МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Институт№8 «Информационные технологии и прикладная математика» **Кафедра** 810Б

Направление подготовки 02.04.02 ФИИТ **Группа** М8О-203М-18

Квалификация (степень) магистр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

На тему: Разработка системы распознавания световых сигналов транспортных средств

Автор диссертации Волков Александр Константинович
(Фамилия, имя, отчество) (подпись)

Научный руководитель Абгарян Каринэ Карленовна
(Фамилия, имя, отчество) (подпись)

Рецензент Думин Павел Николаевич
(Фамилия, имя, отчество) (подпись)

К з а щ и т е д о п у с т и т ь

Зав. кафедрой Абгарян Каринэ Карленовна
(Фамилия, инициалы) (подпись)

« 24 » мая 2020 г.

Москва 2020 г.

РЕФЕРАТ

Магистерская диссертация содержит 63 страницы, 22 рисунка, 4 таблицы, 26 использованных источников.

РАСПОЗНАВАНИЕ СВЕТОВЫХ СИГНАЛОВ, ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ, ДЕТЕКТИРОВАНИЕ ОБЪЕКТОВ, КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ, PYTHON, PYTORCH.

В данной магистерской диссертации проведен анализ, реализована и протестирована система для распознавания световых сигналов транспортных средств с помощью нейронных сетей.

Содержание

ВВЕДЕНИЕ

Автоматизация всевозможных процессов в современном мире играет ключевую роль в научном и техническом прогрессе человечества, начиная от программно управляемых станков на производстве и умных домов, заканчивая электронным документооборотом и беспилотными автомобилями. В частности, создание беспилотного, автономного транспорта является сложной задачей, требующей коллективной работы специалистов из разных областей профессиональной деятельности.

В данной работе рассматривается задача реализации системы распознавания световых сигналов транспортных средств, которая используется в составе платформы интеллектуального анализа дорожной обстановки. Подобная платформа может использоваться как для подсказок водителю, так и для создания беспилотного транспортного средства.

Реализация такой системы позволит реагировать автомобилю не только на фактическое изменения скорости или направления движения впереди идущего транспортного средства, но и на цели его водителя - будь то перестроение, поворот, либо торможение. Необходимая для анализа информация поступает с камеры, например, с видеорегистратора.

Актуальность разрабатываемой системы находится на высоком уровне. Летом 2019 года в Москве и Татарстане началось тестирование высокоавтоматизированных транспортных средств, а с 1 марта 2020 года к тестированию подключились еще 11 регионов Российской Федерации.

Целью диссертации является проектирование и реализация системы распознавания световых сигналов транспортных средств.

Для достижения поставленной цели необходимо решить следующие задачи:

- Проектирование системы распознавания;
- Обнаружение транспортных средств;
- Классификация светового сигнала;
- Фильтрация сигнала и получение его семантик

ОСНОВНАЯ ЧАСТЬ

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Задача распознавания сигналов автомобилей

1.1.1 Особенности задачи

Для безопасного перемещения по дорогам в современных условиях нужно не только хорошо понимать дорожную ситуацию, но и быстро оценивать возможные сценарии ее развития. Своевременное реагирование на изменяющееся окружение также необходимо, однако не менее важным навыком является возможность предугадать последующие изменения с целью предупреждения опасных ситуаций. Водители с опытом интуитивно предсказывают следующие действия других участников движения, и проблема заключается в том, что их приобретенный опыт очень сложно или невозможно формализовать.

Понятно, что не все водители обладают достаточным опытом, поэтому правилами дорожного движения предусмотрено заблаговременное предупреждение других участников дорожного движения о своих дальнейших действиях, которые потенциально могут привести к дорожно-транспортному происшествию – торможение и повороты, не предусмотренные изгибом дороги.

Торможение обозначается сигналом красного цвета, который излучают задние фонари. Мощность излучения стоп-сигнала должна быть выше мощности излучения габаритных огней, но при ярком солнце и недостаточной выразительности фонарей становится затруднительно понять состояние стоп-сигналов. По этой причине с 1990-х годов в Европе, Северной Америке и ряде других стран стала обязательной установка третьего, верхнего сигнала. Он должен находиться по центру за некоторыми исключениями и быть выше линии основных стоп-сигналов. Введение дополнительного фонаря обеспечило однозначность нажатия педали тормоза, особенно в странах, где допускаются сигналы поворота красного цвета. Третий стоп-сигнал может находиться на багажнике или под задним стеклом вверху и может быть реализован в как одной лампой, так и набором светодиодов.

Указатели поворота, как уже было сказано, могут быть красного цвета в некоторых странах, например, в США, а в странах ЕС, СНГ и в большинстве стран Азии и Африки сигналы поворота желтого или оранжевого цвета. Благодаря реле в старых моделях транспортных средств и электронному блоку управления в современных происходит переключение указателя поворота с постоянной периодичностью, примерно раз в секунду.

Необходимо отметить, что несмотря на требования правил дорожного движения использовать указатели поворота, ими пользуются не все водители и не во всех ситуациях. Чуть лучше ситуация обстоит со стоп-сигналами – в этом случае водителю ничего делать не нужно, и поэтому о торможении предупреждают все. Тем не менее, важно понимать цель разработки системы распознавания световых сигналов транспортных средств. Такую систему можно будет использовать в качестве вспомогательной, предупредительной, или информационной, но не в качестве системы безопасности по нескольким причинам:

Первая – световая индикация может не работать у другого автомобиля, и, хотя такие случаи редки, в вопросах безопасности не должно быть вероятности, что система никак не поможет предотвратить дорожно-транспортное происшествие и возможно, травмы или гибель водителя при условии исправности самой системы.

Вторая – системы, основанные исключительно на компьютерном зрении, могут работать с очень высокой точностью и надежностью, но всегда есть вероятность, что при определенных обстоятельствах компьютерное зрение покажет некорректный результат. Это также касается вопросов безопасности, поэтому использование такой системы без дополнения другими источниками данных недопустимо.

Следует отметить, что даже если надежность такой системы превысит уровень среднего водителя, уровень доверия водителей в среднем еще долго останется на достаточно низком уровне. Особенно сильный негативный эффект на

доверие автоматизированным транспортным средствам оказывают аварии с участием «автопилота», поэтому системы помощи водителю не должны выдаваться за гарантию безопасного вождения или беспилотный автомобиль. Тем не менее, в сочетании с другими системами может достигаться определенная степень автономности автомобиля.

1.1.2 Ограничения и требования при решении задачи

Предметная область данной задачи накладывает определенные ограничения на реализацию и работу системы.

Во-первых, нет возможности для каждого функционала беспилотного автомобиля или системы помощи водителю в частности создавать отдельный источник данных, поэтому в данной диссертации будем исходить из предположения, что в распоряжении системы имеется одна обычная RGB-камера. Впрочем, стереопара в задаче распознавания сигналов поможет несильно, как и RGBD-камера (с каналом глубины).

Во-вторых, система должна работать достаточно быстро. Существует условное деление на работу в онлайн режиме, то есть в реальном времени, и в офлайн режиме. Обычно эта граница находится на уровне 25-30 кадров в секунду, что человеческим глазом воспринимается как плавный видеопоток. В переводе на время исполнения это означает, что вся обработка кадра должна проходить менее чем за 30-40 миллисекунд. Таким образом, необходимым требованием к реализуемой системе является ее быстродействие в заданных пределах. Часто для ускорения вычислений используются аппаратные ускорители, например, Nvidia Drive AGX Xavier, включающий в себя отдельный акселератор для нейронных сетей [1]. Такое решение является достаточно дорогостоящим и предназначенным для автомобилей с уровнем автономности 2 или 3, то есть требующих реакции водителя только в экстренных ситуациях. Поэтому разработка и тестирование

системы распознавания будет проводиться на графическом ускорителе Nvidia потребительского уровня, таких как Nvidia GTX 1060 и Nvidia GTX 1080Ti.

В-третьих, по-настоящему значимыми для системы будут не все участники дорожного движения, а только те, кто так или иначе может повлиять на траекторию автомобиля с установленной системой распознавания. Этот момент необходимо учитывать при оценке качества модели.

1.2 Существующие подходы к решению задачи

Прежде чем разрабатывать свою систему распознавания световых сигналов транспортных средств, необходимо рассмотреть и проанализировать существующие решения и научные статьи на тему данной задачи. В них следует выявить как проверенные подходы с хорошими результатами, так и слабые стороны, которые следует учитывать при разработке своей системы.

1) *Vision-Based Method for Forward Vehicle Brake Lights Recognition [2], 2015*

Распознавание стоп-сигналов происходит в два этапа – сначала каскадом Хаара определяется местоположение транспортного средства, а затем методами классического компьютерного зрения, такие как выделение по порогу, бинаризация анализ формы и разница между кадрами, определяется состояние педали тормоза.

2) *Real-time Vehicle Signal Lights Recognition with HDR Camera [3], 2016*

Авторы статьи используют нейросетевой детектор SSD и данные, полученные с лидара для детектирования автомобилей, а также High Definition Range камеру для лучшего выделения ярких фонарей стоп-сигналов. По словам авторов, не существует бенчмарка по распознаванию стоп-сигналов, поэтому они собрали свой датасет для оценки качества. Датасет в открытый доступ выложен не был, поэтому проблема данных для обучения и тестирования сохраняется.

3) *HSV Color Space Based Lighting Detection for Brake Lamps of Daytime Vehicle Images [4], 2018*

В предложенном способе отсутствует детектирование, что приводит к невозможности обрабатывать изображения более чем с одним автомобилем в кадре. Для определения стоп-сигналов используется перевод изображения из RGB представления в HSV, и далее применяется бинаризация ярких красных участков.

4) *Vision-based Two-step Brake Detection Method for Vehicle Collision Avoidance [5], 2015*

Для определения торможения в данной статье применяется пороговая сегментация для нахождения регионов со стоп-сигналами, а для их классификации используется Support Vector Machine. При этом опущен вопрос детектирования транспортных средств.

5) *Daytime Preceding Vehicle Brake Light Detection Using Monocular Vision [6], 2016*

Данная работа похожа на остальные: пороговое выделение, определение яркости и так далее. Отличие заключается в дополнительной эвристике: в проверке симметричности найденных регионов.

6) *On Addressing Driving Inattentiveness: Robust Rear Light Status Classification Using Hierarchical Matching Pursuit [7], 2014*

Авторы статьи с помощью Deformable Part Model (DPM) производят детектирование автомобилей, и в дальнейшем находят кластера пикселей красного цвета, которые затем классифицируют методом иерархического соответствия. В статье приводятся результаты сравнения их подхода и подхода, основанного на анализе яркости, и сообщается, что их подход работает лучше. Однако стоит отметить, что DPM является достаточно медленным алгоритмом детектирования, что не позволяет использовать его в системах, работающих в реальном времени.

7) *Appearance-based Brake-Lights recognition using deep learning and vehicle detection [8], 2016*

В данной работе предлагается использовать детектор на основе нейронных сетей, и оценивать стоп-сигналы также с помощью методов глубокого обучения.

8) *Robust Vision-Based Daytime Vehicle Brake Light Detection Using Two-Stage Deep Learning Model [9], 2019*

Авторы этой статьи предлагают решать задачу распознавания световых сигналов в два этапа: на первом этапе с помощью детектора RetinaNet выделяются прямоугольники на изображении, содержащие автомобили, а затем классификатором DenseNet происходит определение статуса стоп-сигналов. Примечательно, что в то время, как существует несколько больших датасетов, предназначенных для детектирования, для классификации световых сигналов подобного датасета нет, поэтому авторы статьи за несколько месяцев собрали около 4000 обучающих примеров. В заключении сообщается о 99,9% точности правильных ответов, однако следует отметить, что авторы не сообщают о времени работы всей системы, и также не предоставляют код обучения моделей и датасет для анализа.

9) *Learning to tell brake and turn signals in videos using CNN-LSTM structure [10], 2017*

В рамках данной работы было предложено использование двух моделей – одной для классификации стоп-сигналов, а второй для классификации указателей поворота. Во второй модели используется рекуррентная сеть по типу LSTM, что позволяет учитывать изменение признаков во времени.

10) *An Attention-based Recurrent Convolutional Network for Vehicle Taillight Recognition [11], 2019*

Авторы статьи использовали CNN-LSTM для распознавания сигналов автомобиля. В качестве основы сети была выбрана архитектура ResNet50. Примечательно, что авторы вообще не решают задачу детектирования автомобиля, основываясь только на анализе датасета.

11) Performance Evaluation of Region-Based Convolutional Neural Networks Toward Improved Vehicle Taillight Detection [12], 2019

В данной работе используется нейросетевой детектор транспортных средств, такой как Faster-RCNN, а для распознавания световых сигналов с помощью пороговой сегментации находятся регионы с задними фонарями и по анализу гистограммы принимается решение о принадлежности к тому или иному классу.

12) DeepSignals: Predicting Intent of Drivers Through Visual Signals [13], 2019

Авторы статьи рассматривают задачу распознавания указателей поворота. Для решения задачи они применили CNN-LSTM нейронную сеть, а в качестве датасета используют собственноручно размеченный набор данных.

Таким образом, были рассмотрены двенадцать работ, так или иначе относящихся к теме диссертации. Первым интересным наблюдением является то, что чем новее статьи, тем чаще используется нейронные сети. Этот пункт следует рассмотреть отдельно.

Классическое компьютерное зрение, которое интенсивно применялось в ранних статьях, и на данный момент часто используется, но в областях, которые не потеснило глубокое обучение. Есть две причины происходящего, которые на самом деле вытекают одна из другой.

Во-первых, нейросетевое компьютерное зрение часто справляется качественнее с задачами классификации, детектирования, сегментирования и рядом других, чем обработка вручную заданных признаков, основанных на алгоритмическом компьютерном зрении. Традиционное компьютерное зрение по своей сути – это возможность писать программы, работающие с изображениями, и ключевым моментом здесь является слово «программы», то есть некоторый алгоритм действий, созданных программистом. Этот алгоритм постоянен для любых входных данных, и при добавлении новых данных программа никак не

изменится, если ее не переписать или дополнить. То есть для повышения качества работы программы программист должен придумывать улучшения, и с каждым разом улучшить результат становится все сложнее, поскольку изображения – не строго детерминированная сущность, как, к примеру, база данных, а крайне вариативная. Человек, как и многие живые существа, в процессе эволюции научились очень эффективно и качественно обрабатывать визуальную информацию, однако формализация этого процесса на данный момент не может похвастаться выдающимися результатами.

Во-вторых, один раз придумав и реализовав метод машинного обучения с использованием искусственных нейронных сетей, становится возможным добавлять новые данные в обучающую выборку и запускать переобучение модели, не затрачивая ресурсы на усложнение алгоритма, который в других условиях (поменяются обрабатываемые объекты, изменится качество и структура изображения) перестанет работать, и который придется создавать заново. В то же время нейронная сеть постарается сама найти необходимые признаки в исходных данных и наилучшим образом отвечать требованиям обучающего набора данных.

Третьим наблюдением по рассмотренным статьям является различие в поставленных задачах. В некоторых реализуются все этапы распознавания, включая детектирование транспортного средства, а в других исследуется исключительно распознавание световых сигналов. При этом чаще распознаются стоп-сигналы, чем указатели поворота. В существенном числе работ используются свои наборы данных, что не позволяет сравнивать результаты между собой даже примерно. По этой причине следует уделить особое внимание нахождению или составлению датасета со световыми сигналами.

Таким образом, было установлено, что наилучшие результаты показывают методы с использованием нейронных сетей. Также существует проблема с отсутствием набора данных для распознавания световых сигналов, что приводит к невоспроизводимости результатов и невозможности их сравнения между собой.

1.3 Нейросетевые классификаторы

Для определения типа светового сигнала автомобиля необходимо классифицировать его изображение. Классификация – это процесс определения одного или нескольких классов, к которому принадлежит объект. В данной главе будут рассмотрены самые значимые классификаторы, идеи которых легли в основу почти всех современных нейронных сетей, а также варианты, работающие в режиме реального времени.

1.3.1 AlexNet

AlexNet - одна из самых известных нейронных сетей, а научная статья, в которой она представлена, является одной из самых цитируемых среди статей на тему нейронных сетей – на данный момент на нее сослались около 61 тысячи раз [14]. Представленная нейронная сеть была предложена для классификации датасета ImageNet – 15 миллионов изображений, разбитых на 1000 классов.

Нейросеть представляла собой сверточную сеть, принимающую на вход изображения размером 224 на 224 пикселей с тремя каналами. Модель состояла из восьми слоев – пять сверточных и три полносвязных. В качестве функции активации был применен ReLU, а слои подвыборки (pooling) применялись с перекрытием областей действия.

Для борьбы с переобучением авторы статьи применили несколько техник. Первая – аугментация изображений, которая включала в себя отражение по горизонтальной оси и случайный выбор участка из изображения для обучения. Вторая – использование дропаута (dropout), который во время обучения случайным образом отключает передачу активаций между некоторыми нейронами двух соседних слоев. Без дропаута сеть заучивала исходные данные, то есть переобучалась, а с ним – нет, но при этом примерно вдвое вырастало число итераций для сходимости сети.

Для обучения использовался стохастический градиентный спуск с размером минибатча, равным 128 изображениям. Что примечательно, во время

тестирования применялась аугментация с выборками участков изображений, тем самым усредняя предсказания модели и делая ее более устойчивой к выбросам.

1.3.2 Inception

Восьмислойная сверточная нейронная сеть AlexNet заняла первое место в соревновании по компьютерному зрению в 2012 году, и для реализации такой сети потребовалась использование графического ускорителя, за счет чего многие векторные и матричные операции выполнялись быстрее, чем на центральном процессоре. Однако даже при этом нейронная сеть обучалась пять суток. Таким образом, остро стоит вопрос эффективности вычисления и обучения искусственных нейронных сетей.

С целью уменьшить вычислительные затраты при сохранении качества сети, сотрудники компании Google в 2014 году предложили новую архитектуру – Inception[15]. Основной мотивацией создания такой архитектуры послужило стремление уменьшить потребление памяти и электроэнергии для использования во встраиваемых системах. Согласно исследованиям, теоретически существуют разреженные нейронные сети, которые оптимально выполняют поставленную задачу при меньшем числе параметров, однако на текущий момент программное обеспечение и аппаратные архитектурные решения не позволяют эффективно вычислять и обрабатывать разреженные нейронные сети, что делает их неприменимыми в реальности.

Основной идеей новой архитектуры было исследование, насколько возможно достичь оптимальности разреженных сетей с помощью существующих плотных вариантов. Для проверки этого было предложено использовать параллельно три слоя свертки с разным размером фильтра: 1x1, 3x3 и 5x5, а также дополнительно слой максимальной подвыборки, что отражает идею обработки изображения на разных машстабах. После этого выводы всех слоев конкатенируются в один массив данных, который обрабатывается дальше.

Однако при таком способе фильтры 5x5 отнимают слишком много вычислительных ресурсов, поэтому было предложено понижение размерности перед фильтрами 3x3 и 5x5 путем свертки 1x1, как показано на рисунке 1.1. Использование едичничных сверток позволяет увеличивать нейронную сеть как в ширь, так и в глубь без взрывного роста вычислительной сложности.

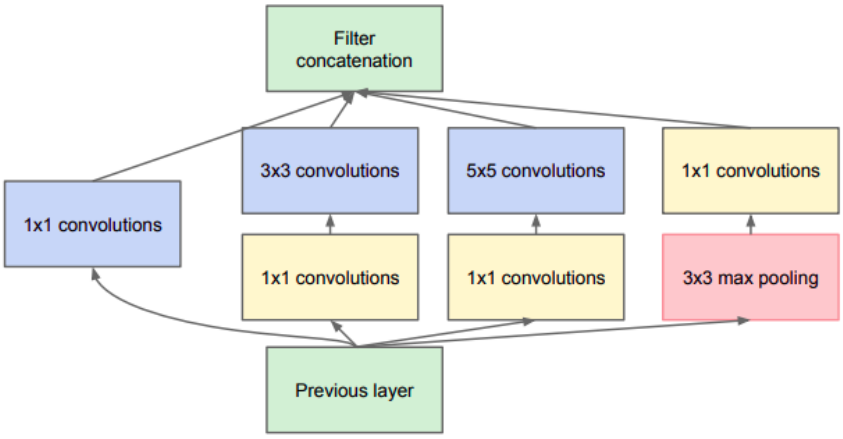

Рис. 1.1 Структура Inception блока

Таким образом, благодаря таким блокам удалось сократить вычислительную сложность при сохранении качества. В частности, предложенная модель GoogLeNet, использующая Inception блоки, содержит в 9 раз меньше параметров, чем AlexNet.

1.3.3 ResNet

Еще одна прорывная статья, вышедшая в 2015 году и открывшая путь к по-настоящему глубокому обучению [16]. Авторы сообщают, что при добавлении все большего числа слоев возникает проблема затухания градиентов. Это приводит к ситуации, когда более глубокая сеть обучается хуже и дольше, чем менее глубокая. Сравнение приводится для сверточной нейронной сети из 20 слоев и сети, состоящей из 56 слоев (рисунок 1.2).

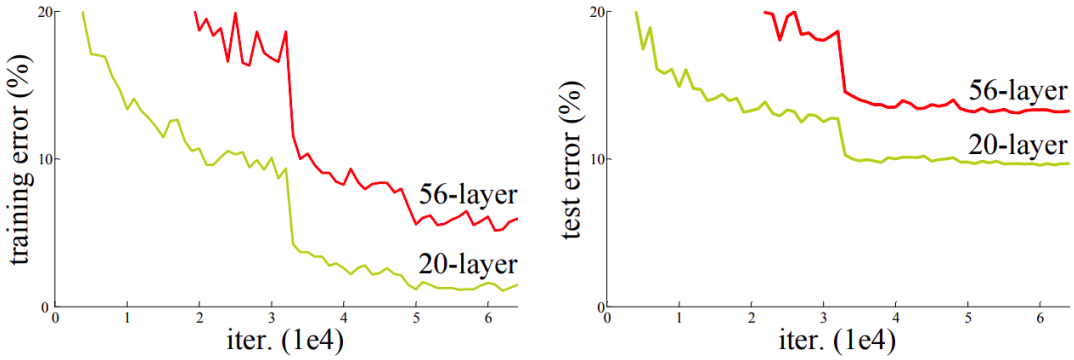

Рис. 1.2 Деградация качества сети из-за слишком большого числа слоев

Для борьбы с этим явлением была предложена архитектура остаточных сетей (residual networks). Их особенность заключается в том, что вместо традиционного последовательного соединения всех слоев используются тождественное отображение (identity mapping) совместно с остаточной функцией, как показано на рисунке 1.3. Знак (+) обозначает поэлементное суммирование.

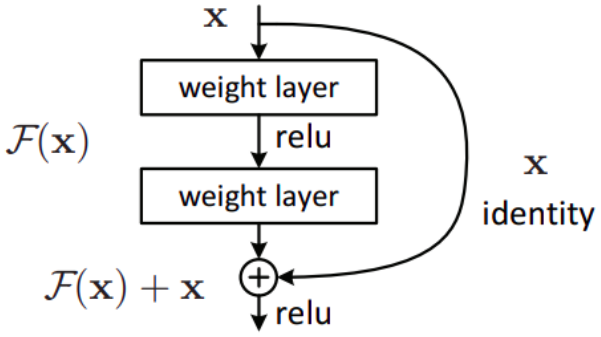

Рис. 1.3 Структура Residual соединения

В такой конфигурации нейронная сеть старается выучить не прямое преобразование данных на каждом слое, а “корректировку” тождественного отображения, что дает ряд преимуществ.

Во-первых, если тождественное отображение является оптимальным (что на самом деле бывает редко, поэтому корректнее говорить об близком к оптимальному), то для нейронной сети будет проще сделать веса остаточного соединения нулевыми, чем выучить идентичное отображение в слоях с нелинейными преобразованиями. Эта гипотеза подтверждается анализом силы активации в слоях обычной сети и остаточной. В случае остаточной сети

активация происходит слабее, что отражает идею поправки, “остатка”, в то время как прямое соединение является главным элементом в вычислении и обучении.

Во-вторых, градиент ошибки может распространяться без существенного затухания. Это позволяет обучать глубокие сети. Если в показанном выше примере сеть с 56 слоями уже уступала сети с 20 слоями, то благодаря остаточной архитектуре становится возможным обучить и 56, и 100, и 152 слоя, и даже больше.

В-третих, большим преимуществом остаточным сетей является то, что идентичное отображение практически не добавляет нагрузки на вычисления.

Использование ResNet-101 (101 слой в сети) в качестве основы вместо VGG-16 для алгоритма детектирования Faster-RCNN увеличило качество детектирования на 28%, что является хорошим результатом и говорит об улучшении представления объектов в признаковом пространстве.

На основе ResNet в последствии вышло много статей и вариаций: ResNext, ResNest, Wide ResNet, а сама идея такого рода соединений нашла отражение в очень многих работах, поскольку сейчас обучение глубоких сетей без “прямых путей” практически невозможно.

1.3.4 MobileNetV2

В 2019 году научными сотрудниками компании Google была предложена архитектура нейронной сети, еще больше понижающая требования к вычислительным мощностям и пригодная для использования в мобильных устройствах [17].

Представленная архитектура опирается на следующие изобретения:

1) Разделяемые свертки с учетом глубины. Классический сверточный слой применяется одновременно и к пространству самого изображения, и к пространству его представлений (каналов), тем самым производя большое число вычилений. Было предложено разделить такую свертку на два этапа – свертка 3x3 по пространственным координатам и затем свертка 1x1 по каналам “в глубину”.

Такой подход позволяет существенно снизить вычислительные затраты за счет уменьшения числа умножений. Существует и минус – такая свертка при использовании вместо стандартной приводит к некоторой потере качества сети, однако выигрыш во времени и используемой памяти значительно выше, что оправдывает применение разделяемой свертки.

2) Инвертированные остаточные сверточные блоки. Основные блоки вычислений в MobileNetV2 похожи на блоки вычислений из Reidual Network. Однако есть существенное отличие: если в ResNet происходило сужение количества каналов, свертка, и затем расширение числа каналов, то в MobileNetV2 процесс инвертирован. Благодаря этому достигается более эффективное использование вычислительных ресурсов.

3) Линейные узкие места. В работе сообщается, что при использовании нелинейной функции в конце блока сверток качество сети будет хуже, чем при отсутствии нелинейности. Это связано с тем, что популярные функции активации, такие как ReLU, отбрасывают значения меньше нуля. Для борьбы с этим часто увеличивают число каналов, тем самым увеличивая запоминающую способность сети.

Общая структура блока MobileNetV2 представлена на рисунке 1.4.

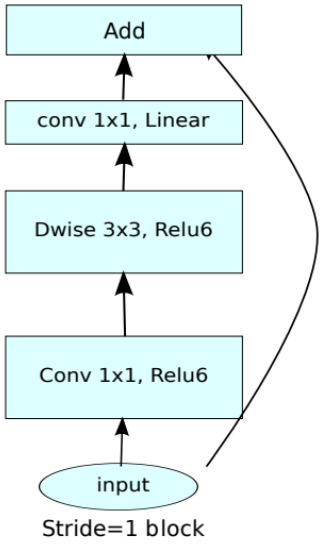

Рис. 1.4 Структура разделяемой свертки

Опыты исследователей показывают, что при сравнимом качестве данная модель работает значительно быстрее, позволяя использовать центральный процессор для вычислений нейронных сетей.

Среди рассмотренных классификаторов наилучшим с точки зрения задачи быстрого распознавания является MobileNetV2. Благодаря использованным нововведениям эта модель может работать в режиме реального времени с достаточно высокой точностью.

1.4 Нейросетевые детекторы

Зачастую изображения, полученные с камер в реальном мире, содержат несколько объектов разного типа, которые представляют интерес для анализа. В таком случае классификация изображения ничего полезного не даст, так как нет правильного ответа на вопрос, к какому классу принадлежит объект на изображении, если их несколько. По этой причине необходимо уметь локализовать объекты на изображении и классифицировать их.

Под локализацией понимается задача определения их расположения на изображении, а также таких параметров, как ширина и высота на кадре. Наиболее распространенным является способ определять объект на изображении через bounding box – прямоугольник, ограничивающий объект, то есть такой прямоугольник, внутри которого лежат все пиксели изображения, принадлежащие объекту. При этом внутрь прямоугольника может попадать значительная часть фона, которая не относится к изображению, и это не считается ошибкой детектора. При этом прямоугольники могут пересекаться, как для объектов одного класса, так и для объектов разного класса. Для детектирования через прямоугольники существует большое число размеченных наборов данных, различающихся своей тематикой, способом получения, разнообразием качества и классами и прочим. Наиболее популярными являются Pascal VOC, OpenImages, MS COCO.

Вторым по распространенности является способ определять объект через маску. Маска – это область изображения, как правило невыпуклой формы, характеризующая объект. В отличие от ограничивающего прямоугольника, все пиксели внутри маски принадлежат только одному объекту. Такой подход позволяет гораздо точнее отделять объект от фона, однако присутствуют существенные недостатки – выделение маски объекта является достаточно сложной и медленной операцией. Детектирование через маски больше похоже на задачу сегментации, однако при сегментации объекты одного класса не отделяются друг от друга, поэтому такая задача называется instance segmentation –

сегментация с учетом отдельных объектов. Пример детектирования через прямоугольники и через маски показан на рисунке 1.5.

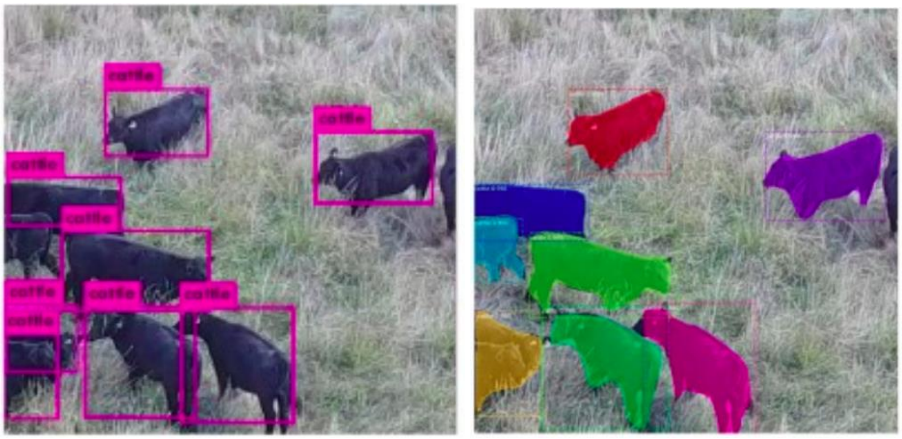

Рис. 1.5 Сравнение разных подходов к детектированию

Для оценки качества детекторов используются различные метрики, и наиболее популярной является mAP – mean Average Precision. Эта метрика измеряет площадь под графиком precision-recall кривой, при этом чем метрика выше, тем выше качество модели.

Так как в данном исследовании важным пунктом является быстродействие, то далее будут рассмотрены некоторые алгоритмы детектирования, работающие с прямоугольниками. Это будет очень малое число детекторов из всех существующих, и они были отобраны, как наибольшим образом повлиявшие на развитие моделей детектирования.

1.4.1 R-CNN

Одна из самых базовых и ключевых статей по задаче детектирования была написана сотрудниками Калифорнийского университета в Беркли в 2013 году. С начала нулевых и до 2010-2012 года доминирующим направлением в задачах визуального распознавания были SIFT и HOG – блочные направленные гистограммы. Однако в 2012 году нейросетевая архитектура AlexNet показала значительное превосходство в качестве над другими методами. Авторы статьи

приближают задачи классификации и детектирования, применяя сверточные сети для создания детектора.

Ссылаясь на результаты другого исследования, авторы сообщают, что рассмотрение задачи локализации как задачи регрессии не показывает хороших результатов, а применение сверточной нейронной сети для скользящего окна получается либо неточным, либо очень дорогим с точки зрения вычислений, в зависимости от размеров сети. Для решения проблемы недостаточности размеченных данных было предложено обучение на больших датасетах, а затем тонкое дообучение (fine-tuning) на имеющихся данных целевой области применения.

Представленная система распознавания состоит из трех модулей (рисунок). Первый генерирует набор регионов – областей изображения, предложенных для распознавания в количестве около 2000. Генерация осуществляется с помощью метода выборочного поиска, который итеративно объединяет похожие регионы, тем самым снижая их число. Второй модуль представляет собой сверточную нейронную сеть AlexNet, производящую вектор признаков, состоящий из 4096 элементов для каждого предложенного региона. Третий модуль с помощью линейных SVM классифицирует данные вектора признаков. Помимо SVM-классификатора применяется регрессия для уточнения позиции детектируемого объекта. На рисунке 1.6 представлена вся последовательность операций.

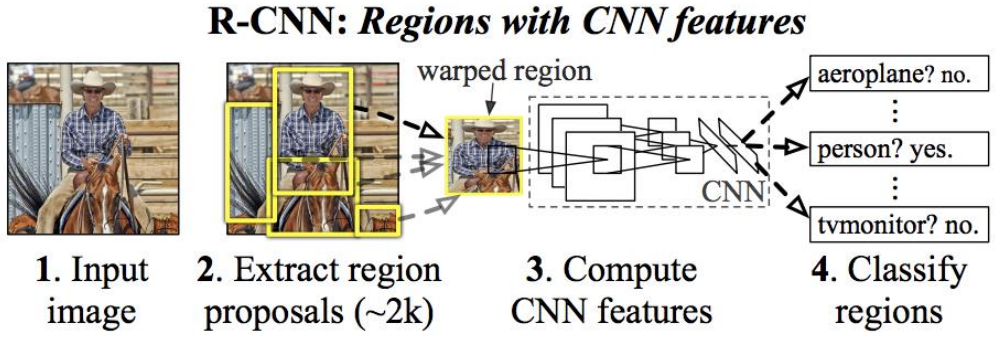

Рис. 1.6 R-CNN архитектура.

Обучение детектора происходит поэтапно – сначала обучается сверточная сеть, генерирующая признаки, а затем на полученных признаках линейные классификаторы.

На всю работу детектора требуется порядка 40 секунд при использовании видеокарты. В соревнованиях по компьютерному зрению, а точнее в задачах детектирования, данная модель в 2012-2013 году показала наилучший результат по метрике Mean Average Precision с большим опережением других участников (более 4%).

Этими же авторами в 2015 году был представлен детектор под названием Fast R-CNN, который снижал время работы детектора примерно в 20 раз до 2-3 секунд на изображение. Этого удалось достичь за счет применения сверточной нейронной сети на всем изображении, и после этого производить отбор регионов.

1.4.2 Faster R-CNN

В 2016 году была продолжена работа над идеей детекторов R-CNN и Fast R-CNN, что привело к созданию Faster R-CNN [18]. Наиболее медленным этапом в алгоритмах детектирования была генерация регионов для распознавания. Fast R-CNN без учета времени на генерацию регионов работал почти в режиме реального времени. Для преодоления этого узкого места в вычислительном контексте было предложено заменить жадные алгоритмы (Selective Search, EdgeBoxes) на сверточную нейронную сеть, которая сможет использовать преимущества графического ускорителя, которая получила название Region Proposal Network (RPN).

В отличие от других алгоритмов, которые стараются учесть информацию в изображении на разных мастшабных уровнях с помощью масштабирования исходного изображения или карт признаков, Faster R-CNN использует новые опорные прямоугольники (anchors), которые служат основой для детектирования объектов различного размера.

Для обучения использовался следующий алгоритм: поочередно обучаются то RPN, то сеть детектирования, причем во время обучения одной из сетей другая не изменяет свои веса. Такой метод обучения позволяет достичь быстрой сходимости.

Таким образом, Faster R-CNN представляет собой алгоритм детектирования, состоящий исключительно из нейронных сетей. Примечательно, что и для генерации регионов, и для классификации используются одни и те же карты признаков, что ускоряет работу системы. На рисунке 1.7 показана структура детектора.

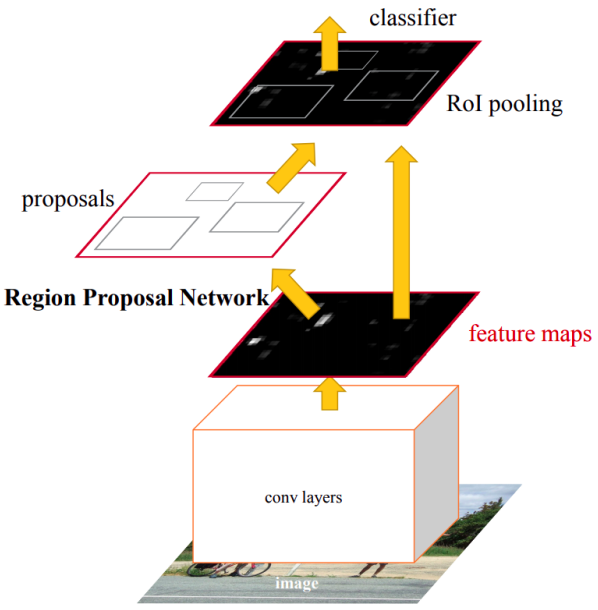

Рис. 1.7 Архитектура Faster R-CNN.

Источник: https://arxiv.org/abs/1506.01497

Задача RPN состоит в регрессии координат прямоугольника и определении наличия объекта в нем. Для этого скользящим окном анализируется каждая точка карты признаков, и для каждой точки применяется все опорные прямоугольники. При этом генерируется WxHxK всевозможных регионов, где W – ширина изображения, H – высота, а K – количество опорных регионов. В работе было предложено использование K равным девяти. Для обучения RPN использовался следующий подход – для каждого региона присваивалась метка наличия объекта в

случае если у него было наибольшее пересечение с истинным прямоугольником по метрике Intersection over Union (IoU, коэффициент Жаккара), либо пересечение по IoU со значением больше 0,7. В большинстве случаев второй критерий является достаточным, но в редких случаях, например, при маленьких объектах необычной формы, он пропускает объект и необходимо применение первого критерия. В качестве отрицательных примеров, то есть регионов, на которых представлен только фон, использовались все регионы, у которых IoU метрика была меньше 0,3. Все остальные регионы, не попавшие ни в положительные ни в отрицательные примеры, в обучении не участвовали. Таким образом, функция ошибки для RPN выглядит следующим образом:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

где i – номер изображения в обучающей выборке, pi – предсказанная вероятность наличия объекта, а p*i – истинная вероятность, ti – предсказанные параметры ограничивающего прямоугольника, t*i – истинные параметры. Lcls – логистическая функция потерь, а Lreg – сглаженная L1-норма. За счет умножения на p*i ошибка предсказания параметров учитывается только для положительных примеров. Ncls, Nreg и lambda – регуляризационные коэффициенты.

Так как одному объекту может принадлежать несколько регионов, то для устранения повторов используется non-maximum suppression (NMS). На обработку каждого изображения требуется порядка 200 миллисекунд.

1.4.3 YOLO v1, v2, v3

В мае 2016 года был выпущен препринт статьи под названием You Only Look Once: Unified, Real-Time Object Detection (YOLO v1) [19], в котором предлагалась простая по своей сути, но в то же время новая идея – решать задачу детектирования целиком как задачу регрессии. В отличие от предыдущих подходов, таких как R-CNN и DPM, использовавших алгоритмы генерации

регионов, которые в дальнейшем классифицировались, в YOLO v1 одна нейронная сеть за один проход предсказывает и параметры рамки объекта, и его класс в рамках задачи регрессии. Использование одной нейросети позволяет добиться end-to-end обучения, то есть обучения всей системы как единого целого.

Преимуществами сети выделяют высокую скорость, позволяющую работать в режиме реального времени, при этом точность детектирования выше в 2 раза по сравнению с другими алгоритмами, работающими с такой же скоростью. За счет того, что сверточная нейронная сеть рассматривает всё изображение целиком, YOLO v1 совершает в несколько раз меньше ошибок, в которых фон принимается за некоторый объект, в отличие от методов со скользящим окном, где анализируется только часть изображения без учета его окружения. Еще одним преимуществом является более высокая обобщающая способность сети. YOLO v1 ошибается меньше чем R-CNN или DPM на новых областях применений (например, сеть обучалась на реальных фотографиях, а распознает объекты на картинах).

К недостаткам же относится слабая точность локализации, то есть определения позиции объекта в кадре.

Вывод нейронной сети представляет собой регулярную сетку SxS, и в каждой ячейке сетки предсказываются:

1) N параметров ограничивающего прямоугольника в формате x, y, w, h, где x, y – координаты цента прямоугольника относительно регулярной сетки, а w, h – его ширина и высота относительно всего изображения;

2) Для каждого из N прямоугольников уверенность в том, что в прямоугольнике есть объект. При обучении этот параметр равен метрике IoU с истинным прямоугольником, при этом неважно, каким именно и какому классу он принадлежит;

3) вероятность объекта принадлежать одному из C классов. Каждая ячейка сетки предсказаний может принадлежать только одному классу, и учитываются

только те предсказания, для которых уверенность в наличие объекта выше заданного порога.

На рисунке 8 показан пример работы такой архитектуры детектора.

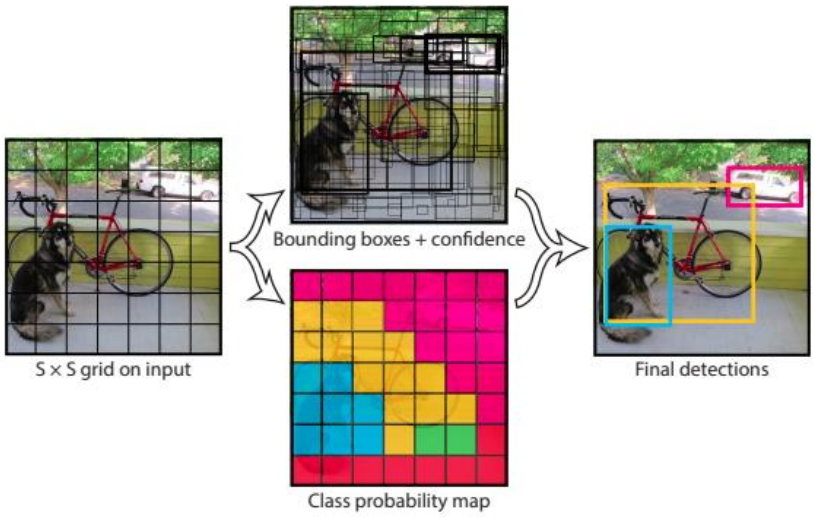

Рис. 1.8 Детектирование с помощью YOLOv1.

Таким образом, вывод сети YOLO v1 составляет SxSx(N*5+C) значений. Для датасета Pascal VOC S = 7, N = 2. Модель состоит из сверточной сети из 24 слоев с двумя полносвязными слоями в конце. Для достижения еще большей скорости работы была обучена модель всего с 9 сверточными слоями.

Скорость обучения сначала постепенно росла в течении первой эпохи с 0,001 до 0,01, а затем снижалась в десять раз каждые 45 эпох в течении 135. Для борьбы с переобучением авторы использовали дропаут и различные аугментации изображения, такие как случайные повороты, смещения, изменения цвета. NMS позволяет удалить повторное детектирование в случаях, когда большой объект детектируется в соседних ячейках.

В конце 2016 года было предложено обновление модели под названием YOLO v2 [20]. Были изменены следующие аспекты детектора:

1) Использование батч-нормализации вместо дропаута. Это позволило улучшить сходимость сети и ускорить обучение.

2) Увеличение размера входного изображения. Для сохранения небольших по размеру деталей изображения было необходимо увеличить размер входного изображения с 224 до 448.

3) Применение опорных прямоугольников. Опыты показали, что предсказывать смещение и поправки опорного прямоугольника проще, чем координаты и параметры напрямую, поэтому в YOLO v2 были использованы

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

опорные прямоугольники. Однако они были использованы с учетом регулярной сетки из YOLO v1. Таким образом, параметры предсказанного прямоугольника с объектом вычисляются следующим образом:

где cx, cy – координаты сетки, tx, ty, tw, th – выход сети, pw, ph – ширина и высота опорного прямоугольника, sigma – логистическая функция. Благодаря использованию опорных ограничивающих прямоугольников, YOLO v2 смогла детектировать более тысячи объектов вместо 98. Несмотря на небольшое падение точности с 69.5 mAP до 69.2 mAP, метрика полноты детектирования выросла с 81% до 88%.

4) Отказ от полносвязных слоев в пользу сверточных на последних слоях сети.

5) Вместо созданных вручную опорных прямоугольников использовалась кластеризация всех истинных прямоугольников в датасете. Она позволила упростить задачу для нейросети, так как уже изначально такие опорные прямоугольники лучше описывают объекты в обучающем датасете, чем заданные вручную.

6) Для повышения качества предсказаний использовалось прямое соединение с предыдущих слоев, аналогично тождественному отображению в ResNet.

7) Обучение с различными размерами изображения. Поскольку сеть состоит только из слоев подвыборки и сверточных слоев, то изменение размера входного изображение возможно без каких-либо затрат. Это позволяет обучить модель хорошо детектировать на различных входных разрешениях, что в свою очередь дает свободу выбора между точностью детектора и скоростью работы.

В 2018 году была выпущена модель YOLO v3 [21]. Основным отличием от YOLO v2 был переход от одной детектирующей “головы” к трем на разных масштабах, тем самым повышая точность детектирования как мелких, так и крупных объектов. Число опорных прямоугольников было увеличено до 9 и распределено по 3 на каждый детектор.

Также была заменена основная часть нейросети Darknet-19 на Darknet-53, состоящей из 53 слоев с residual соединениями, что позволило повысить качество распознавания.

Авторы статьи подчеркивают, что более правильными с точки зрения восприятия являются графики с осями, идущими из нулевой точки в линейном масштабе. При этом становится видно, что с точки зрения точности многие детекторы достаточно близки, однако по скорости работы YOLOv3 значительно опережает некоторый набор других детекторов (рисунок 1.9)

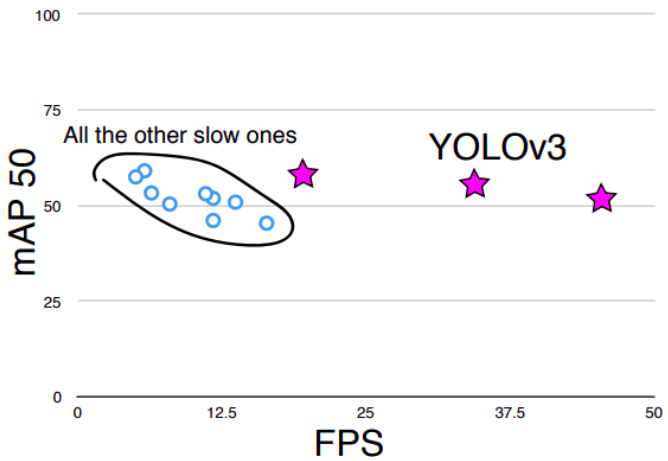

Рис. 1.9 Преимущество YOLOv3 по скорости относительно других детекторов.

Таким образом, были рассмотрены различные нейросетевые детекторы объектов. Двустадийные детекторы демонстрируют высокую точность, однако низкую скорость работы, что делает их непригодными для использования в

данной работе. Первые одностадийные детекторы решали проблему быстродействия, но при этом сильно проигрывали в точности. Однако одностадийные детекторы были улучшены, и, сохранив высокую скорость работы, стали детектировать значительно лучше, приблизившись по качеству к медленным двустадийным детекторам.

В данной диссертации будет использоваться детектор YOLOv3 из-за наилучшего баланса между качеством и скоростью работы.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Наборы обучающих данных

Нейронные сети, независимо от их размера и сложности, являются очень требовательными к объему и разнообразию обучающих данных. Набор обучающих данных часто называется “датасетом”. Чем больше датасет, тем лучше. Pascal VOC содержит около 10 000 изображений с объектами двадцати различных классов, MS COCO – более 200 000 изображений, 80 классов, а OpenImages – около 9 000 000 изображений с 600 классами. Однако стоит учитывать, что не все датасеты имеют идеальную разметку, а нейронные сети “не любят” ошибочные примеры, так как они ухудшают процесс сходимости обучения. В данной главе будут рассмотрены использованные датасеты для решения задачи диссертации.

Частой темой исследования является сохранение качества работы нейронной сети при ее применении на датасете, отличном от тренировочного. При этом есть разница между отложенной тестовой выборкой и другим датасетом – тестовая выборка составляется из того же распределения, что и тренировочная, то есть изображения будут иметь схожие характеристики, а в случае другого датасета распределение параметров изображений может быть совершенно другим. Нейронные сети, качество которых не сильно падает при переходе к другому датасету, считаются более предпочтительными, так как реальные условия применения с высокой вероятностью в чем-то будут отличаться от тренировочного и тестового датасета.

В этой связи компоненты системы распознавания при возможности будут обучаться на различных датасетах, с целью увеличения не только объема выборки, но и разнообразия примеров и характеристик изображений.

2.1.1 Датасеты для детектирования

Существуют различные датасеты для детектирования объектов как прямоугольников. Из всего разнообразия датасетов необходимо выбрать те, которые позволяют детектировать транспортные средства.

MS COCO

Microsoft Common Objects in Context - один из самых популярных датасетов, собранных из обычных фотографий различного происхождения. Как уже было сказано, в нем содержится более 200 тысяч изображений, однако не все их них содержат автомобили. Для отбора исключительно автомобилей был написан скрипт на языке Python, который в зависимости от файла разметки для конкретного изображения решает, оставлять картинку в выборке или нет. Таким образом было отобрано 14 тысяч изображений, на которых встречается автомобиль, для обучающей выборки и 2650 для тестовой. При этом были отфильтрованы слишком маленькие прямоугольники – нет смысла пытаться детектировать очень далекие автомобили, так как их световые сигналы не окажут никакого влияния на автомобиль с системой распознавания, и более того, скорее всего будут неразличимы вдали. Фильтрация происходила путем закрашивания черным цветом прямоугольника объекта, так как удалить только информацию о расположении конкретного объекта из разметки изображения нельзя – в таком случае получится, что объект, пусть и маленький, физически присутствует на кадре, но не отмечен. Такое противоречие мешает нейронной сети корректно обучаться. При этом если часть удаляемого участка попадает на видимый объект, то закрашивание производится таким образом, чтобы никакая часть корректного объекта не оказалась удалена.

Помимо изображений с транспортными средствами, из MS COCO были взяты изображения без единого транспортного средства. Это было сделано с целью обучить нейросеть тому, что не обязательно на изображении должен присутствовать искомый объект. Без такого “негативного” обучения нейросеть иногда выдает прямоугольник размером почти со всё изображение, что является некорректным ответом детектора. Таких изображений в тренировочной выборке 18 тысяч и 2 тысячи в тестовой.

Сообщалось, что разметка в MS COCO не всегда верная, однако использование всего одного класса и фильтрация маленьких объектов должны снизить влияние ошибок в разметке.

KITTI датасет

датасет KITTI - Karlsruhe Institute of Technology and Toyota Technological Institute – серия последовательных фотографий, сделанных из машины на различных улицах одного города в Германии. Особенностью данного датасета является необычное соотношение сторон изображения – три к одному, то есть очень широкий формат изображения. YOLOv3 может принимать на вход не квадратные изображения, но в таком случае все остальные изображения в обучающей выборке, а также при тестировании с камеры, должны быть такие же широкие, чтобы не сильно искажать соотношения сторон у объектов. В реальности большинство изображений имеют формат 4x3 или 16x9, поэтому было принято решение совместить три изображения из датасета в одно таким образом, чтобы соотношение сторон стало примерно один к одному. Это позволит избежать искажений при масштабировании, пример такого изображения показан на рисунке 2.1.

Рис. 2.1 Пример изображения из KITTI.

В датасете KITTI также были удалены слишком маленькие объекты, как и в MS COCO. Стоит отметить, что изображения из этого датасета лучше отражают область применения системы, чем изображения из MS COCO. С учетом

компоновки по три изображения в одно датасет KITTI представлен в общем датасете в размере 2 тысяч в обучающей выборке и 500 в тестовой.

BDD100K

Следующим, и наиболее важным с точки зрения детектирования, является датасет Berkley Drive Dataset от университета Berkley. Это наиболее диверсифицированный датасет с точки зрения как трафика и местоположения, так и погодных условий и времени суток. Все изображения в нем сделаны с помощью камеры, установленной на автомобиль, то есть ракурс максимально приближен к реальным условиям использования системы распознавания. Проводилась такая же обработка изображений, как и в случае других датасетов – удаление слишком маленьких объектов.

Для обучающей выборки было использовано примерно 70 тысяч изображений, а для тестовой – 10 тысяч.

Boxy

Датасет от компании Bosch в основном нацелен на детектирование трехмерного представления объектов, вместо плоских прямоугольников, однако их можно с помощью несложного алгоритма превратить в обычные прямоугольники без учета глубины. Из данного датасета было взято 10 тысяч изображений для обучающей выборки и 1,5 тысячи – для тестовой. Все изображения сняты камерой, установленной в автомобиле.

Однако данный датасет обладает некоторыми недостатками. Во-первых, в нем почти все изображения – это кадры с однообразного шоссе, то есть модель, обучившись на таком датасете, будет испытывать трудности в отличающихся условиях. Во-вторых, размечены только те транспортные средства, которые движутся в том же направлении, что и снимающая машина, а встречное направление не размечено (рисунок 2.2). Несмотря на то, что его почти не видно за разделителем, это может оказать негативное влияние на детектирование,

поэтому было принято решение об исключении данных из этого датасета из общей тренировочной выборки.

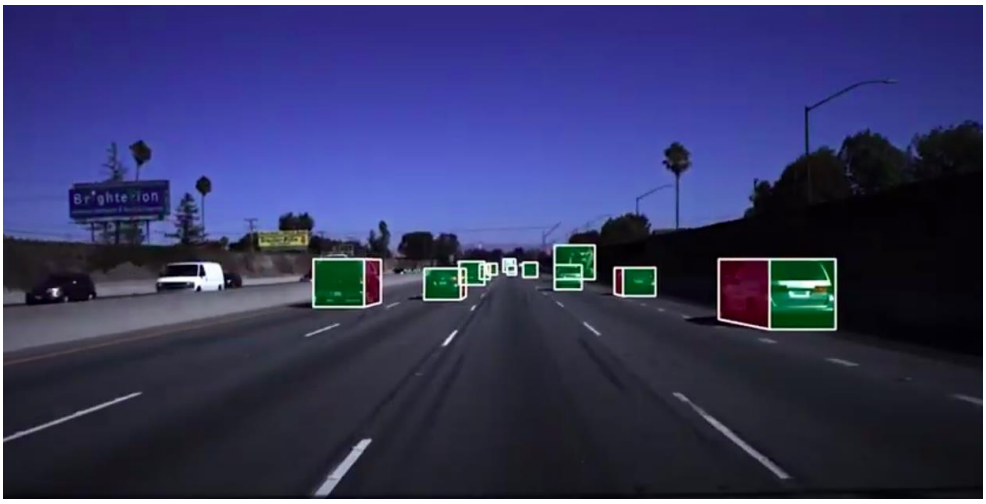

Рис. 2.2 Кубоиды из датасета Воху.

Таким образом, обучающая и тестовая выборки были собраны из трех датасетов, размером 104 тысячи и 15 тысяч соответственно. При обработке изображений учитывались особенности задачи:

1) были удалены маленькие объекты путем закрашивания их в черный цвет с целью обеспечения схождения процесса обучения и удалением информации о них из файлов разметки. Маленькие объекты, а применительно к изображению с камеры автомобиля – далекие объекты, не оказывают влияния на поведение автомобиля с системой распознавания световых сигналов.

2) в некоторых датасетах транспортные средства представлены несколькими классами – такими как обычный автомобиль, грузовик, автобус, фургон и так далее. В рамках данной диссертации не имеет значения это различие, поэтому все классы были приведены к одному – транспортное средство.

2.1.2 Датасеты классификации

После того, как модель детектирования обучится находить транспортные средства на изображении, необходимо классифицировать их световые сигналы.

Для этого необходимо найти или собрать набор данных, который будет отражать анализируемые световые сигналы.

В данной диссертации анализируются стоп-сигналы и указатели поворота. Как известно, сигналы поворота могут быть включены одновременно для сигнализации о какой-либо неисправности. Следовательно, всего существует восемь комбинаций, представленных в таблице 2.1.

Таблица 2.1 Сочетания световых сигналов

Стоп-сигнал	Левый указатель поворота	Правый указатель поворота
Нет	Нет	Нет
Нет	Нет	Да
Нет	Да	Нет
Нет	Да	Да
Да	Нет	Нет
Да	Нет	Да
Да	Да	Нет
Да	Да	Да

Rear Signal Dataset

Данный датасет – единственный датасет, который позволяет классифицировать сигналы транспортных средств. Он был создан в 2017 году силами сотрудников Калифорнийского университета и исследовательской лаборатории компании Тойота. В нем представлены все восемь классов, наблюдается дисбаланс классов в сторону классов с выключенными сигналами поворота, их значительно больше (рисунок 2.3). В датасете принято следующее обозначение: O – отсутствие сигнала, B – стоп-сигналы, L – левый указатель поворота, R – правый указатель поворота.

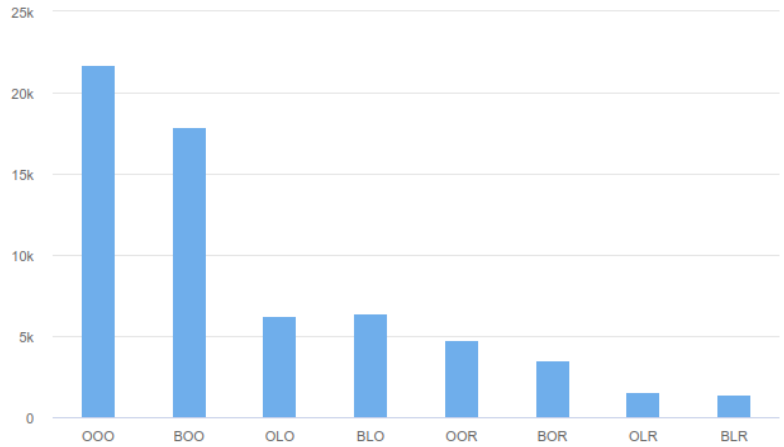

Рис. 2.3 Распределение классов в датасете Rear Signal Dataset.

Такой дисбаланс не удивителен – гораздо чаще водители применяют торможение или едут без активации рассматриваемых световых сигналов, чем пользуются указателями поворота. Пример изображений из датасета показан на рисунке 2.4.

Рис. 2.4 Пример обучающих данных.

Датасет представляет собой набор папок, в каждой из которой хранятся от 20 до 800 изображений, полученных из видео, то есть изображения организуют последовательности.

При детальном изучении датасета выяснилось, что некоторые последовательности содержат большое число практически одинаковых кадров,

поскольку автомобиль находится на одном месте перед световофором, и только окружающий фон незначительно меняется. В случае с поворотниками использование последовательностей изображений приводит к ситуации, когда на изображении в папке, обозначенной как класс, содержащий включенный указатель поворота, по факту показан автомобиль с выключенным указателем поворота. Датасет создавался для статьи, в которой авторы использовали LSTM сеть, и такое явление не создавало проблем, однако в данной диссертации LSTM нейронные сети не применяются, поэтому необходимо точно разделить изображения на соответствующие классы. Более того, поскольку классификатор обучается на одном изображении, то десятки и сотни одинаковых изображений не принесут новой информации, следовательно, их необходимо отфильтровать.

Для этого был написан скрипт на языке Python, использующий классическое компьютерное зрение и определяющий силу сигнала поворотника на каждом изображении из датасета. По результату анализа, показанного на рисунке 2.5, выбирались изображения с локально наиболее активированным состоянием и наиболее деактивированном состоянии поворотника. После этого визуально выбирались от двух до пяти примеров, которые уже использовались в обучении классификатора.

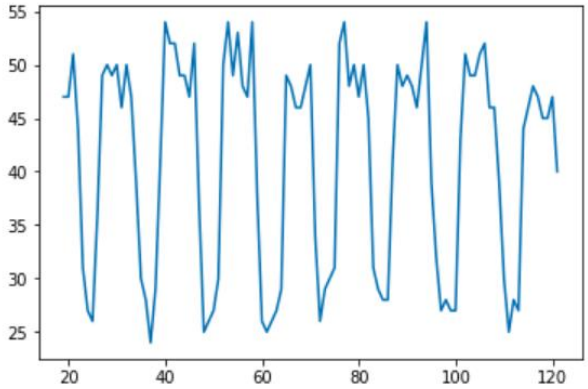

Рис. 2.5 – Работа включенного указателя поворота.

Кроме того, около трети всех изображений принадлежат всего двум автомобилям, что может привести к сильному переобучению. Таким образом, из

63 тысяч изображений реально применимы только 660 без опасения переобучения.

График на рисунке выше может навести на мысль, что если классическое компьютерное зрение хорошо может отделять включенный поворотник от выключенного, то зачем использовать нейронную сеть, которая будет работать дольше? На самом деле, у представленного датасета есть еще один недостаток. На всех изображениях отображен не весь автомобиль, а только его задняя часть. Из-за этого правила по локализации указателей поворота, которые работают с таким ограничением, не будут работать в реальный условиях, где, во-первых, транспортные средства могут быть повернуты различными углами к наблюдателю, а во-вторых, ограничивающие прямоугольники получены с помощью сети-детектора, имеющего не идеальную точность, в отличие от людей, разметивших датасет. Эти особенности и приводят к тому, что задать правила алгоритмом становится достаточно сложно, и такой алгоритм будет ненадежным.

Возможным решением данной проблемы могло быть обучение детектора находить прямоугольник не для всего автомобиля, а только для задней части, однако оказалось, что таких датасетов нет, а трехмерные кубоиды вносят слишком большую погрешность.

Авторский датасет

По описанным выше причинам использование исключительно одного датасета не представляется возможным. По этой причине был составлен свой датасет из двух источников – по различным видео из Youtube, которые являются записями с видеорегистраторов, и из «Яндекс.Зеркала» – сервиса, в котором пользователи выкладывают части своих поездок, также снятых на видеорегистраторы или на смартфоны.

В этом датасете присутствуют не только задние части машин, что позволит классификатору лучше работать в реальных условиях. Разметка осуществлялась в механизированном режиме, то есть необходимо было нажать всего одну клавишу,

чтобы определить, в какой класс будет помещено изображение. После этого автоматически показывалось следующее изображение.

Для классификации был использован существующий датасет, однако его пришлось значительно изменить для соответствия поставленной задаче и методу решения. Из-за изменения датасета в нем осталось всего 660 изображений, что может быть недостаточно для обучения хорошего классификатора, поэтому были добавлены дополнительно размеченные данные. Суммарно для обучения классификатора теперь будет использоваться 2000 изображений. С учетом техник аугментации формально различных изображений будет в сотни раз больше, однако это будет разнообразие существующей информации, но не добавление новой.

2.2 Разработка системы распознавания световых сигналов

2.2.1 Проектирование системы распознавания

Прежде чем реализовывать систему распознавания, необходимо ее спроектировать. Поскольку система основана на нейросетевом компьютерном зрении, то следует спроектировать два основных этапа – обучение нейронных сетей и алгоритм работы самой системы.

Алгоритм обучения.

Обучение детектора и классификатора происходит похожим образом, поэтому будет приведена общая схема процесса. Основным отличием являются данные и их разметка, а также метрики, по которым определяется качество модели. Процесс обучения представлен на рисунке 2.6.

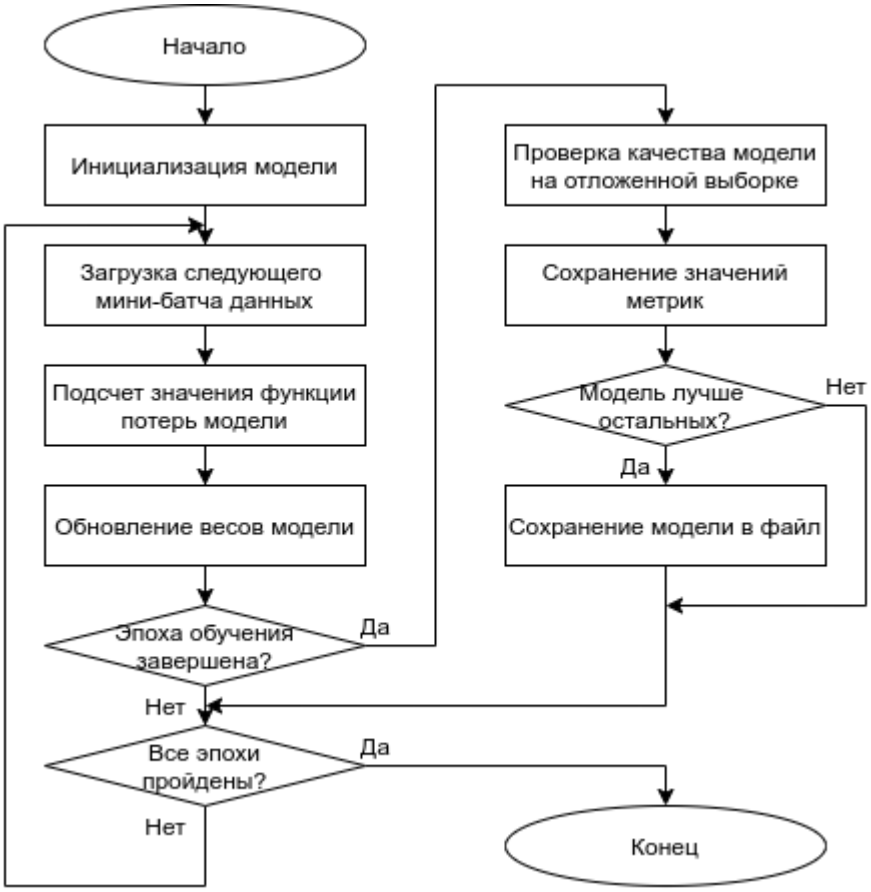

Рис. 2.6 Структурная схема процесса обучения.

Мини-батч – это несколько обучающих примеров, обрабатываемых вместе. Такой подход позволяет увеличить стабильность градиента и ускорить обучение.

Под эпохой понимается обучение на всех обучающих данных один раз. Количество эпох может варьироваться от количества данных и скорости обучения. С помощью отложенной выборки, то есть данных, которые не участвуют в обучении, проверяется качество модели, и если качество долгое время не растет (несколько эпох), то обучение можно завершать.

И для детектора, и для классификатора существует несколько метрик качества. Для детектора является стандартной метрикой качества mAP – mean Average Precision, который показывает точность детектирования как с точки зрения позиционирования объектов, так и с точки зрения определения их классов. Другим способом оценки модели являются precision и recall метрики. Precision показывает долю найденных настоящих объектов среди всех найденных, а recall –

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

долю найденных настоящих объектов среди всех существующих.

Известно, что легко получить почти идеальный precision путем увеличения порога уверенности сети в своем ответе, то есть снизить до минимума ложноположительные срабатывания (FP). Однако при этом и количество верных срабатываний (TP) также уменьшится, что приведет к сильному падению recall. Аналогично повышение чувствительности приведет к высокому recall и низкому precision.

Обычно требуется адекватная модель, у которой будет высокий и precision, и recall, поэтому была создана метрика, объединяющие эти два параметра. F1-score представляет собой гармоническое среднее между precision и recall.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Для детектора основной мерой качества будет метрика F1-score, а при близких значениях этой метрики у двух разных моделей лучше будет считаться та, у которой выше mAP.

Для классификатора будет также учитываться F1-score. Precision, recall вычисляется относительно одного класса, поэтому для учета нескольких классов существуют две стратегии усреднения – микро и макро. Микро-усреднение означает вычисление по всем объектам, независимо от класса. При таком подходе дисбаланс классов влияет на метрику. Макро-усреднение использует не учитывает дисбаланс и использует precision и recall по каждому классу независимо.

Алгоритм распознавания.

Самая главная часть в системе распознавания – это сам алгоритм распознавания. В нем используется модели, обученные на созданных датасетах. В процессе работы модели не дообучаются, что позволяет зафиксировать веса модели и ускорить выполнение программы. Сам алгоритм показан на рисунке 16.

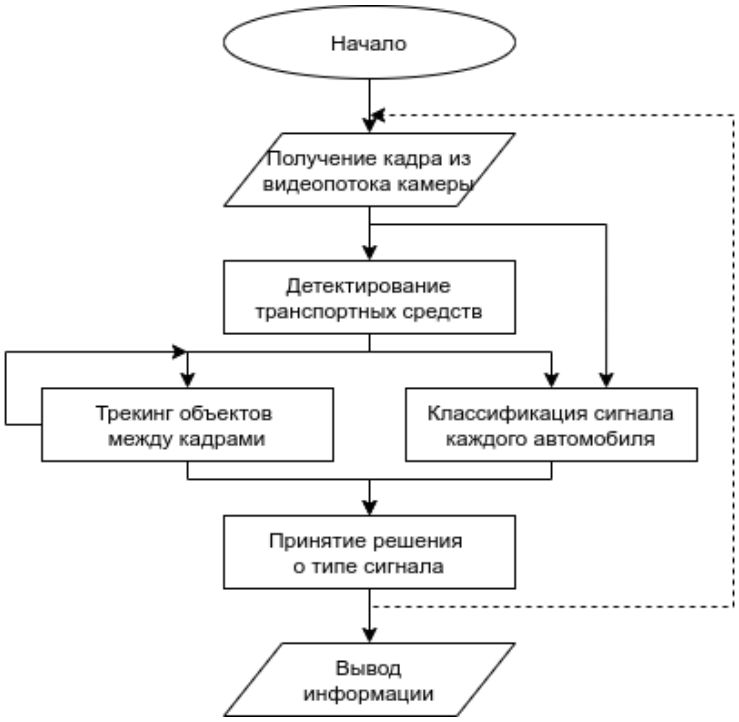

Рис. 2.7 Структурная схема алгоритма работы системы распознавания.

Глобально алгоритм не содержит условий и выполняется достаточно линейно. Разберем каждый этап работы алгоритма.

В начале происходит считывание изображения из видеопотока. Программными средствами возможно получать кадры с камеры в режиме реального времени.

Прежде чем классифицировать световые сигналы транспортных средств, необходимо найти сами транспортные средства на кадре. Для этого используется детектор YOLOv3. Результатом детектирования является список прямоугольников, в которых на кадре находятся автомобили. Данный список попадает в два модуля – модуль классификации сигналов и модуль трекинга (отслеживания).

Модуль классификации также принимает на вход изображение с камеры, чтобы используя координаты прямоугольников и исходное изображение получить изображения каждого транспортного средства отдельно. Затем эти изображения классифицируются, то есть определяется тип светового сигнала.

Модуль отслеживания (трекинга) используется для сопоставления автомобилей между кадрами. Это необходимо для получения информации о сигналах с учетом времени, что вызвано особенностью сигнализации указателя поворота. Также пользой от трекинга является возможность фильтровать выбросы в сигналах. Помимо передачи данных об идентификационных номерах транспортных средств дальше для обработки, на структурной схеме алгоритма обозначена циклическая связь модуля отслеживания с самим собой. Таким образом обозначен тот факт, что данный модуль использует информацию с предыдущего шага вычислений.

Модуль принятия решений о типе сигнала занимается фильтрацией сигнала, поступающего с классификатора, так как он может быть “шумным”, а также определяет тип сигнала указателя поворота. Благодаря трекингу становится известна история состояний световых сигналов конкретного транспортного средства, что позволяет точнее определять намерения водителей.

Вывод информации осуществляется в структурированном виде – для каждого объекта на кадре выдается его текущее состояние. В дальнейшем эта

информация может быть выведена на экран, на стекло водителю или быть передана далее в систему автопилота или систему помощи водителю.

Пунктирной линией обозначен переход к первому шагу – считыванию следующего кадра из камеры.

Использованные технологии.

Для реализации системы распознавания основным инструментом является язык программирования Python 3. Он является де-факто стандартом в разработках систем с использованием искусственного интеллекта и нейронных сетей. Благодаря большому числу уже написанных эффективных библиотек быстродействие упадет не сильно по сравнению с компилируемыми языками, такими как Java или C++. Все основные фреймворки для разработки нейронных сетей основным интерфейсом взаимодействия считают именно Python.

Обучение детектора YOLOv3 будет производиться во фреймворке DarkNet [22]. В отличие от большинства репозиториев с кодом, созданным для научных статей, данный репозиторий не был заброшен и активно развивается. Благодаря постоянным улучшениям и грамотным инженерным решениям модель YOLOv3 из данного репозитория по факту работает быстрее, и точнее заявленного изначально в статье. Многие новые методы детектирования сравнивают свои результаты с YOLOv3, однако делают это с цифрами из статьи, и из-за этого иногда случается так, что на бумаге новый метод является лучшим в своей области, а на самом деле он не превзошел хорошо проработанный уже существующий подход.

Тем не менее, даже хорошо проработанный репозиторий уступает специализированному ПО, которое нацелено на максимальную скорость исполнения. Таковым является TensorRT – разработка компании Nvidia, которая является одновременно и производителем графических ускорителей, используемых в ускорении выполнении нейронных сетей [23]. С использованием TensorRT в зависимости от различных параметров возможно ускорение работы нейронной сети от нескольких процентов до нескольких раз. Так, на видеокарте

Nvidia GTX 2080Ti в среде DarkNet модель YOLOv3 работает около 20 миллисекунд, а с использованием TensorRT – 10 миллисекунд, что означает ускорение в два раза.

Для модели классификации будет использован фреймворк PyTorch. Существует множество различных фреймворков, но наиболее популярными и развитыми являются TensorFlow/Keras и PyTorch, причем последний активно набирает популярность и признание в научном сообществе за счет своей гибкости и удобства. Данный фреймворк имеет Python и C++ интерфейсы, возможно сохранение обученных моделей. Большим плюсом PyTorch является “зоопарк” моделей, который позволяет за несколько строк кода загрузить уже обученную модель из облака и начать ее использовать.

Из используемых библиотек на языке Python стоит отметить следующие:

1) OpenCV – библиотека для компьютерного зрения и эффективной обработки изображений;

2) NumPy – библиотека для векторных и матричных вычислений;

3) Albumentations – библиотека для аугментации изображений.

Детектирование транспортных средств.

Как уже было сказано, в качестве детектора будет использоваться модель YOLOv3 из-за оптимального компромисса между качеством и скоростью работы. Однако следует отметить, что оригинальная модель обучалась для 80 классов согласно разметке датасета MS COCO.

Облегчение нейронных сетей (pruning) является одним из способов ускорить выполнение модели. В процессе итеративного облегчения путем отбрасывания ненужных нейронов качество нейронной сети будет снижаться, поэтому ее дообучают для восстановления качества. Однако, если существует возможность обучить модель за разумное время «с нуля», то можно сразу задать необходимую архитектуру и уже не облегчать ее итеративно.

Эмпирическим путем было установлено, что если снизить число карт признаков в два раза в каждом сверточном слое сети, за исключением трех первых, то при детектировании одного класса качество практически не изменится по сравнению с полной версией. При этом скорость выполнения увеличится почти в два раза, а также понизится вес модели, что может быть особенно актуально на встраиваемых устройствах. Параметры работы на видеокарте Nvidia GTX 1080Ti показаны в таблице 2.2:

Таблица 2.2 Сравнение оригинальной и облегченной модели

Модель	Время выполнения, миллисекудны	Видеопамять, мегабайт	Количество операций, BFlops
YOLOv3	15	819	65 879
YOLOv3 halved	8	505	20 376

Никаких дополнительных слоев добавлено или убрано не было, только изменено количество карт признаков.

Процесс обучения был описан в соответсвующей главе. В качестве аугментации применялся поворот, отзеркаливание, смещения в пространстве цветов.

Для ускорения работы использовался программный продукт TensorRT. Обученная модель в DarkNet описывается двумя файлами – текстовым файлом, задающим архитектуру модели, и бинарным файлом, содержащим параметры модели (веса). Перед созданием TensorRT-представления модели, ее необходимо трансформировать в подходящий формат. Таковым является ONNX (Open Neural Network Exchange), который позволяет конвертировать модели и переносить их между различными фреймворками [24]. Затем TensorRT из ONNX представления генерирует “вычислительный движок” - набор команд и CUDA-ядер, оптимизированных под конкретное физическое устройство для достижения максимальной производительности.

После создания такого движка, в него подаются изображения и производится соответствующая пост-обработка. На выходе модуля

детектирования получается список прямоугольников, охватывающих все распознанные объекты в кадре.

Классификация световых сигналов.

В качестве классификатора была выбрана модель MobileNetV2. Она также может работать в режиме реального времени, при этом показывая хорошую точность.

Для разработки модели классификации использовался фреймворк PyTorch. В “зоопарке моделей” уже была реализованная модель MobileNetV2, что позволило ускорить разработку. Более того, она уже предобучена, и несмотря на необходимость переобучения на свой датасет, обученны на ImageNet веса помогут модели быстрее сойтись.

По умолчанию в конце модели находится классификатор на 1000 классов. Он представляет собой один полносвязный слой с 1000 выходами. Поскольку в задаче диссертации максимально может быть представленно 8 классов, то становится понятно, что такой классификатор не нужен и требуется внести соответствующие изменения.

Также стоит отметить, что в задаче классификации сигналов стоп-сигналы и указатели поворотников не исключают друг-друга, как классы в MS COCO датасете. То есть на наличие признака “стоп-сигнал” не должно влиять включение или выключение поворотника. Чтобы это учесть, необходимо аккуратно подойти к вопросу выбора функции ошибки и разметки.

Базовым вариантом является кросс-энтропия. Она измеряет качество классифицирующей модели в диапазоне от 0 до 1, и чем выше значение кросс-энтропии, тем хуже модель работает.

$$Cross\ Entropy = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}),$$

Где M – число классов, c – текущий класс, o – текущий объект, y – истинный ответ (0, если объект o не принадлежит классу c, и 1, если принадлежит), p – ответ модели. Обычно кросс-энтропия применяется для

унитарного кодирования, когда объект принадлежит только одному классу, но в данном случае объекту может принадлежать несколько классов. Это один из способов задать независимость стоп-сигналов от указателей поворота.

Другой способ заключается в создании двух “голов” у нейросети. Базовая нейросеть, MobileNetV2, остается неизменной, однако вместо одного полносвязного классифицирующего слоя, данные поступают в две небольших нейронных сети, каждая из которых классицифирует свой тип сигнала. Такой подход позволяет нейронной сети сфокусироваться на признаках, характерных для каждого типа сигнала, и также появляется возможность получить вероятности для каждого класса, независимо от классов другого типа сигналов. Вероятности сигналов необходимы при фильтрации классификации. Архитектура адаптированной модели MobileNetV2 представлена на рисунке 2.9.

Рис. 2.9 Адаптированный MobileNetV2.

Так этот модуль реализуется в python-коде с использованием фреймворка PyTorch. Слои torch.nn.Linear в конце каждого блока head означает классификатор, причем последнее число в параметрах слоя отвечает за количество выходных классов.

```
      class ClassificationModel(torch.nn.Module):
          def __init__(self):
              super().__init__()
              self.basemodel =
torchvision.models.mobilenet_v2(pretrained=True).features

              self.brake_head = torch.nn.Sequential(
                  torch.nn.Conv2d(1280, 256, kernel_size=(3, 3), bias=False),
                  torch.nn.BatchNorm2d(256),
                  torch.nn.ReLU(inplace=True),
                  torch.nn.Conv2d(256, 256, kernel_size=(3, 3), bias=False),
                  torch.nn.BatchNorm2d(256),
                  torch.nn.ReLU(inplace=True),
                  torch.nn.AdaptiveAvgPool2d((1, 1)),
                  torch.nn.Flatten(start_dim=1),
                  torch.nn.Linear(256, 2)
              )
              self.turns_head = torch.nn.Sequential(
                  torch.nn.Conv2d(1280, 256, kernel_size=(3, 3), bias=False),
                  torch.nn.BatchNorm2d(256),
                  torch.nn.ReLU(inplace=True),
                  torch.nn.Conv2d(256, 256, kernel_size=(3, 3), bias=False),
                  torch.nn.BatchNorm2d(256),
                  torch.nn.ReLU(inplace=True),
                  torch.nn.AdaptiveAvgPool2d((1, 1)),
                  torch.nn.Flatten(start_dim=1),
                  torch.nn.Linear(256, 4)
              )


          def forward(self, x):
              x = self.basemodel(x)
              return {'brake': self.brake_head(x), 'turns': self.turns_head(x)}


          def get_loss(self, net_output, ground_truth):
              brake_loss = torch.nn.functional.cross_entropy(net_output['brake'],
ground_truth['brake_labels'].cuda())
              turns_loss = torch.nn.functional.cross_entropy(net_output['turns'],
ground_truth['turns_labels'].cuda())
              loss = brake_loss + turns_loss
              return loss
```

Ошибка при обучении считается как сумма ошибок каждого классификатора, заданных через кросс-энтропию.

При обучении применялись такие аугментации изображения, как повороты на случайный угол, изменения насыщенности, тона, контраста и отражение по горизонтали. При этом учитывалась метка указателя поворота, чтобы после аугментации класс менялся соответствующим образом. Следует отметить, что

многие популярные техники аугментации, такие как обрезание изображения и параллельные переносы, неприменимы в данной задаче, так как основная информация о световых сигналах может находиться на краю изображения, и если этот край не будет присутствовать на изображении, то обучать модель на таких данных не имеет смысла.

Модуль классификации выдает вероятность быть включенным для стоп-сигнала и вероятность для каждого из четырех состояний указателей поворота.

Отслеживание объектов.

Для отслеживания объектов между кадрами существует несколько различных техник. Некоторые опираются на сопоставление ключевых точек у объектов, другие оценивают гистограммы и распределения цветов. Одним из современных и активно развивающихся направлений является нейросетевая реидентификация. Ее суть заключается в том, что обученная нейронная сеть по заданному изображению выдает эмбеддинг изображения – вектор чисел фиксированной длины, который обладает следующими характеристиками:

1) Расстояние между этим вектором и другими векторами того же объекта должно стремиться к нулю;

2) Расстояние между этим векторов и векторами других объектов должно быть больше, чем расстояния до любых других векторов того же объекта.

Эти два требования записаны в функции ошибки – triplet loss, который в отличие от обычных методов классификации принимает не одно изображение, а три – анализируемую картинку, для которой строится представление; картинку с объектом того же класса или сущности; картинку с объектом другой сущности.

Примером такой модели является OsNet [25]. Она работает в режиме реального времени, однако уже обученная версия существует только для реидентификации людей, а не автомобилей. К тому же лишняя нагрузка на графический ускоритель ни к чему.

По этой причине целесообразно использовать алгоритмы, основанные на анализе местоположений и перемещений объектов. Одним из таких алгоритмов является SORT – Simple, Online, and Realtime Tracking [26]. Под термином Online подразумевается тот факт, что для своей работы он не использует данные из будущего. С помощью фильтра Калмана алгоритм предсказывает новые положения найденных ранее объектов в кадре, и затем соотносит их с реальными объектами по метрике IoU.

За счет высокой скорости работы и достаточной точности в качестве алгоритма трекинга в данной диссертации будет применяться SORT.

Обработка сигналов.

Последним модулем в очереди обработки информации является модуль обработки световых-сигналов. На вход он принимает уже не “сырые” данные – изображение, а вероятности работы каждого типа светового сигнала и идентификационный номер каждого объекта, для которого распознан сигнал.

В задачи модуля входят:

1) Небольшая фильтрация сигнала, сглаживание выбросов с целью уменьшить случайные ошибки системы;

2) Распознавание временных паттернов сигналов;

3) Выдача структурированной информации.

Фильтрация осуществляется с помощью скользящего экспоненциального среднего с окном в 20 кадров. Такой подход позволяет, во-первых, убрать редкие выбросы и сгладить уровень сигналов, а во-вторых, достаточно оперативно реагировать на реальные изменения сигнала. При этом чем более уверенный сигнал, то есть вероятность близка к 1, тем с большим весом учитываются новые данные.

Распознавание паттернов во времени позволяет корректно обработать чередование активаций указателей поворота, а также понять изменение статуса стоп-сигналов даже при первоначальном превышении некоторого порога, как показано на рисунке 2.10. Для этого использовалась неглубокая нейронная сеть,

принимающая на вход вектор состояний, который содержит в себе историю состояний конкретного автомобиля.

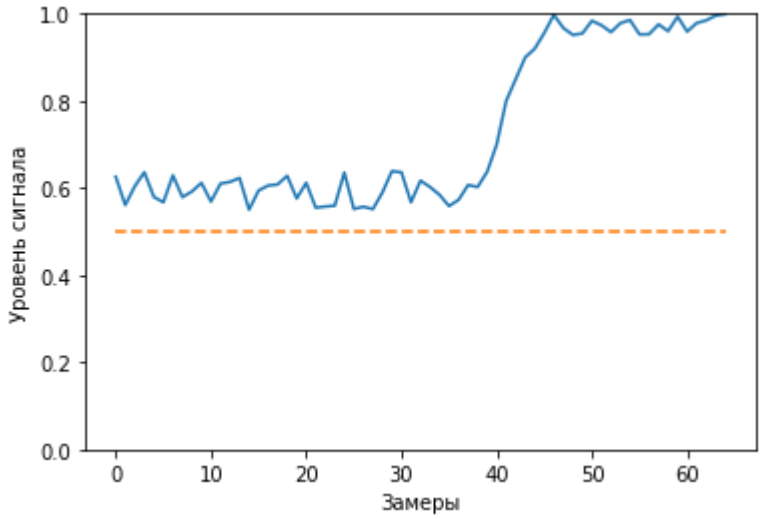

Рис. 2.10 Анализ временных паттернов.

Выдача структурированной информации производится в формате JSON. В JSON-сообщении содержится массив элементов со следующими полями:

1) bbox_x, float – центр прямоугольника по оси X в диапазоне от 0 до 1, относительно изображения;

2) bbox_y, float – центр прямоугольника по оси Y в диапазоне от 0 до 1, относительно изображения;

3) bbox_w, float – ширина прямоугольника в диапазоне от 0 до 1, относительно изображения;

4) bbox_h, float – высота прямоугольника в диапазоне от 0 до 1, относительно изображения;

5) braking, bool – флаг торможения транспортного средства;

6) turning_left, bool – флаг активности левого указателя поворота;

7) turning_right, bool – флаг активности правого указателя поворота.

Данные в таком формате поступают туда, где они будут требоваться. Это может быть более глобальная система помощи водителю, в которой учитывается состояние водителя, или же система автоматического вождения без человека.

2.3 Обучение и тестирование

2.3.1 Обучение моделей

Длительность обучения детектора измеряется в количествах итераций обучения – такова особенность фреймворка. YOLOv3 обучалась в течении 30 тысяч итераций, в каждой из которых было по 64 изображения, что эквивалентно примерно 20 эпохам. Процесс уменьшения ошибки и роста метрики mAP показан на рисунке 2.11.

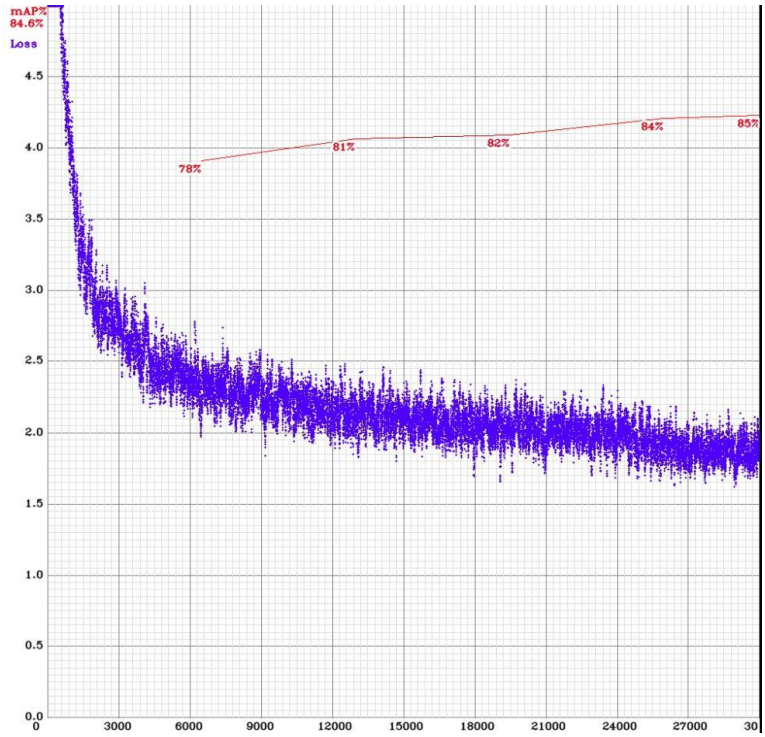

Рис. 2.11 График обучения YOLOv3.

В результате обучения была получена модель со следующими характеристиками:

Таблица 2.3 Результаты обучения детектора

Метрика	Значение
MAP@0.5	84.23%
Recall	0.76
Precision	0.89
F1-score	0.82

Данные показатели являются достаточно высокими и позволяют использовать модель в качестве детектора транспортных средств. Пример работы детектора показан на рисунке 2.12.

Рис. 2.12 Пример детектирования.

Обучение классификатора длилось в десять раз дольше – 200 эпох. Это связано с тем, что данных для обучения классификатора в разы меньше, чем для детектора. График снижения ошибки на обучающей и отложенной выборке показан на рисунке 2.13.

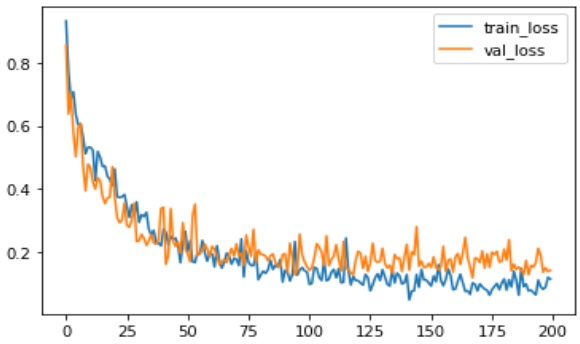

Рис. 2.13 График обучения MobileNetV2.

Для классификатора были получены следующие метрики:

	precision	recall	f1-score	support
Нет торможения	0.983	0.991	0.987	466
Торможение	0.969	0.941	0.955	135
accuracy			0.980	601
macro avg	0.976	0.966	0.971	601
weighted avg	0.980	0.980	0.980	601

	precision	recall	f1-score	support
Нет указателей	0.988	0.994	0.991	517
Налево	0.919	0.895	0.907	38
Направо	0.966	0.875	0.918	32
Аварийка	0.800	0.857	0.828	14
accuracy			0.978	601
macro avg	0.918	0.905	0.911	601
weighted avg	0.978	0.978	0.978	601

Здесь следует отметить высокий F1-score как для стоп-сигналов, так и для указателей поворота. При этом даже худший показатель – precision для аварийного сигнала – не опустился ниже 0.8, что является хорошим значением.

2.3.2 Скорость работы

Замеры времени производились на двух различных по мощности компьютерах. Сравнение представлено в таблице 2.4. Все замеры производились в миллисекундах.

Таблица 2.4 Сравнение быстродействия

Модуль	Intel i5-7500, Nvidia GTX 1060	Intel i7-7820X, Nvidia GTX 1080Ti
Детектор	16-17	10-12
Классификатор	4-12	3-7
Трекер	2-4	2-3
Обработка сигналов	2-4	1-3
Вся система в целом	25-33	18-25

Таким образом, даже на более слабом компьютере система может работать в режиме реального времени. Такой разброс объясняется тем, что время некоторых модулей существенно зависит от количества обрабатываемых объектов.

ЗАКЛЮЧЕНИЕ

В данной диссертации решалась задача распознавания световых сигналов транспортных средств. В ходе выполнения работы были достигнуты следующие цели:

- Рассмотрена специфика задачи и изучены существующие решения;
- Проанализированы нейросетевые архитектуры компьютерного зрения;
- Разработана архитектура системы распознавания световых сигналов;
- Собран репрезентативный датасет;
- Обучены модели детектора и классификатора.

В качестве детектора транспортных средств использовалась облегченная версия YOLOv3, а классификатором световых сигналов выступала адаптированная под условия задачи архитектура MobileNetv2. Модели обучались на нескольких датасетах, приведенных к единому формату.

Полученное качество моделей является достаточно хорошим для доказательства работоспособности системы. Дальнейшее улучшение качества возможно за счет увеличения объема обучающей выборки и более точного подбора гиперпараметров модели.

С точки зрения скорости работы система распознавания показала приемлемый результат. Весь процесс обработки кадра может быть выполнен за разумное время на бюджетном по меркам вычислительных центров оборудовании. Тем не менее, для полноценного встраиваемого решения следует еще сильнее ускорить процесс обработки, при этом снизив потребление графической памяти и нагрузку на процессор.

При проверке работоспособности на записях с видеорегистратора система показала себя хорошо, однако иногда наблюдались некорректные результаты для далеких автомобилей.

Список использованных источников

[1] NVIDIA DRIVE AGX / nvidia.com – URL: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/ (Access date: 18.04.2020).

[2] Wei L., Hong B. Vision-Based Method for Forward Vehicle Brake Lights Recognitio. // International Journal of Signal Processing. – 2015. – Vol.8. – No.6. – pp. 167-180.

[3] Jian-Gang W., Lubing Z. Real-time Vehicle Signal Lights Recognition with HDR Camera. / Conference: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) // ieeexplore.ieee.org. – URL: https://ieeexplore.ieee.org/document/7917112 (Access date: 18.04.2020).

[4] Yuki O., Yoshihiro S. HSV Color Space Based Lighting Detection for Brake Lamps of Daytime Vehicle Images. // Journal of Computers. – 2019. – Vol.14. – No.1. – pp. 25-30.

[5] Xueming W., Jinhui T. Vision-based two-step brake detection method for vehicle collision avoidance. // Neurocomputing. – 2016. – Vol.173. – Part 2. – pp.450-461.

[6] Hua-Tsung C., Yi-Chien W. Daytime Preceding Vehicle Brake Light Detection Using Monocular Vision. // IEEE Sensors Journal. – 2016. – Vol.16. – Issue 1. – pp. 120-131.

[7] Zhiyong C., Shao-Wen Y. On Addressing Driving Inattentiveness: Robust Rear Light Status Classification Using Hierarchical Matching Pursuit (Extended Abstract). / 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). // ieeexplore.ieee.org. – URL: https://ieeexplore.ieee.org/document/6958037 (Access date: 19.04.2020).

[8] Jian-Gang W., Lubing Z. Appearance-based Brake-Lights recognition using deep learning and vehicle detection. / 2016 IEEE Intelligent Vehicles Symposium (IV).

// ieeexplore.ieee.org. – URL: https://ieeexplore.ieee.org/document/7535481 (Access date: 19.04.2020).

[9] Duan-Yu C., Tsu-Yang L. Robust Vision-Based Daytime Vehicle Brake Light Detection Using Two-Stage Deep Learning Model. // BDIOT 2019: Proceedings of the 3rd International Conference on Big Data and Internet of Things. – 2019. – pp. 47-50.

[10] Hsu H.-K., Tsai Y.-H. Learning to tell brake and turn signals in videos using CNN-LSTM structure. // 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). – 2017. – pp. 1-6.

[11] Lee K.-H., Tagawa T. An Attention-based Recurrent Convolutional Network for Vehicle Taillight Recognition. // 2019 IEEE Intelligent Vehicles Symposium (IV). – 2019. – pp. 1-6.

[12] Wang Z., Huo W. Performance Evaluation of Region-Based Convolutional Neural Networks Toward Improved Vehicle Taillight Detection. // Applied Sciences. – Vol.9. – No.18:3753. – 2019. – pp. 1-19.

[13] Frossard D., Kee E. DeepSignals: Predicting Intent of Drivers Through Visual Signals. // 2019 International Conference on Robotics and Automation (ICRA). – 2019. – pp. 1-7.

[14] Krizhevsky A., Sutskever I. ImageNet Classification with Deep. // Advances in neural information processing systems. – 2012. – Vol.25. – No.2. – pp. 1-9.

[15] Szegedy C., Liu W. Going deeper with convolutions. // 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2015. – pp. 1-9.

[16] He K., Zhang X. Deep Residual Learning for Image Recognition. // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – pp. 770-778.

[17] Sandler M., Howard A. MobileNetV2: Inverted Residuals and Linear Bottlenecks. // The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2018. – pp. 4510-4520.

[18] Ren S., He K. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2017. – Vol.39. – Issue 6. – pp. 1137 - 1149.

[19] Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. – URL: https://arxiv.org/abs/1804.02767 (Access date: 05.06.2020).

[20] Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. – URL: https://arxiv.org/pdf/1612.08242.pdf. (Access date: 05.06.2020).

[21] Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. – URL: https://arxiv.org/abs/1804.02767 (Access date: 05.06.2020).

[22] YOLOv4 – Neural Networks for Object Detection (Windows and Linux version of Darknet). / github.com – URL: https://github.com/AlexeyAB/darknet (Access date: 05.06.2020).

[23] NVIDIA TensorRT / developer.nvidia.com – URL: https://developer.nvidia.com/tensorrt (Access date: 18.04.2020).

[24] Open Neural Network Exchange / onnx.ai – URL: https://onnx.ai/ (Access date: 18.04.2020).

[25] Zhou K., Yang Y. Omni-Scale Feature Learning for Person Re-Identification. // 2019 IEEE/CVF International Conference on Computer Vision (ICCV). – 2019. – pp. 3702-3712.

[26] Bewley A., Ge Z. Simple Online and Realtime Tracking. – URL: https://arxiv.org/pdf/1602.00763.pdf (Access date: 05.06.2020).