разделение функциональности на отдельные слои, каждый из которых отвечает за определенный этап взаимодействия с другими слоями для обеспечения комплексного решения задачи.

Первый слой «Слой UI» отвечает за предоставление пользователю графического интерфейса для работы с системой. Этот слой играет ключевую роль в пользовательском опыте, обеспечивая удобство и понятность взаимодействия с функционалом системы.

Второй слой «Слой обработки данных» расположен на серверной части системы, и обеспечивает сбор и обработку данных, а также обслуживание запросов, приходящих со слоя UI. Здесь располагается модуль камер, который занимается обработкой видеопотока, генерацией событий и взаимодействием с моделью видеофиксации событий. Важным аспектом этого слоя является наличие очереди сообщений, которая обеспечивает асинхронное взаимодействие между различными модулями системы. После добавления камеры пользователем системы, начинается анализ потока видеоданных и ожидание указанного события. Когда происходит детекция события, начинается запись видео отрывка, который после окончания события записывается в слой хранения .После чего пользователь может увидеть информацию о событии в слое UI.

Третий слой «Слой хранения» отвечает за хранение информации о системе, загрузку и выгрузку данных. Он является фундаментом системы, обеспечивая безопасное хранение и доступность всей информации. Здесь также располагается файловое хранилище для изображений и других медиафайлов, обеспечивающее их безопасное хранение и доступность. Благодаря этому слою система может эффективно управлять объемом данных, обеспечивая их сохранность и быструю доступность при необходимости.

Диаграмма компонентов системы «Окошко» представлена на рисунке 13.

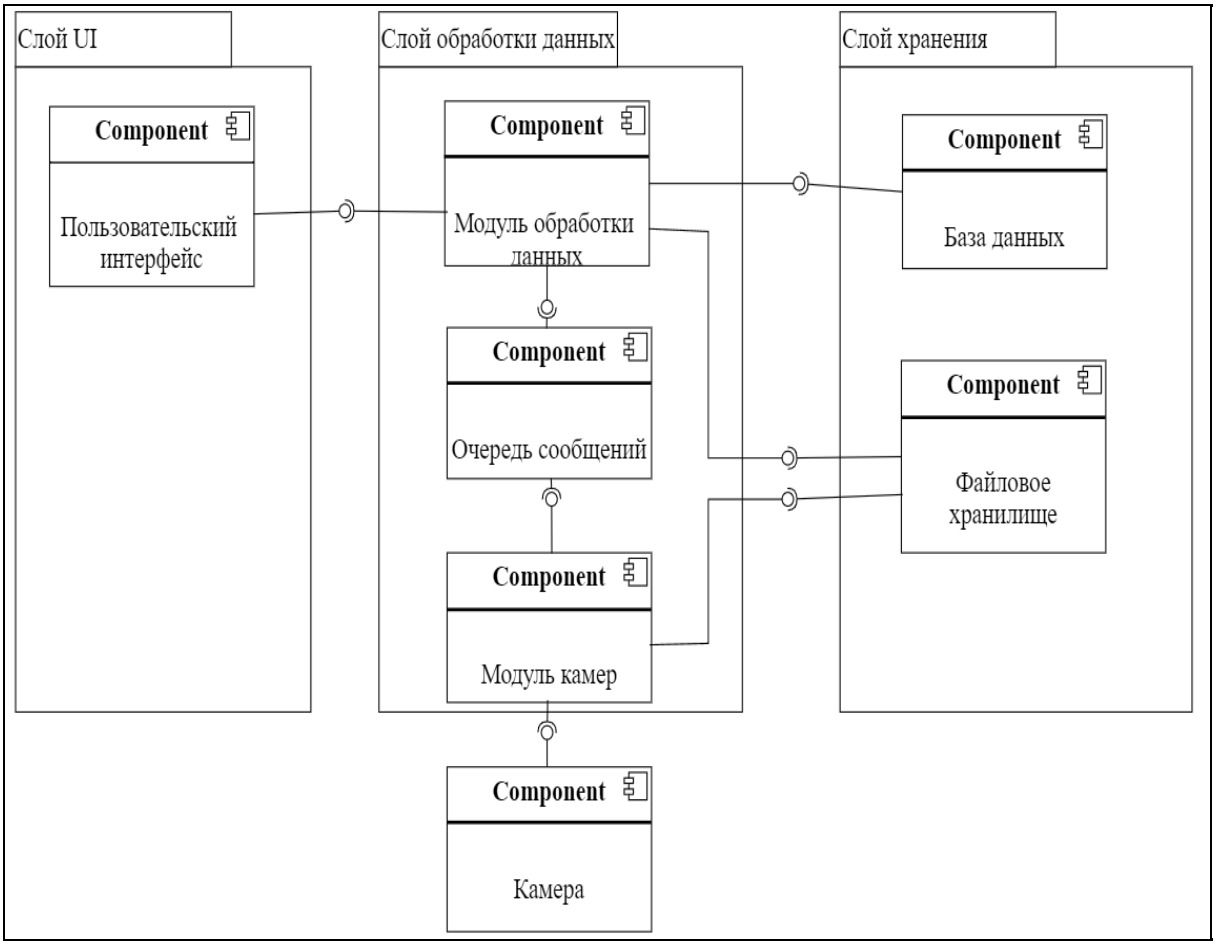

Рисунок 13 – Диаграмма компонентов системы «Окошко»

Представленная на рисунке 13 диаграмма компонентов системы «Окошко» состоит из следующих компонентов.

1. Пользовательский интерфейс – этот компонент является точкой взаимодействия между пользователем и системой. Он отвечает за прием запросов от пользователей и их обработку на основе входящих данных. Пользовательский интерфейс обеспечивает интуитивное управление функциями системы и визуализацию обратной связи.

2. Модуль обработки данных – ключевой элемент системы, который служит центром обработки данных. API принимает запросы от пользователей, обрабатывает их, выполняет необходимые операции

(создание, обновление, удаление, чтение данных) и управляет передачей обработанных данных.

3. Модуль камер – компонент, задачей которого является обнаружение и распознавание объектов, а также детекция событий. Модуль принимает видеопоток от камер, обрабатывает его с использованием алгоритмов нейронных сетей и выводит результаты в виде видеофиксации событий с информацией о распознанных объектах.

4. Камера – компонент системы, предназначенный для предоставления видеопотока. Установленные в ключевых местах, камеры передают сырое видео на обработку в Модуль камер, предоставляя системе необходимые данные для последующего анализа и принятия решений. Одной из ключевых характеристик камер является их разрешение и качество видеозаписи.

5. Файловое хранилище – является ключевым компонентом в информационных системах, обеспечивающим эффективное хранение и управление видеофайлами. Оно предоставляет структурированное пространство для хранения больших объемов данных, позволяя эффективно организовывать, индексировать и обеспечивать доступ к видеоматериалам.

6. База данных – является фундаментальным компонентом любой информационной системы, отвечающим за эффективное хранение и управление данными, необходимыми для функционирования системы. Она обеспечивает структурированное хранение информации в удобном для доступа.

Выводы по третьей главе

Разработанная архитектура системы представляет собой тщательно спроектированный набор модулей, нацеленных на обеспечение высокой производительности, точности и надежности при обработке и анализе видеоданных. Она учитывает особенности работы с большим объемом данных, обеспечивая эффективное и быстрое выполнение задач анализа и принятия решений.

4. РЕАЛИЗАЦИЯ

4.1. Программные средства реализации

Для разработки программной части системы «Окошко», был выбран высокоуровневый язык программирования Python версии 3.8.13 и высокоуровневый язык программирования JavaScript версии ES6. Оба языка предоставляют широкие возможности для создания современных веб-приложений, обладают богатым набором инструментов и библиотек, что обеспечивает удобство и эффективность разработки.

В процессе разработки использовали редактор исходного кода Visual Studio Code, который предоставляет удобную и мощную среду разработки с широким спектром функциональных возможностей и инструментов для работы с кодом.

Для обеспечения изолированной среды разработки и управления зависимостями использовали платформу контейнеризации Docker. Контейнеры разработки содержат все необходимые компоненты, включая зависимости, инструменты и настройки, что обеспечивает единое и удобное окружение для работы над проектом.

Серверная часть приложения была реализована с использованием фреймворка Flask [21] версии 2.0.3. Flask предоставляет простой и гибкий способ создания веб-приложений на языке Python, что делает его идеальным выбором для разработки серверных компонентов приложения. Для хранения видеоданных использовался сервисом Amazon S3 [22]. Для хранения пользовательской информации использовался Postgres [23] версии 14.2. Для удобного общения между модулями использовался сервис RabbitMQ [24] версии 3.9.

Пользовательская часть приложения была реализована с использованием библиотеки ReactJs [25] версии 17.0.2. ReactJs позволяет создавать динамичные и интерактивные пользовательские интерфейсы, что делает его одним из наиболее популярных инструментов для разработки фронтенда веб-приложений.

Для работы с нейронными сетями использовали следующие библиотеки: Flask версии 2.0.3 и OpenCV версии 4.6.0.66. Эти библиотеки позволили интегрировать нейронные сети в систему и обеспечить обработку видеопотока в реальном времени.

Для обучения основной модели использовались веса, полученные из модели YOLOV5. Это позволило нам достичь высокой точности и эффективности в обнаружении объектов на видеопотоке.

Для управления всеми зависимостями и пакетами в проекте использовали библиотеку pip версии 24.0, которая является стандартным инструментом для установки и управления пакетами Python.

4.2. Реализация пользовательской части

Основным инструментом для разработки пользовательской части выступила библиотека ReactJS. Этот выбор был обусловлен высокой популярностью ReactJS в среде разработки интерактивных веб-приложений, а также его способностью обеспечивать быструю перерисовку компонентов интерфейса. Использование ReactJS позволило нам создать эффективный и динамичный пользовательский интерфейс, который легко масштабировать и поддерживать.

Для обеспечения взаимодействия между клиентской и серверной частями приложения были разработаны специализированные интерфейсы и функции. Эти компоненты занимаются обработкой данных, поступающих от сервера и отправляемых на сервер, что включает в себя обработку текстовых данных и видео. Каждый тип данных обрабатывается с помощью отдельно созданной функции, что значительно упрощает процесс интеграции и последующую поддержку системы.

Важной частью клиентской логики является возможность пользователями создавать и управлять предприятиями через веб-интерфейс на рисунке 14. Пользователи могут задавать необходимые параметры и настройки для каждого предприятия.

Рисунок 14 – Добавление предприятия

Далее в интерфейсе пользователя представлена возможность просмотра только что созданного предприятия вместе с его уникальным токеном. Токен играет ключевую роль в архитектуре системы, так как используется для защищенного общения между камерами и серверной частью приложения. Это обеспечивает безопасный обмен данными и контроль доступа к информации.

Кроме того, пользователь имеет возможность добавлять камеры к конкретному предприятию. Пример созданного приятия можно увидеть на рисунке 15.

Рисунок 15 – Созданное предприятие

Для добавления новой камеры в систему, пользователю необходимо предоставить несколько ключевых данных.

Во-первых, необходимо указать URL-адрес, с которого будет транслироваться поток видео (stream). Это обеспечит прямую связь с

камерой и позволит серверной части приложения обрабатывать видеопоток в реальном времени.

Во-вторых, требуется задать имя камеры. Это имя помогает отличать одну камеру от другой в системе управления, облегчая тем самым процесс навигации и управления множеством подключенных устройств.

В-третьих, пользователь может выбрать тип события, которое необходимо детектировать с помощью данной камеры. Это может быть, например, детекция движения, распознавание лиц или другие специфические виды анализа видео. Выбор типа события позволяет адаптировать работу системы под конкретные требования безопасности и мониторинга на объекте.

Интерфейс добавления камеры представлен на рисунке 16.

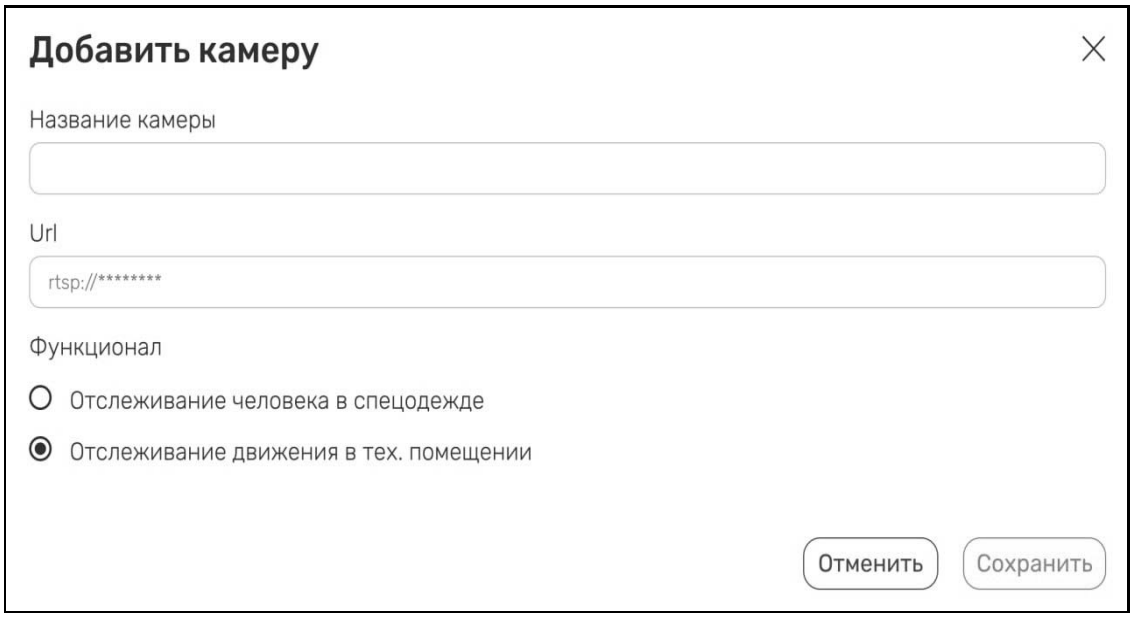

Рисунок 16 – Добавление камеры

Пользователю предоставляется возможность просмотра списка всех добавленных камер в конкретном предприятии. Данный список включает информацию о каждой камере, в том числе тип события, который она способна детектировать (рисунки 17 и 18). Эта функциональность значительно упрощает управление системой видеонаблюдения, позволяя

оперативно оценивать настройки каждой камеры и быстро вносить необходимые корректировки.

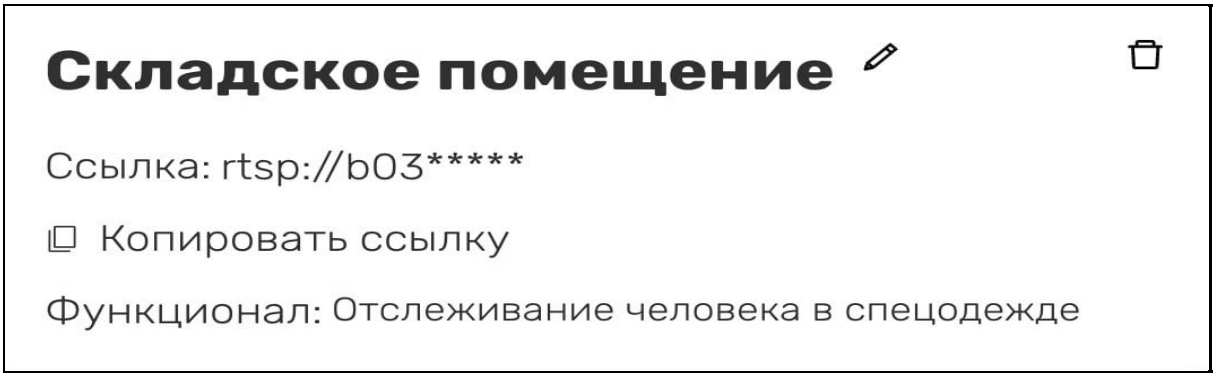

Рисунок 17 – Камера с отслеживанием человека в спецодежде

Рисунок 18 – Камера с детекцией движения

Пользователю предоставляется возможность просмотра списка всех событий, которые произошли на предприятиях. Для удобства организации и анализа эта информация представлена в виде таблицы, в которой пользователь может увидеть детали каждого события, такие как время и дата возникновения, тип события, и другие релевантные характеристики.

Кроме того, в таблице предусмотрена функциональность просмотра видео, связанного с каждым событием (рисунок 19). Пользователь имеет возможность просмотреть запись с момента срабатывания триггера события, что обеспечивает полное представление о происходящем и помогает быстро реагировать на возникшие ситуации.

Такой подход к организации данных о событиях делает систему более прозрачной и удобной для мониторинга, позволяя пользователям быстро и

эффективно анализировать произошедшие инциденты и принимать соответствующие меры.

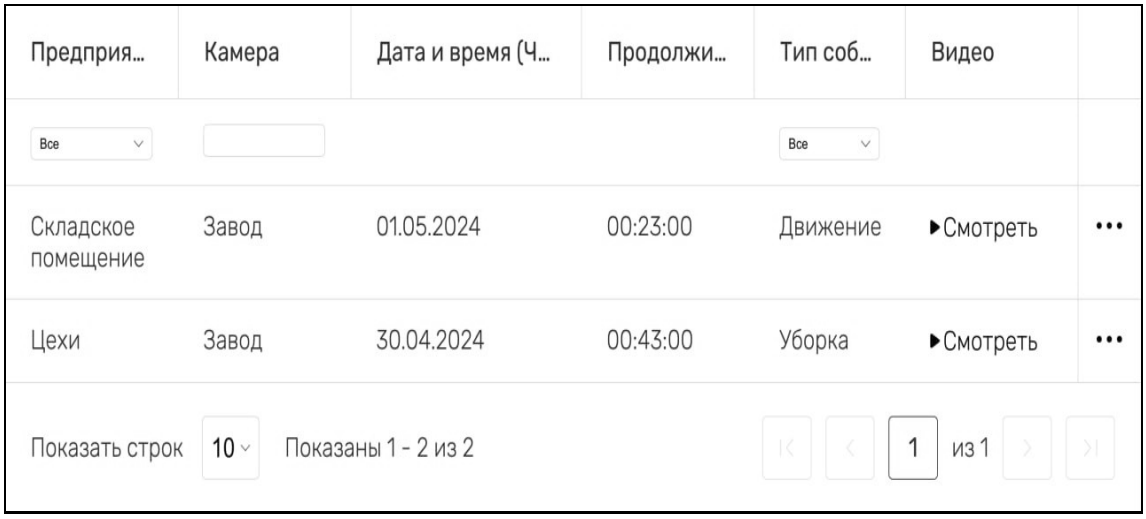

Рисунок 19 – Таблица событий

4.3. Реализация модуля обработки данных

Для реализации серверной части проекта использовали комплекс технологий, включающий в себя фреймворк Flask, базу данных Postgres, брокер сообщений RabbitMQ и сервис хранения файлов Amazon S3. Этот подбор технологий позволил создать мощную и гибкую платформу, способную эффективно обрабатывать и управлять большими объемами данных и взаимодействиями между различными компонентами системы.

Flask использовался как основа для серверной логики, благодаря его простоте и расширяемости. Этот фреймворк идеально подходит для создания REST-сервисов, что является краеугольным камнем в архитектуре нашего приложения. С его помощью разработали API, который обеспечивает создание, обновление, удаление и получение данных о производствах и камерах, которые используются в нашем проекте.

База данных Postgres была выбрана для хранения всех данных, связанных с производствами и камерами. Она обеспечивает высокую производительность и надежность, которые необходимы для управления критически важными данными в реальном времени.

Для взаимодействия между различными сервисами в архитектуре системы был использован RabbitMQ как брокер сообщений. RabbitMQ позволяет надежно обмениваться сообщениями между компонентами системы, что является важным аспектом в поддержке распределенной обработки и обеспечении отказоустойчивости.

Сервис хранения файлов Amazon S3 использовался для сохранения и предоставления доступа к большим файлам, таким как видеозаписи с камер. Взаимодействие с Amazon S3 организовано таким образом, что серверная часть формирует пути к файлам, позволяя клиентской части отображать видео и предоставляя пользователям возможность их скачивания.

Весь этот комплекс технологий сделал модуль обработки данных не только функциональной, но и масштабируемой, поддерживающей строгие требования к безопасности и производительности, которые предъявляются современными веб-приложениями. В приложении С представлен код, иллюстрирующий реализацию описанных функций серверной части.

4.4. Реализация модуля камер

Обучение нейронной сети

Обучение нейронной сети производилось в облачном сервисе Google Colab [26] со следующими характеристиками, перечисленными ниже.

1. GPU NVIDIA Tesla T4 (15 Гб).
2. ОЗУ 12 Гб.

Набор данных был собран с помощью видео и фотографии предоставленных владельцем предприятия. Видео материал был разбит на кадры, что позволило увеличить количество изображений.

Всего набор данных составил 2000 изображений, из них 1400 (70%) изображения использовалось для обучения и 600 (30%) – для тестирования. Обучение проводилось на наборе данных из 1400 (70%) изображений, на которых содержится 2 398 аннотированных людей в спецодежде и 738 аннотированных людей без спецодежды. Для оценки модели было

использовано 600 (30%) изображений, содержащих 942 человека в спецодежде и 186 человек без спецодежды.

Реализация отслеживания с использованием нейронных сетей

Для начальной разработки проекта была выбрана модель YOLOv5, провели ее обучение и тестирование, достигнув точности около 90%. Однако, для работы с реальными камерами решили использовать библиотеку OpenCV, сохраняя при этом архитектуру и веса модели YOLOv5. Таким образом, система сочетает преимущества обученной модели с оптимизацией процесса работы с видеопотоком через OpenCV.

Принцип работы системы заключается в предоставлении пользователем доступа к видеопотоку камеры через URL и указании типа события, который необходимо обнаруживать. Однако, при первых тестированиях обнаружили проблему ложных срабатываний. Например, проезжающая мимо помещения машина с включенными фарами могла вызвать ложное срабатывание модели, так как ее тень могла быть ошибочно интерпретирована как вход человека.

Для решения этой проблемы был внедрен механизм анализа изменений между кадрами видеопотока. Для модели детекции движения используем порог в 25% изменений от общего числа кадров в пуле, который составляет 200 кадров. Аналогично, для модели обнаружения человека в спецодежде был применен анализ с порогом в 25% от 20 кадров. Если порог превышен, считаем, что произошло событие, и начинаем запись видеоматериала.

Выбор модели для использования определяется типом события, указанным пользователем. После записи видеофрагмента данные отправляются на серверную часть системы через очередь сообщений, для дальнейшей обработки и анализа.

Таким образом, реализация этой части работы с нейросетями представляет собой комплексный подход к обнаружению и отслеживанию событий на основе компьютерного зрения, объединяющий в себе

преимущества обученных моделей и оптимизированные методы работы с видеопотоком. На рисунках 20 и 21 изображен пример вывода результата работы нейронной сети.

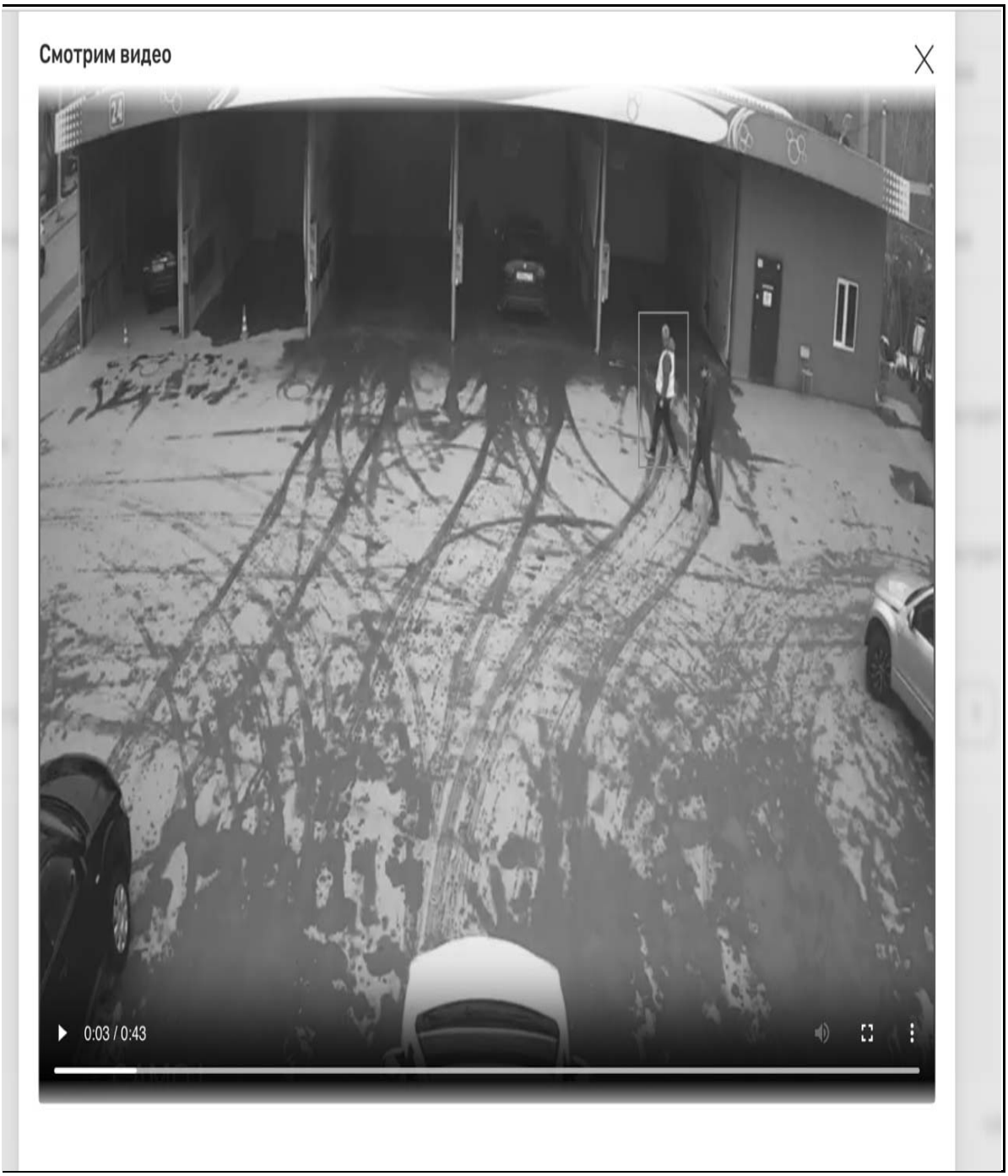

Рисунок 20 – Отслеживание человека в спецодежде

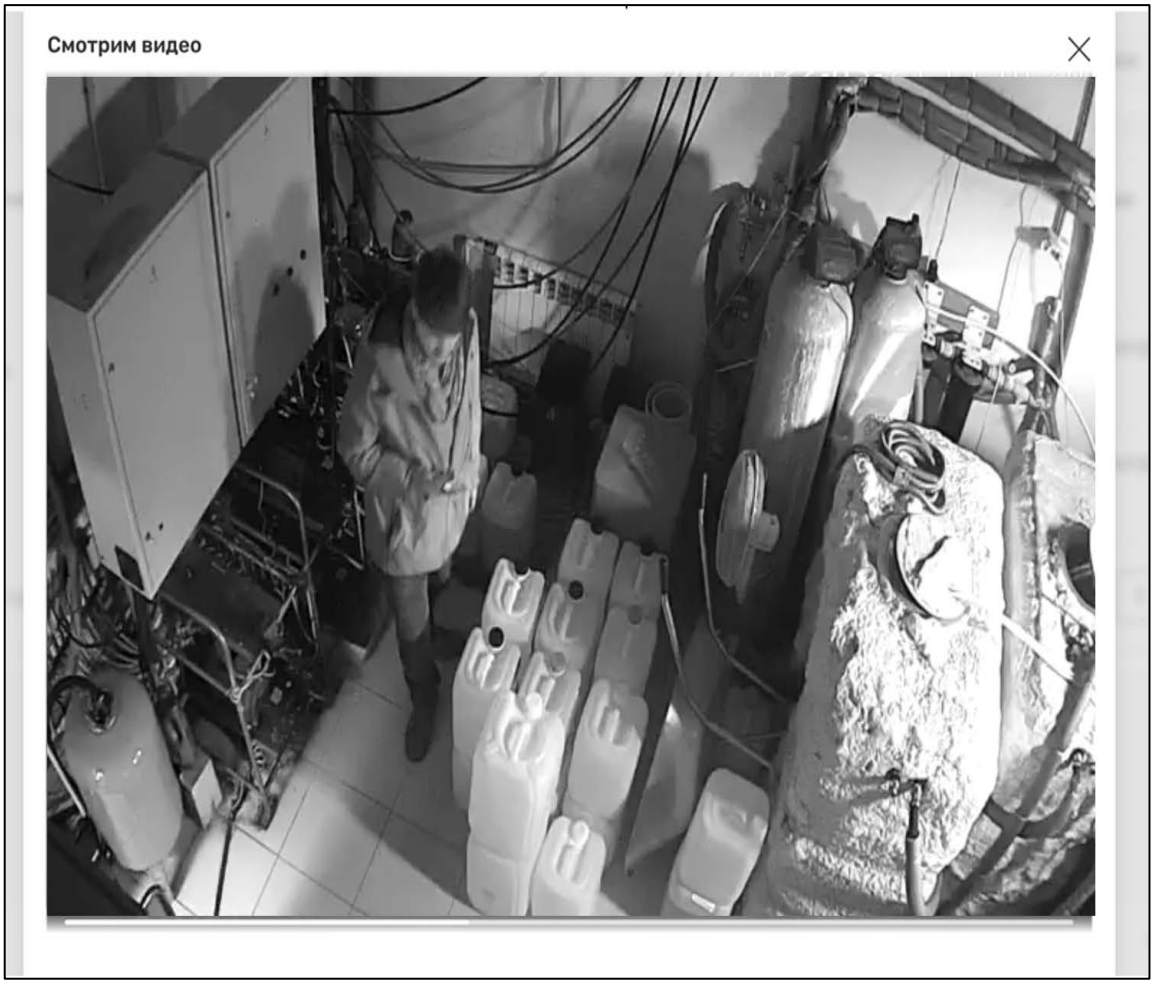

Рисунок 21 – Детекция движения

4.5. Создание Docker контейнеров

Для обеспечения стабильности и удобства развертывания, система использует технологию контейнеризации с помощью Docker. В настройке системы задействованы три различных Dockerfile.

Описание Dockerfile

1. Пользовательский интерфейс – для этого компонента используется образ Node 18. Dockerfile описывает процесс установки зависимостей из файла package.json, который находится в рабочей директории проекта. Затем проект запускается с использованием команды npm run start. Подробная конфигурация этого контейнера представлена в листинге 1.

2. Модуль обработки данных – контейнер для сервера построен на базе образа Python:3.8. В Dockerfile указаны шаги для перехода в рабочую директорию и установки зависимостей из Pipfile.

3. Модуль камер – аналогично серверной части, этот компонент также базируется на образе Python:3.8.

Эти настройки Docker обеспечивают гибкость и масштабируемость системы, позволяя легко развертывать и обновлять компоненты в различных средах с минимальными настройками.

Конфигурация Dockerfile для пользовательского интерфейса представлена в листинге 1.

Листинг 1 – Код Dockerfile пользовательского интерфейса

```
FROM node:14.16.0 as build
ENV PATH /node_modules/.bin:$PATH

COPY package.json /
COPY package-lock.json /
RUN npm ci

COPY / /

RUN npm run build
```

Конфигурация Dockerfile для модуля обработки данных представлена в листинге 2.

Листинг 2 – Код Dockerfile серверной части

```
FROM python:3.8.10 as base

# Setup env
ENV LANG C.UTF-8
ENV LC_ALL C.UTF-8
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONFAULTHANDLER 1
ENV PYTHONUNBUFFERED 1
# Install pipenv
RUN pip install pipenv==2022.1.8

ENV WORKON_HOME="/.venvs"
FROM base as python-virtual-environment
WORKDIR /app-workspace

ADD Pipfile .
ADD Pipfile.lock .
RUN PIP_IGNORE_INSTALLED=1 pipenv sync --keep-outdated
FROM base AS runtime
COPY --from=python-virtual-environment "${WORKON_HOME}" "${WORKON_HOME}"

COPY . /app-workspace
WORKDIR /app-workspace

ENV FLASK_RUN_PORT 80
EXPOSE 80
CMD pipenv run gunicorn --preload --workers 1 --bind 0.0.0.0:80 --capture-
output --enable-stdio-inheritance wsgi:app
```

Конфигурация Dockerfile для модуля камер приведена в листинге 3.

Листинг 3 – Код Dockerfile модуля камер

```
FROM python:3.8.10 as base

# Setup env
ENV LANG C.UTF-8
ENV LC_ALL C.UTF-8

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONFAULTHANDLER 1
ENV PYTHONUNBUFFERED 1
RUN python -m pip install --upgrade pip
RUN pip install pipenv==2022.7.4
ENV WORKON_HOME="/.venvs"
FROM base as python-virtual-environment
WORKDIR /app-workspace

ADD Pipfile .
ADD Pipfile.lock .

RUN PIP_IGNORE_INSTALLED=1 pipenv sync --keep-outdated
FROM base AS runtime
ARG DEV=False
ARG BROKER_HOST
ARG BROKER_PORT
ARG BROKER_USER
ARG BROKER_PASSWORD
ENV RABBITMQ_HOST=$BROKER_HOST
ENV RABBITMQ_PORT=$BROKER_PORT
ENV RABBITMQ_DEFAULT_USER=$BROKER_USER
ENV RABBITMQ_DEFAULT_PASS=$BROKER_PASSWORD
RUN apt-get -y update
RUN apt-get -y upgrade
RUN apt-get install -y ffmpeg
COPY --from=python-virtual-environment "${WORKON_HOME}" "${WORKON_HOME}"
COPY . /app-workspace
WORKDIR /app-workspace

VOLUME ["/records"]
CMD pipenv run python application.py
```

Вывод по четвертой главе

В данной главе была описана реализация данной системы, включая этапы разработки и интеграции компонентов. В процессе изучения исследования работы модели был уделен особый акцент на тщательное обучение и тестирование полученных алгоритмов. Кроме того, важным шагом стала подготовка и использование Docker контейнеров для обеспечения надежной и масштабируемой работы каждого модуля системы.

5. ТЕСТИРОВАНИЕ

5.1. Тестирование системы

Функциональное тестирование выполняется с целью верификации соответствия системы заявленным функциональным требованиям. Для оценки реализации функциональных аспектов системы был разработан специализированный набор тестов. Результаты тестирования представлены в таблице 1, что позволяет наглядно оценить соответствие разработанной системы предъявляемым к ней требованиям [27].

Таблица 1 – Функциональное тестирование

Название теста	**Шаги**	**Ожидаемый результат**	**Тест пройден?**
Заполнил форму для создания предприятия.	1. Перейти на страницу предприятий. 2. Заполнить форму создания предприятия. 3. Отправить данные в API.	Страница обновится и появится созданное предприятие с данными, которые были заполнены в форме.	Да
Заполнил форму для подключения камеры.	1. Перейти на страницу предприятий. 2. Перейти на страницу для подключения камеры. 3. Заполнить форму для подключения камеры.	Страница обновится и появится добавленная камера с данными, которые были заполнены в форме.	Да
Увидеть все существующие события.	1. Перейти на страницу событий. 2. Дождаться ответа сервера. 3. Посмотреть таблицу.	Таблица с событиями обновится и можно увидеть существующие события.	Да

Для оценки и сравнения эффективности различных моделей нейронной сети применялось A/B тестирование. Этот метод позволяет не только анализировать показатели каждой модели, но и выявить оптимальный вариант для задач системы [28]. В ходе тестирования было проведено ряд экспериментов с моделями, обученными на разное количество эпох. Цель этих экспериментов заключалась в определении

наиболее эффективного количества эпох обучения, что позволило бы достичь оптимальной точности и полноты в ответах модели.

Для объективной оценки результатов использовались метрики Precision и Recall. Метрика Recall отражает долю правильно идентифицированных объектов из всех реальных объектов, что позволяет оценить полноту системы, а Precision демонстрирует точность, т.е. долю правильно идентифицированных объектов среди всех определенных системой как объекты. Результаты A/B тестирования были зафиксированы и систематизированы в таблице 2.

Таблица 2 – Результаты A/B тестирования

Количество эпох обучения	**Модель**	**Precision**	**Recall**
10	YOLOv5s	0,79	0,79
	YOLOv5m	0,89	0,87
	YOLOv5l	0,89	0,89
15	YOLOv5s	0,89	0,91
	YOLOv5m	0,91	0,90
	YOLOv5l	0,88	0,85
20	YOLOv5s	0,88	0,84
	YOLOv5m	0,94	0,94
	YOLOv5l	0,89	0,90

Как показывает таблица, при увеличении числа эпох обучения модель YOLOv5m продемонстрировала стабильное улучшение как в Precision, так и в Recall по сравнению с YOLOv5s и YOLOv5l. Например, после 20 эпох YOLOv5m достигает самого высокого уровня Precision (0,92) и Recall (0,93), что говорит о его высокой эффективности. Эта модель была выбрана как наиболее подходящая для дальнейшего использования в рамках разрабатываемой системы.

Вывод по пятой главе

В данной главе было проведено тестирование приложение на соответствие функциональным требование и A/B тестирование нейронной сети, в результате которого была выявлена лучшая модель.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана система для фиксации перемещения рабочих на производстве на основе алгоритмов компьютерного зрения. Работа включала несколько ключевых этапов, направленных на создание и оптимизацию системы, обеспечивающей надежный и точный мониторинг перемещения персонала на производственных площадках. В ходе выполнения проекта были решены следующие задачи.

1. Произведен обзор литературы и существующих приложений по предметной области. На этом этапе была изучена актуальная научная и техническая литература, а также проведен анализ существующих решений в области мониторинга перемещений с использованием компьютерного зрения.

2. Выполнено проектирование архитектуры приложения. На этом этапе были разработаны основные компоненты системы, определены их взаимодействие и функциональные требования.

3. Разработка системы для фиксации перемещения рабочих на производстве на основе алгоритмов компьютерного зрения. Этот этап включал в себя программирование всех необходимых компонентов системы, начиная от захвата видеопотока и заканчивая обработкой данных с использованием алгоритмов компьютерного зрения.

4. Проведено тестирование системы. На заключительном этапе была проведена серия тестов для проверки корректности работы системы.

ЛИТЕРАТУРА

1. Rojas R.A Systematic Introduction. // Springer-Verlag, March 1996, Berlin, Germany. – 1996. – 506 p.

2. Géron A. Hands On Machine Learning with Scikit-Learn and TensorFlow. // O'Reilly Media March 2017, Sebastopol. – 2017. – 718 p.

3. Ripley B. Pattern Recognition and Neural Networks – Cambrige University Press, 1st edition, 2008. – 416 p.

4. Chun-Wei Y., Thanh Hai P. Mask or Non-Mask? Robust Face Mask Detector via Triplet-Consistency Representation Learning. // ACM Trans. Multimedia Comput. Commun, 2021 – 19 p.

5. Docker. [Электронный ресурс] URL: https://docs.docker.com/ (дата обращения: 10.04.2024 г.).

6. Джоши П. Искусственный интеллект с примерами на Python. // Packt, 2019. – 200 с.

7. Cielen D., Meysman A. D. B., Ali M. Introducing Data Science. // Manning publications, 2016 – 320 p.

8. Redmon, J., Divvala, S., Girshick, R., Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection [Электронный ресурс] // arXiv.org. 2016. Дата обновления: 09.05.2016. URL: https://arxiv.org/abs/1506.02640 (дата обращения: 10.04.2024 г.).

9. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. SSD: Single Shot MultiBox Detector [Электронный ресурс] // arXiv.org. 2016. Дата обновления: 29.12.2016. URL: https://arxiv.org/abs/1512.02325 (дата обращения: 10.04.2024 г.).

10. Ren, S., He, K., Girshick, R., Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [Электронный ресурс] // arXiv.org. 2015. Дата обновления: 26.01.2016. URL: https://arxiv.org/abs/1506.01497 (дата обращения: 10.04.2024 г.).

11. Xu, S., Guo, Z., Liu, Y., Fan, J., Liu, X. An Improved Lightweight YOLOv5 Model Based on Attention Mechanism for Face Mask Detection. In:

Artificial Neural Networks and Machine Learning – ICANN 2022. ICANN 2022. Lecture Notes in Computer Science, vol 13531. Springer, Cham, September 15, 2022, 2022. – 531–543 pp. DOI: /10.1007/978-3-031-15934-3_44

12. Набор данных AIZOOTech. [Электронный ресурс] URL: https://github.com/AIZOOTech/FaceMaskDetection (дата обращения: 07.05.2024 г.)

13. Sun, C.; Zhang, S.; Qu, P.; Wu, X.; Feng, P.; Tao, Z.; Zhang, J.; Wang, Y. MCA-YOLOV5-Light: A Faster, Stronger and Lighter Algorithm for Helmet-Wearing Detection. Appl. Sci. 2022, 12, 9697. [Электронный ресурс] URL: https://www.mdpi.com/2076-3417/12/19/9697 (дата обращения: 07.05.2024 г.).

14. Xu Q, Zhu Z, Ge H, Zhang Z, Zang X. Effective Face Detector Based on YOLOv5 and Superresolution Recon-struction. Comput Math Methods Med. 2021 Nov. [Электронный ресурс] URL: https://pubmed.ncbi.nlm.nih.gov/34824599 (дата обращения: 07.05.2024 г.).

15. Ku, B.; Kim, K.; Jeong, J. Real-Time ISR-YOLOv4 Based Small Object Detection for Safe Shop Floor in Smart Factories. Electronics 2022, 11, 2348. [Электронный ресурс] URL: https://www.mdpi.com/2079-9292/11/15/2348 (дата обращения: 07.05.2024 г.).

16. PyTorch. [Электронный ресурс] URL: https://pytorch.org (дата обращения: 10.04.2024 г.).

17. OpenCV [Электронный ресурс] URL: https://docs.opencv.org/4.x/ (дата обращения: 10.04.2024 г.)

18. TensorFlow. [Электронный ресурс] URL: https://www.tensorflow.org/?hl=ru (дата обращения: 10.04.2024 г.).

19. Гудфеллоу Я., Бенджио И. Глубокое обучение. // ДМК Пресс, 2018. – 652 с.

20. UML [Электронный ресурс] URL: https://www.omg.org/ https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling (дата обращения: 10.04.2024 г.).

21. Flask [Электронный ресурс] URL: https://flask.palletsprojects.com/en/latest/ (дата обращения: 10.04.2024 г.).

22. Amazon S3. [Электронный ресурс] URL: https://docs.amazonaws.cn/en_us/AmazonS3/latest/API/s3-api.pdf (дата обращения: 10.04.2024 г.).

23. Postgres. [Электронный ресурс] URL: https://www.postgresql.org/about/ (дата обращения: 10.04.2024 г.).

24. RabbitMQ. [Электронный ресурс] URL: https://www.rabbitmq.com/docs (дата обращения: 10.04.2024 г.).

25. ReactJs [Электронный ресурс] URL: https://react.dev/learn (дата обращения: 10.04.2024 г.).

26. Colaboratory. [Электронный ресурс] URL: https://colab.research.google.com (дата обращения 10.04.2024 г.).

27. Machine Learning Lifecycle. [Электронный ресурс] URL: https://www.geeksforgeeks.org/machine-learning-lifecycle/ (дата обращения: 18.05.2024 г.).

28. Breaking down mean average Precision (maP). [Электронный ресурс] URL: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52 (дата обращения: 10.04.2024 г.).