«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.04 Программная инженерия
Профиль	Разработка программно-информационных систем
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой А.А. Лисс

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: РАЗРАБОТКА АЛГОРИТМА ОБНАРУЖЕНИЯ СВОБОДНЫХ ПАРКОВОЧНЫХ МЕСТ РЯДОМ С ЖИЛЫМИ ДОМАМИ С ИСПОЛЬЗОВАНИЕМ НЕЙРОСЕТЕЙ

Студентка _____ Костебелова Е. К.
подпись

Руководитель к.т.н., доцент _____ Борисенко К. А.
(Уч. степень, уч. звание) *подпись*

Санкт-Петербург
2024

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ А.А. Лисс
«___»_____2024 г.

Студентка Костебелова Е. К. Группа 0303

Тема работы: Разработка алгоритма обнаружения свободных парковочных мест рядом с жилыми домами с использованием нейросетей

Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» кафедра МО ЭВМ

Исходные данные (технические требования):

Необходимо обучить несколько нейросетей детектированию нового объекта «Свободное парковочное место», сравнить результаты скорости обучения и качества итоговой способности обнаружения каждой модели, после чего сделать выводы о целесообразности использования переобученных нейросетей в выбранной сфере.

Содержание ВКР:

Введение, Обзор предметной области, Формулировка требований к решению поставленной задачи, Программная реализация обучения нейросетей, Исследование и оценка обученных моделей, Заключение

Перечень отчетных материалов: пояснительная записка, иллюстративный материал, приложения А, Б, В, Г, Д, Е, Ж, З.

Дополнительные разделы: Информационный маркетинг

Дата выдачи задания
«___»_____20___ г.

Дата представления ВКР к защите
«___»_____20___ г.

Студентка _____ Костебелова Е. К.

Руководитель к.т.н., доцент _____ Борисенко К. А.
(Уч. степень, уч. звание)

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ А.А. Лисс
«___»_____2024 г.

Студентка Костебелова Е. К. Группа 0303

Тема работы: Разработка алгоритма обнаружения свободных парковочных мест рядом с жилыми домами с использованием нейросетей

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	01.04 – 03.04
2	Обзор предметной области	03.04 – 07.04
3	Формулировка требований к решению поставленной задачи	07.04 – 08.04
4	Программная реализация обучения нейросетей	08.04 – 20.04
5	Исследование и оценка обученных моделей	20.04 – 23.04
6	Информационный маркетинг	23.04 – 26.04
7	Оформление пояснительной записки	01.04 – 02.05
8	Оформление иллюстративного материала	01.04 – 02.05
9	Предзащита	14.05

Студентка _____ Костебелова Е. К.

Руководитель _____ Борисенко К. А.
(Уч. степень, уч. звание)

РЕФЕРАТ

Пояснительная записка 75 стр., 32 рис., 3 табл., 22 ист.

ДЕТЕКТИРОВАНИЕ, YOLO, FASTER R-CNN, ОБУЧЕНИЕ, ПАРКОВОЧНЫЕ МЕСТА.

Объектом исследования являются нейросети YOLO и алгоритмы обучения нейросетей для детектирования предметов.

Предметом исследования является эффективность обучения нейросетей YOLO на обширном датасете фотографий для детектирования свободных парковочных мест во дворе дома. Это включает в себя анализ и сравнение различных параметров обучения, архитектур нейросети, размера датасета, а также оценку точности и скорости работы обученной нейросети.

Цель работы: рассмотрение возможности обучения нейросетей детектировать свободные парковочные места во дворе дома и составление сравнительной характеристики для каждой рассмотренной нейросети.

В ходе выполнения работы были представлены подходы к разработке алгоритма для эффективного поиска и мониторинга свободных парковочных мест во дворах жилых домов с использованием обучения нейросетей.

Для разработки алгоритма был взят датасет, содержащий размеченные изображения парковочных мест во дворе дома. Затем был рассмотрен процесс обучения нескольких нейросетей, включающий разметку большого набора данных и дальнейшую настройку на выбранном датасете. Далее был подробно рассмотрен процесс обучения нейросетей, в результате чего сформировалась сравнительная статистика качества детектирования и скорости обучения каждой нейросети.

ABSTRACT

During the course of the work, approaches to developing an algorithm for effectively searching and monitoring free parking spaces in the courtyards of residential buildings using neural network training were presented.

To develop the algorithm, a dataset was taken containing marked images of parking spaces in the courtyard of a house. Then the process of training several neural networks was considered, including marking a large data set and further adjustment on the selected dataset. Next, the process of training neural networks was examined in detail, as a result of which comparative statistics of the quality of detection and the learning speed of each neural network were formed.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЯП – язык программирования;

IDE – Integrated Development Environment;

IoU – Intersection over Union, отношение площадей ограничивающих рамок;

mAP – mean Average Precision, усреднённая по всем категориям величина средней точности;

AP – Average Precision, величина средней точности;

YOLO – «You Only Look Once», нейронная сеть, которая за один проход прогнозирует для изображения положение ограничивающих прямоугольников и вероятности классификации;

XML – eXtensible Markup Language, расширяемый язык разметки;

JSON – JavaScript Object Notation, текстовый формат обмена данными, основанный на JavaScript;

YAML – Yet Another Markup Language, язык для сериализации данных.

Wandb – Weights & Biases, платформа для отслеживания результатов глубокого обучения

R-CNN – Region-based Convolutional Network

Датасет – набор данных

ВВЕДЕНИЕ

В современных городах проблема дефицита парковочных мест становится все более актуальной, что приводит к усложнению управления транспортными потоками и затрудняет поиск парковочных мест для водителей. Автоматическое обнаружение свободных парковочных мест играет важную роль в сфере управления городской инфраструктурой и повышении эффективности парковки автомобилей.

Целью данной работы является исследование возможности обучения нейронных сетей для детектирования свободных парковочных мест во дворах жилых домов. Применение нейросетей позволяет эффективно и точно обнаруживать объекты на основе анализа фотографий, что может быть весьма полезным для автоматизации систем парковки и улучшения доступности парковочного пространства.

Задачами, поставленными для достижения цели, являются:

- Разбор аналогов и выбор наиболее подходящих нейросетей для обнаружения свободных парковочных мест.

- Создание датасета с исходными данными для обучения моделей.

- Обработка исходных данных, включая удаление шумов и создание обучающей выборки.

- Обучение каждой нейросети на собранном датасете.

- Оценка обученных моделей по общепринятых в сфере обнаружения объектов метрикам.

- Составление сравнительной характеристики по качеству детектирования и скорости обучения различных моделей.

Объектом исследования в данной работе являются нейросети YOLO и Faster R-CNN, использующие алгоритмы детектирования объектов.

Предметом исследования является эффективность обучения нейросетей YOLO и Faster R-CNN на обширном датасете фотографий для детектирования свободных парковочных мест во дворе дома. Это включает в

себя анализ и сравнение различных параметров обучения, архитектур нейросети, размера датасета, а также оценку точности и скорости работы обученной нейросети.

Практическая ценность работы: реализация алгоритма обнаружения свободных парковочных мест во дворах жилых домов. Ожидается, что разработанный алгоритм поможет улучшить жизнь горожан, снизить загруженность улиц автомобилями и сократить время поиска свободного парковочного места.

Дополнительно, проведенное исследование может пролить свет на практическую применимость нейросетей в области автоматического обнаружения объектов и дать понимание потенциальных перспектив в улучшении городской транспортной инфраструктуры.

1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Существующие методы учёта парковочных мест

1.1.1 Математические вычисления

На просторах интернета существует большое количество приложений, где любой водитель может «забронировать» парковочное место, однако такие разработки направлены исключительно на платные парковки, которые не пользуются спросом для длительной стоянки автомобиля.

Рассмотрим несколько примеров приложений, оказывающих услуги про бронированию парковочного места на платной парковке в городе Санкт-Петербурге.

Приложение «Парковки Санкт-Петербург» - мобильное приложение для поиска парковки в г. Санкт-Петербург [1]. Оно позволяет: находить парковки разных типов и получать дополнительную информацию о парковках (название, адрес, стоимость, вместимость и др.). Есть возможность выбрать любое платное парковочное место в Санкт-Петербурге и оплатить его. Отсутствуют данные о бесплатных парковках. Основная направленность приложения – учёт платных парковочных мест. Исходя из последних отзывов – приложение часто даёт сбой, что вызывает понижение спроса. Детектирование свободных парковочных мест происходит за счёт вычисления оплаченных мест на парковке, никакой алгоритм не применяется.

Приложение «Парковки России» - персональный мобильный паркомат с набором полезных функций [2]. Предоставляется информация о городских и коммерческих парковках, их тарифах. Ведётся учёт свободных платных парковочных мест. Рассматриваются только платные парковки. Детектирование свободных парковочных мест происходит за счёт вычисления оплаченных мест на парковке, никакой алгоритм не применяется.

Приложение «GetPark: паркшеринг сервис» - сервис помогает арендовать паркинг в нескольких жилых комплексах Санкт-Петербурга [3]. Доступны на выбор 7 жилых комплексов. Рассматриваются только платные

парковки. Ведётся учёт свободных парковочных мест. Детектирование свободных парковочных мест происходит за счёт вычисления оплаченных мест на парковке, никакой алгоритм не применяется.

Каждый из рассматриваемых аналогов не использует никакого алгоритма детектирования, только лишь математические вычисления по формуле: *кол-во свободных мест = общее кол-во мест – кол-во занятых мест*

Самый главный недостаток рассматриваемого подхода – невозможность вести учёт бесплатных парковок, что приводит к необходимости создания нового метода контроля парковочных мест.

1.1.2 Техническое определение

Рассмотрим аналоги с использованием широко известных и старых технологий для определения свободных парковочных мест.

Парковка ТРК «Европолис» - в ТРК присутствует подземная парковка на 1300 мест, при этом над каждым парковочным местом расположена лампочка, которая горит зелёным цветом, в случае, если место свободно, и красным цветом, если парковочное место занято автомобилем [4]. Позволяет издалека автолюбителям видеть свободные места. Заранее неизвестно количество свободных мест. Нет автоматизации, детектирование происходит механически, с помощью датчиков, установленных на каждом парковочном месте.

Сервис «Яндекс.Парковки» - приложение для поиска свободного места на парковке, которым можно воспользоваться, загрузив приложение «Яндекс.Навигатор» [5]. Система определяет недавно освободившиеся места. На карте появляется зеленый значок, который сигнализирует о том, что недавно транспортное средство уехало. Значок остается активен в течение пяти минут. Рассматриваются платные и бесплатные парковки. Неизвестно количество свободных мест на парковках. Алгоритм определения недавно освободившихся мест использует GPS-данные пользователей приложения и

информации с датчиков, встроенных в смартфон. Алгоритм не всегда работает корректно, скорее всего за счёт неточного определения геолокации владельца смартфона или отсутствия постоянного доступа к GPS-данным телефона.

Рассмотренные аналоги обладают большим количеством недостатков, хоть и частично выполняют заявленные функции – учёт бесплатных парковочных мест. В случае с парковками в торговых центрах проблема очевидна – не предоставляется возможным заранее установить наличие свободного места, а также технология распространяется только на один конкретный торговый центр.

Если говорить о сервисе от компании «Яндекс» - то все недостатки упирается в ненадёжность определения геопозиции, что подтверждается негативными отзывами пользователей о частой неработоспособности сервиса.

1.1.3 Применение нейросетей

Технология обучения нейросетей широко используется во многих сферах деятельности, однако в качестве распознавания парковочных мест её использовала лишь одна компания «Интерсвязь».

Сервис «Умные парковки» - это система для определения наличия свободных мест на парковках городов Челябинской области [6]. Воспользоваться ей можно загрузив мобильное приложение «Интерсвязь». Сервис показывает ближайшие незанятые парковки, используя GPS устройство пользователя, для распознавания свободных мест применяется нейронная сеть. При желании можно открыть изображение с парковки и узнать местоположение каждого свободного места. Обновление изображения происходит каждые тридцать секунд. Недостатком данного приложения является то, что детектирование в случаях некорректно занятого места или при полностью свободной парковке происходит некорректно. Неизвестно какая именно нейронная сеть была обучена, сколько эпох было проведено, а также какого размера и содержания использовался датасет. Дополнительно можно

добавить к списку изьянов тот факт, что сервис «Умные парковки» («Интерсвязь») решает проблему поиска бесплатных парковочных мест во дворах дома только в городах в пределах Челябинской области.

Исходя из анализа найденного аналога можно сделать вывод, что попытка внедрить технологию обучения нейросетей детектировать свободные парковочные места уже была, однако количество недостатков свидетельствует о необходимости разработки более устойчивого алгоритма на базе нейросети и сравнении качества обучения нескольких существующих нейронных сетей, направленных на распознавание объектов.

1.1.4 Сравнение методов

Можно выделить следующие критерии сравнения аналогов:

- Учёт бесплатных парковок

  Учет бесплатных парковок является одним из важных критериев сравнения, поскольку большинство автолюбителей паркует свои автомобили рядом с домом, что создаёт проблему поиска свободного места, которое никак нельзя забронировать.

- Актуальные данные о количестве свободных парковочных мест

  Актуальные данные о количестве свободных мест позволят автолюбителям заранее планировать маршрут к той или иной парковке, потому что, если количество парковочных мест небольшое – возможно стоит поискать другой вариант парковки.

- Применение детектирования по фото или видео

  Детектирование предметов по фото или видео является основным критерием, поскольку идея обучения нейросети строится именно на анализе кадров и последующем обнаружении свободных парковочных мест на них.

Сравнение аналогов по критериям представлено в таблице 1.

Таблица 1 – Сравнение аналогов по критериям

Критерий/Аналог	Учёт бесплатных парковок	Актуальные данные о количестве свободных парковочных мест	Применение детектирования по фото или видео
«Парковки Санкт-Петербург»	Не ведётся	Доступны	Не применяется
«Яндекс.Парковки»	Ведётся	Не всегда актуальны	Не применяется
«Умные парковки»	Ведётся	Доступны	Применяется, но присутствуют частые затруднения
«Парковки России»	Не ведётся	Доступны	Не применяется
«GetPark: паркшеринг сервис»	Не ведётся	Доступны	Не применяется

Проанализировав данные, представленные в табл. 1, можно сделать вывод, что только один из рассматриваемых аналогов использует алгоритм детектирования свободных парковочных мест по фото или видео, однако Сервис «Умные парковки» («Интерсвязь») решает проблему поиска бесплатных парковочных мест во дворах дома только в городах в пределах Челябинской области, а также сервис может показывать некорректную работу при определённых условиях.

Приложения «Парковки Санкт-Петербург», «Парковки России» и «GetPark: паркшеринг сервис» обладают примерно идентичным функционалом, за исключением того, что «Парковки России» работает не

только на территории Санкт-Петербурга, но и в нескольких городах России, а приложение «GetPark: паркшеринг сервис» распространяется только на платные парковки в нескольких жилых комплексах Санкт-Петербурга. Также они не используют никакого алгоритма детектирования, основываясь просто на математических вычислениях оплаченных парковочных мест.

Сервис «Яндекс.Парковки» является вторым сервисом, использующим алгоритм для детектирования парковочных мест, однако он основывается на GPS-данных автолюбителей, а также распространяется только на пользователей приложения.

Таким образом, для устранения недостатков вышеописанных аналогов, необходима разработка алгоритма детектирования свободных парковочных мест во дворе дома по фото или видео с помощью нейросети. Данный алгоритм позволит более точно и своевременно определять освободившиеся места на бесплатных парковках.

1.2 Нейросети

1.2.1 Модели YOLO

YOLOv8 – это последняя версия YOLO от Ultralytics [7]. YOLOv8 представляет собой передовую, современную (state-of-the-art (SOTA)) модель, основанную на успехе предыдущих версий, с новыми функциями и улучшениями для повышения производительности, гибкости и эффективности. YOLOv8 поддерживает полный спектр задач искусственного интеллекта, включая детектирование (detection), сегментацию (segmentation), оценку позы (pose estimation), отслеживание (tracking) и классификацию (classification).

YOLOv8 использует сверточную нейронную сеть, которую можно разделить на две основные части: позвоночник (backbone) и голову (head).

Позвоночник – это модифицированная версия архитектуры CSPDarknet53. Эта архитектура состоит из 53 сверточных слоев и использует

частичные межэтапные соединения для улучшения информационного потока между различными слоями. Голова YOLOv8 состоит из нескольких сверточных слоев, за которыми следует ряд полносвязных. Эти слои отвечают за прогнозирование ограничивающих прямоугольников (bounding boxes), оценки объектности (objectness scores) и вероятности классов для объектов, обнаруженных на изображении. Одной из ключевых особенностей YOLOv8 является использование механизма самоконтроля в голове сети.

Этот механизм позволяет модели сосредоточиться на разных частях изображения и устанавливать важность различных признаков в зависимости от их применимости к задаче. Еще одной важной особенностью YOLOv8 является его способность выполнять многомасштабное обнаружение объектов. Модель использует пирамидальную сеть признаков (feature pyramid network) для обнаружения объектов разных размеров и масштабов на изображении. Она состоит из нескольких слоев, которые обнаруживают объекты в разных масштабах, что позволяет модели обнаруживать большие и маленькие объекты на изображении.

На рис. 1 представлена подробная визуализация архитектуры нейронной сети YOLOv8.

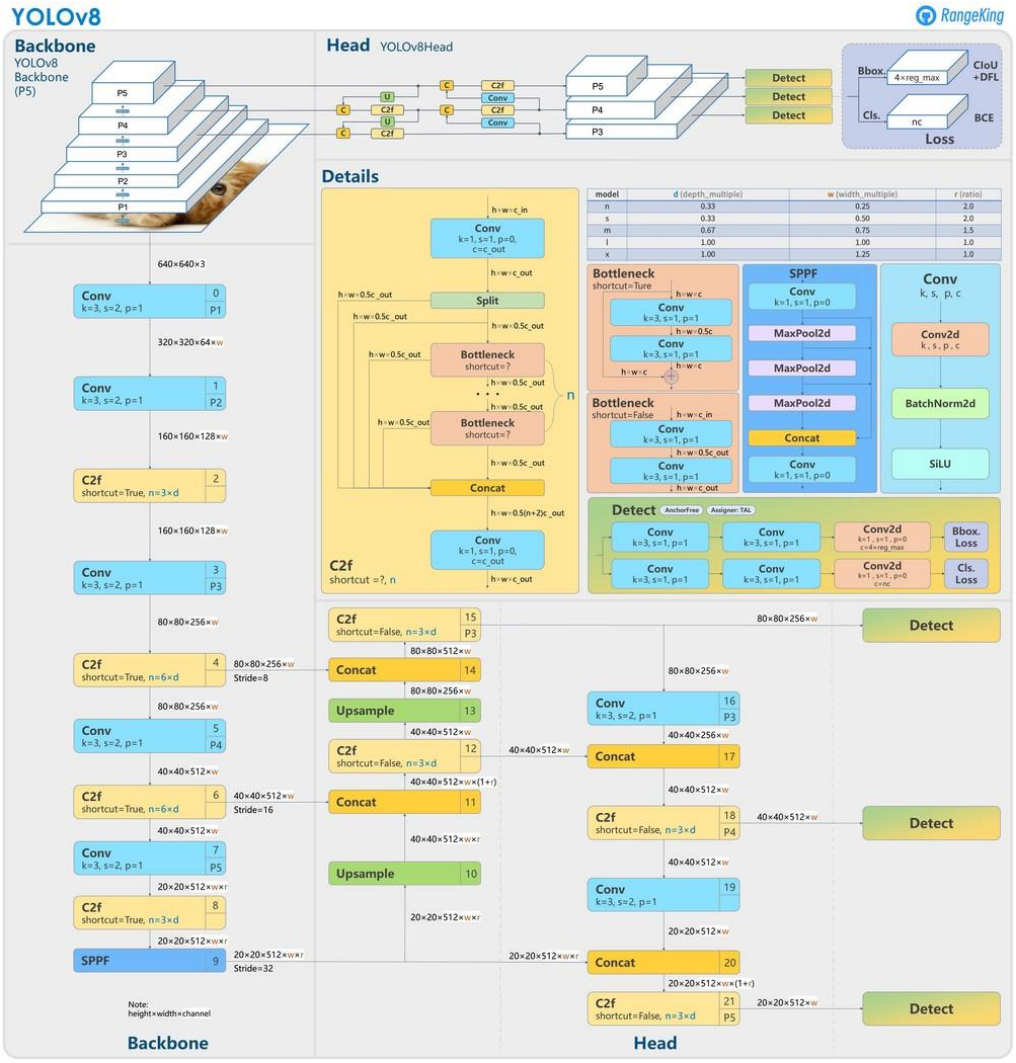

Рисунок 1 – Архитектура нейронной сети YOLOv8

На официальном репозитории Ultralytics представлены графики по результатам сравнения последних версий алгоритма YOLO [8]. Результаты можно увидеть на рис. 2.

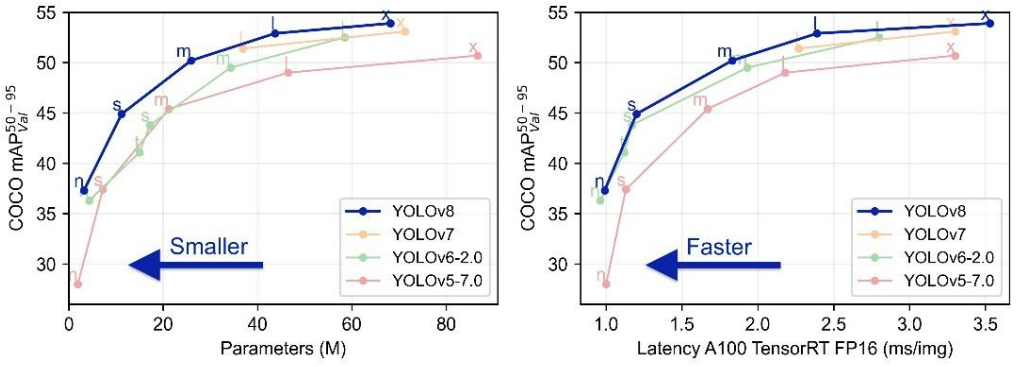

Рисунок 2 – Графическое сравнение моделей YOLOv5 — YOLOv8

Исходя из графиков можно сделать вывод о том, что лучшей по эффективности моделью является YOLOv8.

Именно поэтому в качестве нейросети для обучения детекции свободных парковочных мест было решено использовать модели YOLOv8, поскольку это новейшая версия алгоритма YOLO, которая добавляет обширный функционал и не была использована для решения задачи детектирования свободных парковочных мест во дворе дома.

1.2.2 Модели Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Network) - это один из передовых алгоритмов компьютерного зрения, предназначенный для обнаружения объектов и их границ на изображениях [9]. Этот метод сочетает в себе два основных компонента: сверточную нейронную сеть для извлечения признаков и детектор объектов на основе регионов для точного выявления объектов.

В Faster R-CNN применяется типичная архитектура нейронной сети для сегментации изображений [10]. Сначала изображение проходит через сверточные слои, которые извлекают признаки с разных уровней детализации. Затем используется Region Proposal Network (RPN), который предлагает области, где могут находиться объекты. RPN генерирует области кандидатов (анкоры), которые потенциально содержат объекты, и оценивает вероятность их наличия.

Далее эти области передаются в RoI (Region of Interest) pooling слой, который позволяет извлечь фиксированное количество признаков из каждой области для дальнейшего анализа. Полученные признаки проходят через полносвязные слои и классификатор для точного обнаружения объектов и их классификации.

Основной принцип работы Faster R-CNN заключается в том, что сначала сеть предсказывает области, где могут находиться объекты, а затем

классифицирует их. Это позволяет значительно увеличить скорость обнаружения объектов за счет оптимизации процесса генерации регионов. Схематически принцип работы изображен на рис. 3.

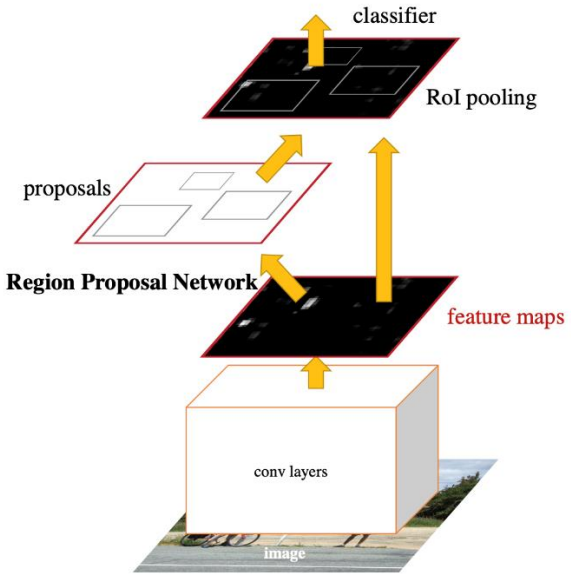

Рисунок 3 – Принцип работы Faster R-CNN

Применение Faster R-CNN для обнаружения свободных парковочных мест рядом с жилыми домами может значительно упростить процесс поиска и определения доступных мест и повысить удобство для водителей. Анализ изображений с помощью нейросетей поможет эффективно автоматизировать эту задачу и повысить ее точность.

1.3 Критерии оценки эффективности обучения

В задачах классификации с локализацией и детекции объектов для определения достоверности местоположения ограничивающей рамки в качестве метрики чаще всего используется отношение площадей ограничивающих рамок (англ. Intersection over Union, IoU):

$$IoU = \frac{S(A \cap B)}{S(A \cup B)},$$

где A и B – предсказанная ограничивающая рамка и настоящая ограничивающая рамка соответственно [11]. IoU равно нулю в случае

непересекающихся ограничивающих рамок и равно единице в случае идеального наложения. Пример можно наблюдать на рис. 4.

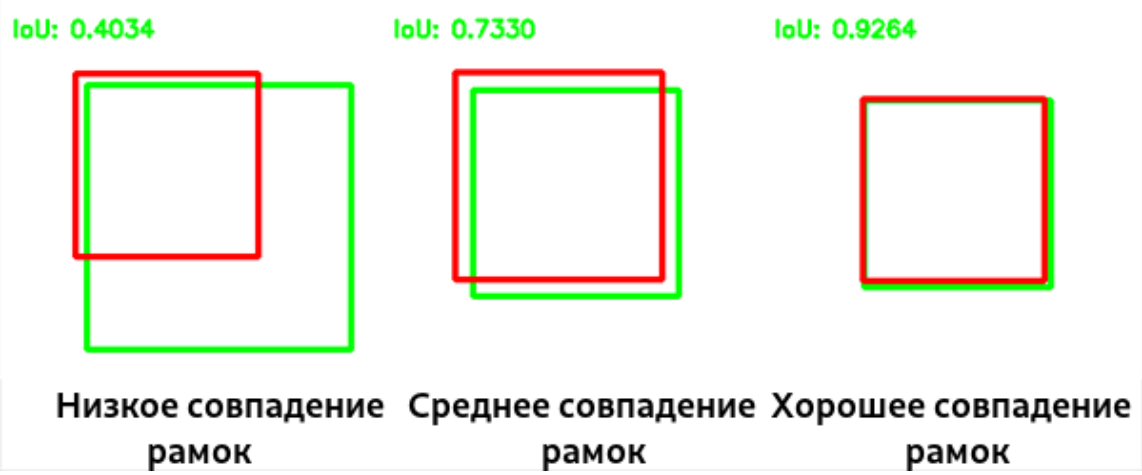

Рисунок 4 – Примеры влияния взаимного положения ограничивающих рамок на метрику IoU

Также в задачах детекции объектов в качестве метрики используется mAP (англ. mean Average Precision) — усреднённая по всем категориям величина средней точности (англ. Average Precision, AP)

$$AP = \int_0^1 p(r)dr,$$

где p – точность, r – полнота из предположения, что ограничивающая рамка определена верно, если $IoU \geq 0.5$. Поскольку точность и полнота находятся в промежутке от 0 до 1, то AP, а следовательно, и mAP также находится в пределах от 0 до 1. На практике, AP часто считают по точкам, значения полноты которых равномерно распределены в промежутке [0;1]:

$$AP_c = \frac{1}{11} \times (AP_c(0) + AP_c(0.1) + \cdots + AP_c(1))$$

$$mAP = \overline{AP_c}$$

Зависимость точности от полноты, необходимая для вычисления AP можно увидеть на рис. 5.

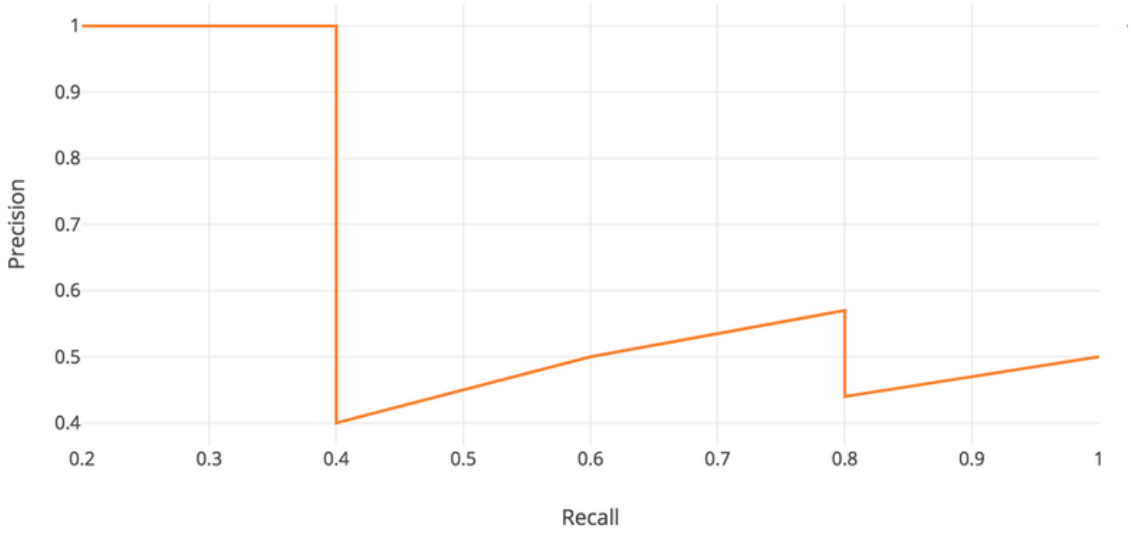

Рисунок 5 – Зависимость точности от полноты, необходимая для вычисления Average Presicion (AP)

train_loss (потери при обучении) — это показатель, используемый для оценки того, насколько модель глубокого обучения соответствует данным обучения [12]. То есть он оценивает ошибку модели на обучающем наборе. Важно уточнить, что обучающий набор — это часть набора данных, используемая для первоначального обучения модели. В вычислительном отношении потери при обучении рассчитываются путем взятия суммы ошибок для каждого примера в обучающем наборе. Также важно отметить, что потери на обучение измеряются после каждой партии. Обычно это визуализируется путем построения кривой потерь при обучении.

Напротив, val_loss (потери при проверке) — это показатель, используемый для оценки производительности модели глубокого обучения в наборе проверки. Набор проверки — это часть набора данных, предназначенная для проверки производительности модели. Потери при проверке аналогичны потерям при обучении и рассчитываются из суммы ошибок для каждого примера в наборе проверки.

2. ФОРМУЛИРОВКА ТРЕБОВАНИЙ К РЕШЕНИЮ И ПОСТАНОВКА ЗАДАЧИ

2.1 Постановка задачи

Необходимо обучить несколько нейросетей детектированию нового объекта «parking», сравнить результаты скорости обучения и качества итоговой способности обнаружения каждой модели, после чего сделать выводы о целесообразности использования переобученных нейросетей в выбранной сфере.

2.2 Требования к решению

Результат должен обладать следующим набором свойств:

1. Каждая модель должна быть обучена на обширном датасете фотографий
2. Проведено большое количество итераций обучения, для более корректной работы каждой модели и для составления более практичной статистики
3. Алгоритм должен детектировать свободные парковочные места
4. Результатом работы каждой обученной нейросети является выделение рамкой свободных парковочных мест, обнаруженных на поданном алгоритму на вход кадре
5. В конце необходимо провести сравнение всех обученных моделей, для того чтобы выявить наиболее работоспособный вариант алгоритма

2.3 Обоснование требований к решению

1. Обучение на обширном наборе данных поможет моделям адекватно оценивать разнообразные сценарии свободных парковочных мест возле жилых домов и повысит общую точность детектирования.

2. Большое количество итераций обучения обеспечит моделям достаточное время для "усвоения" множества различных вариантов свободных парковочных мест, что повысит общую производительность и точность результатов.
3. Обнаружение свободных парковочных мест является ключевым функциональным требованием реализуемого алгоритма, поскольку именно это позволит оптимизировать процесс нахождения свободного места для водителей.
4. Выделение рамкой на изображении свободных парковочных мест, обнаруженных алгоритмом, позволяет визуально представить информацию о расположении парковочных мест и упрощает восприятие результатов.
5. Сравнение всех обученных моделей необходимо для выбора наиболее эффективного варианта алгоритма, учитывая его точность детектирования, скорость работы и другие критерии.

2.4 Обоснование выбора метода решения

После анализа существующих методов учета парковочных мест (аналогов), можно отметить, что имеющиеся подходы к обнаружению свободных парковок обладают рядом недостатков. В частности, часто применяемые алгоритмы оказываются недостаточно эффективными при детектировании, что снижает общую точность системы. Кроме того, многие из них не используют алгоритмы распознавания парковочных мест на основе нейронных сетей по изображениям или видео.

Использование нейронных сетей, несмотря на свои технологические преимущества, представляет собой относительно новый подход в данной области. Возможно, нарастание популярности этой методики в будущем будет обусловлено не только постоянным улучшением алгоритмов машинного

обучения, но и увеличением доступности вычислительных ресурсов для обучения нейросетей.

Однако необходимо учитывать, что использование нейронных сетей требует значительных временных затрат на подготовку датасета и обучение модели. Правильно сконструированный датасет и настроенные параметры для обучения могут повлиять на результат работы модели, позволяя ей адаптироваться к конкретной задаче обнаружения свободных парковочных мест с высокой точностью.

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ОБУЧЕНИЯ НЕЙРОСЕТЕЙ

3.1 Обучение моделей YOLO

3.1.1 Настройка ClearML

Из-за того, что вручную отслеживать эксперименты с глубоким обучением очень неудобно, было принято решение воспользоваться интеграцией ClearML, которую Ultralytics YOLOv8 поддерживает по умолчанию [13]. Установка интеграции была осуществлена путем выполнения команды pip install clearml, а затем ее инициализации.

Прежде чем начать использовать ClearML, необходимо было зарегистрироваться на платформе и создать рабочее пространство (Workspace). Для этого были выполнены соответствующие шаги, и затем рабочее пространство было проинициализировано с помощью команды clearml-init, вводом секретных ключей, обеспечивающих безопасное взаимодействие с платформой.

С момента инициализации ClearML все обучающие эксперименты YOLOv8, запускаемые в терминале, автоматически регистрируются на панели управления ClearML. Это обеспечивает удобное и централизованное отслеживание всех этапов обучения модели, а также позволяет анализировать результаты и принимать решения на основе собранной статистики. Пример можно увидеть на рис. 6.

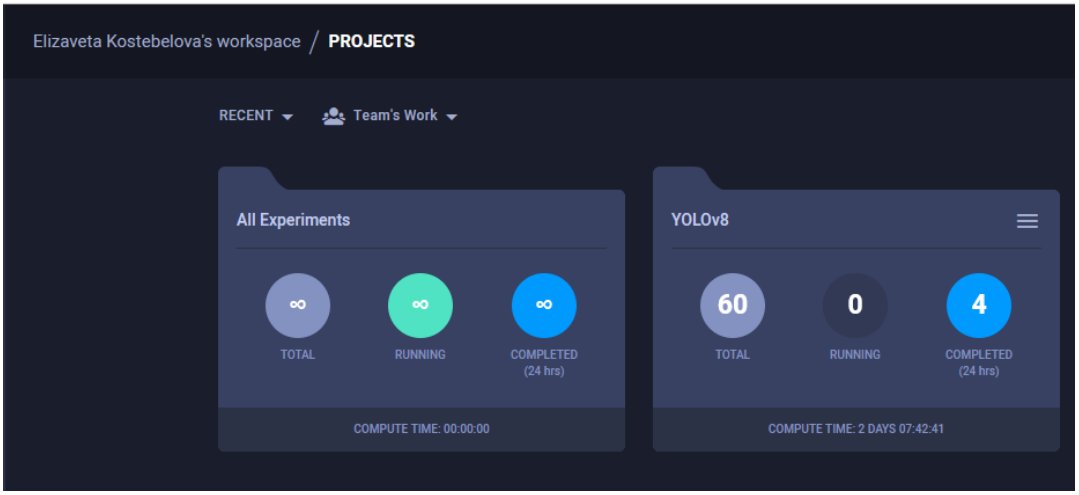

Рисунок 6 – Скриншот настроенной и уже использованной панели управления ClearML.

3.1.2 Подготовка и обработка датасета

Так как необходимо проверить реальную работоспособность нейросети YOLO детектировать любое свободное парковочное место, то размер датасета должен быть довольно обширным, поэтому было принято решение воспользоваться наборами изображений, подходящих по параметрам, из интернета.

Выбранный датасет содержит наборы изображений с нескольких камер под разным углом, а также с разными погодными условиями: солнечно, дождливо, облачно. Для более успешного обучения моделей детектировать свободные парковочные места, было принято решение создать для обучения датасет, включающий в себя большое количество фотографий с каждой камеры в разные погодные условия. Примеры изображений, входящих в сформированный датасет для обучения, можно наблюдать на рис. 7-9.

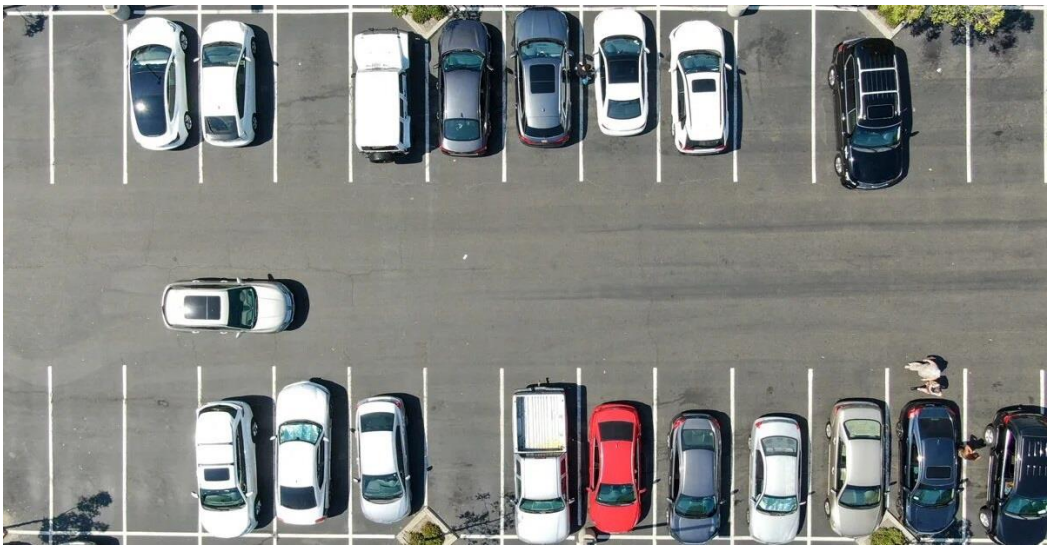

Рисунок 7 – Изображение во время солнечной погоды

Рисунок 8 – Изображение во время дождливой погоды

Рисунок 9 – Изображение во время облачной погоды

Главная проблема при подготовке и обработке найденного датасета заключается в приведении данных в необходимый для обучения YOLO формат, а именно: необходимо создать два каталога с изображениями для тренировки и проверки соответственно. Дополнительно для каждого изображения должен быть составлен текстовый файл, содержащий метки с нормализованными xmin, ymin, width, height, которые $\in [0; 1]$. Это необходимо для корректной обработки обучающих изображений и выявлении признаков для успешного обучения.

Важным моментом является составление датасета YAML для определения путей к изображениям и имен классов. Файл parking_v8.yaml находится в корневом каталоге проекта, для удобства и простоты доступа к нему при написании скрипта на Python для обучения модели YOLO. Содержимое этого YAML файла см. рис. 10.

```yaml
path: parking_dataset_v8/ # путь к исходным данным датасета
train: 'train/images' # путь к тренировочным изображениям
val: 'valid/images' # путь к проверочным изображениям
names: # имена классов для обнаружения
  0: 'parking'
```

Рисунок 10 – Содержимое файла parking_v8.yaml

YAML часто используется для структурированной записи информации, так как он обладает более простым синтаксисом, чем, например, JSON или XML [14]. Сами разработчики нейросети YOLO рекомендуют пользоваться именно форматом YAML при работе с их моделями, так что для корректного обучения следуем указаниям Ultralytics.

В каталоге train, содержащем тренировочные изображения, находятся подкаталоги images и labels, в них фотографии свободных парковочных мест и описание меток в текстовых файлах соответственно.

В исходном датасете используется описание изображения в формате XML, причем учитываются как занятые парковочные места, так и свободные. Также описание области, в которой находится парковочное место, реализовано с помощью <rotatedRect> и <contour>, которые содержат больше параметров, чем нужно. <rotatedRect> содержит координаты x и y центра сегмента и необходимые ширину (w) и высоту (h) кусочка изображения. <contour> содержит координаты x и y всех четырёх точек прямоугольника по периметру, причём не всегда в отсортированном порядке.

Было принято решение использовать 1122 тренировочных изображений, которые состоят из полностью занятых, полностью свободных и частично занятых парковок в разное время и в разные погодные условия.

Внушительный размер датасета вынуждает написать скрипт для эффективного парсинга .xml файлов и создания необходимых файлов с приведёнными координатами.

Необходимо подробнее рассмотреть способ нормализирования координат, поскольку в случае создания меток для сегментированных изображений не было необходимости применять какие-либо вычисления, так как целиком всё изображение содержит объект «parking», здесь же на одном изображении может быть от 0 до 100 свободных парковочных мест.

Для начала были получены координаты объекта на изображении в формате (x, y, w, h), где (x, y) - координаты верхнего левого угла объекта, а (w, h) - его ширина и высота.

Произведена нормализация координат следующим образом:

- xmin = x / image_width
- ymin = y / image_height
- width = w / image_width
- height = h / image_height

Где: image_width - ширина изображения, image_height - высота изображения.

В результате получим значения в диапазоне от 0 до 1, что соответствует стандартам обучения нейросетей, упрощает расчеты и обеспечивает инвариантность к размерам изображений [15].

Для переименования необходимых файлов-изображений был написан скрипт rename_jpg.py (см. приложение А), он с помощью подключённой библиотеки os получает список файлов с расширением .jpg в папке, путь к которой указан, после чего сортирует и переименовывает все файлы .jpg числами в порядке возрастания, сохраняя при этом расширение .jpg.

Для переименовывания файлов с разметкой .xml использовался немного модифицированный скрипт rename_xml.py (см. приложение Б) Это было

сделано для упрощения итерации по файлам при написании парсера xml-разметок.

Был написан скрипт parser_xml.py (см. приложение В), в нём написан шаблон поиска строк с подходящим форматом (pattern), с помощью которого, используя также подключённую библиотеку re, запускается поиск в файле всех совпадений. Значения, которые нас интересуют это координаты x, y, и значения w и h сегмента. Ширина и высота кусочка изображения идут в единственном экземпляре, их считывание не составило труда, а вот для координат x и y было добавлено выявление минимума. После нормализации координат получатся кортеж из 4 значений (xmin, ymin, w, h), которые добавляется в список всех координат values. После окончания поиска всех совпадений в файле .xml программа создаёт заданное количество файлов, в каждом из которых len(values) строк с метками.

В каталоге valid, содержащем проверочные изображения, находятся подкаталоги images и labels, в них фотографии целой парковки и описание меток в текстовых файлах соответственно.

Для более точных результатов оценки обучения модели, было принято решение использовать равное количество изображений разных погодных условий с разных камер, конкретно по 20 фотографий. Итого с 3 камер по 20 изображений с 3 видами погодных условий – сформировался датасет, содержащий 180 фотографий.

Аналогично обработке датасета каталога train, в каталоге valid использовались скрипты: rename_jpg.py (см. приложение А), для переименовывания файлов .jpg, rename_xml.py (см. приложение Б), для переименовывания файлов .xml, parser_xml.py (см. приложение В), для парсинга xml-разметок и создания текстовых файлов .txt с метками в соответствующем формате.

Таким образом процесс подготовки датасета занял более 3 часов. Сэкономить время получилось благодаря написанию вспомогательных

программ, которые ускорили процесс приведения данных в нужный для обучения моделей формат. Самый же затратный по времени этап заключался в ручной выборке изображений из имеющихся в датасете, найденном в интернете.

3.1.3 Обучение модели YOLO Nano

Для обучения модели был написан скрипт train_nano.py (см. приложение Г), который подключает библиотеку разработчиков YOLO – Ultralytics, загружает выбранную модель размера nano и запускает обучение, используя заданные параметры [16].

Необходимо пояснить значение каждого параметра и причину выбора того или иного значения [17]:

- «data» – путь к файлу-датасету YAML.

  Заданное значение – “parking_v8.yaml”, нет необходимости указывать полный путь до файла, поскольку он содержится в одной директории со скриптом train_nano.py.
- «imgsz» – размер изображения.

  Заданное значение – 640. Несмотря на то, что исходный размер изображения составляет 1280, было принято решение уменьшить значение до стандартного для модели YOLO, так как это увеличит скорость обучения, пусть и с небольшими потерями в точности.
- «epochs» – количество эпох, для которых будет производится обучение.

  Заданное значение – 50. Было выбрано именно это количество итераций, поскольку сами разработчики рекомендуют его как минимально достаточное число эпох. Значения больше вызывали отказ операционной системы, на которой производилось обучение, в силу недостаточно мощных компонентов компьютера.
- «batch» – размер пакета для загрузчика данных.

Заданное значение – -1. Модель самостоятельно подбирает наиболее подходящий размер пакета, если задать в качестве параметра отрицательную единицу. Для оптимизации обучения было принято решение воспользоваться данным свойством модели, в итоге автоматически было выбрано значение равное 16.

- «name» – имя каталога результатов обучения.

  Заданное значение: 'yolov8n_50e_640s_autob'. Указание всех параметров обучения в названии каталога с результатами упростит его поиск среди множества проведённых экспериментов.

Обучение проводилось на ОС Windows 10 с использованием вычислений на CPU. Время обучения заняло 7 часов и 2 минуты.

Характеристики устройства:

- Профессор – Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
- Оперативная память – 16,0 ГБ

По окончанию тренировки модель предоставляет изображения с промежуточными результатами обучения, их можно увидеть на рис. 11.

Рисунок 11 – Файл train.jpg

Результат запуска обученной модели для задачи детектирования можно увидеть на рис. 12. При этом для тестирования бралась фотография, не входящая в датасет для обучения и проверки.

Рисунок 12 – Результат детектирования обученной модели Nano

3.1.4 Обучение модели YOLO Small

Для обучения модели был написан скрипт train_small.py (см. приложение Д), который подключает библиотеку разработчиков YOLO – Ultralytics, загружает выбранную модель размера small и запускает обучение, используя заданные параметры.

Параметры были выбраны аналогично модели nano, за исключением имени каталога с результатами обучения модели ('yolov8s_50e_640s_autob').

Обучение проводилось на ОС Windows 10 с использованием вычислений на CPU. Время обучения заняло 15 часов и 6 минут.

По окончанию тренировки модель предоставляет изображения с промежуточными результатами обучения, их можно увидеть на рис. 13.

Рисунок 13 – Файл train.jpg

Результат запуска обученной модели для задачи детектирования можно увидеть на рис. 14. При этом для тестирования бралась фотография, не входящая в датасет для обучения и проверки.

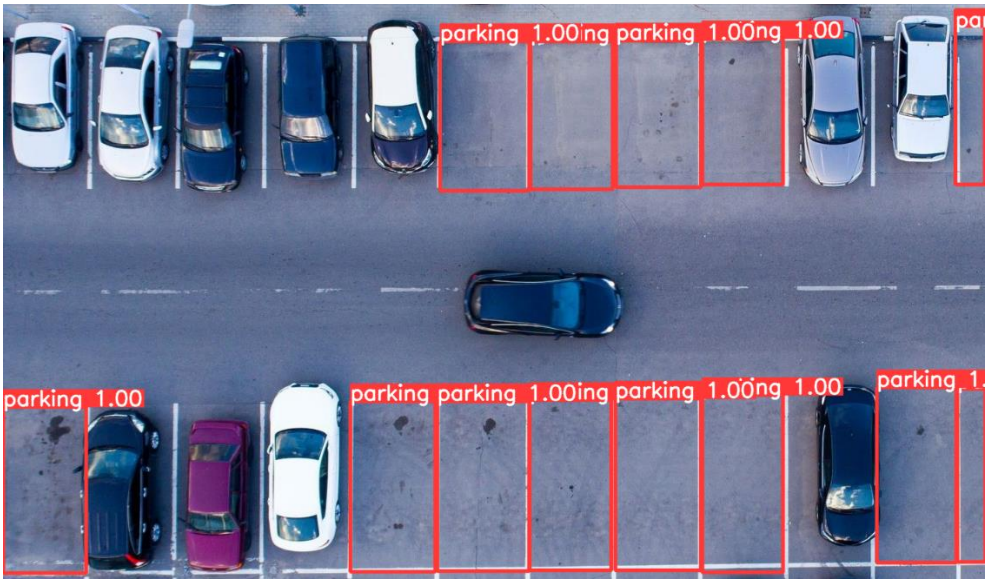

Рисунок 14 – Результат детектирования обученной модели Small

3.1.5 Обучение модели YOLO Medium

Для обучения модели был написан скрипт train_medium.py (см. приложение Д), который подключает библиотеку разработчиков YOLO – Ultralytics, загружает выбранную модель размера medium и запускает обучение, используя заданные параметры.

Параметры были выбраны аналогично модели nano, за исключением имени каталога с результатами обучения модели ('yolov8m_50e_640s_autob').

Обучение проводилось на ОС Windows 10 с использованием вычислений на CPU. Время обучения заняло 28 часов и 3 минуты.

По окончанию тренировки модель предоставляет изображения с промежуточными результатами обучения, их можно увидеть на рис. 15.

Рисунок 15 – Файл train.jpg

Результат запуска обученной модели для задачи детектирования можно увидеть на рис. 16. При этом для тестирования бралась фотография, не входящая в датасет для обучения и проверки.

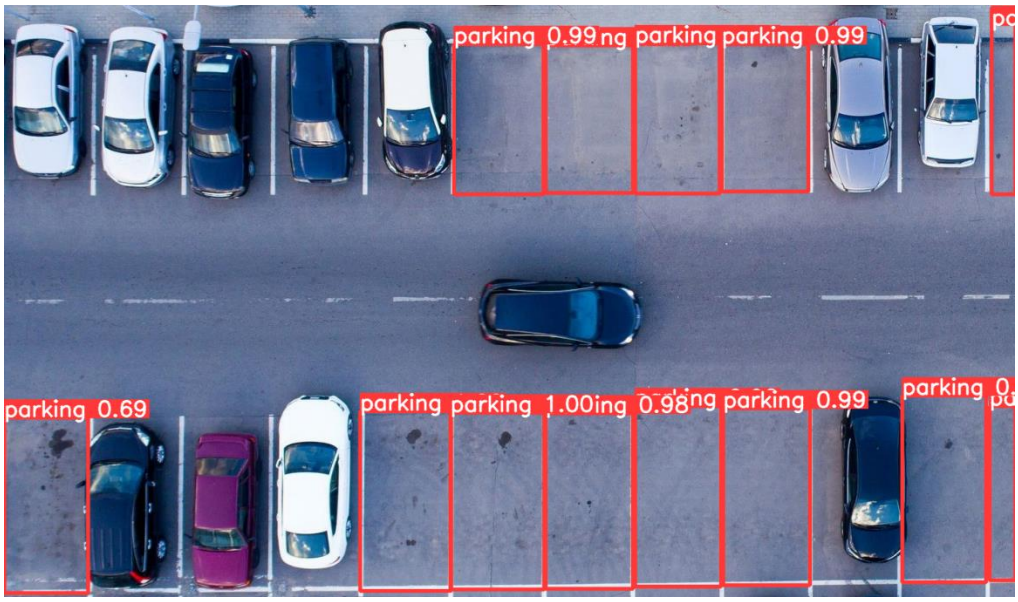

Рисунок 16 – Результат детектирования обученной модели Medium

3.2 Обучение моделей Faster R-CNN

3.2.1 Настройка Weights & Biases

Так как Faster R-CNN не поддерживает интеграцию ClearML, то было принято решение использовать другую платформу для отслеживания и визуализации процесса глубокого обучения, а именно – Weights & Biases [18].

Прежде чем начать использовать Wandb, необходимо было зарегистрироваться на платформе и создать рабочее пространство. Для этого были выполнены соответствующие шаги, после чего в командную строку была введена команда wandb.login(), которая запустила процесс аутентификации созданного окружения в W&B. После ввода API-ключа было обеспечено безопасное взаимодействие с платформой.

С момента аутентификации все обучающие эксперименты с Faster R-CNN, запускаемые в терминале, автоматически регистрируются на платформе Wandb. Использование централизованного инструмента отслеживания

обучения обеспечивает удобное и систематизированное отслеживание всех этапов процесса обучения модели. Данный инструмент позволяет анализировать результаты обучения, выявлять области для улучшения и принимать обоснованные решения на основе собранной статистики, что способствует оптимизации процесса обучения. Пример можно увидеть на рис. 17.

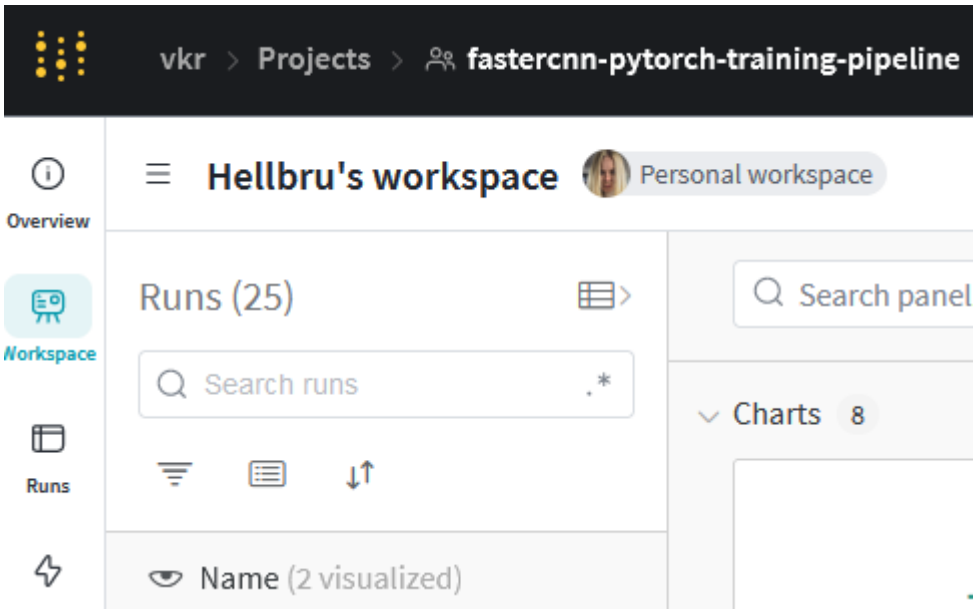

Рисунок 17 – Скриншот настроенного и уже использованного рабочего пространства Wandb.

3.2.2 Подготовка и обработка датасета

Использовать для обучения Faster R-CNN датасет, составленный для работы с моделями YOLO, необходимо не только для ускорения работы, но и для корректного сравнения результатов обучения моделей двух нейросетей.

Проблема заключается в том, что Faster R-CNN принимает разметку изображений в формате VOC, что отличается от формата данных для YOLO [19]. Чтобы ускорить процесс перевода данных в необходимый формат был написан скрипт parser_txt_to_xml.py (см. приложение Ж). Эта программа считывает построчно из файла параметры, запоминая их в списке кортежей, после чего формирует новый файл .xml, куда записывает в нужном формате данные.

В отличие от моделей YOLO, Faster R-CNN не требует нормализовать координаты, нейросети необходимы следующие параметры: xmin, xmax, ymin, ymax, имя класса. Дополнительно в файле разметки также указывается путь к изображению.

Пример файла-разметки для изображения, на котором находится одно свободно парковочное место, находится в приложении З.

В качестве подготовки можно также считать необходимость клонирования репозитория разработчиков Faster R-CNN, на котором хранятся все предобученные модели, а также скрипты для обучения модели, проверки детектирования и отображения результатов.

Аналогично с подготовкой датасета для YOLO, необходимо создать YAML-файл для описания параметров датасета. В отличие от уже написанного ранее файла parking_v8.yaml новый файд custom_data.yaml должен содержать два класса, один из которых отмечается как '__background__', то есть фон [20]. Также требуется указать 4 пути: к папкам с тестовыми и проверочными изображениями и с тестовыми и проверочными метками. Содержимое итогового YAML-файла можно увидеть на рис. 18.

```yaml
# Images and labels direcotry should be relative to train.py
TRAIN_DIR_IMAGES: 'custom_data/train/images'
TRAIN_DIR_LABELS: 'custom_data/train/labels'
VALID_DIR_IMAGES: 'custom_data/valid/images'
VALID_DIR_LABELS: 'custom_data/valid/labels'

# Class names
CLASSES:
  - '__background__'
  - 'parking'

# Number of classes (object classes + 1 for background class in Faster RCNN)
NC: 2

# Whether to save the predictions of the validation set while training
SAVE_VALID_PREDICTION_IMAGES: True
```

Рисунок 18 – Файл custom_training.yaml

Параметр SAVE_VALID_PREDICTION_IMAGES выставлен True, поскольку сохранение промежуточных результатов детектирования модели в

процессе обучения позволит лучше отслеживать прогресс модели в задаче детектирования нового объекта «свободное парковочное место».

3.2.3 Обучение модели ResNet-50

Запуск обучения модели ResNet-50 осуществлялся с помощью следующей команды [21]:

python train.py --data data_configs/custom_data.yaml --epochs 50 --model fasterrcnn_resnet50_fpn_v2 –name custom_training_50e_2b_resnet50_fpn_v_new --batch 2

Разберём каждый параметр и причину выбора того или иного значения:

- «data» – путь к файлу-датасету YAML.
  Заданное значение – “data_configs/custom_data.yaml”, необходимо указать полный путь до файла, поскольку он содержится в разных директориях со скриптом train.py.
- «epochs» – количество эпох, для которых будет производится обучение.
  Заданное значение – 50. Было выбрано аналогичное значению при обучении моделей YOLO число, для корректного сравнения результатов.
- «batch» – размер пакета для загрузчика данных.
  Заданное значение – 2. Так как возможности GPU устройства, на котором производились вычисления, ограничены – размер пакета в разы меньше, чем для модели YOLO, но этот параметр не должен сильно повлиять на результат обучения, только лишь на время для его получения.
- «name» – имя каталога результатов обучения.
  Заданное значение: 'custom_training_50e_2b_resnet50_fpn_v_new'. Указание всех параметров обучения в названии каталога с

результатами упростит его поиск среди множества проведённых экспериментов.

Обучение проводилось на ОС Linux Ubuntu 22.04 с использованием вычислений на GPU. Время, затраченное на обучение, равно 10 часов 58 минут.

Характеристики устройства:

- Профессор – Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
- Оперативная память – 16,0 ГБ
- Видеокарта – NVIDIA GeForce GTX 1650

По окончанию тренировки модель предоставляет изображения с промежуточными результатами обучения, их можно увидеть на рис. 19.

Рисунок 19 – Файл train.jpg

Результат запуска обученной модели для задачи детектирования можно увидеть на рис. 20. При этом для тестирования бралась фотография, не входящая в датасет для обучения и проверки.

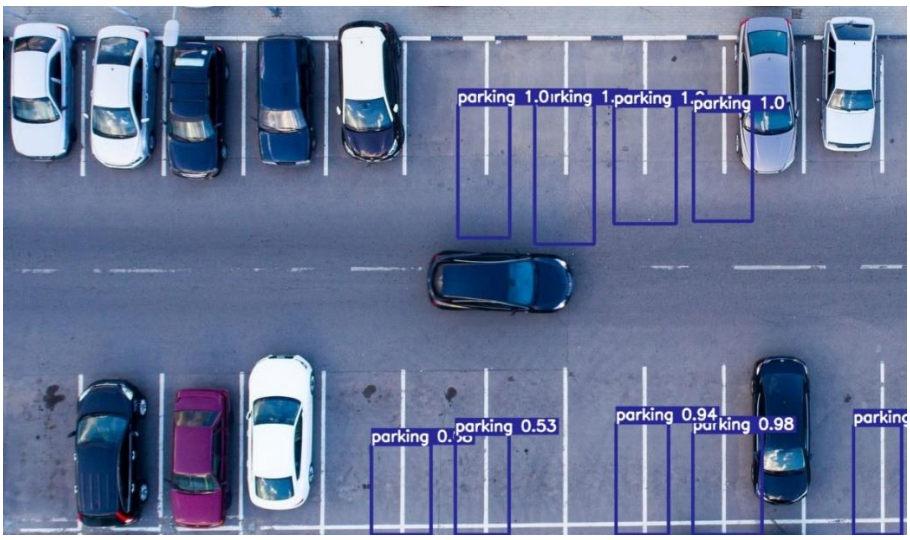

Рисунок 20 – Результат детектирования обученной модели ResNet-50

Можно заметить, что модель плохо справляется с детектированием, акцентируя внимание на разметке. Это может быть связано с особенностью применяемого моделью алгоритма обнаружения объектов.

3.2.4 Обучение модели VGG16

Запуск обучения модели VGG16 осуществлялся с помощью следующей команды:

python train.py --data data_configs/custom_data.yaml --epochs 50 --model fasterrcnn_vgg16 --name custom_training_fasterrcnn_vgg16 --batch 1

Разберём каждый параметр и причину выбора того или иного значения:

- «data» – “data_configs/custom_data.yaml”, аналогично модели ResNet-50
- «epochs» – 50. Стандартное значение, для корректного сравнения результатов.
- «batch» – 1. Так как возможности GPU устройства, на котором производились вычисления, ограничены – размер пакета пришлось выбрать меньше, чем для обучении модели ResNet-50, в случае увеличение размера пакета модель выбрасывала ошибку о нехватке памяти для CUDA.
- «name» –' custom_training_fasterrcnn_vgg16 '.

Обучение проводилось на ОС Linux Ubuntu 22.04 с использованием вычислений на GPU. Время, затраченное на обучение, равно 1 час 35 минут.

По окончанию тренировки модель предоставляет изображения с промежуточными результатами обучения, их можно увидеть на рис. 21.

Рисунок 21 – Файл train.jpg

Результат запуска обученной модели для задачи детектирования можно увидеть на рис. 22. При этом для тестирования бралась фотография, не входящая в датасет для обучения и проверки.

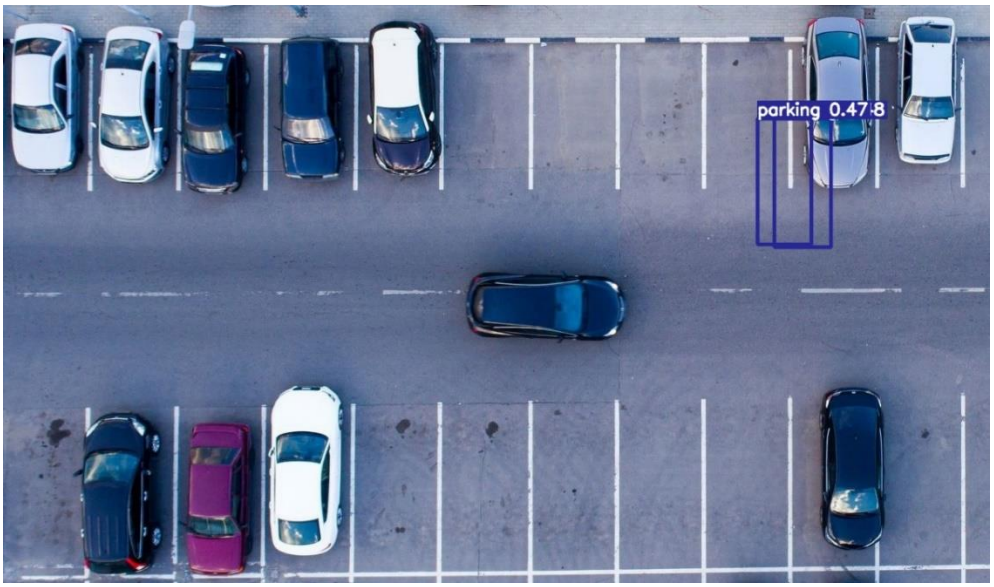

Рисунок 22 – Результат детектирования обученной модели Medium

Можно заметить, что модель не справляется с детектированием. Это может быть связано с особенностью применяемого моделью алгоритма обнаружения объектов, а также с недостаточно большим количеством эпох для обучения именно для данной модели.

4. ИССЛЕДОВАНИЕ И ОЦЕНКА ОБУЧЕННЫХ МОДЕЛЕЙ

4.1 Оценка моделей YOLO

4.1.1 Оценка модели YOLO Nano

Используя интеграцию ClearML, которую Ultralytics YOLOv8 поддерживает по умолчанию, были получены графики изменения метрик оценки качества обучения модели, их можно увидеть на рис. 23.

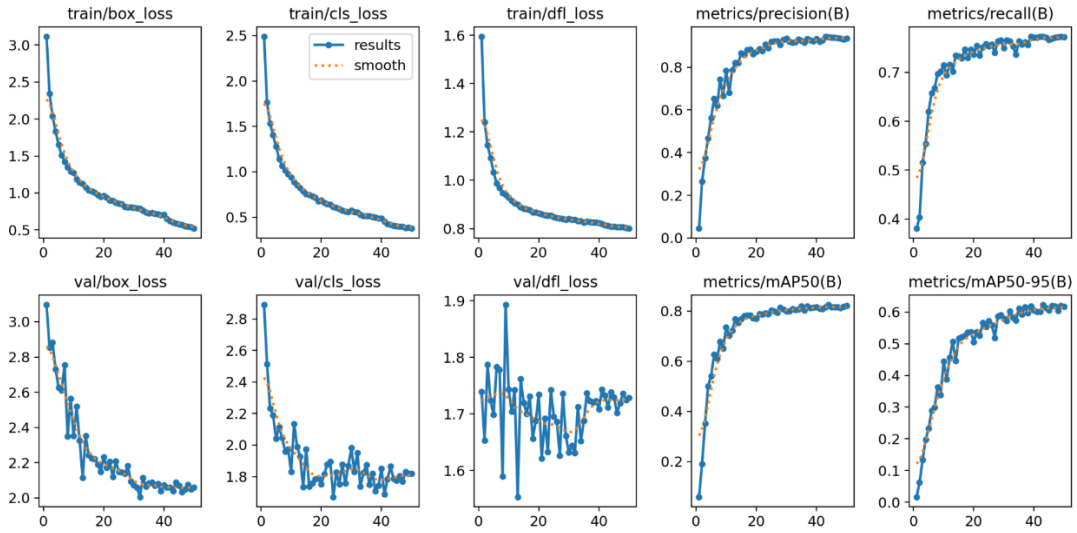

Рисунок 23 – Графики изменения метрик оценки качества обучения

Теперь можно оценить лучшую модель, используя следующую команду.

yolo task=detect mode=val model=runs/detect/yolov8n_50e_640s_autob/weights/best.pt name=yolov8n_50e_640s_autob_best data=parking_v8.yaml imgsz=640

Был получен следующий результат:

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	180	6319	0.939	0.684	0.738	0.56

Самая высокая mAP at 0.50 IoU равняется 0.738 при 0.50:0.95 IoU равным 0.56. Этот результат уже можно считать неплохим, учитывая, что это модель Nano.

4.1.2 Оценка модели YOLO Small

Используя ClearML, были получены графики изменения метрик оценки качества обучения модели, их можно увидеть на рис. 24.

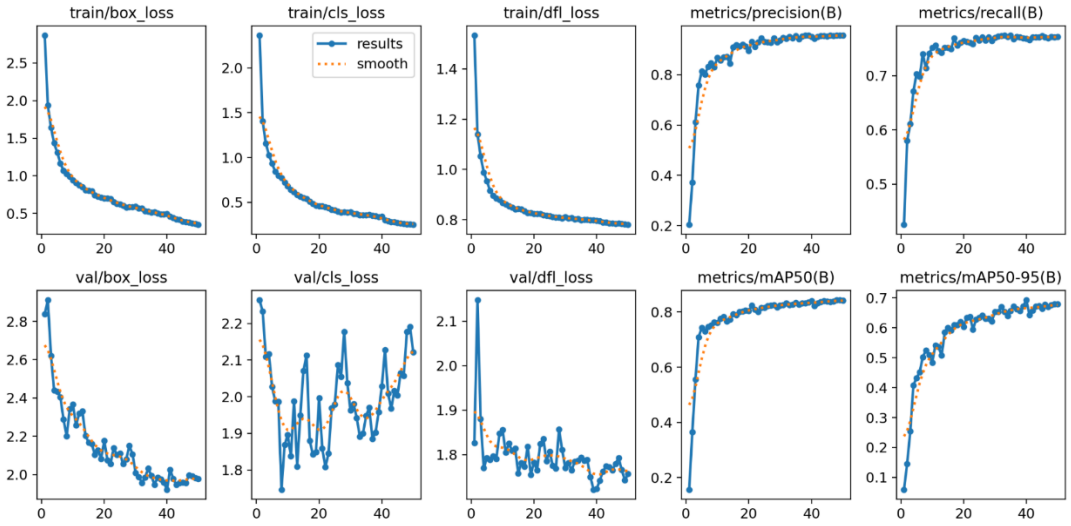

Рисунок 24 – Графики изменения метрик оценки качества обучения

Теперь можно оценить лучшую модель, используя следующую команду.

yolo task=detect mode=val model=runs/detect/yolov8s_50e_640s_autob/weights/best.pt name=yolov8s_50e_640s_autob_best data=parking_v8.yaml imgsz=640

Был получен следующий результат:

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	180	6319	0.951	0.768	0.841	0.693

Самая высокая mAP at 0.50 IoU равняется 0.841 при 0.50:0.95 IoU равным 0.693. Результаты немного превосходят модель Nano, это было ожидаемо, поскольку модель Small по размерам превосходит Nano, а также потребовала времени в 2,14 раз больше на обучение.

4.1.3 Оценка модели YOLO Meduim

Используя ClearML, были получены графики изменения метрик оценки качества обучения модели, их можно увидеть на рис. 25.

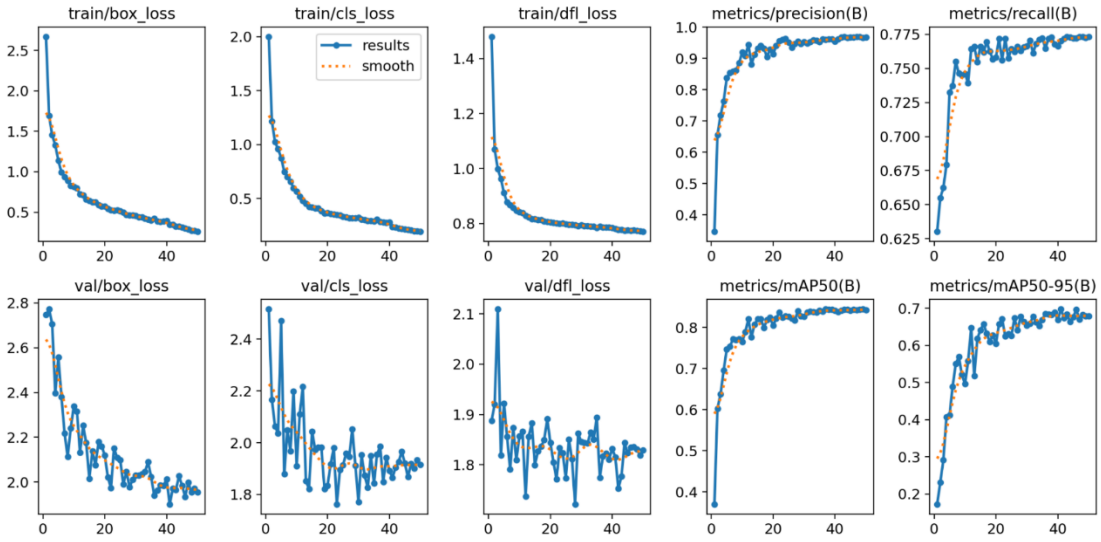

Рисунок 25 – Графики изменения метрик оценки качества обучения

Теперь можно оценить лучшую модель, используя следующую команду.

yolo task=detect mode=val model=runs/detect/yolov8s_50e_640s_autob/weights/best.pt name=yolov8m_50e_640s_autob_best data=parking_v8.yaml imgsz=640

Был получен следующий результат:

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	180	6319	0.963	0.77	0.838	0.698

Самая высокая mAP at 0.50 IoU равняется 0.838 при 0.50:0.95 IoU равным 0.698. Результаты почти идентичны показателям модели Small, разница доходит до сотых.

Отсюда можно сделать вывод о нецелесообразности рассмотрения моделей больших весов для сформированного датасета и выбранных параметров, поскольку обучение модели Medium заняло в 1,86 раз больше времени, чем модель Small, а результат при этом не особо отличается.

4.1.4 Выявление лучшей модели

Сравнительные графики и диаграммы для трёх моделей YOLO приведены на рис. 26-27.

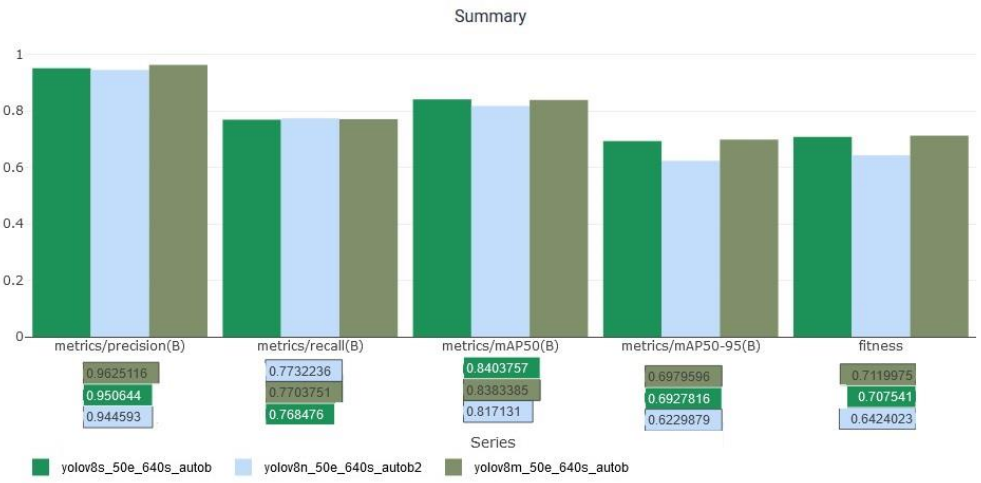

Рисунок 26 – Сравнительная диаграмма метрик трёх моделей YOLO.

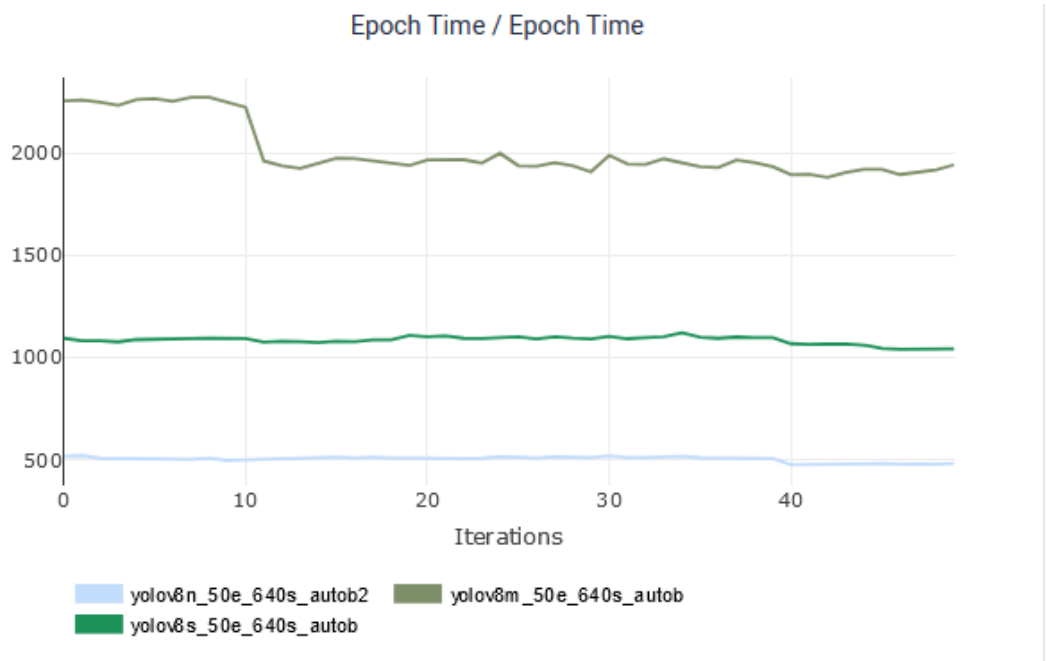

Рисунок 27 – Сравнительный график затраченного времени на итерацию обучения для трёх моделей YOLO.

Исходя из полученных сравнительных данных можно сделать вывод о том, что модель Small обладает наилучшими показателями, проигрывая модели Nano только лишь по времени обучения. Модель Medium добилась почти идентичных результатов, побеждая лишь на несколько сотых, чего

явно недостаточно для определения ее лучшей моделью, поскольку среди всех исследуемых – время для ее обучения было самым долгим. Именно модель Small имеет смысл сравнивать с лучшей обученной моделью нейросети Faster R-CNN в дальнейшем. Это обусловлено тем, что она демонстрирует сопоставимую точность с моделью Medium, но требует значительно меньше времени для обучения. Кроме того, модель Small имеет более компактный размер, что может быть важным фактором в приложениях с ограниченными ресурсами.

4.2 Оценка моделей Faster R-CNN

4.2.1 Оценка модели ResNet-50

Используя Wandb, были получены графики изменения метрик оценки качества обучения модели, их можно увидеть на рис. 28.

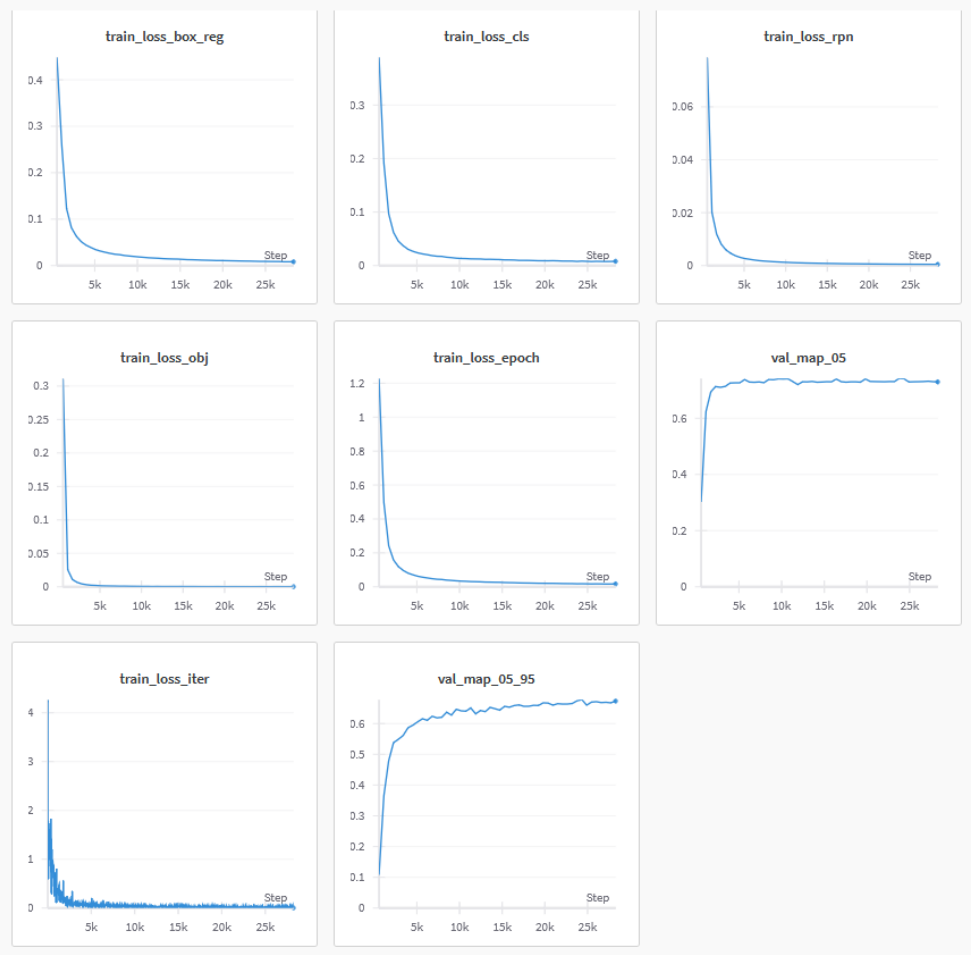

Рисунок 28 – Графики изменения метрик оценки качества обучения

Теперь можно оценить лучшую модель, воспользовавшись результатами на платформе Wandb. Был получен следующий результат:

val_map_05:0.7295012402637114

val_map_05_95:0.6733188992655875

Самая высокая mAP at 0.50 IoU равняется 0.73 при 0.50:0.95 IoU равным 0.673. Результаты обучения модели получились неплохими, однако не смогли дотянуть до уровня модели YOLO_nano.

4.2.2 Оценка модели VGG16

Используя Wandb, были получены графики изменения метрик оценки качества обучения модели, их можно увидеть на рис. 29.

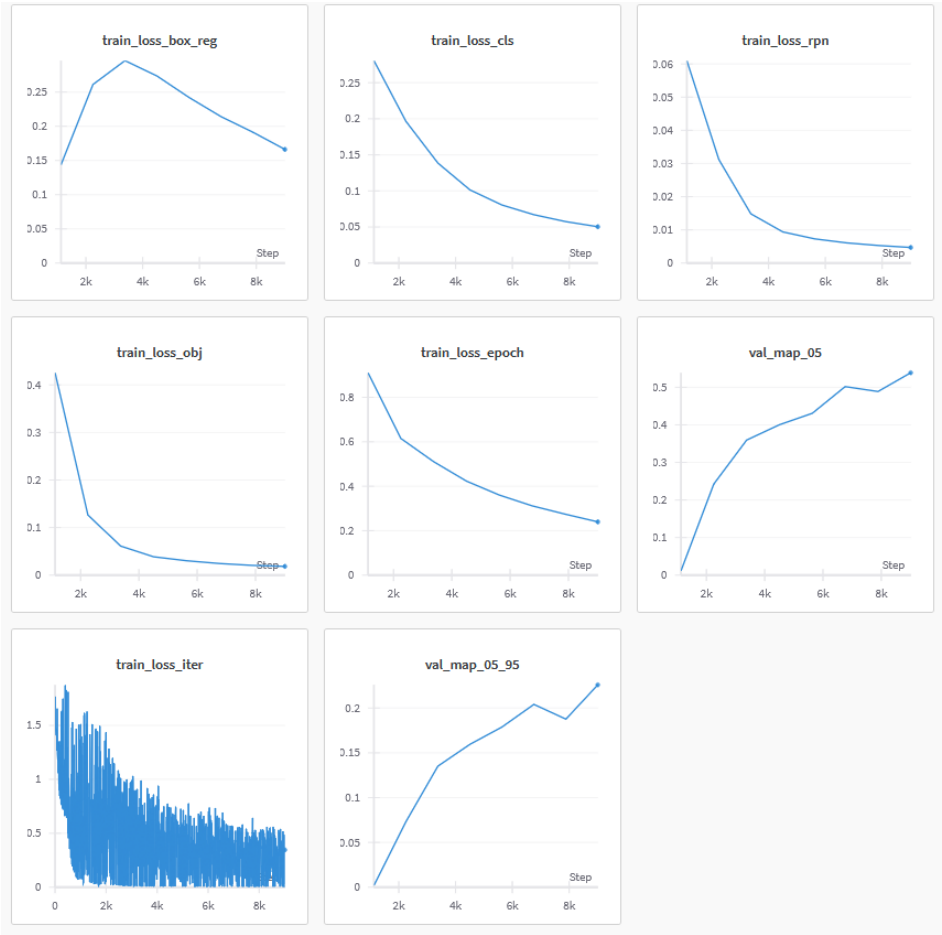

Рисунок 29 – Графики изменения метрик оценки качества обучения

Теперь можно оценить лучшую модель, воспользовавшись результатами на платформе Wandb. Был получен следующий результат:

val_map_05:0.5388997160691524

val_map_05_95:0.22598267498587232

Самая высокая mAP at 0.50 IoU равняется 0.54 при 0.50:0.95 IoU равным 0.223. Результаты обучения оказались неконкурентоспособными, по сравнению с моделью ResNet-50.

4.2.3 Выявление лучшей модели

Сравнительные графики для двух моделей Faster R-CNN приведены на рис. 30-31.

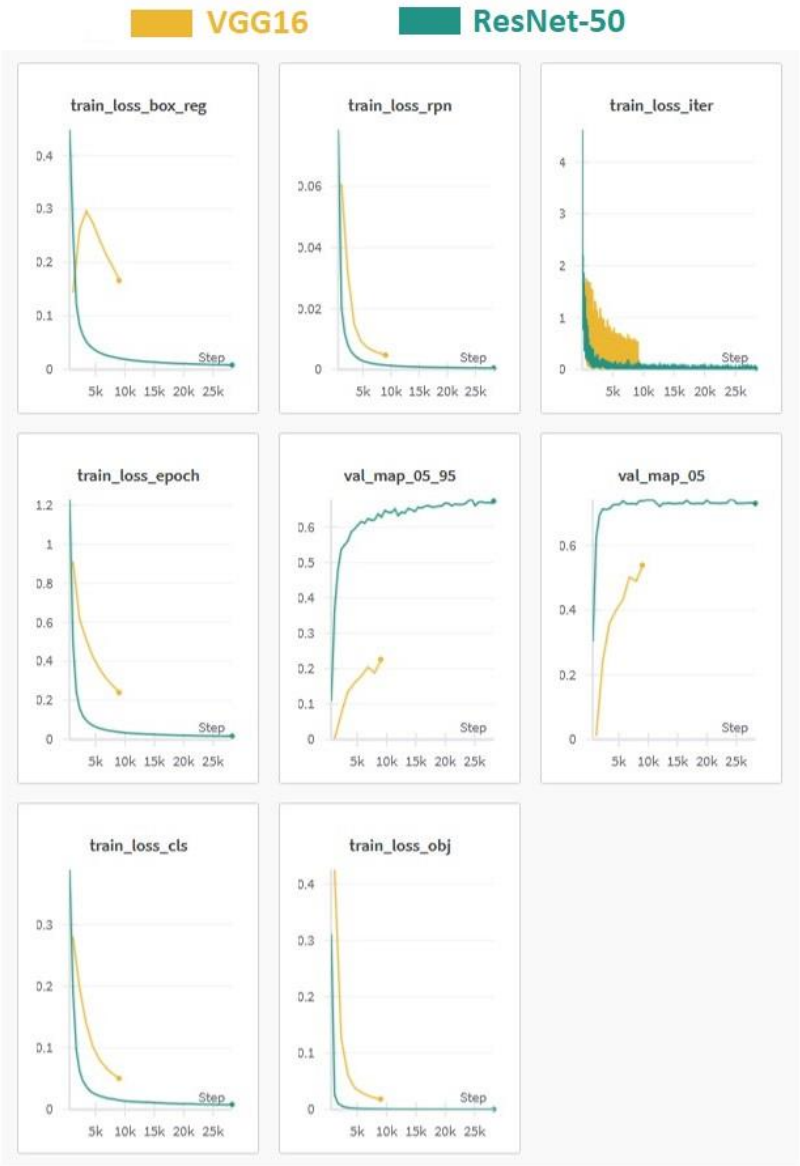

Рисунок 30 – Сравнительные графики изменения метрик оценки качества обучения

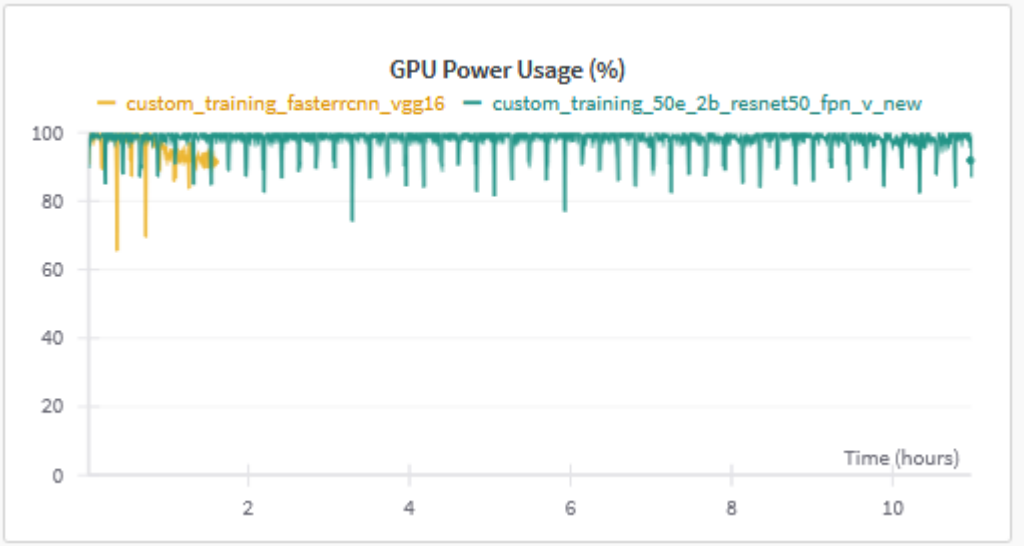

Рисунок 31 – Сравнительный график использования GPU в зависимости от времени обучения

Исходя из полученных сравнительных данных можно сделать вывод о том, что модель ResNet-50 обладает лучшими показателями, за исключением времени обучения модели. Однако ResNet-50 допускает намного меньше потерь в процессе обучения, что приводит к высокой точности, поэтому целесообразно затратить в 10 раз больше времени на обучение, по сравнению с моделью VGG16.

Есть несколько причин, по которым модель VGG16 может обучаться хуже, чем модель ResNet-50, при обучении нейросети Faster R-CNN, основная из них – это архитектурные особенности. ResNet-50 имеет более глубокую архитектуру с большим количеством сверточных слоев, что позволяет извлекать более сложные пространственные особенности. Скиповые соединения в ResNet-50 облегчают передачу градиентов и предотвращают проблему градиентного исчезания. VGG16 не имеет таких соединений.

Также одной из причин является спецификация задачи, конкретно обнаружение и классификация объектов, которая является очень сложной, требующей использовать модели с большой емкостью [22]. ResNet-50 имеет более высокую емкость благодаря своей глубине и скиповым соединениям,

когда VGG16 может быть менее эффективна для этой задачи из-за своей ограниченной емкости.

4.3 Сравнение YOLO и Faster R-CNN

Данные для сравнения всех рассматриваемых моделей предоставлены в таблице 2.

Таблица 2 – Сравнение моделей

Модель/Критерий	mAP at 0.50 IoU	mAP at 0.50:0.95 IoU	Время обучения
YOLO Nano	0.738	0.56	7 часов 2 минуты
YOLO Small	**0.841**	**0.693**	**15 часов 6 минут**
YOLO Medium	0.838	0.698	28 часов 3 минуты
Faster R-CNN ResNet-50	0.73	0.673	10 часов 58 минут
Faster R-CNN VGG16	0.539	0.226	1 час 35 минут

Модель Small нейросети YOLO превосходит модель ResNet-50 нейросети Faster R-CNN по результатам обучения, демонстрируя более высокую точность обнаружения объектов.

Модель Small достигает mAP at 0.50 IoU 0,841 и mAP at 0.50:0.95 IoU 0,693, что значительно выше, чем 0,73 и 0,673 для ResNet-50 соответственно. Более высокие значения mAP указывают на то, что модель Small может более точно обнаруживать и локализовывать объекты в изображениях.

Стоит учитывать, что модель Small требует больше времени обучения (15 часов 6 минут), чем ResNet-50 (10 часов 35 минут). Однако, учитывая ее уровень точности, дополнительные инвестиции во время обучения оправданы.

Модель ResNet-50 имеет меньший показатель потерь (train_loss), однако на финальную точность детектирования этот параметр никак не повлиял. Причина в том, что нейросеть YOLO имеет более подходящие

гиперпараметры и архитектуру для конкретной задачи обнаружения, использующую более эффективный алгоритм обнаружения, что приводит к лучшим результатам детектирования.

В целом, с точки зрения производительности и эффективности обучения, модель Small нейросети YOLO является лучшим выбором в данной ситуации. Ее высокая точность обнаружения объектов делает ее более подходящей для приложений, требующих надежного обнаружения и локализации объектов.

Модель YOLO Small следует использовать для дальнейшего обучения на более крупном датасете с более высокими параметрами по следующим причинам:

- Высокая точность.

  Модель YOLO Small уже продемонстрировала превосходную точность обнаружения объектов по сравнению с другими моделями, такими как ResNet-50. Более крупный датасет с более высокими параметрами позволит еще больше улучшить ее точность.

- Эффективность обучения.

  Хотя время обучения модели YOLO Small больше, чем у других моделей, ее превосходная производительность оправдывает эти дополнительные инвестиции во время.

- Низкая вычислительная стоимость.

  Модель YOLO Small имеет относительно низкую вычислительную стоимость по сравнению с другими моделями обнаружения объектов. Это делает ее подходящей для развертывания на устройствах с ограниченными вычислительными ресурсами, таких как мобильные телефоны и встроенные системы.

Обученная модель YOLO Small может быть использована для создания приложения, которое позволяет удаленно узнать количество свободных парковочных мест. Такое приложение может значительно упростить процесс поиска парковки, сэкономить время водителей и уменьшить стресс, связанный с поиском места для парковки.

5. ИНФОРМАЦИОННЫЙ МАРКЕТИНГ

5.1 Расчет затрат на выполнение и внедрение проекта, расчет цены

- Оборудование:
  - Видеокамеры: 5 камер х 35 000 рублей = 175 000 рублей
  - Сервер для обработки изображений: 1 сервер х 140 000 рублей = 140 000 рублей
- Разработка программного обеспечения:
  - Интеграция с существующей системой: 140 000 рублей
- Внедрение:
  - Установка оборудования: 70 000 рублей
  - Настройка и обучение нейросети: 70 000 рублей
- Тестирование и техническое обслуживание:
  - Тестирование и устранение неполадок: 70 000 рублей
  - Техническое обслуживание и обновления: 35 000 рублей в год

Итого:

- Затраты на выполнение: 620 000 рублей
- Ежегодные затраты на техническое обслуживание: 35 000 рублей

Цена проекта будет зависеть от нескольких факторов, включая:

- Стоимость оборудования и программного обеспечения
- Затраты на внедрение
- Затраты на техническое обслуживание
- Прибыль

Допустим, получаемая прибыль должна быть не менее 20% от стоимости проекта. Тогда цена проекта будет рассчитываться следующим образом:

$$\text{Цена} = \text{Затраты на выполнение} / (1 - \text{Прибыль})$$

$$\text{Цена} = 620\,000 \text{ рублей} / (1 - 0{,}20) = 775\,000 \text{ рублей}$$

Таким образом, можно установить цену проекта в размере 775 000 рублей.

5.2 Расчет показателей конкурентоспособности

Для расчета показателей конкурентоспособности разработанной продукции можно использовать следующие ключевые показатели:

1. Точность детекции
2. Затраты на разработку и внедрение продукта

Оценка конкурентоспособности, как известно, может проводиться на основе сравнения комплексных показателей конкурентоспособности товаров или услуг-аналогов с показателями анализируемого товара или услуги. Показатель общей конкурентоспособности K может быть при этом определен по формуле:

$$K = \frac{Q}{E},$$

где Q – показатель конкурентоспособности по характеристикам качества;

E – показатель конкурентоспособности по экономическим характеристикам.

- Показатель конкурентоспособности по характеристикам качества (Q):

Для данного продукта показатель Q может быть определен как среднее значение точности детекции свободных парковочных мест на изображениях, полученное в ходе тестирования системы на различных наборах данных. Используя полученные в ходе оценки моделей значения, мы получили значение точности детекции равное 85%.

- Показатель конкурентоспособности по экономическим характеристикам (E):

Для данного продукта значение E может быть определено как отношение общей стоимости разработки и внедрения системы к ожидаемой экономической выгоде от использования данной системы. Стоимость разработки и внедрения системы, рассчитанная ранее, составила 620 000 рублей, а ожидаемая экономическая выгода за покупку системы - 775 000 рублей.

Теперь можем рассчитать показатель общей конкурентоспособности К по формуле $K = \frac{Q}{E}$:

$$K = \frac{85\%}{620000\text{р}/775000\text{р}} = \frac{0.85}{0.8} = 1.0625$$

Итак, общий показатель конкурентоспособности данной продукции составляет 1.0625. Этот показатель поможет оценить уровень конкурентоспособности системы на основе качества и экономических характеристик.

На рис. 32 можно увидеть схему, которая позволяет выделить области (зоны) I÷VI, отображающие различные ситуации, которые могут возникнуть в процессе управления конкурентоспособностью товара или услуги.

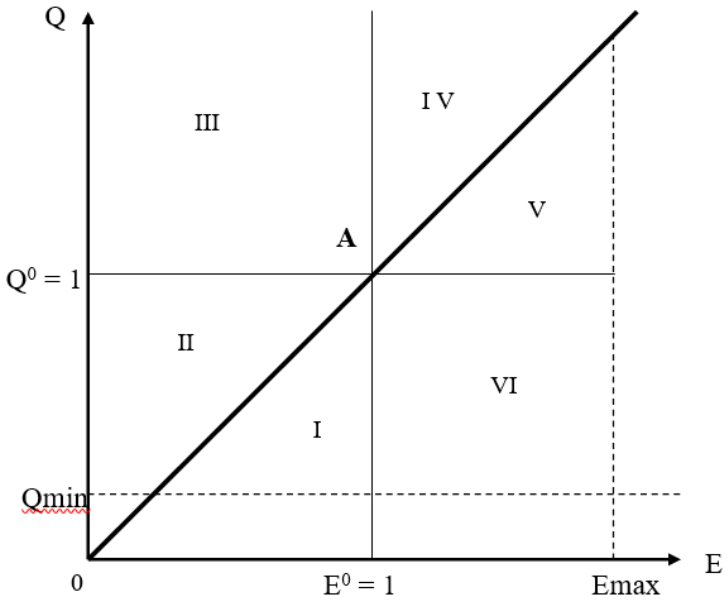

Рисунок 32 – Варианты соотношения показателей конкурентоспособности предлагаемого и базового товара или услуги

Характеристика ситуаций, отображаемых областями I÷VI, предоставлена в таблице 3.

Таблица 3 – Ситуации, возникающие в процессе управления конкурентоспособностью платной образовательной услуги

Область на рис. 32	Показатели конкурентоспособности			Характеристика ситуации	Оценка
	Q	E	K		
I	Q<1	E<1	K <1	Снижение качества товара или услуги не компенсируется снижением ее стоимости	Неконкурентоспособна
II	**Q<1**	**E<1**	**K >1**	**Снижение качества товара или услуги компенсируется снижением ее стоимости**	**Конкурентоспособна**
III	Q>1	E<1	K >1	Качество товара или услуги повышается при одновременном снижении ее стоимости	Конкурентоспособна
IV	Q>1	E>1	K >1	Повышение качества товара или услуги компенсирует повышение ее стоимости	Конкурентоспособна
V	Q>1	E>1	K <1	Повышение качества товара или услуги не компенсирует повышение ее стоимости	Неконкурентоспособна
VI	Q<1	E>1	K <1	Качество товара или услуги снижается при одновременном росте ее стоимости	Неконкурентоспособна

Основываясь на данных в таблице, можно сделать вывод о том, что разработанный продукт является конкурентоспособным.

5.3 Предложения по продвижению разработанной продукции

1. Контент-маркетинг
    - Создать ценный контент, демонстрирующий преимущества и ценность продукта.
    - Опубликовать этот контент на веб-сайте, в социальных сетях и на сторонних отраслевых сайтах.
2. Социальные сети
    - Активно продвигать продукт в социальных сетях, используя соответствующие хэштеги и участвуя в отраслевых обсуждениях.
    - Запустить таргетированные рекламные кампании в социальных сетях для охвата потенциальных клиентов.
3. Инфлюенсер-маркетинг
    - Сотрудничать с влиятельными лицами в отрасли для демонстрации и продвижения продукта их подписчикам.
4. Поисковая оптимизация (SEO)
    - Оптимизировать веб-сайт и контент для релевантных ключевых слов для повышения видимости продукта в результатах поиска.
    - Создать обратные ссылки на веб-сайт с авторитетных отраслевых сайтов.
5. Партнерский маркетинг
    - Сотрудничать с другими компаниями в отрасли для проведения перекрестных промоакций и получения доступа к новой аудитории.
6. Рекламные кампании
    - Запустить целевые рекламные кампании в поисковых системах, социальных сетях и отраслевых публикациях.
    - Использовать баннерную рекламу и видеорекламу.

7. Событийный маркетинг
    - Участвовать в отраслевых конференциях и выставках для демонстрации продукта и общения с потенциальными клиентами.
    - Организовать собственные мероприятия или вебинары для представления продукта и предоставления ценной информации о нем.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была достигнута поставленная цель – было успешно обучено несколько нейросетей детектированию нового объекта «parking», было проведено сравнение результатов скорости обучения и качества итоговой способности обнаружения каждой модели, после чего были сделаны выводы о целесообразности использования переобученных нейросетей для задачи обнаружения свободных парковочных мест.

Для достижения цели было исследовано несколько моделей нейронных сетей, включая YOLO, модели: nano, small, medium, и Faster R-CNN, модели: ResNet-50, VGG16.

Были выполнены следующие условия:

1. Каждая модель должна быть обучена на обширном датасете фотографий
2. Проведено большое количество итераций обучения, для более корректной работы каждой модели и для составления более практичной статистики
3. Алгоритм детектирует свободные парковочные места
4. Результатом работы каждой обученной нейросети является выделение рамкой свободных парковочных мест, обнаруженных на поданном алгоритму на вход кадре
5. В конце проведено сравнение всех обученных моделей, для того чтобы выявить наиболее работоспособный вариант алгоритма

В результате проведённых экспериментов по обучению и оценке каждой модели на наборе данных изображений парковочных мест был сделан вывод о том, что модель YOLO Small демонстрирует наилучшие показатели точности обнаружения свободных парковочных мест при относительно низком времени обучения.

Таким образом, на основе полученных результатов можно сделать вывод о том, что YOLO Small является наиболее подходящей моделью нейронной

сети для решения поставленной задачи. Однако для достижения более высокой точности обнаружения требуется использование вычислительной техники с большей мощностью, чем та, которая была доступна в рамках данной работы.

Дальнейшие исследования могут быть направлены на оптимизацию модели YOLO Small для повышения точности обнаружения, а также на разработку приложений, использующих обученную модель, для создания удобного средства удалённого определения свободных парковочных мест во дворах жилых домов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. «Парковки Санкт-Петербург» [Электронный ресурс]. URL: https://parking.spb.ru/ru/ (дата обращения: 01.12.2023).
2. «Парковки России» [Приложение]. URL: https://apps.apple.com/ru/app/парковки-россии/id1434426876 (дата обращения: 01.12.2023).
3. «GetPark: паркшеринг сервис» [Электронный ресурс]. URL: https://getpark.ru/ (дата обращения: 01.12.2023).
4. Сервисы в ТРК «Европолис» [Электронный ресурс]. URL: https://trk-europolis.ru/services/ (дата обращения: 01.12.2023)
5. «Яндекс.Парковки» [Электронный ресурс]. URL: https://yandex.ru/support/navigator/parking.html (дата обращения: 01.12.2023).
6. «Умные парковки» («Интерсвязь») [Электронный ресурс]. URL: https://vision.is74.ru/parking (дата обращения: 15.12.2023).
7. YOLOv8 [Электронный ресурс]. URL: https://github.com/ultralytics/ultralytics/blob/main/README.md (дата обращения: 15.12.2023).
8. Ultralytics [Электронный ресурс]. URL: https://github.com/ultralytics (дата обращения: 01.04.2024).
9. Faster R-CNN [Электронный ресурс]. URL: https://github.com/sovit-123/fasterrcnn-pytorch-training-pipeline (дата обращения: 12.04.2024).
10. Faster R-CNN — Torchvision main documentation [Электронный ресурс]. URL: https://pytorch.org/vision/main/models/faster_rcnn.html (дата обращения: 12.04.2024).
11. Задача нахождения объектов на изображении [Электронный ресурс] URL: https://neerc.ifmo.ru/wiki/index.php?title=Задача_нахождения_объектов_на_изображении (дата обращения: 12.04.2024)

12. Training and Validation Loss in Deep Learning [Электронный ресурс] URL: https://www.baeldung.com/cs/training-validation-loss-deep-learning (дата обращения: 12.04.2024)
13. ClearML | The Continuous Machine Learning Company [Электронный ресурс]. URL: https://clear.ml/ (дата обращения: 01.04.2024).
14. YAML за 5 минут: синтаксис и основные возможности [Электроннный ресурс] URL: https://tproger.ru/translations/yaml-za-5-minut-sintaksis-i-osnovnye-vozmozhnosti (дата обращения: 12.04.2024)
15. Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, Bo Ma, A Review of Yolo Algorithm Development // Procedia Computer Science – 2022 – Volume 199 – C. 1066-1073.
16. Xiangwu Ding, Ruidi Yang, Vehicle and Parking Space Detection Based on Improved YOLO Network Model // Journal of Physics: Conference Series – 2019 – Volume 1325.
17. YOLO-World (Real-Time Open-Vocabulary Object Detection) [Электронный ресурс]. URL: https://docs.ultralytics.com/ru/models/yolo-world/ (дата обращения: 04.04.2024).
18. Weights & Biases: The AI Developer Platform [Электронный ресурс]. URL: https://wandb.ai/site (дата обращения: 12.04.2023).
19. Статья «Обнаружение объектов с помощью более быстрого R-CNN» [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/cognitive-toolkit/object-detection-using-faster-r-cnn (дата обращения: 12.04.2024).
20. Quanfu Fan, Lisa Brown, John Smith, A closer look at Faster R-CNN for vehicle detection // 2016 IEEE Intelligent Vehicles Symposium (IV).
21. Статья «Faster R-CNN» [Электронный ресурс]. URL: https://roboflow.com/model/faster-r-cnn (дата обращения: 12.04.2024).
22. Raj Kirtibhai Patel, Praveen Meduri, Faster R-CNN based Automatic Parking Space Detection // MLMI '20: Proceedings of the 2020 3rd International

Conference on Machine Learning and Machine IntelligenceSeptember – 2020 – C. 105–109.

ПРИЛОЖЕНИЕ А

Файл: rename_images.py

```
import os
folder_path =
'/home/liza/vkr/datasets/parking_dataset_v8/train/images'
files = [f for f in os.listdir(folder_path) if
f.endswith('.jpg')]
files.sort()
for idx, file_name in enumerate(files):
    new_name = f"v{idx+1}.jpg"
    os.rename(os.path.join(folder_path, file_name),
os.path.join(folder_path, new_name))
```

ПРИЛОЖЕНИЕ Б

Файл: rename_xml.py

```
import os
folder_path =
'/home/liza/vkr/datasets/parking_dataset_v8/train/labels’
files = [f for f in os.listdir(folder_path) if
f.endswith('.xml')]
files.sort()
for idx, file_name in enumerate(files):
new_name = f"v{idx+1}.xml"
os.rename(os.path.join(folder_path, file_name),
os.path.join(folder_path, new_name))
```

ПРИЛОЖЕНИЕ В

Файл: parser_xml.py

```
import re
img_w = 1280
img_h = 720

def extract_values_from_file(file_path):
    with open(file_path, 'r') as file:
        content = file.read()
    pattern = r'<space id="\d+"
occupied="0">\s*<rotatedRect>\s*<center x="\d+" y="\d+"
/>\s*<size w="(\d+)" h="(\d+)" />\s*<angle d="-?\d+"
/>\s*</rotatedRect>\s*<contour>\s*<point x="(\d+)" y="(\d+)"
/>\s*<point x="(\d+)" y="(\d+)" />\s*<point x="(\d+)" y="(\d+)"
/>\s*<point x="(\d+)" y="(\d+)" />\s*</contour>\s*</space>'
    matches = re.findall(pattern, content)
    extracted_values = []
    for match in matches:
        w, h, x1, y1, x2, y2, x3, y3, x4, y4 = map(int,
match)
        list_x = list([x1, x2, x3, x4])
        list_y = list([y1, y2, y3, y4])
        extracted_values.append((min(list_x)/img_w,
min(list_y)/img_w, w/img_w, h/img_h))

    return extracted_values

for i in range(1, 1122):
    file_path = str("v"+str(i)+".xml")
    f_name_txt = str("v"+str(i)+".txt")
    values = extract_values_from_file(file_path)
    with open(f_name_txt, 'w') as file:
        for val in values:
            file_content = "0 " + " ".join(map(str, val))
            file.write(file_content + "\n")
```

ПРИЛОЖЕНИЕ Г

Файл: train_nano.py

```
from ultralytics import YOLO

# Load the model.
model = YOLO('yolov8n.pt')

# Training.
results = model.train(
   data='parking_v8.yaml',
   imgsz=640,
   epochs=50,
   batch=-1,
   name='yolov8n_50e_640s_autob')
```

ПРИЛОЖЕНИЕ Д

Файл: train_small.py

```
from ultralytics import YOLO

# Load the model.
model = YOLO('yolov8s.pt')

# Training.
results = model.train(
   data='parking_v8.yaml',
   imgsz=640,
   epochs=50,
   batch=-1,
   name='yolov8s_50e_640s_autob')
```

ПРИЛОЖЕНИЕ Е

Файл: train_medium.py

```
from ultralytics import YOLO

# Load the model.
model = YOLO('yolov8m.pt')

# Training.
results = model.train(
   data='parking_v8.yaml',
   imgsz=640,
   epochs=50,
   batch=-1,
   name='yolov8m_50e_640s_autob')
```

ПРИЛОЖЕНИЕ Ж

Файл: parser_txt_to_xml.py

```
data = []
img_w = 1280
img_h = 720

for i in range(1, 1122):
    file_path = str("v"+str(i)+".txt")
    f_name_txt = str("v"+str(i)+".xml")
    with open(file_path, "r") as file:
        lines = file.readlines()
        for line in lines:
            values = [float(val) for val in
line.strip().split()]
            data.append(tuple(values))
    with open(f_name_txt, 'w') as file:

file.write("<annotation>\n\t<folder><folder/>\n\t<filename>" +
str("v"+str(i)+".jpg") + "</filename>\n\t<path>" +
str("v"+str(i)+".jpg") +
"</path>\n\t<size>\n\t\t<width>1280</width>\n\t\t<height>720</he
ight>\n\t\t<depth>24</depth>\n\t</size>\n\t<segmented>0</segment
ed>\n")
        for tup in data:
    file.write("\t<object>\n\t\t<name>parking</name>\n\t\t<pose>
Unspecified</pose>\n\t\t<truncated>0</truncated>\n\t\t<difficult
>0</difficult>\n\t\t<occluded>0</occluded>\n\t\t<bndbox>\n\t\t\t
<xmin>" + str(int(tup[1]*img_w)) + "</xmin>\n\t\t\t<xmax>" +
str(int(tup[1]*img_w)+int(tup[3]*img_w)) +
"</xmax>\n\t\t\t<ymin>" + str(int(tup[2]*img_h)) +
"</ymin>\n\t\t\t<ymax>" +
str(int(tup[2]*img_h)+int(tup[4]*img_h)) +
"</ymax>\n\t\t</bndbox>\n\t</object>\n")
        file.write("</annotation>")
    data = []
```

ПРИЛОЖЕНИЕ 3

Файл: example.xml

```
<annotation>
     <filename>1.jpg</filename>
     <path>1.jpg</path>
     <size>
          <width>1280</width>
          <height>720</height>
          <depth>24</depth>
     </size>
     <segmented>0</segmented>
     <object>
          <name>parking</name>
          <pose>Unspecified</pose>
          <truncated>0</truncated>
          <difficult>0</difficult>
          <occluded>0</occluded>
          <bndbox>
               <xmin>145</xmin>
               <xmax>226</xmax>
               <ymin>514</ymin>
               <ymax>609</ymax>
          </bndbox>
     </object>
</annotation>
```