Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
Институт радиоэлектроники и информационных технологий - РТФ
Кафедра информационных технологий и систем управления

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

Зав. кафедрой ИТиСУ
_____ Е.В. Кислицын
(подпись) (Ф.И.О.)
«31» 05 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПРИМЕНЕНИЕ АЛГОРИТМОВ КОМПЬЮТЕРНОГО ЗРЕНИЯ ДЛЯ СИСТЕМЫ КОНТРОЛЯ ОТГРУЗКИ ГОРЯЧЕКАТАННЫХ РУЛОНОВ

Научный руководитель: Борисов Василий Ильич
к.т.н., доцент
_____
подпись

Нормоконтролер: Бредихина Наталья Сергеевна
_____
подпись

Студент группы РИМ-220906: Васильев Антон Евгеньевич
_____
подпись

Екатеринбург
2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ
Кафедра информационных технологий и систем управления
Направление подготовки 09.04.01 Информатика и вычислительная техника
Образовательная программа Инженерия искусственного интеллекта

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента Васильева Антона Евгеньевича **группы** РИМ-220906
(фамилия, имя, отчество)

1. Тема выпускной квалификационной работы Применение алгоритмов компьютерного зрения для системы контроля отгрузки горячекатанных рулонов
Утверждена распоряжением по институту от «4» декабря 2023 г. № 33.02-05/298

2. Научный руководитель Борисов Василий Ильич, к.т.н., доцент
(Ф.И.О., должность, ученая степень, ученое звание)

3. Исходные данные к работе

4. Перечень демонстрационных материалов *презентация в MS PowerPoint*

5. Календарный план

№ п/п	Наименование этапов выполнения работы	Срок выполнения этапов работы	Отметка о выполнении
1.	Глава 1. Анализ предметной области	до 23.03.2024 г.	
2.	Глава 2. Техническая реализация	до 29.04.2024 г.	
	Глава 3. Анализ результатов	до 10.05.2024 г.	
3.	ВКР в целом	до 20.05.2024 г.	

Научный руководитель _____ _____
Ф.И.О. (подпись)

Студент задание принял к исполнению 22.05.2024 _____
дата (подпись)

7. Допустить Васильева Антона Евгеньевича к защите выпускной квалификационной работы в экзаменационной комиссии

Зав. кафедрой ИТиСУ _____ Е.В. Кислицын
(подпись) Ф.И.О.

РЕФЕРАТ

Васильев А.Е. Использование алгоритмов компьютерного зрения для системы контроля отгрузки горячекатанных рулонов.

Магистерская диссертация 53 с., 15 рис., 10 табл., 48 источн.

КОМПЬЮТЕРНОЕ ЗРЕНИЕ, ДЕТЕКЦИЯ ОБЪЕКТОВ, ТРЕКИНГ ОБЪЕКТОВ, НЕЙРОННЫЕ СЕТИ, ОТГРУЗКА ПРОДУКЦИИ, «УМНЫЙ СКЛАД»

Объект исследования – подвижная единица железнодорожного состава.

Предмет исследования – процесс идентификации и отслеживания отгружаемой продукции.

Цель работы – разработка инструмента, использующего методы компьютерного зрения, для контроля и идентификации отгружаемой продукции в производственных условиях промышленного предприятия.

Результатом работы является разработанный инструмент для автоматического сбора и обработки данных для формирования отгрузочных документов на основе видеопотока с камеры наблюдения за проходящим железнодорожным составом.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

CPU – Central Processing Unit

FPS – Frames Per Second

GPU – Graphic Processing Unit

MES – Manufacturing Execution System

OCR – Optical Character Recognition

SOTA – State-of-the-art

IOU – Intersection Over Union

АС – Автоматизированная Система

АСУ – Автоматизированная Система Управления

ЛПЦ – Листопрокатный Цех

МК –Металлургический Комбинат

СКП – Система Контроля Продукции

СКНВ – Система Контроля Номеров Вагонов

ВВЕДЕНИЕ

В последние годы в промышленности эффективное управление производственными процессами становится неотъемлемой составляющей обеспечения конкурентоспособности предприятий. Наиболее важными задачами являются точное и своевременное выполнение операций по отгрузке и доставке продукции. В металлургической отрасли, где скорость и качество производства играют решающую роль, управлению процессом отгрузки продукции уделяется особое внимание. Системы контроля отгрузки горячекатанных рулонов на металлургическом предприятии обеспечивают не только безопасность и эффективность производства, но и минимизацию ошибок, связанных с ошибками в оформлении отгрузочных документов, оптимизацию логистических затрат, ведение претензионной работы, исключение хищений.

Актуальность данного исследования обусловлена рядом факторов. Во-первых, автоматизация и точность системы контроля отгрузки позволит оптимизировать логистические процессы, обеспечивая ориентированность на конечного потребителя и предотвращая случаи доставки не тому адресату. Во-вторых, автоматическое формирование документов на основе данных, полученных от системы контроля, существенно повышает оперативность и достоверность учета и отчетности на предприятии.

В рамках данной магистерской работы исследуется проблема оптимизации системы контроля отгрузки горячекатанных рулонов с использованием передовых технологий компьютерного зрения на примере ЛПЦ МК. Предполагается, что внедрение данных технологий на производство позволит эффективно контролировать и оптимизировать процесс отгрузки и отправки продукции заказчику, при этом минимизируя затраты.

Целью данного исследования является разработка инструмента, использующего методы компьютерного зрения, для контроля и идентификации отгружаемой продукции в производственных условиях

промышленного предприятия. Разработанный инструмент предназначен для автоматического сбора и обработки данных для формирования отгрузочных документов на основе видеопотока с камеры наблюдения за проходящим железнодорожным составом, вследствие чего вытекают следующие задачи:

1) провести анализ существующих решений для решения задачи детекции и идентификации объектов по видеопотоку;

2) собрать и разметить необходимые данные;

3) обучить выбранную модель детекции с последующим анализом ее результатов работы;

4) разработать единый алгоритм учета проезжающей продукции с помощью подобранного алгоритма трекинга.

Результаты данного исследования могут вносить значительный вклад в область автоматизации и интеллектуальных систем контроля, а также иметь практическое применение на металлургических предприятиях.

1 Анализ предметной области

1.1 Детекция объектов

Детекция объектов — это компьютерная технология, относящаяся к компьютерному зрению и обработке изображений, которая занимается обнаружением экземпляров семантических объектов определенного класса (например, людей, зданий или автомобилей) в цифровых изображениях и видео. Хорошо изученные области обнаружения объектов включают в себя обнаружение лиц и обнаружение пешеходов. Обнаружение объектов применяется во многих областях компьютерного зрения, например, видеонаблюдение. С развитием технологий в области компьютерного зрения и машинного обучения, детекция объектов становится все более востребованной и имеет широкий спектр применений в различных сферах современного общества.

Одной из ключевых особенностей моделей детекции является их способность выявлять несколько объектов на одном кадре. Это отличает их от простых классификаторов, которые могут определять только один объект на изображении. Выделение объектов происходит за счёт прогонки кадра в модели, которая является либо двухстадийным, либо одностадийным детектором. При этом задача нахождения объектов на изображении — задача машинного обучения, в рамках которой выполняется определение наличия или отсутствия объекта определённого домена на изображении, нахождение границ этого объекта в системе координат пикселей исходного изображения. В зависимости от алгоритма обучения, объект может характеризоваться координатами ограничивающей рамки, ключевыми точками и контуром объекта.

Для рассмотрения вопроса детекции объектов было проанализировано исследование "Road Object Detection: A Comparative Study of Deep Learning-Based Algorithms"[1, 11, 36, 41], которое содержит подробный анализ пяти

моделей обнаружения объектов на основе глубокого обучения: R-FCN[26], Mask R-CNN[16], SSD[30], RetinaNet[8] и YOLOv4[4], с использованием набора данных Berkeley Deep Drive (BDD100K), который содержит большой объем разнообразных данных, собранных с улиц городов[2]. Данное исследование направлено на повышение производительности систем автоматизированного вождения и безопасности транспортных средств за счет повышения точности обнаружения объектов, устойчивости к различным условиям и скорости выполнения.

В нем подробно описывается весь процесс проведения эксперимента, применяют аугментации для обучения моделей. Результаты представлены в виде кривых "precision-recall" и метрики средней точности (mAP).

Результаты исследования показывают, что YOLOv4 значительно превосходит другие модели по точности и скорости обнаружения, что делает ее наиболее подходящим выбором для обнаружения дорожных объектов в режиме реального времени в сложных сценариях и погодных условиях. Mask R-CNN также демонстрирует многообещающую производительность, особенно на аугментированных изображениях. SSD, хотя и работает быстрее всех, но демонстрирует более высокий процент ошибок в обнаружении (таблица 1).

Таблица 1 – Сравнение моделей детекции в исследовани "Road Object Detection: A Comparative Study of Deep Learning-Based Algorithms"

Модель	mAP (%)	mAP_{WOT}* (%)	CPU inference (сек)	GPU inference (сек)
R-FCN	52.86	41.86	2.96	0.17
Mask R-CNN	67.61	60.69	3.32	0.35
SSD	41.52	32.62	0.04	0.01
RetinaNet	62.57	54.02	3.68	0.20
YOLOv4	88.67	73.09	0.10	0.02
*Примечание – WOT - с техниками аугментации (окклюзия и усечение)				

В исследовании делается вывод о том, что, несмотря на значительный прогресс в технологиях обнаружения объектов, сохраняются проблемы, связанные с балансом между скоростью, точностью и надежностью, особенно

в динамичных и сложных средах. Производительность YOLOv4 подчеркивает ее потенциал в качестве лучшего решения для обнаружения объектов в реальном времени.

Следующее исследование производительности алгоритмов обнаружения объектов[14, 45, 48] предлагает обзор и анализ алгоритмов глубокого обучения для детекции объектов в условиях, которые усложняют этот процесс. Авторы работы подчеркивают значительные достижения в глубоком обучении, которые привели к разработке эффективных и точных систем детекции объектов. Однако эти системы часто сталкиваются с проблемами при наличии препятствий, низкой освещенности или когда объекты сливаются с фоном - ситуации, которые в этом исследовании называются "сложными условиями"[10, 29].

Для этого исследования авторы проводили исследования современных архитектур для задачи детекции на следующих датасетах:

– ExDark[15] – это специализированный датасет, созданный для исследований в области компьютерного зрения, фокусирующихся на задачах детекции и классификации объектов в условиях низкой освещенности. Датасет был разработан с целью преодоления ограничений традиционных методов компьютерного зрения, которые часто испытывают трудности при работе с изображениями, сделанными в темное время суток или при слабом освещении.

– CURE-STD (Challenging Unreal and Real Environments for Surveillance and Traffic Detection)[32] - датасет, созданный для исследований в области компьютерного зрения, направленных на задачи обнаружения и классификации объектов в сложных условиях. Этот датасет разработан для проверки устойчивости алгоритмов компьютерного зрения к различным искажениям и неблагоприятным условиям, которые могут возникнуть в реальных сценариях наблюдения и мониторинга дорожного движения.

– RESIDE (Realistic Single Image Dehazing)[3] - специализированный датасет, разработанный для исследований в области тумана (dehazing) на изображениях. Датасет RESIDE предназначен для улучшения алгоритмов

компьютерного зрения, которые сталкиваются с задачей обработки изображений, страдающих от природных явлений, таких как смог или туман.

Из обзора статьи следует, что нет единой модели, которая была бы идеальной для всех сценариев детекции объектов в сложных условиях. Однако, YOLOv3 и Mask R-CNN выделяются своей эффективностью в реальном времени и точностью соответственно, делая их предпочтительными выборами для разных приложений (таблица 2).

Таблица 2 – Результаты исследования статьи "Survey and Performance Analysis of Deep Learning Based Object Detection in Challenging Environments"

Модель	ExDark		CURE-STD		RESIDE		FPS
	AP	AP^{50}	AP	AP^{50}	AP	AP^{50}	
Mask R-CNN	0.54	0.84	0.2	0.35	0.51	0.79	8
Faster R-CNN	0.53	0.82	0.25	0.43	0.49	0.78	12
YOLOv3	0.67	0.93	0.16	0.32	0.37	0.78	50
Retina-Net	0.36	0.67	0.14	0.25	0.48	0.75	11
Cascade Mask R-CNN[5]	0.49	0.78	0.28	0.38	0.5	0.76	8

YOLOv3[23] демонстрирует высокую скорость обработки, что делает его идеальным для задач, требующих обработки в реальном времени, в то время как Mask R-CNN[16] обеспечивает высокую точность за счет более глубокого и тщательного анализа каждого кадра, что делает его подходящим для приложений, где точность является ключевым фактором.

В заключение, выбор модели зависит от специфических требований задачи, включая скорость обработки, точность и специфику данных. Однако YOLOv3 и Mask R-CNN выделяются как две особенно мощные модели для детекции объектов в широком спектре сложных условий.

1.1.1 YOLO

YOLO (You Only Look Once)[47], в отличие от традиционных систем обнаружения объектов, которые адаптируют классификаторы или применяют их на разных масштабах для выполнения обнаружения, рассматривает обнаружение объектов как единую задачу регрессии, где за аргументы взяты

пиксели изображений, координаты ограничивающих рамок и вероятности классов. Такое допущение позволяет выполнить весь процесс обнаружения одной нейронной сетью за один «проход», выполняя поиск значительно быстрее существующих методов, и при этом сохраняя высокую точность.

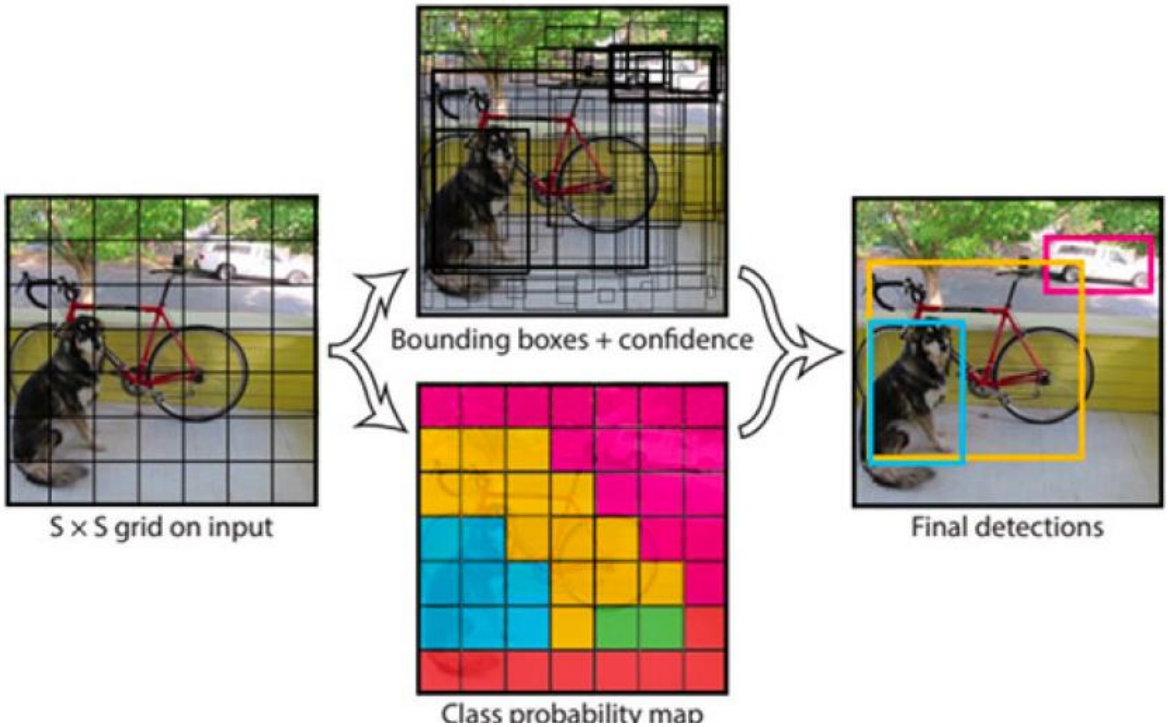


Рисунок 1 – Принцип работы модели YOLO

Алгоритм YOLO был первой попыткой сделать возможной детекцию объектов в реальном времени. В рамках алгоритма YOLO исходное изображение сначала разбивается на сетку из N×N ячеек (обычно 13x13 или 19x19) и для каждой ячейки определяет вероятность наличия в ней объекта и его координаты. Также для каждого объекта определяется его класс (например, автомобиль, человек, собака и т.д.). Если центр объекта попадает внутрь координат ячейки, то эта ячейка считается ответственной за определение параметров местонахождения объекта. Каждая ячейка описывает несколько вариантов местоположения ограничивающих рамок для одного и того же объекта. Каждый из этих вариантов характеризуется пятью значениями — координатами центра ограничивающей рамки, его шириной и

высотой, а также степени уверенности в том, что ограничивающая рамка содержит в себе объект. Также необходимо для каждой пары класса объектов и ячейки определить вероятность того, что ячейка содержит в себе объект этого класса.

После этого происходит фильтрация результатов – удаляются объекты с низкой вероятностью и объединяются близкие объекты. На выходе получается список обнаруженных объектов с их координатами и классами. Преимуществом YOLO является высокая скорость работы алгоритма. Он может обрабатывать изображения со скоростью до 45 кадров в секунду на современном GPU. Это делает его очень эффективным для использования в реальном времени – например, для распознавания объектов на видеопотоке. Модель достигает более чем двухкратную среднюю точность предсказания по сравнению с другими системами реального времени.

Однако, как и любой другой алгоритм, несмотря на его скорость и эффективность, YOLO имеет ограничения и недостатки. Например, он может иметь проблемы с обнаружением мелких объектов или с объектами, которые перекрываются другими объектами. Также YOLO может допускать ошибки при классификации объектов – например, если объект очень похож на объект другого класса.

На данный момент существует 8 версий YOLO. Наиболее популярной версией, обладающей наилучшим соотношением скорость-качество среди практически всех остальных нейросетевых детекторов, является модель YOLOv5, которую реализовали на языке PyTorch[13]. В данной работе предложено использовать модель YOLOv8 превосходящую пятую версию в точности и скорости[12].

1.1.2 RCNN

Архитектура R-CNN (Region-Based Convolutional Neural Network) была разработана в 2014 году Ross Girshik и другими [27]. В основе этого метода лежит следующий алгоритм:

Нахождение потенциальных объектов на изображении и разбиение их на регионы спомощью метода selective search[42]. Извлечение признаков каждого полученного региона с помощью сверточных нейронных сетей. Классифицирование обработанных признаков с помощью метода опорных векторов (SVM, Support Vector Machine) и уточнение границ регионов с помощью линейной регрессии.

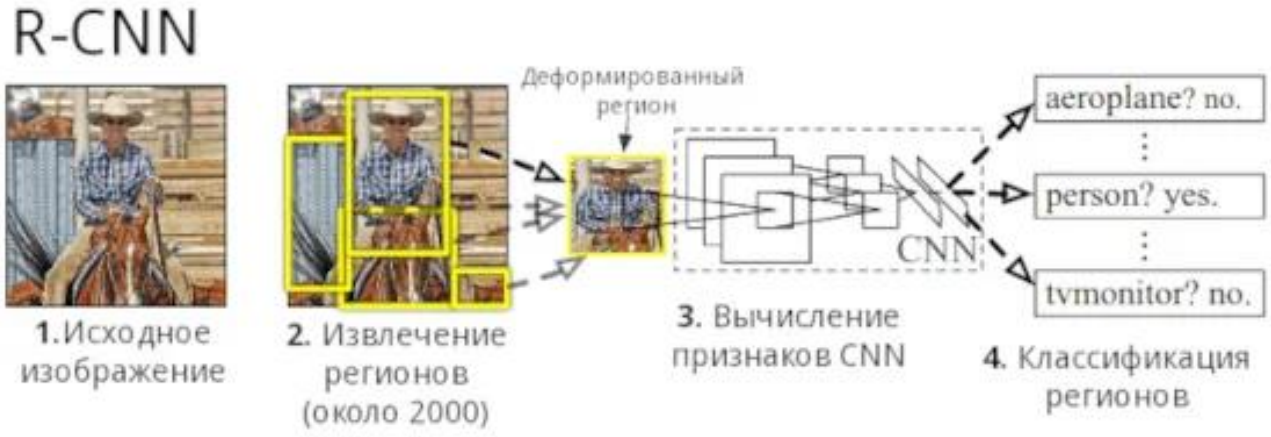


Рисунок 2 – Принцип работы архитектуры R-CNN

В итоге, получаем отдельные регионы с объектами и их классами. В центре стоят сверточные нейронные сети, которые показывают хорошую точность на примере изображений. Но у такой архитектуры есть основной недостаток: энергозатратность. Данная архитектура требует большого количество времени на обучение. В методе selective search изображение сначала сегментируется на 2000 регионов, которые затем в ходе итерирования с помощью жадного алгоритма объединяются в более крупные регионы. Кроме того, сами сверточные сети тоже требует вычислительных мощностей. Не может быть использован для видео. Опять же из вытекает из недостатка выше, так как все промежуточные методы энергозатратны, поэтому кадры просто не будут успевать обрабатываться. Selective search не является алгоритмом

машинного обучения, поэтому могут возникнуть проблемы с выявлением потенциальных объектов на разных изображениях.

На данный момент модель R-CNN устарела и не применяется.

1.1.2 Fast R-CNN

Недостатки R-CNN привели авторов в 2015 году к улучшению модели. Они назвали ее Fast R-CNN[9]. В ее основе так же лежит multi-stage архитектура (рисунок 3).

Изображение подается на вход сверточной нейронной сети и обрабатывается с помощью алгоритма selective search. В итоге получается карта признаков и регионы потенциальных объектов. Координаты регионов потенциальных объектов преобразуются в координаты на карте признаков. Полученная карта признаков с регионами передается слою RoI (Region of Interest) polling layer. Здесь на каждый регион представлен сеткой размером HxW. Затем применяется MaxPolling для уменьшения размерности. Так, все регионы потенциальных объектов имеют одинаковую фиксированную размерность. Полученные признаки подаются на вход полносвязного слоя (Fully-conectedlayer), который передается двум другим полносвязным слоям. Первый с функцией активацией softmax определяет вероятность принадлежности классу, второй — границы (смещение) региона потенциального объекта.

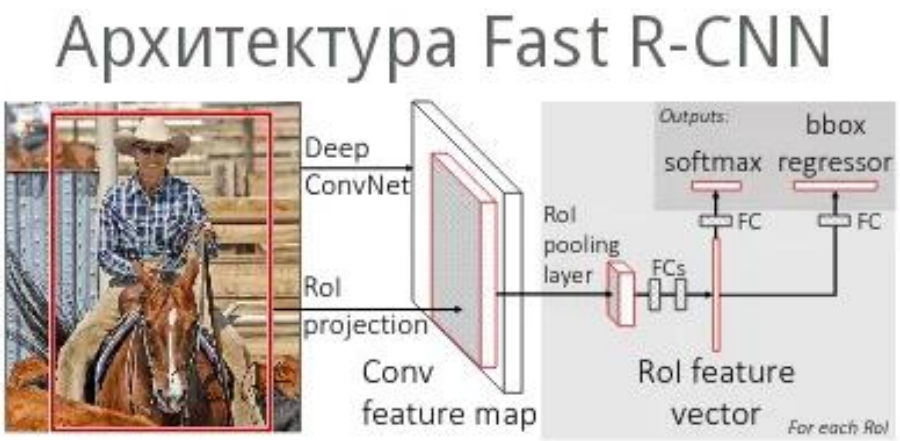


Рисунок 3 – Принцип работы архитектуры Fast R-CNN

Fast R-CNN показывает чуть более высокую точность и большой прирост времени обработки в отличие от R-CNN, так как не требуется подавать все регионы на сверточный слой. Но тем не менее, данный метод использует затратный Selective Search. Поэтому авторы пришли к Faster R-CNN.

1.1.3 Faster R-CNN

Авторы продолжили улучшение над Fast R-CNN и в 2016 предложили Faster R-CNN[7]. Они разработали собственный метод локализации объекта взамен Selective Search — RPN (Region Proporsal Networks). В основе RPN лежит система якорей. Архитектура Faster R-CNN образована следующим образом:

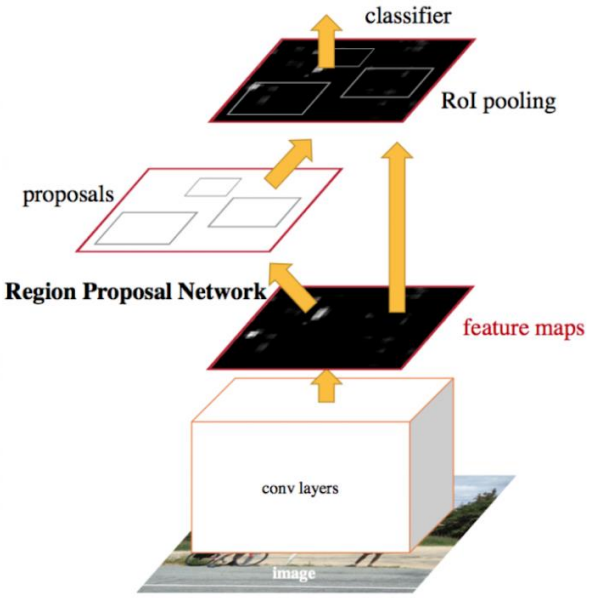


Рисунок 4 – Архитектура Faster R-CNN

Изображение подается на вход сверточной нейронной сети. Так, формируется карта признаков. Карта признаков обрабатывается слоем RPN. Здесь скользящее окно проходится по карте признаков. Центр скользящего окна связан с центром якорей. Якоря — это области, имеющие разные

соотношения сторон и разные размеры. Авторы используют 3 соотношения сторон и 3 размера. На основе метрики IoF(intersection-over-union), степени пересечения якорей и истинных размеченных прямоугольников, выносится решение о текущем регионе — есть объект или нет. Далее используется алгоритм FastCNN: карта признаков с полученными объектами передаются слою RoI с последующей обработкой полносвязных слоев и классификацией, а также с определением смещения регионов потенциальных объектов.

1.1.4 SSD

В своё время оригинальная версия YOLO [21] достигла значительного прогресса по сравнению с Faster R-CNN, обеспечивая более высокую скорость обработки при незначительном снижении точности. Авторы статьи продолжили эту линию работы, стремясь улучшить качество без ущерба скорости. SSD был разработан для обнаружения объектов в реальном времени[30].

Как упоминалось ранее, Faster R-CNN использует RPN для создания ограничивающих рамок и их последующей классификации. Несмотря на то, что эта модель считается ведущей по точности, её скорость составляет всего 7 кадров в секунду, что недостаточно для реального времени. Авторы SSD ускорили процесс, удалив RPN и применив несколько улучшений для компенсации потери точности.

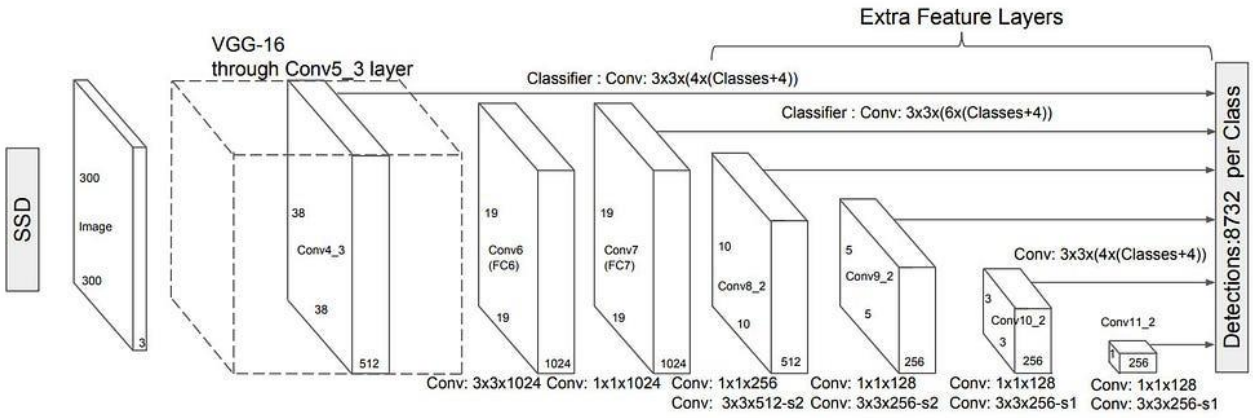


Рисунок 5 – Архитектура SSD

Подход SSD, аналогично YOLO, состоит из двух частей:

- Извлечение карт признаков
- Применение сверточных фильтров к обнаруженным объектам.

Также авторы предложили фиксированный набор прямоугольников для поиска объектов в изображении, но в YOLO недостаточно кандидатов на роль объекта, и используемые признаки брались с последнего свёрточного слоя, что приводило к потере мелких объектов и ухудшению качества. Решение было найдено в использовании признаков с разных свёрточных слоёв, что автоматически увеличивало число кандидатов. Для избежания обучения сети с применением одного предсказания на объект было решено применить NMS после детектирования.

Эти улучшения позволили SSD достичь точности Faster R-CNN при использовании изображений меньшего разрешения, что ещё больше повысило скорость обработки. Согласно исследованию, SSD достигает скорости обработки в реальном времени и даже превосходит Faster R-CNN по точности.

Для извлечения карты признаков SSD применяет VGG16 []. Принцип работы и основное отличие заключаются в следующем. Сначала получается карта признаков размером m×n с помощью одного из свёрточных слоёв. Затем к ней применяется свёрточный слой размером 3×3, результатом которого становится выход с 4+Cl каналами, где Cl – количество классов, подлежащих детектированию, а 4 канала отводятся для определения прямоугольника, описывающего объект. Это позволяет разбить изображение таким образом, что каждая характеристика на выходе свёрточного слоя содержит информацию о пикселях квадрата на исходном изображении, что, в свою очередь, позволяет обнаруживать объект, находящийся в этом квадрате. Использование более раннего слоя для создания карты признаков приводит к увеличению её размера и уменьшению по размеру детектируемых объектов.

Предсказания SSD классифицируются как положительные или отрицательные совпадения. При расчёте точности локализации объекта SSD

учитывает только положительные совпадения. Совпадение считается положительным, если ограничивающее поле по умолчанию имеет IoU больше 0,5 с реальным местоположением объекта. В противном случае оно считается отрицательным.

Подход, который был сначала внедрен в оригинальной YOLO, а затем улучшен в SSD, продемонстрировал лучшее качество при высокой скорости по сравнению с первоначальной YOLO. Этот подход может быть успешно применен вместе с Faster R-CNN, особенно в ситуациях, где требуется обнаружение объектов в реальном времени. Однако SSD проявляет себя менее эффективно по сравнению с Faster R-CNN в обнаружении мелких объектов. В SSD мелкие объекты могут быть обнаружены только на более высокоразрешенных слоях (наиболее левых слоях). Однако эти слои содержат низкоуровневые признаки, такие как края или цветовые пятна, которые менее информативны для классификации. Кроме того, на текущий момент данное решение уступает более поздним версиям модели YOLO.

1.2 Трекинг объектов

Трекинг объектов — это определение местоположения движущегося объекта (нескольких объектов) во времени с помощью камеры. Это крайне важная область компьютерного зрения и машинного обучения, которая находит широкое применение в различных сферах, включая видеонаблюдение, автономные транспортные системы и многое другое. Цель трекинга объектов – это создание траектории объекта во времени путем локализации его позиции в каждом кадре видео. Применение видеоаналитических систем дает возможность снизить участие в процессе наблюдения оператора, что, в свою очередь, уменьшает влияние человеческого фактора, сокращает время реакции на нештатные ситуации и позволяет своевременно распознавать угрозы. Кроме того, такая система

позволяет анализировать большое количество поступающих данных в режиме реального времени, а не обрабатывать их уже после события.

Очевидно, что автономный трекинг объектов является объективно необходимым решением. Для его реализации необходимо изучить различные существующие методы и подходы, а также проанализировать их преимущества, недостатки и области применения, чтобы выбрать наилучший.

1.2.1 SORT

Алгоритм SORT[38] — это эффективный алгоритм отслеживания в реальном времени. Он сочетает в себе концепции фильтров Калмана и венгерских алгоритмов отслеживания объектов.

Все начинается с использования алгоритма обнаружения, такого как детектор объектов на основе глубокого обучения, для идентификации объектов в первом кадре видеопоследовательности. Каждому обнаруженному объекту присваивается уникальный идентификационный номер. Обнаружения из текущего кадра передаются на следующий кадр путем оценки положения объектов с использованием модели с постоянной скоростью. Эти компоненты скорости решаются фильтром Калмана. Фильтр Калмана – это рекурсивный алгоритм оценки состояния динамической системы на основе последовательности неполных и зашумленных измерений. Он используется для оценки вектора состояния системы на основе предыдущих состояний и измерений. После оценки будущих кадров берется IOU основной истины и оценка, и если IOU меньше порога, то прогнозы отклоняются. Уже после установления ассоциаций алгоритм SORT обновляет состояние каждого отслеживаемого объекта, включая вновь обнаруженную информацию. Это включает в себя обновление положения, скорости и других соответствующих атрибутов объекта[43].

Алгоритм SORT — это простой алгоритм отслеживания в реальном времени, что делает его подходящим для приложений, требующих

высокоскоростного и надежного отслеживания объектов. Данный алгоритм отличается своей простотой применения, а также высокой скоростью работы. SORT не нуждается в детальной настройке, поэтому подходит для абсолютного большинства случаев. Именно эта несложность и обеспечивает его широкое применение в области трекинга объектов – это самый применяемый и «базовый» алгоритм в области компьютерного зрения.

Однако у этого алгоритма есть свои недостатки, в частности он легко теряет объект при мультитрекинге[19] – если в работу взято большое количество объектов, то алгоритм может потерять некоторые из виду, а после идентифицировать их как совершенно новые объекты. Поскольку в работе не подразумевается трекинг слишком большого количества объектов, которые бы заслоняли друг друга, алгоритм SORT может быть применен.

1.2.2 ByteTrack

Современный алгоритм трекинга, который заслуживает отдельного внимания из-за своих положительных особенностей работы с мультитрекингом. ByteTrack — это алгоритм для многокадрового отслеживания объектов в видеопотоках, который особенно эффективен при работе с видео, где объекты могут частично или полностью перекрываться, а также исчезать из поля зрения на короткое время. Данный метод выделяется своей способностью объединять данные о движении и внешнем виде объектов для повышения точности и стабильности отслеживания.

Весь алгоритм ByteTrack[39] можно разделить на несколько основных шагов. Все начинается с детекции объектов, прежде всего, алгоритму требуются координаты обнаруженных объектов - прямоугольные рамки (bounding boxes), которые указывают на положение и размеры объектов. Далее для каждого обнаруженного объекта алгоритм извлекает вектор признаков, который описывает внешний вид предмета. Эти признаки помогают различать объекты друг от друга и отслеживать их в последующих кадрах.

Далее алгоритм использует методы ассоциации данных для сопоставления обнаруженных объектов на текущем кадре с уже отслеживаемыми объектами. Этот процесс обычно включает в себя вычисление расстояний между векторами признаков объектов и применение венгерского метода или других методов оптимизации для нахождения оптимального сопоставления. Далее организуется список активных треков (отслеживаемых объектов), которые поддерживает алгоритм, он же обновляет их состояние на основе результатов ассоциации данных. Если объект не может быть сопоставлен с существующим треком, создается новый трек. Треки, для которых не найдены соответствующие детекции в течение определённого количества кадров, могут быть закрыты.

Уже после обнаружения и трекинга объектов, ByteTrack применяет различные фильтры, такие как фильтр Калмана, для сглаживания траекторий треков и коррекции их положений на основе наблюдаемых движений и изменений во времени. Это позволяет устойчиво отслеживать объекты даже при временных перекрытиях или исчезновениях из поля зрения.

На последнем этапе для оптимизации алгоритм может применять дополнительные методы для улучшения точности отслеживания, включая уточнение границ объектов и корректировку идентификаторов треков для уменьшения ошибок идентификации.

Как становится понятно из определения, ByteTrack отличается от других алгоритмов трекинга своей способностью эффективно работать в сложных условиях, таких как плотные сцены с множеством перекрывающихся объектов, благодаря своей продвинутой системе управления треками и оптимизации. В дополнение к этому, данный алгоритм может обрабатывать кадры (проводить трекинг) с частотой 30 кадров в секунду, что быстрее, чем некоторые другие алгоритмы. Из этих положительных особенностей вытекает главный недостаток алгоритма – он требует больших вычислительных мощностей, поскольку лучше справляется с задачей мультитрекинга. Также из недостатков у этого метода можно выделить детальную настройку под каждую

конкретную задачу – сложности в конфигурировании отталкивают, алгоритм явно не отличается простотой, как SORT.

Таблица 3 – Сравнение Byte Track в контексте других алгоритмов на датасете MOT17[18] и BDD100K[2]

Трекер	С RE-ID	MOT 17			BDD100K			FPS
		MOTA	IDF1	IDs	MOTA	IDF1	IDs	
SORT	-	74.6	76.9	291	30.9	41.3	10067	30.1
DeepSORT	+	75.4	77.2	239	24.5	38.2	10720	13.1
MOTDT	+	75.8	77.6	273	26.7	39.8	14520	11.1
BYTE	-	76.6	79.3	159	39.4	48.9	27902	29.6
BYTE	+	76.3	80.5	216	45.5	54.8	9140	11.8

Исходя из таблицы ByteTrack демонстрирует отличные результаты на таких наборах данных, как MOT17, MOT20, HiEve и BDD100K, достигая высоких значений метрик MOTA, IDF1, и HOTA при скорости работы 30 FPS на одном GPU V100. Особенно стоит отметить достижение 80.3 MOTA, 77.3 IDF1 и 63.1 HOTA на тестовом наборе MOT17.

1.2.3 BoT-SORT

Также из более современных алгоритмов стоит рассмотреть, BoT-SORT[37] - алгоритм слежения за несколькими объектами, который использует преимущества информации о движении и внешнем виде, а также компенсирует движения камеры для повышения точности слежения. Этот подход отличается от предыдущих алгоритмов отслеживания тем, что вектор состояния фильтра Калмана (KF) напрямую оценивает ширину и высоту ограничительного поля, что повышает производительность. Ковариационные матрицы шума процесса и измерений в BoT-SORT выбираются зависящими от времени, в отличие от независимого от времени выбора в предыдущих реализациях SORT. Эти изменения, как было продемонстрировано, привели к более высоким показателям HOTA (Higher Order Tracking Accuracy).

Существенным новшеством в BoT-SORT является техника компенсации движения камеры (CMC). Это очень важно в сценариях с нестабильным

положением камеры, где местоположение ограничительной рамки может значительно смещаться из-за движений камеры, что потенциально может привести к увеличению числа переключений идентификации или ложных срабатываний. BoT-SORT использует глобальную компенсацию движения для более точной оценки движения фона, что делает трекер устойчивым к движению камеры.

Кроме того, BoT-SORT представляет инновационный метод IoU - Re-ID Fusion для объединения информации о движении и внешнем виде, он позволяет реиндентифицировать объект с высокой точностью. Этот метод направлен на повышение устойчивости трекера в переполненных или сложных сценах за счет тщательного управления компромиссом между информацией о движении и внешнем виде для ассоциации объектов.

Резюмируя, данный алгоритм так же, как ByteTrack, точно идентифицирует объекты даже в рамках мультитрекинга. Его дополнительным преимуществом является точность реиндентификации и трекинга (объект не «теряется из виду», а всегда остается опознанным даже в рамках большого количества предметов в задаче). Однако, в отличие от прошлых методов, BoT-SORT работает намного медленнее (обычно на частоте 6-10 кадров в секунду) и, соответственно, так же требует больших вычислительных мощностей. Этот недостаток, с другой стороны, и обеспечивает крайне четкую работу алгоритма.

Таблица 4 – Сравнение BoT-SORT в контексте других алгоритмов на датасете MOT20

Трекер	MOTA	IDF1	HOTA	FP	FN	IDs	FPS
MLT	48.9	54.6	43.2	45660	216803	2187	3.7
FairMOT[6]	61.8	67.3	54.6	103440	88901	5243	13.2
TransCenter[33]	61.9	50.4	–	45895	146347	4653	1.0
TransTrack[34]	65.0	50.4	48.5	27197	150197	3608	7.2
CorrTracker[35]	65.2	69.1	–	79429	95855	5183	8.5
Semi-TCL[28]	65.2	70.1	55.3	61209	114709	4139	–
CSTrack[25]	66.6	68.6	54.0	25404	144358	3196	4.5
GSDT[32]	67.1	67.5	53.6	31913	135409	3131	0.9
SiamMOT[21]	67.1	69.1	–	–	–	–	4.3

RelationTrack[24]	67.2	70.5	56.5	61134	104594	4243	2.7
SOTMOT[40]	68.6	71.4	–	57064	101154	4209	8.5
MAATrack[44]	73.9	71.2	57.3	24942	108744	1331	14.7
OCSORT[20]	75.7	76.3	62.4	19067	105894	942	18.7
StrongSORT++[31]	73.8	77.0	62.6	16632	117920	770	1.4
ByteTrack[39]	77.8	75.2	61.3	26249	87594	1223	17.5
BoT-SORT	77.7	76.3	62.6	22521	86037	1212	6.6
BoT-SORT-ReID	77.8	77.5	63.3	24638	88863	1257	2.4

С точки зрения производительности, BoT-SORT добился заметных результатов на наборах данных MOTChallenge, заняв место среди лучших трекеров на тестовых наборах MOT17 и MOT20 по нескольким метрикам: MOTA, IDF1 и HOTA. Например, на MOT17 он достиг MOTA 80,5, IDF1 80,2 и HOTA 65,0, войдя в число лучших.

1.3 Вывод

На основе анализа области детекции и существующих в ней решений в качестве основного детектора было решено взять модель YOLOv8. Выбор обусловлен тем, что это самая современная модель детекции YOLO на данный момент[22], а также для рассмотренной задачи не требуется слишком точное позиционирование и поиск мелких объектов.

При выборе метода трекинга для работы, безусловно, стоит опираться на сравнительные характеристики разных алгоритмов для подбора наилучшего. Сравнительный анализ приведен в соответствии с вышеупомянутым описанием каждого из методов.

Таблица 5 – Сравнительная характеристика рассмотренных алгоритмов трекинга на датасете MOT20[17]

Трекер	Сложность настройки	MOT20			FPS
		MOTA	IDF1	HOTA	
SORT[43]	–	42.7	45.1	36.1	57.3
ByteTrack[39]	+	77.8	75.2	61.3	17.5
BoT-SORT[37]	+	77.8	77.5	63.3	5.6

Таким образом, наиболее простым и базовым решением является SORT, который использует простой метод ассоциации на основе предсказания следующего состояния объекта с помощью фильтра Калмана и сопоставления с детекциями на основе метрики IoU. В силу своей простоты, SORT может не всегда обеспечивать высокие показатели точности и идентификации, но выигрывает за счет своей простоты и производительности.

BoT-SORT представляет собой улучшенную версию SORT, включающую дополнительные механизмы для более эффективного справления с перекрытиями и обеспечения стабильности идентификаторов (ID) объектов.

ByteTrack выделяется среди трекеров своей способностью эффективно использовать детекции с низкими уверенностями, что позволяет улучшить трекинг в условиях, когда объекты частично перекрыты или имеют низкую видимость, что особенно актуально при трекинге в ночное время суток.

2 Техническая реализация

2.1 Описание системы контроля отгрузки

2.1.1 Подготовка к отгрузке продукции

Отгрузка продукции состоит из следующих этапов:

1) Рулон складируется на «ромашки» для обработки и упаковки (снимаются внешние и внутренние витки, наносится повторно маркировка резиной на рулон, осуществляется обвязка рулона);

2) Формируются карточки на отгрузку в АРМ Бригадира отгрузки. При этом, если состав еще не пришел, то номера вагонов заполняются уникальными цифрами или знаками в рамках одной постановки (1, 2, 3, $, % и прочие знаки);

3) Печатаются бирки на рулон:

- две товарные бирки существующего образца (крепятся по бокам на площадку);

- две идентификационные (контрольные) бирки размером 200х200 мм, содержащие QR-код (максимально большой по площади)

4) Товарные бирки крепятся на рулон по существующим правилам (после упаковки на подготовленные площадки). Идентификационные бирки крепятся на верхнюю сторону рулона (торец), на защитные листы так, чтобы текстовая часть располагалась с краю.

5) После упаковки и нанесения товарной и идентификационной бирок бригадир отгрузки подписывает согласно сформированным формировочным карточкам на каждом рулоне мелом номер вагона (1, 2, 3, $, % и прочие знаки, понятные машинисту крана – своего рода транспортное задание на погрузку в вагоны);

6) Контролер ОТК проверяет соответствие подготовленных к отгрузке рулонов формировочной карточке (сверяет номер рулона и номером

вагона, подписанный мелом). Проверяет внешний вид рулона, его качественные характеристики, соответствие маркировки резиной товарной и идентификационной бирки. Если все соответствует квитирует прием рулонов к отгрузке;

7) Машинист крана транспортирует рулоны в вагоны согласно транспортным заданиям (подписанные мелом цифры и знаки на рулоне, например 1 – первый вагон от головы состава и т.д.);

8) Бригадир отгрузки вписывает в формировочные карточки фактически зашедшие номера вагонов и подтверждает отгрузку, формируя реестр сдачи.

2.1.2 Контроль отгрузки

Контроль правильности отгрузки осуществляется при прохождении составом через портал системы контроля продукции (СКП). Для этого портал оснащается камерой сверху. При необходимости, устанавливается дополнительный прожектор.

Камеры СКП синхронизируются с существующим фотобарьером системы контроля номера вагона (СКНВ), отвечающим за считывание номера вагона с камеры расположенной с борта подвижного состава.

По данным работы СКП и СКНВ фиксируется массив следующих данных по каждому вагону:

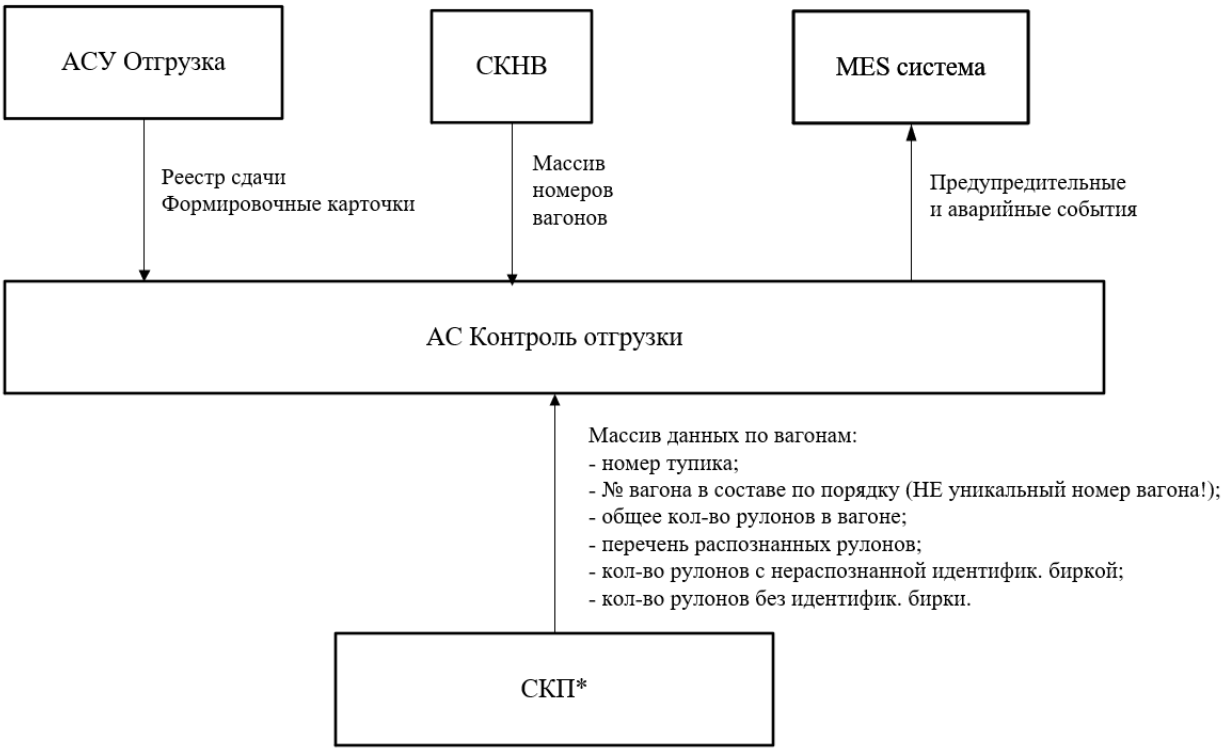
– № вагона по порядку;

– количество рулонов в вагоне;

– количество рулонов с/без идентификационной бирки;

– перечень распознанных рулонов.

Полученная информация передается в систему контроля отгрузки.



* - Разрабатываемая система

Рисунок 6 – Схема работы автоматизированной системы контроля отгрузки

1) Система контроля отгрузки получает данные с СКП, СКНВ (перечень распознанных вагонов), данные с АСУ Отгрузка (реестр сдачи), агрегирует полученные данные и формирует события:

– Предупредительные (не требующие возврата состава в тупик);

– Аварийные (требующие возврат рулона в тупик).

Сформированные события передаются в MES систему предприятия для отображения выявленных событий (таблица …).

Таблица 6 – События системы контроля отгрузки

№	Событие	Описание события
Предупредительные события		
1	Рулон не распознан	рулон в вагоне распознан, наличие идентификационной бирки распознано, QR-код не распознан. В реестре сдачи в вагоне есть нераспознанный рулон
2	Вагон не распознан	Вагон не распознан
Аварийные события		
1	Рулон без идентификационной бирки	рулон в вагоне распознан, идентификационная бирка отсутствует (загружен нетоварный рулон)
2	Несоответствие количества рулонов в вагоне	рулоны в вагоне распознаны, количество рулонов не соответствует реестру сдачи (рулонов больше или меньше)
3	Ошибка размещения рулона	рулон в вагоне распознан, идентификационная бирка распознана, QR-код распознан, рулон по реестру сдачи в другом вагоне
4	Распознан рулон, которого нет в реестре сдачи	рулон в вагоне распознан, идентификационная бирка распознана, QR-код распознан, рулона по реестру сдачи нет

2) Выявленные события отображаются в АРМ MES системе и требуют обязательного квитирования факта просмотра. Факт квитирования логируется с указанием ФИО, отметившего факт просмотра события.

При выявлении аварийных событий, состав должен быть остановлен и возвращен в тупик для выяснения причин возникновения события.

После проверки состав либо возвращается в тупик, и выполняется корректировка отгрузки, либо выполняется ручное квитирование ошибки с АРМ бригадира отгрузки, и состав покидает тупик. Факт квитирования сохраняется в системе.

2.2 Детекция сцепок, рулонов и рулонных бирок

2.2.1 Выбор модели

В качестве модели детекции было решено взять модель YOLOv8. Выбор обусловлен исследованием, проведенным в главе 1, а также текущей популярностью данной модели при решении задачи детекции.

Преимущества модели YOLOv8:

– Наличие официальной open-source библиотеки ultralytics[46].

– Модель представлена в пяти вариациях: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, которые отличаются количеством параметров и временем работы (рисунок 7).

– Возможность работы в режиме реального времени

– Баланс между скоростью работы и производительностью

В связи ограниченными ресурсами GPU и тем, что разработанная модель должна работать в режиме реального времени, решено было обучить и сравнить 3 модели: YOLOv8n, YOLOv8s, YOLOv8m.

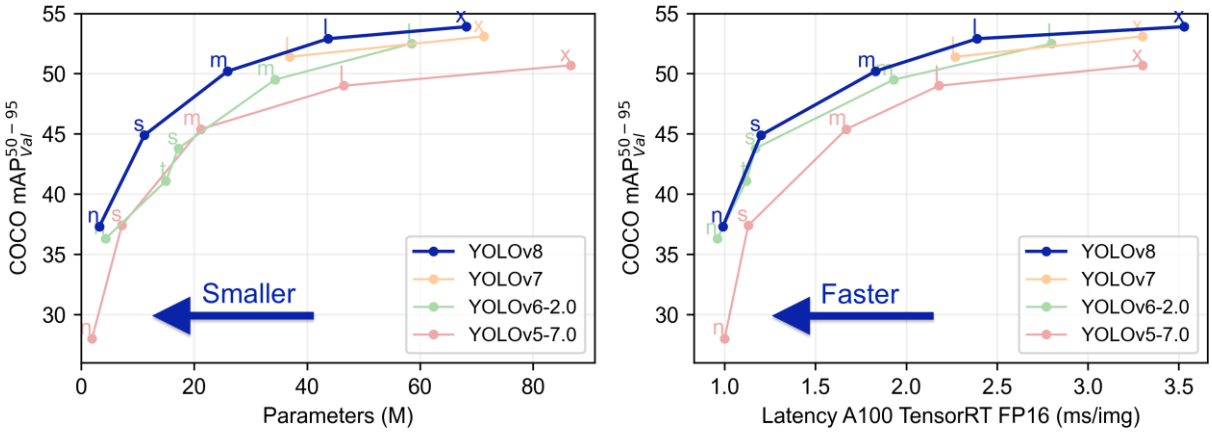


Рисунок 7 – Сравнение версий YOLO

2.2.2 Датасет

2.2.2.1 Сбор данных

Сбор данных происходил в 2 этапа: первичный сбор и сбор с учетом ошибок обученной модели.

Первоначальный этап сбора данных был направлен на формирование исходного набора изображений для последующего обучения модели детекции объектов. В этом процессе было собрано некоторое количество изображений с производственной камеры в местах, где проезжают вагоны с продукцией (вид сверху). Логика сбора данных представляла из себя запись каждого 10 кадра из видеопотока при наличии движения на изображении. Обработка видеопотока и факт наличия движения на видеопотоке был реализован программно с помощью библиотеки opencv.

После первоначального сбора данных модель детекции yolov8n была обучена на этом наборе изображений. Затем эта модель была протестирована на том же видеопотоке на новых объектах. В ходе этого тестирования были выявлены случаи, когда модель не смогла достаточно точно определить объекты из-за низкого уровня уверенности (confidence score).

С учетом недостатков, выявленных в предыдущем этапе, был проведен второй этап сбора данных. В ходе этого этапа внимание было сосредоточено на сценах, где модель демонстрировала низкую уверенность в определении объектов (confidence-score < 0.5). Изображения, соответствующие этим сценам, были сохранены и добавлены в исходный датасет.

После завершения вторичного сбора данных был сформирован итоговый набор изображений, объединяющий изображения как из первичного, так и из вторичного этапов сбора данных. Суммарно было собрано 2278 изображений в различные моменты времени суток.

2.2.2.2 Разметка данных

Разметка данных производилась в программе label-studio. Так как выделение рамок для нескольких объектов на изображении достаточно трудозатратный процесс, то было решено прибегнуть к автоматической разметке, благо label-studio это позволяет. Изначально вручную было размечено примерно 500 изображений, далее на предварительно размеченных данных была обучена модель YOLOv8n и интегрирована в label-studio для автоматической разметки, что заметно ускорило ее процесс.

Таким образом был размечен и сформирован следующий итоговый датасет:

– Формат датасета: YOLO

– Кол-во изображений: 2278 шт.

– Количество экземпляров класса Coupling (сцепка): 1005

– Количество экземпляров класса Roll (рулон):1746

– Количество экземпляров класса LabelRoll (маркировочная бирка): 978

– Количество экземпляров без объектов (фон): 527

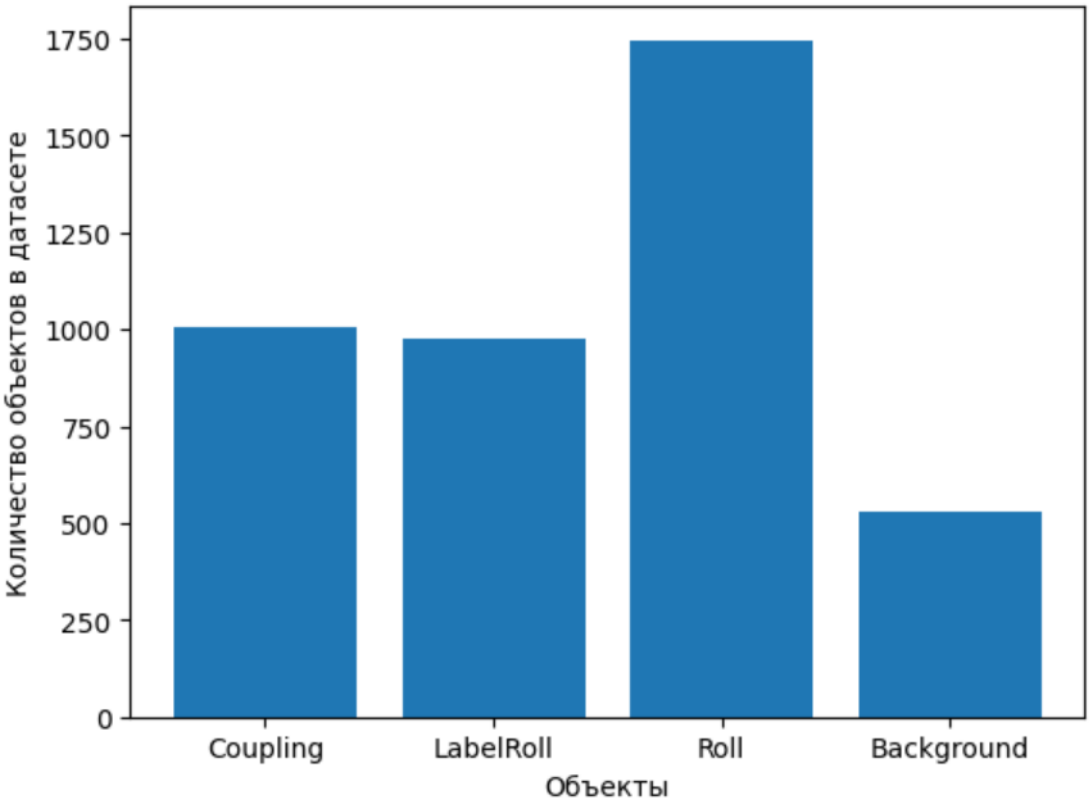


Рисунок 8 – Количественное распределение данных в датасете

Итоговый датасет был поделен на тренировочную и валидационную выборки в соотношении 70% на 30% соответственно.

2.2.3 Обучение модели детекции

Было обучено 3 модели: YOLOv8n, YOLOv8s, YOLOv8m.

Скрипт для обучения был написан на языке Python с использованием библиотеки ultralytics.

Для достижения лучшего качества была применена техника аугментации данных, выбор аугментаций подбирался экспериментально с помощью библиотеки Albumentations.

К обучающей выборке были применены следующие аугментации:

– GaussianBlur. Применяет размытие к изображению с использованием фильтра Гаусса. Фильтр Гаусса работает путем усреднения значений пикселей вокруг каждого пикселя в изображении, применяя весовую

функцию, основанную на распределении Гаусса. Это имитирует эффект размытия, что может быть полезно для обучения модели.

– GaussNoise. Добавляет случайный шум на основе распределения Гаусса к изображению.

– RandomBrightnessContrast. Изменение яркости и контраста изображения.

– RandomSnow. Имитация снега, что может быть очень полезно, так как камера расположена на улице.

– RandomRain. Имитация дождя.

– HorizontalFlip. Зеркальное отображение по горизонтали.

– VerticalFlip. Зеркальное отражение по вертикали.

Параметры обучения модели приведены в таблице 7. Все значения подбирались экспериментально.

Таблица 7 – Параметры обучения модели

Параметр	Значение
Количество эпох (num_epochs)	200
Размер батча (batch_size)	16
Размер изображения	640x640
Optimizer	Adamw
Скорость обучения (lr)	0.0015
Momentum	0.9

Для обучения модель использовалась локальная машина с графическим ускорителем GPU RTX-3060 с видеопамятью 12 Гб.

2.3 Трекинг объектов

Для сравнения алгоритмов трекинга объектов (SORT, BoT-SORT и ByteTrack) была выбрана методика визуального анализа и сравнения их работы на видео. Данная методика позволяет наглядно оценить качество трекинга, а также выявить особенности работы каждого алгоритма в реальных условиях.

В качестве тестового видео были выбраны реальные сцены с потоком вагонов, где двигаются они достаточно быстро. Каждый алгоритм был применен к тем же самым видеофрагментам, и результаты были визуально сравнены. Методика сравнения заключается в правильном отслеживании одного объекта на протяжении всего его нахождения в видео. Правильно затреканным объект считается, если у него не менялся его уникальный ID на протяжении всего его нахождения в видеокадре.

Также каждый алгоритм имеет свои настроечные параметры, для наилучшего инференса были подобраны следующие параметры:

1) SORT:

– max_age = 3. Этот параметр определяет максимальное количество кадров, в течение которых трекер может оставаться необновленным, прежде чем будет удален. Если объект не был обнаружен в течение этого количества кадров, трекер будет удален из списка отслеживаемых объектов.

– min_hits = 1. Этот параметр устанавливает минимальное количество обнаружений, которые трекер должен иметь, прежде чем его маркировка как действительного объекта будет утверждена. Это помогает фильтровать ложные срабатывания и улучшает точность отслеживания объектов.

– Iou_threshold = 0.3. Параметр IOU (Intersection over Union) является пороговым значением, используемым для определения того, считаются ли два прямоугольника, обозначающих объекты в двух последовательных кадрах, одним и тем же объектом или разными. IOU определяется как отношение площади пересечения между двумя прямоугольниками к их общей площади. Если IOU между двумя прямоугольниками превышает установленный порог, эти два прямоугольника будут считаться относящимися к одному и тому же объекту.

2) BoT-SORT:

– track_high_thresh = 0.5. Пороговое значение для первичной ассоциации. Этот порог используется для определения того, считается ли объект соответствующим существующему треку.

– track_low_thresh = 0.1. Пороговое значение для вторичной ассоциации. Если объект не соответствует ни одному существующему треку с помощью первичного порога, вторичный порог используется для поиска наилучшего соответствия.

– new_track_thresh = 0.6. Пороговое значение для инициализации нового трека, если обнаружение не соответствует ни одному треку. Если сходство между обнаружением и треком ниже этого порога, то обнаружение инициирует новый трек.

– track_buffer = 30. Буфер времени для определения, когда треки должны быть удалены. Если трек не обновлялся в течение этого времени, он будет удален из списка отслеживаемых объектов.

– match_thresh = 0.8. Пороговое значение для сопоставления треков. Оно определяет, насколько близко должны находиться два трека, чтобы быть сопоставленными.

3) ByteTrack:

– track_high_thresh = 0.5;

– track_low_thresh = 0.1;

– new_track_thresh = 0.6;

– track_buffer = 30;

– match_thresh = 0.8.

Результаты сравнения алгоритмов с описанными конфигурациями представлены в таблице 8.

Таблица 8 – Сравнение алгоритмов трекинга

Алгоритм трекинга	Сцепка (из 10 шт)	Рулон (из 35 шт)	FPS
SORT	4	15	56
BoT-SORT	10	34	27
ByteTrack	10	34	33

В ходе проверки наилучшим трекером по производительности оказался алгоритм SORT, но худшим по точности трекинга, на протяжении всего видео отслеживаемые объекты не раз меняют свой уникальный ID. Алгоритмы BoT-SORT и ByteTrack гораздо лучше справились с задачей трекинга сцепок и рулонов, но у ByteTrack есть проблемы с ограничивающими рамками объектов в процессе трекинга. Таким образом наилучшим вариантом оказался алгоритм BoT-SORT, который практически не терял отслеживаемые объекты на протяжении всего видео.

3 Анализ результатов

3.1 Результаты обучения модели детекции

Для оценки работы моделей были применены следующие метрики:

– Average Precision (AP) - дает общую оценку качества обнаружения для каждого класса объекта.

– mean Average Precision (mAP) – дает общую оценку качества обнаружения для всех классов.

– Frames Per Second (FPS) – показывает количество кадров в секунду, с которой может работать модель.

Результаты обучения моделей YOLOv8n, YOLOv8s, YOLOv8m с аугментациями и без представлены в таблице 9. Также на рисунке 10 представлена матрица ошибок (confusion matrix) для лучшей модели.

Таблица 9 – Результаты обучения моделей

Модель	AP50%		mAP50%	mAP50-95%	FPS
YOLOv8n	Сцепка	0.96	0.96	0.80	133
	Рулон	0.99			
	Бирка	0.92			
YOLOv8n с аугментациями	Сцепка	0.97	0.96	0.80	133
	Рулон	0.99			
	Бирка	0.93			
YOLOv8s	Сцепка	0.97	0.96	0.80	71
	Рулон	0.99			
	Бирка	0.93			
YOLOv8s с аугментациями	Сцепка	0.97	0.96	0.82	71
	Рулон	0.99			
	Бирка	0.96			
YOLOv8m	Сцепка	0.97	0.97	0.81	56
	Рулон	0.99			
	Бирка	0.96			
YOLOv8m с аугментациями	Сцепка	0.97	0.98	0.83	56
	Рулон	0.99			
	Бирка	0.97			

Исходя из таблицы 8 можно сделать вывод что техника аугментаций дала небольшой прирост к метрикам для каждой модели и повысило ее робастность для различных случаев.

Самой лучшей моделью по качеству оказалась модель YOLOv8m. Процесс обучения данной модели представлен на рисунке 9.

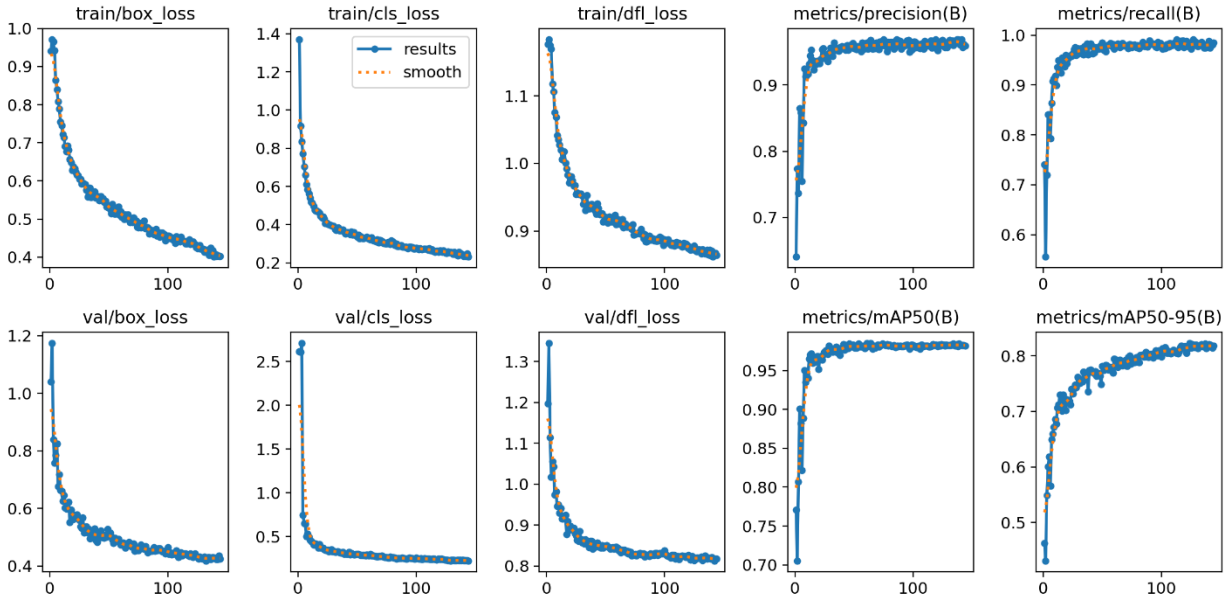


Рисунок 9 – Процесс обучения модели YOLOv8m с аугментациями

Если посмотреть на матрицу ошибок для данной модели (рисунок 10), то можно сделать следующие выводы:

– из 293 сцепок модель предсказала 286 сцепок верно, 7 не обнаружила и обнаружила 3 лишних сцепки;

– из 310 рулонных бирок модель определила 297 бирок верно, 13 не обнаружила и обнаружила 17 лишних бирок;

– из 515 рулонов модель определила 511 штук верно, не обнаружила 4 рулона и обнаружила 4 лишних рулона.

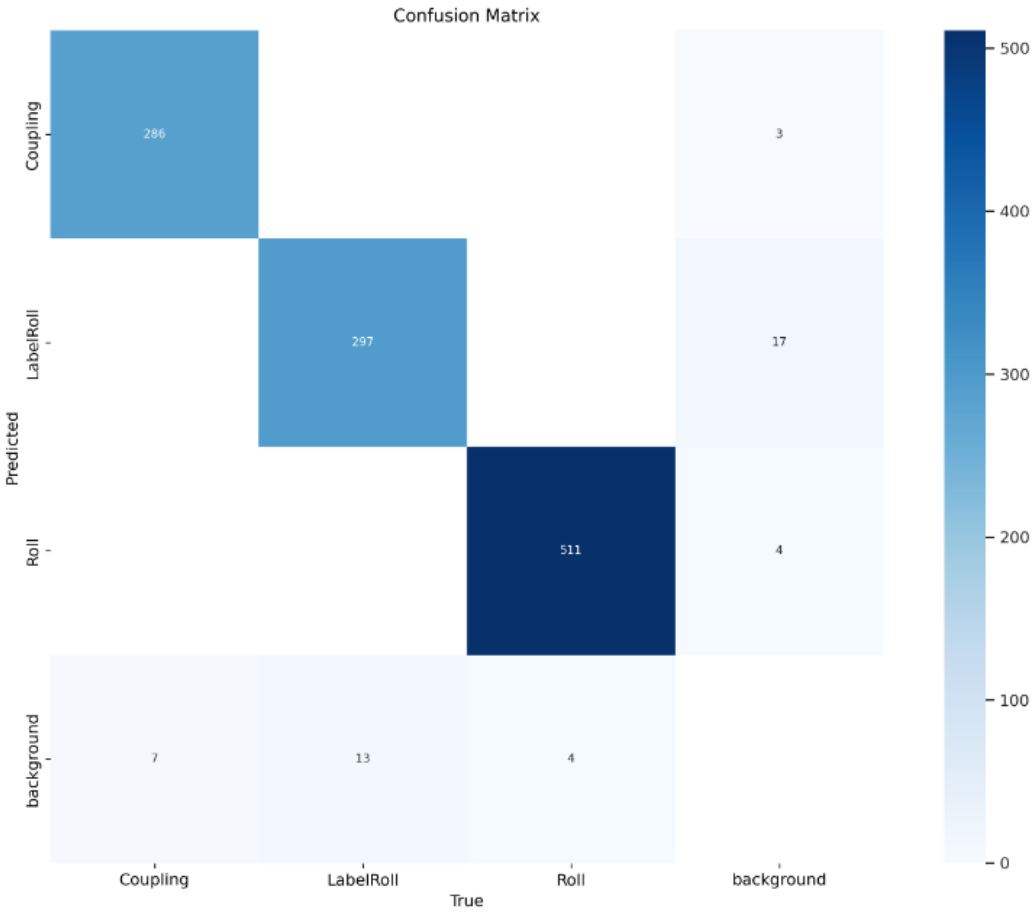


Рисунок 10 – Матрица ошибок для модели YOLOv8m с аугментациями

Рассмотрим примеры изображений, где модель ошибается. Ошибки модели можно классифицировать на 4 категории:

1) Эталонные ошибки – ошибки где модель предсказывает объект правильно, но он не был размечен на изображении (пример рисунок 11).

Рисунок 11 – Пример «эталонных» ошибок модели

2) Объекты, которые находятся в кадре частично (пример рисунок 12)

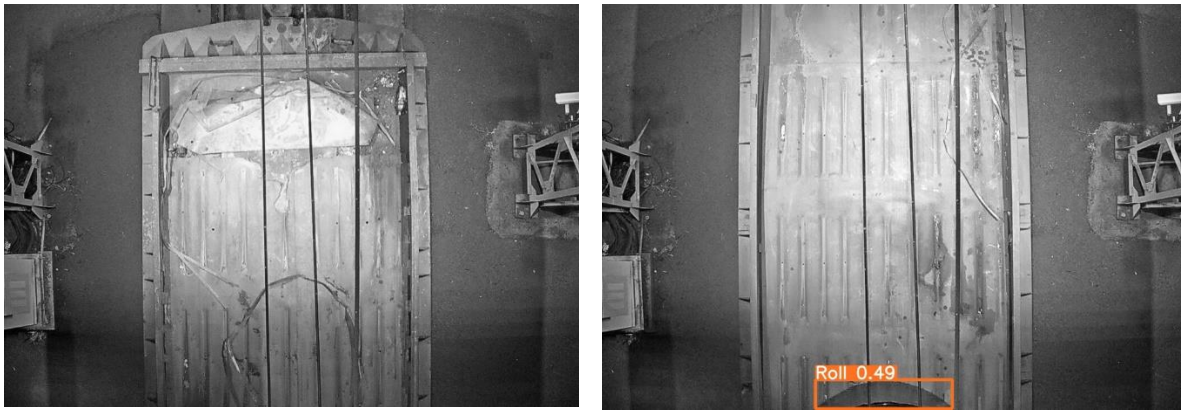


Рисунок 12 – Пример работы модели на изображениях, где объект находится частично

3) Объекты которые плохо видно на изображении (пример рисунок 13)

Рисунок 13 – Пример работы модели на изображениях с плохой видимостью

4) Действительные ошибки модели (пример рисунок 14)

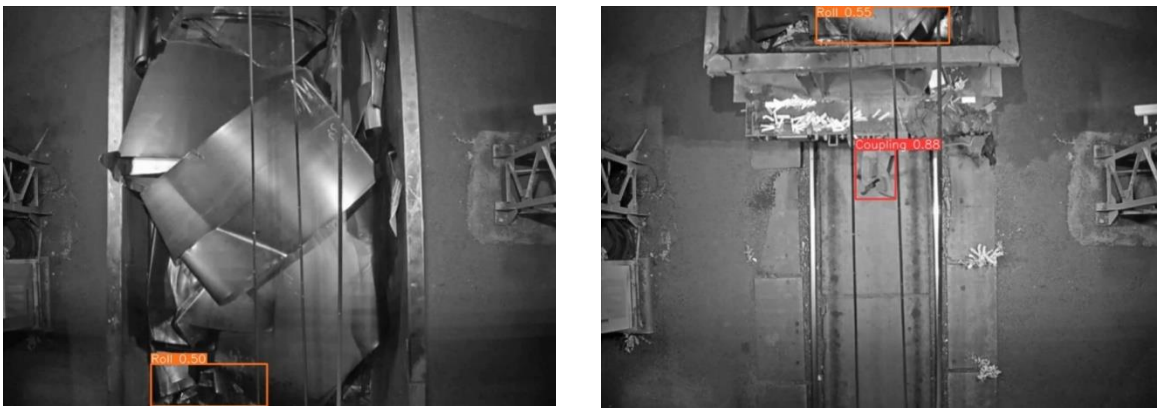


Рисунок 14 – Пример действительных ошибок модели

Обученная модель отлично определяет рулоны и сцепки, и правильно обнаруживает их даже там, где они видны частично. С рулонными бирками модель справляется чуть хуже, скорее всего это связано с тем, что объект маленький и визуально более сложный чем остальные.

Также можно заметить, что места где модель предсказывает объекты неправильно, у этих ложных объектов, как правило, низкий уровень уверенности модели (confidence-score).

Таким образом, экспериментальным путем были подобраны оптимальные пороги (confudence threshold) для каждого объекта (таблица 9), которые должны понизить количество ложных обнаружений.

Таблица 10 – Значения confidence threshold для каждого объекта

Объект	Confidence threshold
Сцепка (Coupling)	0.61
Рулон (Roll)	0.75
Рулонная бирка (LabelRoll)	0.47

3.2 Алгоритм счета сцепок и рулонов для каждого вагона

После разработки модели детекции и выбора оптимального трекера была реализована тестовая программа для подсчета рулонов и рулонных бирок в каждом вагоне.

Алгоритм программы представлен на рисунке 15:

1) Входящий видеопоток покадрово считывается.

2) На изображении задается регион счета для сцепок и рулонов.

3) Как только сцепка пересекла регион, программа регистрирует начало вагона.

4) После регистрации начала вагона идет счет рулонов, где для каждого рулона фиксируется факт наличия или отсутствия рулонных бирок путем сравнения координат объектов.

5) Если на изображении нет никаких объектов в течении 1 минуты, то счетчик обнуляется.

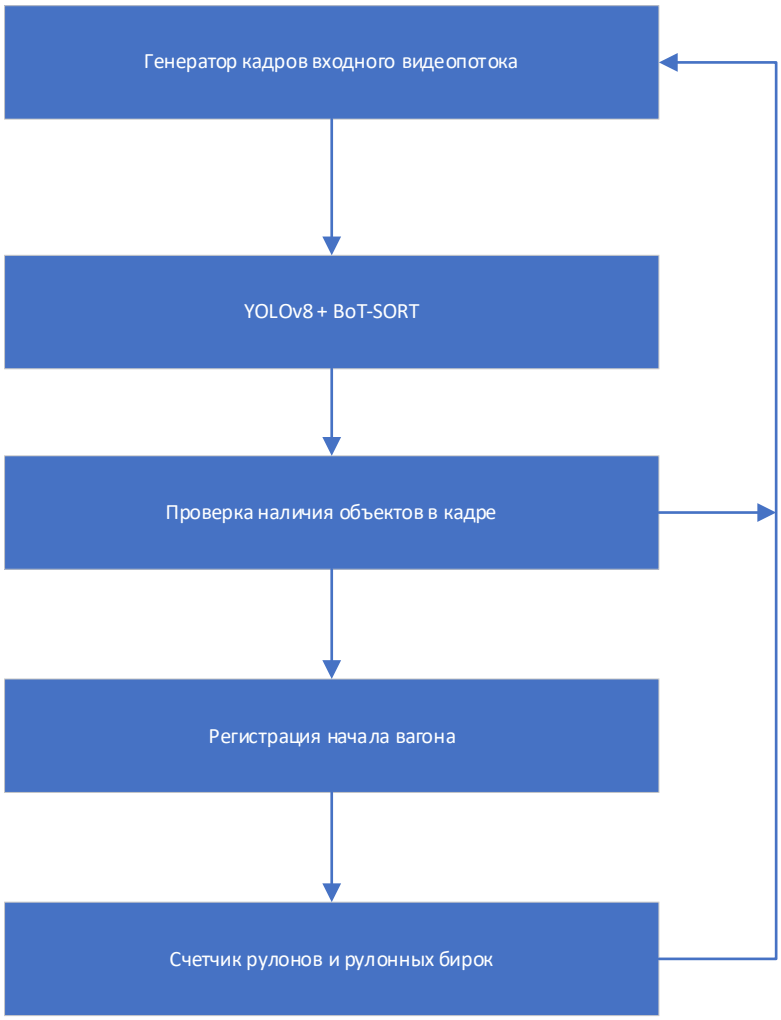


Рисунок 15 – Схема работы разработанной программы

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы поставленная цель была достигнута, а также решены следующие задачи:

– произведен анализ современных решений в области детекции объектов на изображении. Была выбрана модель детекции YOLO в качестве основной модели для решения задачи детекции и 3 алгоритма трекинга SORT, ByteTrack и BoT-SORT;

– собран и размечен собственный датасет для решения задачи детекции сцепок, рулонов и рулонных бирок на изображении размером 2068 изображений;

– обучена модель YOLOv8m с применением техник аугментаций, метрики которой на тестовой выборке составляют mAP50% = 0.98 и mAP50-95% = 0.83, это показывает, что модель может эффективно применяться в промышленности, где требуется надежная детекция объектов в сложных условиях;

– выбран оптимальный алгоритм трекинга объектов по видеопотоку;

– написан прототип программы для формирования отчета по проезжающему составу.

Данный инструмент может быть применен в системе контроля отгрузки продукции любого металлургического предприятия, благодаря чему открываются следующие возможности:

– отслеживание процесса отгрузки продукции в реальном времени;

– автоматическая проверка отгружаемой продукции на соответствие отгрузочным документам;

– оперативное реагирование на ошибки в процессе отгрузки;

– фиксация и хранение истории перемещения грузов в базе данных предприятия;

– предоставление цифрового инструментария для поиска утерянной продукции;

Это позволит сделать процесс отгрузки и перемещения прозрачным для всех уровней управления, снизить количество претензий со стороны клиентов и случаев потери и хищения грузов.

Рекомендации по дальнейшим исследованиям:

– исследование возможностей повышения точности и производительности модели детекции;

– разработка более эффективных методов трекинга объектов;

– адаптация и тестирование системы для других отраслей промышленности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. A Survey of Deep Learning Applications to Autonomous Vehicle Control / S. Kuutti, R. Bowden, Y. Jin, [и др.] arXiv:1912.10773 [cs, eess, stat]. – arXiv, 2019. – URL: http://arxiv.org/abs/1912.10773 (дата обращения: 14.02.2024). – Текст : электронный.

2. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. BDD100K / F. Yu, H. Chen, X. Wang, [и др.] arXiv:1805.04687 [cs]. – arXiv, 2020. – URL: http://arxiv.org/abs/1805.04687 (дата обращения: 20.03.2024). – Текст : электронный.

3. Benchmarking Single Image Dehazing and Beyond / B. Li, W. Ren, D. Fu, [и др.] arXiv:1712.04143 [cs]. – arXiv, 2019. – URL: http://arxiv.org/abs/1712.04143 (дата обращения: 17.05.2024). – Текст : электронный.

4. Bochkovskiy, A. YOLOv4: Optimal Speed and Accuracy of Object Detection. YOLOv4 / A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao arXiv:2004.10934 [cs, eess]. – arXiv, 2020. – URL: http://arxiv.org/abs/2004.10934 (дата обращения: 20.03.2024). – Текст : электронный.

5. Cai, Z. Cascade R-CNN: High Quality Object Detection and Instance Segmentation. Cascade R-CNN / Z. Cai, N. Vasconcelos arXiv:1906.09756 [cs]. – arXiv, 2019. – URL: http://arxiv.org/abs/1906.09756 (дата обращения: 31.01.2024). – Текст : электронный.

6. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking / Y. Zhang, C. Wang, X. Wang [и др.] // International Journal of Computer Vision. – 2021. – Т. 129. – FairMOT. – № 11. – С. 3069-3087.

7. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Faster R-CNN / S. Ren, K. He, R. Girshick, J. Sun arXiv:1506.01497 [cs]. – arXiv, 2016. – URL: http://arxiv.org/abs/1506.01497 (дата обращения: 17.05.2024). – Текст : электронный.

8. Focal Loss for Dense Object Detection / T.-Y. Lin, P. Goyal, R. Girshick, [и др.] arXiv:1708.02002 [cs]. – arXiv, 2018. – URL: http://arxiv.org/abs/1708.02002 (дата обращения: 20.03.2024). – Текст : электронный.
9. Girshick, R. Fast R-CNN / R. Girshick arXiv:1504.08083 [cs]. – arXiv, 2015. – URL: http://arxiv.org/abs/1504.08083 (дата обращения: 17.05.2024). – Текст : электронный.
10. He, Z. Multi-adversarial Faster-RCNN for Unrestricted Object Detection / Z. He, L. Zhang arXiv:1907.10343 [cs]. – arXiv, 2019. – URL: http://arxiv.org/abs/1907.10343 (дата обращения: 01.04.2024). – Текст : электронный.
11. Huang, Y. Autonomous Driving with Deep Learning: A Survey of State-of-Art Technologies. Autonomous Driving with Deep Learning / Y. Huang, Y. Chen arXiv:2006.06091 [cs]. – arXiv, 2020. – URL: http://arxiv.org/abs/2006.06091 (дата обращения: 19.03.2024). – Текст : электронный.
12. Jocher, G. Ultralytics YOLO / G. Jocher, A. Chaurasia, J. Qiu. – 2023. – URL: https://github.com/ultralytics/ultralytics (дата обращения: 20.03.2024). – Текст : электронный.
13. Jocher, G. YOLOv5 by Ultralytics / G. Jocher. – 2020. – URL: https://github.com/ultralytics/yolov5 (дата обращения: 20.03.2024). – Текст : электронный.
14. Kopelowitz, E. Lung Nodules Detection and Segmentation Using 3D Mask-RCNN / E. Kopelowitz, G. Engelhard arXiv:1907.07676 [cs, eess]. – arXiv, 2019. – URL: http://arxiv.org/abs/1907.07676 (дата обращения: 05.05.2024). – Текст : электронный.
15. Loh, Y. P. Getting to Know Low-light Images with The Exclusively Dark Dataset / Y. P. Loh, C. S. Chan arXiv:1805.11227 [cs]. – arXiv, 2018. – URL: http://arxiv.org/abs/1805.11227 (дата обращения: 17.05.2024). – Текст : электронный.

16. Mask R-CNN / K. He, G. Gkioxari, P. Dollár, R. Girshick arXiv:1703.06870 [cs]. – arXiv, 2018. – URL: http://arxiv.org/abs/1703.06870 (дата обращения: 20.03.2024). – Текст : электронный.
17. MOT20: A benchmark for multi object tracking in crowded scenes. MOT20 / P. Dendorfer, H. Rezatofighi, A. Milan, [и др.] arXiv:2003.09003 [cs]. – arXiv, 2020. – URL: http://arxiv.org/abs/2003.09003 (дата обращения: 20.03.2024). – Текст : электронный.
18. MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking. MOTChallenge / P. Dendorfer, A. Ošep, A. Milan, [и др.] arXiv:2010.07548 [cs]. – arXiv, 2020. – URL: http://arxiv.org/abs/2010.07548 (дата обращения: 20.03.2024). – Текст : электронный.
19. Multi-Object Tracking Through Simultaneous Long Occlusions and Split-Merge Conditions. T. 1 / A. G. A. Perera, C. Srinivas, A. Hoogs, [и др.] journalAbbreviation: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. – 2006. – 666 с.
20. Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking. Observation-Centric SORT / J. Cao, J. Pang, X. Weng, [и др.] arXiv:2203.14360 [cs]. – arXiv, 2023. – URL: http://arxiv.org/abs/2203.14360 (дата обращения: 23.03.2024). – Текст : электронный.
21. One More Check: Making «Fake Background» Be Tracked Again. One More Check / C. Liang, Z. Zhang, X. Zhou, [и др.] arXiv:2104.09441 [cs]. – arXiv, 2021. – URL: http://arxiv.org/abs/2104.09441 (дата обращения: 17.02.2024). – Текст : электронный.
22. Real-Time Flying Object Detection with YOLOv8 / D. Reis, J. Kupec, J. Hong, A. Daoudi arXiv:2305.09972 [cs]. – arXiv, 2023. – URL: http://arxiv.org/abs/2305.09972 (дата обращения: 17.05.2024). – Текст : электронный.
23. Redmon, J. YOLOv3: An Incremental Improvement. YOLOv3 / J. Redmon, A. Farhadi arXiv:1804.02767 [cs]. – arXiv, 2018. – URL:

http://arxiv.org/abs/1804.02767 (дата обращения: 20.03.2024). – Текст : электронный.

24. RelationTrack: Relation-aware Multiple Object Tracking with Decoupled Representation. RelationTrack / E. Yu, Z. Li, S. Han, H. Wang arXiv:2105.04322 [cs]. – arXiv, 2021. – URL: http://arxiv.org/abs/2105.04322 (дата обращения: 15.03.2024). – Текст : электронный.

25. Rethinking the competition between detection and ReID in Multi-Object Tracking / C. Liang, Z. Zhang, X. Zhou, [и др.] arXiv:2010.12138 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2010.12138 (дата обращения: 14.02.2024). – Текст : электронный.

26. R-FCN: Object Detection via Region-based Fully Convolutional Networks. R-FCN / J. Dai, Y. Li, K. He, J. Sun arXiv:1605.06409 [cs]. – arXiv, 2023. – URL: http://arxiv.org/abs/1605.06409 (дата обращения: 20.03.2024). – Текст : электронный.

27. Rich feature hierarchies for accurate object detection and semantic segmentation / R. Girshick, J. Donahue, T. Darrell, J. Malik arXiv:1311.2524 [cs]. – arXiv, 2014. – URL: http://arxiv.org/abs/1311.2524 (дата обращения: 16.05.2024). – Текст : электронный.

28. Semi-TCL: Semi-Supervised Track Contrastive Representation Learning. Semi-TCL / W. Li, Y. Xiong, S. Yang, [и др.] arXiv:2107.02396 [cs]. – arXiv, 2021. – URL: http://arxiv.org/abs/2107.02396 (дата обращения: 07.04.2024). – Текст : электронный.

29. Simonyan, K. Very Deep Convolutional Networks for Large-Scale Image Recognition / K. Simonyan, A. Zisserman arXiv:1409.1556 [cs]. – arXiv, 2015. – URL: http://arxiv.org/abs/1409.1556 (дата обращения: 23.02.2024). – Текст : электронный.

30. SSD: Single Shot MultiBox Detector / W. Liu, D. Anguelov, D. Erhan [и др.]. – Текст : электронный // arXiv:1512.02325 [cs]. – 2016. – Т. 9905. – SSD. – С. 21-37. – URL: http://arxiv.org/abs/1512.02325 (дата обращения: 20.03.2024).

31. StrongSORT: Make DeepSORT Great Again. StrongSORT / Y. Du, Z. Zhao, Y. Song, [и др.] arXiv:2202.13514 [cs]. – arXiv, 2023. – URL: http://arxiv.org/abs/2202.13514 (дата обращения: 08.05.2024). – Текст : электронный.

32. Temel, D. Traffic Sign Detection under Challenging Conditions: A Deeper Look Into Performance Variations and Spectral Characteristics / D. Temel, M.-H. Chen, G. AlRegib // IEEE Transactions on Intelligent Transportation Systems. – 2020. – Т. 21. – Traffic Sign Detection under Challenging Conditions. – № 9. – С. 3663-3673.

33. TransCenter: Transformers with Dense Representations for Multiple-Object Tracking. TransCenter / Y. Xu, Y. Ban, G. Delorme, [и др.] arXiv:2103.15145 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2103.15145 (дата обращения: 19.01.2024). – Текст : электронный.

34. TransTrack: Multiple Object Tracking with Transformer. TransTrack / P. Sun, J. Cao, Y. Jiang, [и др.] arXiv:2012.15460 [cs]. – arXiv, 2021. – URL: http://arxiv.org/abs/2012.15460 (дата обращения: 11.03.2024). – Текст : электронный.

35. Wang, Y. Joint Object Detection and Multi-Object Tracking with Graph Neural Networks / Y. Wang, K. Kitani, X. Weng arXiv:2006.13164 [cs]. – arXiv, 2021. – URL: http://arxiv.org/abs/2006.13164 (дата обращения: 29.05.2024). – Текст : электронный.

36. A Survey of Autonomous Vehicles: Enabling Communication Technologies and Challenges / M. N. Ahangar, Q. Z. Ahmed, F. A. Khan, M. Hafeez // Sensors. – 2021. – Vol. 21. – A Survey of Autonomous Vehicles. – № 3. – P. 706.

37. Aharon, N. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. BoT-SORT / N. Aharon, R. Orfaig, B.-Z. Bobrovsky arXiv:2206.14651 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2206.14651 (дата обращения: 20.03.2024). – Text : electronic.

38. Bayesian Multi-object Tracking Using Motion Context from Multiple Objects / J. H. Yoon, M.-H. Yang, J. Lim, K.-J. Yoon. – Текст : электронный // 2015 IEEE

Winter Conference on Applications of Computer Vision 2015 IEEE Winter Conference on Applications of Computer Vision (WACV). – Waikoloa, HI, USA : IEEE, 2015. – P. 33-40. – URL: http://ieeexplore.ieee.org/document/7045866/ (дата обращения: 06.02.2024).

39. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. ByteTrack / Y. Zhang, P. Sun, Y. Jiang, [et al.] arXiv:2110.06864 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2110.06864 (дата обращения: 20.03.2024). – Text : electronic.

40. Improving Multiple Object Tracking with Single Object Tracking / L. Zheng, M. Tang, Y. Chen [et al.]. – Text : electronic // 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – Nashville, TN, USA : IEEE, 2021. – P. 2453-2462. – URL: https://ieeexplore.ieee.org/document/9578284/ (дата обращения: 25.03.2024).

41. Mahaur, B. Road object detection: a comparative study of deep learning-based algorithms / B. Mahaur, N. Singh, K. K. Mishra // Multimedia Tools and Applications. – 2022. – Vol. 81. – Road object detection. – № 10. – P. 14247-14282.

42. Selective Search for Object Recognition / J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, A. W. M. Smeulders // International Journal of Computer Vision. – 2013. – Vol. 104. – № 2. – P. 154-171.

43. Simple Online and Realtime Tracking / A. Bewley, Z. Ge, L. Ott [et al.]. – Text : electronic // 2016 IEEE International Conference on Image Processing (ICIP) / arXiv:1602.00763 [cs]. – 2016. – P. 3464-3468. – URL: http://arxiv.org/abs/1602.00763 (дата обращения: 20.03.2024).

44. Stadler, D. Modelling Ambiguous Assignments for Multi-Person Tracking in Crowds / D. Stadler, J. Beyerer. – Text : electronic // 2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW) 2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW). – Waikoloa, HI, USA : IEEE, 2022. – P. 133-142. – URL: https://ieeexplore.ieee.org/document/9707576/ (дата обращения: 10.01.2024).

45. Survey and Performance Analysis of Deep Learning Based Object Detection in Challenging Environments / M. Ahmed, K. A. Hashmi, A. Pagani [et al.] // Sensors. – 2021. – Vol. 21. – № 15. – P. 5116.

46. Ultralytics. Home. – URL: https://docs.ultralytics.com/ (дата обращения: 17.05.2024). – Text : electronic.

47. You Only Look Once: Unified, Real-Time Object Detection. You Only Look Once / J. Redmon, S. Divvala, R. Girshick, A. Farhadi arXiv:1506.02640 [cs]. – arXiv, 2016. – URL: http://arxiv.org/abs/1506.02640 (дата обращения: 09.01.2024). – Text : electronic.

48. Zhang, Q. Vehicle-Damage-Detection Segmentation Algorithm Based on Improved Mask RCNN / Q. Zhang, X. Chang, S. B. Bian // IEEE Access. – 2020. – Vol. 8. – P. 6997-7004.