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

РАБОТА ПРОВЕРЕНА

Рецензент
Ведущий разработчик
ООО «ВОРТЕКСКОД»
_____ П.А. Михайлов
“___”_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор
_____ Л.Б. Соколинский
“___”_____ 2019 г.

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОПРЕДЕЛЕНИЯ И КЛАССИФИКАЦИИ АВТОТРАНСПОРТА НА ВИДЕО С ПРИМЕНЕНИЕМ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.02.2019.115-029.ВКР

Научный руководитель,
старший преподаватель кафедры СП
_____ К.Ю. Никольская

Автор работы,
студент группы КЭ-405
_____ А.А. Бурая

Ученый секретарь
(нормоконтролер)
_____ О.Н. Иванова
“___”_____ 2019 г.

Челябинск–2019

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

УТВЕРЖДАЮ
Зав. кафедрой СП
_____Л.Б. Соколинский
09.02.2019

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра

студентке группы КЭ-405
Бурой Анастасии Анатольевне,
обучающейся по направлению
09.03.02 «Информационные системы и технологии»

1. **Тема работы** (утверждена приказом ректора от 25.04.2019 № 899)
Разработка приложения для определения и классификации автотранспорта на видео с применением нейросетевых технологий.
2. **Срок сдачи студентом законченной работы:** 05.06.2019.
3. **Исходные данные к работе**
3.1 Хайкин С., «Нейронные сети: полный курс, 2-е издание», 2006.
3.2 Мошкин, В.В. «Распознавание образов транспортных средств на основе эвристических данных и машинного обучения»./ Мошкин, В.В., Сайфудинов, И.Р., Кирпичников, А.П., Шарнин, Л.М. // Вестник технологического университета, 2016. Т.19, №5.
4. **Перечень подлежащих разработке вопросов**
4.1. Провести обзор аналогов и научной литературы.
4.2. Подготовить репрезентативную обучающую выборку.
4.3. Спроектировать топологию искусственной нейронной сети и провести ее обучение.
4.4. Разработать приложение для определения и классификации автотранспорта.
4.5. Провести тестирование приложения.
5. **Дата выдачи задания:** 09.02.2019.

Научный руководитель
Старший преподаватель кафедры СП К.Ю.Никольская

Задание принял к исполнению А.А. Бурая

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Искусственные нейронные сети – математические модели, а также их программные или аппаратные реализации, построенные по принципам организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма [36].

Искусственный нейрон – элементарная ячейка вычислений в искусственной нейронной сети [39].

Синапсы – точки соединения между двумя несколькими нейронами, имеющие единственный параметр – вес. Каждый вес соответствует «силе» одной биологической синаптической связи [38].

Машинное обучение – область научного исследования, которая концентрируется на алгоритмах, способных к обучению [8].

Обучающая выборка – набор примеров, используемых для обучения, а именно для соответствия параметрам классификатора [13].

Контрольная выборка – набор примеров, используемых только для оценки эффективности полностью обученного классификатора [13].

Сверточная нейронная сеть (Convolutional neural network, CNN) – особый вид нейронных сетей для обработки данных, имеющий сеточную топологию [3].

Интенсивность транспортного потока – это число транспортных средств, проезжающих через сечение за единицу времени авт./ч. [30].

Instance-сегментация – определение пикселей, принадлежащих каждому объекту каждого класса по отдельности [19].

АКТУАЛЬНОСТЬ РАБОТЫ

В современном мире количество автомобилей превысило число в 1 млрд. (рис. 1) [9]. В связи с чем проблема регулирования потоков движения автомобильного транспорта является особенно актуальной.

Уровень интенсивности движения определяет требования к эксплуатационным качествам дороги, является важной характеристикой

объектов недвижимости, а также необходим для обеспечения повышения пропускной способности автодороги.

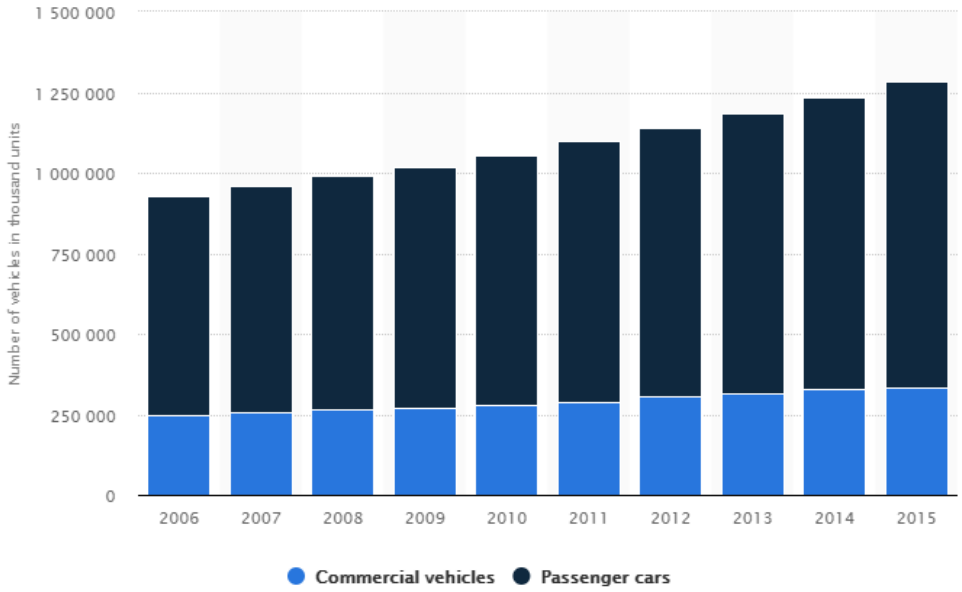

Рис. 1. Количество легковых и коммерческих автомобилей, используемых по всему миру с 2006 по 2015 год (1000 единиц)

Подсчет интенсивности движения очень важен для участков дороги с реверсивным движением (рис. 2). Необходимость введения реверсивных полос на дороге обусловлена повышенной интенсивностью движения, которое в различное время суток меняется с одного направления на другое. Выделение полосы для направления с более интенсивным движением, когда утром все едут на работу, по вечерам – домой, помогает избежать многочасовых пробок.

ЦЕЛЬ И ЗАДАЧИ ИССЛЕДОВАНИЯ

Основной целью данной работы является разработка приложения для распознавания и классификации различного автотранспорта на кадрах по видеопотоку с использованием нейросетевых технологий.

Для выполнения поставленной цели необходимо решить следующие задачи.

1. Сделать обзор аналогов и научной литературы.

2. Подготовить репрезентативную обучающую выборку.

3. Спроектировать топологию искусственной нейронной сети и провести ее обучение.

4. Разработать приложение для определения и классификации автотранспорта.

5. Провести комплексное тестирование приложения.

Рис. 2. Участок дороги с реверсивным движением

СТРУТУРА И ОБЪЕМ РАБОТЫ

Работа состоит из введения, 6 глав и заключение. Объем работы составляет 52 страницы, объем библиографии – 41 источник, объем приложения – 1 страница.

СОДЕРЖАНИЕ РАБОТЫ

Первая глава «Анализ предметной области» описывает предметную область, в рамках которой выполняется данная работа, включает в себя обзор аналогов и существующих решений данной задачи.

Вторая глава «Теоретическая часть» содержит алгоритм предобработки видеофайлов и теоретические сведения об искусственных нейронных сетях, применяющихся для решения данной задачи.

Третья глава «Требования к приложению» описывает функциональные и нефункциональные требования к приложению.

В четвертой главе «Проектирование приложения» содержит общее описание проектирования реализуемого приложения.

В пятой главе «Практическая часть» приводится техническая реализация приложения на основе поставленного списка требований.

В шестой главе «Тестирование» приведены результаты комплексного тестирования реализованного приложения.

В заключении представлены результаты проделанной работы.

Приложения содержат дополнительные материалы такие, как листинги.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Изучение предметной области

Искусственные нейронные сети представляют собой систему соединенных и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединенными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи [35].

Нейронные сети не программируются в привычном смысле этого слова, они обучаются, поэтому главным преимуществом искусственных нейронных сетей перед другими алгоритмами обработки является ее способность к обучению и использованию предыдущего опыта для исправления ошибок. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке.

1.2. Задача распознавания образов

Задачи распознавания легко решаются людьми, причем делается это, как правило, подсознательно. Попытки построить искусственные системы распознавания не столь убедительны.

Распознавание – отнесение предъявляемых объектов к определенным классам с помощью применения известных правил классификации [20]. Перед тем, как система сможет выполнять свою функцию, необходимо провести ее обучение на множестве примеров – обучающей выборке объектов распознавания.

Задача распознавания образов является основной в большинстве интеллектуальных систем. Чаще всего применяется в символьном распознавании и в компьютерном зрении – системах, цель которых заключается в формировании полезных выводов относительно объектов и сцен реального мира на основе изображений, полученных с помощью датчиков [41]. В таких системах распознавание заключается в классификации на множестве признаков, оцениваемых по наблюдаемому изображению.

Для реализации приложения для распознавания движущихся транспортных средств на видеофайле, которое относится к задаче распознавания образов, могут быть применены следующие архитектуры искусственных нейронных сетей.

Обучение с учителем:

1) перцептрон;
2) сверточные нейронные сети.

Обучение без учителя:

1) сети адаптивного резонанса.

Смешанное обучение:

1) сеть радиально-базисных функций.

В качестве входных данных могут быть различные объекты: текстовые символы, звуковые файлы, изображения или видеоряд. Во время обучения нейронной сети предоставляются примеры образов с указанием класса, к которому они относятся. Такие примеры представляют собой вектор значений признаков. Необходимо, чтобы совокупность признаков однозначно определяла класс примера, иначе нейронная сеть будет работать некорректно. После обучения сети предлагаются неизвестные ей примеры для получения ответа об их принадлежности определенному классу.

Топологии сетей, применяемых для задач распознавания образов и классификации, как правило, характерны равным количеством нейронов во входном и выходном слое.

1.3. Обзор аналогов

«Vehicle Detection in Urban Traffic Surveillance Images Based on Convolutional Neural Networks with Feature Concatenation» [17]

Разработка единой глубокой нейронной сети для обнаружения транспортных средств в городском наблюдении за дорожным движением. Авторы предложили структуру обнаружения транспортных средств, которая улучшает производительность обычного многокадрового детектора, который эффективно обнаруживает различные типы автомобилей в режиме реального времени. Предлагается использование различных экстракторов признаков для задач локализации и классификации в одной сети. Сеть обнаружения предсказывает ограничивающие рамки с оценками для каждого транспортного средства и выводит их категории одновременно. В текущее время разрабатывается задача классификации по нескольким меткам.

«Vehicle Detection» [14]

Система подсчитывания автомобилей, реализованная при помощи алгоритма машинного обучения – метода опорных векторов (Support vector machine, SVM) на основе гистограммы направленных градиентов (Histogram of Oriented Gradients, HOG (рис. 3). Данная система имеет показатель точности в 80%. Для реализации системы «Vehicle Detection» были использованы готовые алгоритмы библиотеки компьютерного зрения OpenCV [26].

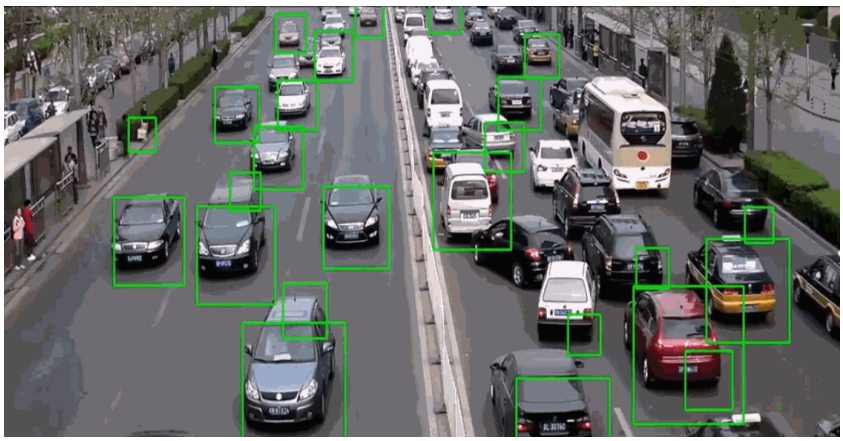

Рис. 3. Система подсчета автомобильных средств «Vehicle Detection»

«Vehicle Detection, Tracking and Counting» [15]

Система обнаружения автомобильных средств по видеопотоку (рис. 4), реализованная при помощи каскада Хаара (набор признаков цифрового изображения, используемых в распознавании образов). Для реализации приложения были использованы библиотеки компьютерного зрения OpenCV, cvBlob [22] и Boost [21].

Рис. 4. Система подсчета автомобильных средств «Vehicle Detection, Tracking and Counting»

«VIERO» [16]

Система подсчета транспортных средств VIERO предоставляет данные о количестве транспортных средств на любом участке дороги при любых погодных условиях на основе методов обработки изображений. Система отправляет все данные в указанный пользователем центр дистанционного управления как проводным, так и беспроводным способом. VIERO генерирует данные о средней скорости транспортных средств, измеряя мгновенную плотность транспортных средств, и работает в двух разных режимах для дня и ночи, достигая результатов с ошибкой менее 5 %.

«**PureActiv**» [10]

PureActiv Vehicle Counting превращает видеокамеры в интеллектуальные датчики подсчета. Данная система широко применяется в многоуровневых парковках. Его автономная конструкция позволяет обнаруживать и подсчитывать транспортные средства, используя видео, полученное с IP-видеокамер. Все данные передаются в системы управления парковкой через XML по TCP / IP. Программное обеспечение даже стабилизирует видеоизображение, чтобы удалить эффекты камеры и вибрации. Как только объект обнаружен, применяется фильтр, чтобы избежать подсчета предметов, не являющихся транспортными средствами, таких как люди.

1.4. Обзор имеющихся решений для реализации проекта

Для корректной работы реализуемого приложения понадобится отслеживать перемещение автомобильных средств. Были найдены следующие алгоритмы реализации:

«**Mean-shift**» [25]

Метод «сдвига среднего» – это непараметрическая техника анализа пространства признаков. Данный алгоритм используется для сегментации изображений и слежения за движущимися объектами (рис. 5). Mean-shift обладает достоинствами, указанными ниже.

1. Масштабируемость (время работы линейно зависит от количества обрабатываемых точек).

2. Робастность (процедура дает устойчивые результаты при изменении входных параметров).

3. Возможность параллельной обработки.

Тем не менее, метод обладает и недостатками:

1) чувствительность к размерам окна;

2) значительное увеличение времени работы алгоритма при росте размеров окна [18] (этот недостаток может быть устранен применением систем параллельных вычислений).

Рис. 5. Результат работы алгоритма Meanshift

«**Camshift**» [25]

Camshift (Continuously Adaptive Meanshift) – алгоритм отслеживания движения, который отслеживает информацию о цвете движущихся объектов по видеопотоку. Данный алгоритм использует метод «сдвига среднего» и является его усовершенствованием, поскольку устойчив к размерам окна (рис. 6).

Рис. 6. Результат работы алгоритма Camshift

«**ICP**» [32]

ICP (Iterative Closest Point) – метод, устанавливающий соответствия между парами точек выравниваемых наборов данных на основании выбранного критерия близости точки до множества (обычно используется метод нескольких ближайших соседей) и вычисляющий выравнивающее преобразование путем минимизации функционала среднеквадратичной ошибки. Затем происходит уточнение соответствия между парами точек, и эти два шага повторяются до удовлетворения критерию локального минимума. Алгоритм весьма чувствителен к выбору начального приближения и мало приспособлен к работе с зашумленными данными.

Вычислительная эффективность этого метода зависит от количества точек для сопоставления.

«Faster R-CNN» [12]

Архитектура искусственных нейронных сетей, вычисляющая регионы, на которых предположительно имеются какие-то объекты, не по изна-

чальному изображению, а по карте признаков. Для этого был добавлен модуль под названием Region Proposal Network. В рамках этого модуля небольшие нейронные сети с небольшим (3x3) окном проходят по извлеченным признакам. Архитектура модели представлена на рис. 7.

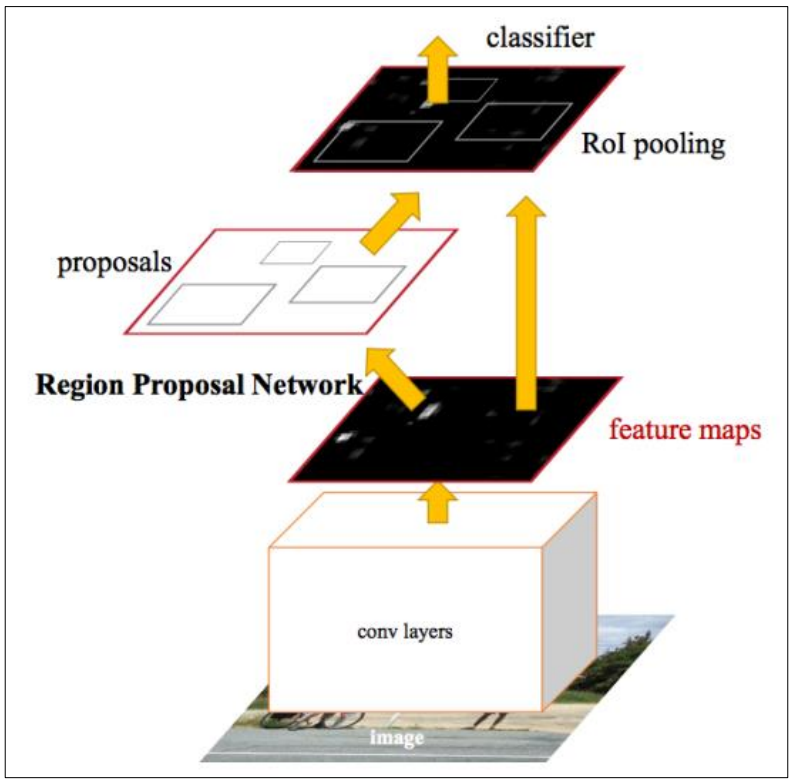

Рис. 7. Архитектура Faster R-CNN

«**Mask R-CNN**» [4]

Модель, обнаруживающая объекты на изображении, и одновременно генерирующая ограничивающие рамки и маски сегментации для каждого экземпляра и являющаяся развитием архитектуры «Faster R-CNN» (рис. 8). Главными достоинствами модели являются простота в обучении, высокая скорость работы и возможность применения ко многим задачам машинного обучения.

Модель имеет дополнительную ветвь для предсказания положения маски объекта (прямоугольной матрицы, где 1 на определенной позиции указывает на принадлежность пикселя объекту заданного класса, 0 на отсутствие принадлежности), работающую параллельно с ветвью для распознавания ограничивающего прямоугольника.

RolAlign – процедура, вычисляющая матрицу признаков для определенного региона, которая в отличии от подобной процедуры в обычных сверточных нейронных сетях не использует округление дробных значений до целых, все числа остаются действительными, а для вычисления значений признаков используется билинейная интерполяция по четырем ближайшим целочисленным точкам. Архитектура модели представлена на рис. 9.

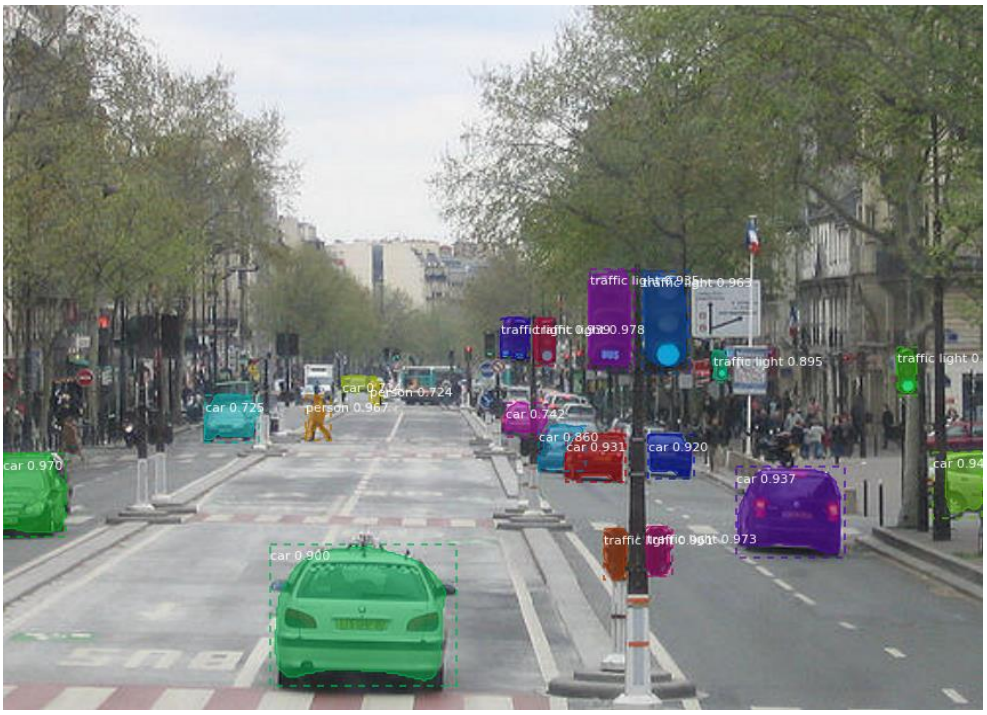

Рис. 8. Результат работы нейронной сети с архитектурой Mask R-CNN

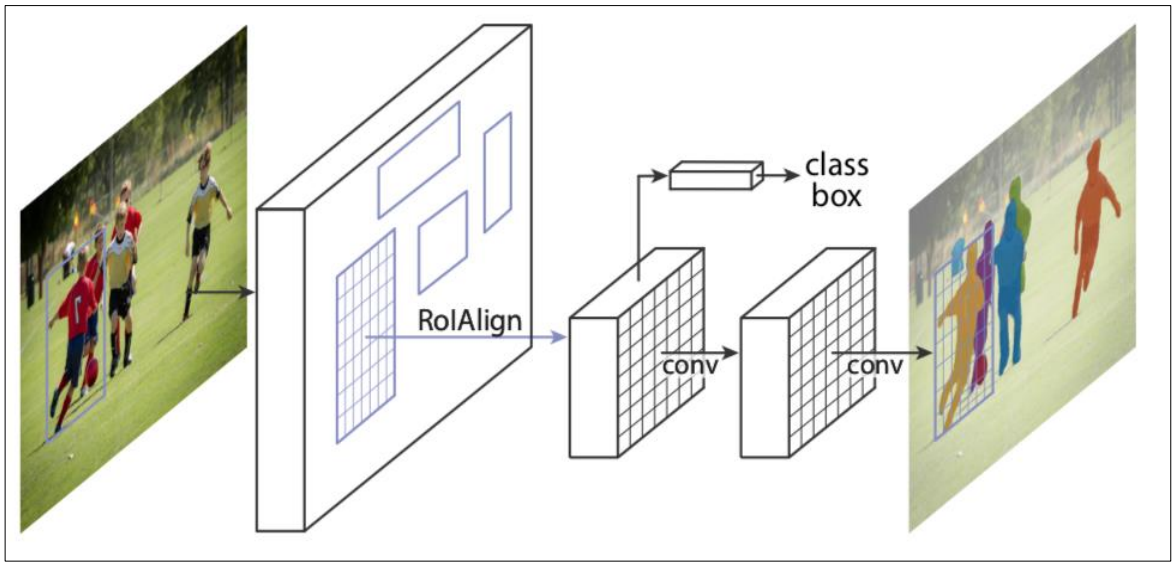

Рис. 9. Архитектура Mask R-CNN

«Yolov3» [11]

Алгоритм обнаружения объектов в реальном времени. Проект использует одну искусственную нейронную сеть для полного изображения, которая делит его на регионы и предсказывает ограничивающие рамки и вероятности для каждого региона (рис. 10), в отличие от других систем предварительного обнаружения, которые изменяют классификаторы или локализаторы для обнаружения. Данный алгоритм имеет преимущества перед схожими системами на основе классификаторов:

1) просмотр всего изображения во время тестирования, поэтому все прогнозы основываются на общем контексте изображения;

2) увеличение скорости работы алгоритма из-за использования единой оценки сети для прогнозов в отличие от систем таких, как R-CNN.

3) просмотр всего изображения во время тестирования, поэтому все прогнозы основываются на общем контексте изображения;

4) увеличение скорости работы алгоритма из-за использования единой оценки сети для прогнозов в отличие от систем таких, как R-CNN.

Алгоритм предусматривает обработку, как изображений, так и данных, поступающих на вход с пользовательской веб-камеры.

Yolo делит входное изображение на сетку размером S×S. Каждая ячейка предсказывает только один объект и фиксированное количество граничных блоков. Однако данное правило одного объекта ограничивает расстояние, насколько близко обнаруженные предметы могут находиться. Для этого у алгоритма имеются ограничения на расстояние между близкими объектами.

Алгоритм использует искусственную нейронную сеть с 24 сверточными слоями, за которыми следуют 2 полносвязных слоя. В некоторых слоях свертки в качестве альтернативы используются слои сокращения 1×1 для уменьшения глубины карт объектов. Также есть более быстрая, но менее точная версия алгоритма Fast Yolo, которая использует только 9 сверточных слоев с более мелкими картами объектов.

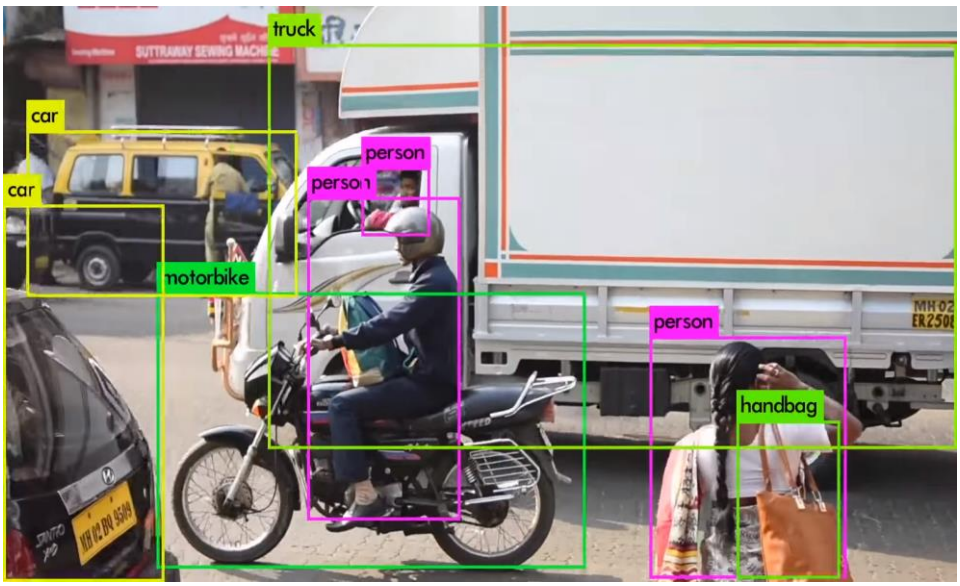

Рис. 10. Пример работы алгоритма Yolov3

1.5. Обзор готовых решений для создания нейронных сетей

Существует множество реализованных библиотек и программных интерфейсов для работы с нейронными сетями:

TensorFlow [28]

Программная библиотека для машинного обучения с открытым исходным кодом, разработанная для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов (Python, C++, Java) достигая качества человеческого восприятия. Поддерживает создание современных моделей, не жертвуя скоростью или производительностью.

Keras [24]

Высокоуровневый программный интерфейс, нацеленный на оперативную работу с сетями глубинного обучения (Python). Основными характеристиками являются минимальность, модульность и расширяемость.

Выводы по первой главе

В соответствии с целью работы в первой главе был проведен обзор аналогов реализуемого приложения и уже имеющихся решений для упрощения реализации проекта. Наличие аналогов реализованных по разным

технологиям подтверждает актуальность поставленной задачи, а большой выбор алгоритмов работы для отслеживания движения, программных продуктов и библиотек, предназначенных для работы с нейронными сетями, демонстрирует широкий спектр проблем, которые можно решить с их помощью, а также их популярность.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1. Сверточные нейронные сети

Для решения поставленной задачи будет применена сверточная нейронная сеть, так как такая архитектура может обеспечить частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. На данный момент сверточные нейронные сети и их модификации считаются лучшими по точности и скорости алгоритмами нахождения объектов.

Сверточные нейронные сети состоят из нескольких слоев: сверточные слои, субдискретизирующие (subsampling, подвыборка) и слои «обычной» нейронной сети (полносвязные) – персептрона (рис. 11) [6].

Первые два слоя (сверточный и субдискретизирующий), чередуясь между собой, формируют входной вектор признаков для персептрона.

Сверточный слой включает в себя ядра свертки, которые являются матрицами весов, обрабатывающие предыдущие слои по фрагментам (рис. 12).

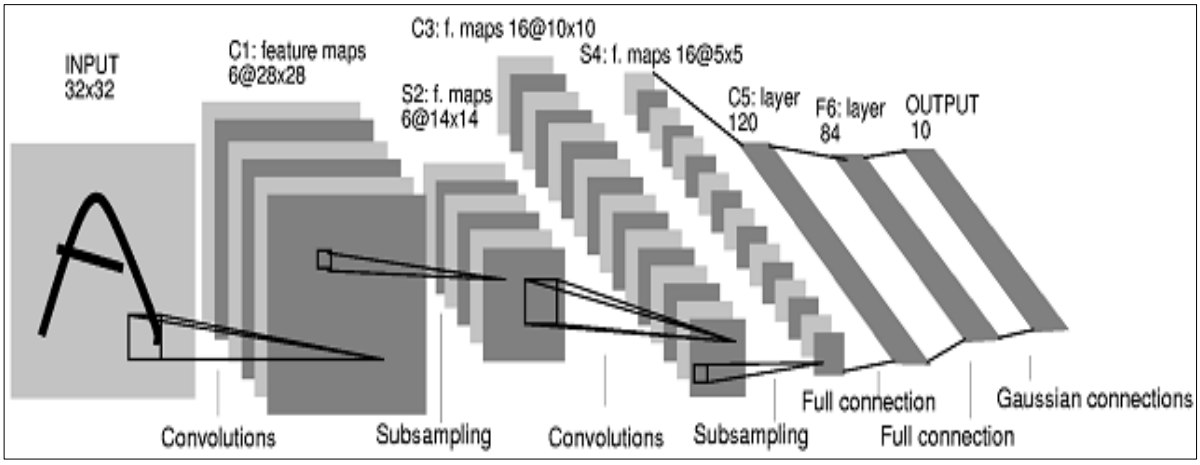

Рис. 11. Топология сверточной нейронной сети

В сверточных нейронных сетях ядро – это фильтр, находящий определенные признаки объектов. Обычно размер ядра варьируется в пределах от 3×3 до 7×7. Ядро с недостаточным размером не сможет выделить необходимые признаки, а слишком большой размер может привести к увеличению количества связей между нейронами.

Ядро представляет собой систему разделяемых весов или синапсов, это одна из главных особенностей сверточной нейронной сети. В обычной многослойной сети очень много связей между нейронами, то есть синапсов, что весьма замедляет процесс детектирования. В сверточной сети – наоборот, общие веса позволяет сократить число связей и позволить находить один и тот же признак по всей области изображения [37].

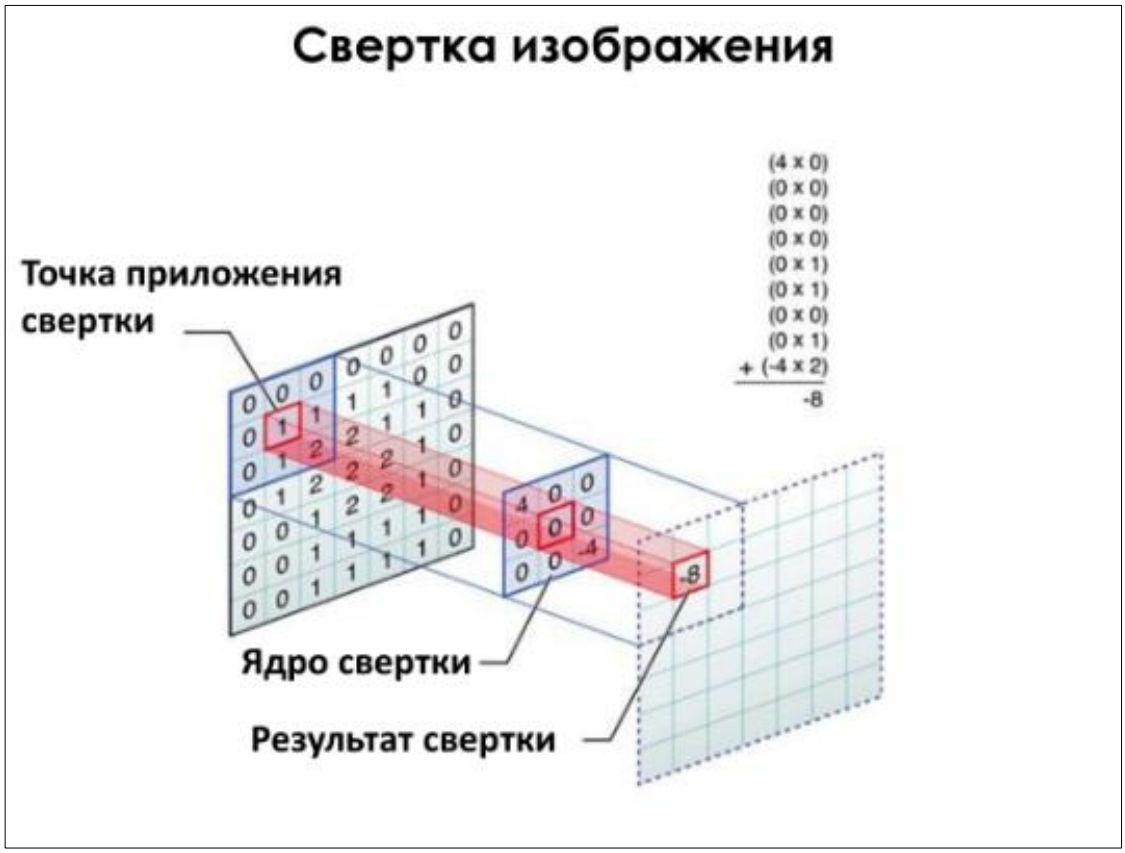

Рис. 12. Применение ядра сввертки

Целью следующего типа слоев – субдискретизирующего, является уменьшение размерности карт признаков предыдущего слоя и фильтрация уже ненужных деталей, помогающая предотвратить переобучение сети. Использование этого слоя позволяет улучшить распознавание образцов с измененным масштабом (уменьшенных или увеличенных).

Последний из типов слоев – полносвязный. Это слой обычного многослойного персептрона. Основной целью слоя является классификация и моделирование сложной нелинейной функции, оптимизируя которую, качество распознавания улучшается.

2.2. Классификация автомобильного транспорта на основе сверточной нейронной сети

Изначально при проектировании приложения планировалось использовать подход, классифицирующий автотранспорт на изображении с применением сверточных нейронных сетей. Была выбрана и разработана топология сверточной нейронной сети, состоящая их четырех сверточных слоев, четырех слоев субдискретизации и двух полносвязных слоев. Для реализации данной топологии искусственной нейронной сети был выбран язык программирования Python 3 и программные библиотеки Keras и Tensorflow.

Во время предварительной обработки данных, предоставленных ЗАО «Интерсвязь», необходимые изображения транспортных средств были вырезаны с кадров видеофайлов с помощью бесплатной программы для ОС Windows Monosnap (рис. 13).

Рис. 13. Пример обучающих и тестирующих данных

Обучение нейронной сети производилось в течение 20 эпох. В среднем на одну эпоху тратилось 27 секунд. По итогам тестирования обученная модель показала качество в 95,13 %. Это означало, что нейронная сеть ошиблась в 60 элементах тестовой выборки из 1 245 изображений.

Однако данный метод показал свою неприменимость к работе с видеопотоком онлайн и оказался крайне неэффективен из-за подачи на вход нейронной сети изображения только отдельного автомобиля, что означало невозможность детекции всех автотранспортных средств на общем кадре. Поэтому было принято решение не развивать этот подход, а использовать другой, а именно модель сверточных нейронных сетей Mask R-CNN для эффективного решения задачи instance-сегментации.

2.3. Предварительная обработка видеофайла

В данной работе обнаружение и классификация автотранспорта будет производиться по видеофрагменту с транспортным движением. Поскольку для решения поставленной задачи будут применяться нейронные сети, которые обрабатывают последовательность изображений, то поступающий на вход нейронной сети видеофрагмент необходимо разбить на последовательные кадры. Для решения этой подзадачи существует множество готовых программных библиотек таких, как OpenCV, которые позволят приложению производить покадровую обработку данных.

Из-за ограниченной памяти графического процессора все изображения будут изменены в размерах до разрешения 1280×520. Для того, чтобы повысить устойчивость работы искусственной нейронной сети, была проведена аугментация кадров с помощью следующих методов:

1) отражение;

2) изменение контраста и яркости;

3) введение искусственных окклюзий.

Окклюзии были сгенерированы посредством двух методов: добавлением частичной маски произвольного объекта, который занимает минимум 40 пикселей, или с помощью сильного размытия на видимой части дорожного полотна. Чтобы воспроизвести неисправности камеры, были случайным образом размыты высокие горизонтальные участки вблизи центра изображения.

2.4. Алгоритм детекции автотранспорта

Для реализации приложения был использован алгоритм отслеживания на видео Simple Online and Real-time Tracking (Sort) [1], поскольку он имеет хороший компромисс между скоростью и точностью. Это алгоритм, предназначенный для отслеживания нескольких объектов, основанный на элементарной ассоциации данных и методах оценки состояния. Sort обеспечивает необходимую функциональность для отслеживания согласованности объектов обнаружения в нескольких кадрах и подходит для приложений онлайн-отслеживания, в которых доступны только прошлые и текущие кадры, а метод создает идентификацию объектов на лету.

Работа алгоритма состоит из двух этапов. Первый этап извлекает карты признаков и предлагает регионы для второго этапа, который затем классифицирует объект в предложенном регионе. Преимущество этой структуры состоит в том, что параметры разделяются между двумя этапами, создавая эффективную структуру для обнаружения. Сама сетевая архитектура может быть заменена на любую другую для повышения производительности обнаружения.

Для распространения идентификатора цели в следующем кадре используется модель движения, которая аппроксимирует межкадровые смещения каждого объекта линейной моделью постоянной скорости, которая не зависит от других объектов и движения камеры. Состояние каждой цели моделируется как:

$$X = [u, v, s, r, \dot{u}, \dot{r}, \dot{s}]^T,$$

где u и v представляют горизонтальное и вертикальное расположение пикселей в центре цели, s и r представляют масштаб (площадь) и соотношение сторон ограничительной рамки цели соответственно, а $\dot{u}, \dot{r}, \dot{s}$ представляют производную по времени перечисленных ранее параметров относительно предыдущего кадра. Соотношение сторон считается постоянным. Если обнаружение не связано с целью, ее состояние просто прогнозируется без коррекции с использованием линейной скоростной модели.

Выбранный подход обеспечивает точность, сопоставимую с более сложными онлайн-трекерами. Кроме того, благодаря простоте данного метода отслеживания, трекер обновляется с частотой 260 Гц, что более чем в 20 раз быстрее, чем другие современные трекеры.

Выводы по второй главе

Во второй главе были приведены теоретические сведения об искусственных нейронных сетях и их разновидностях для решения поставленной задачи. Также был определен этап предобработки изображений для искусственной нейронной сети и описан алгоритм детекции автотранспортных средств.

3. ТРЕБОВАНИЯ К ПРИЛОЖЕНИЮ

Для распознавания и классификации различного автотранспорта на кадрах по видеопотоку должно быть создано приложение, обеспечивающее возможность автоматического распознавания и классифицирования большого количества видеофрагментов.

3.1. Функциональные требования

Функциональные требования – это требования к поведению системы, они определяют функциональность разрабатываемого программного обеспечения, т.е. описывают возможности, которые предоставляет система [34, 40]. Разрабатываемое приложение должно удовлетворять следующим функциональным требованиям:

1) приложение должно принимать на вход пользовательский видеофрагмент в формате mp4;

2) приложение должно предоставлять возможность запуска процесса определения и классификации автотранспорта на выбранном видеофайле;

3) приложение должно определять и классифицировать обнаруженный автотранспорт на видеофайле;

4) приложение должно предоставить пользователю видеофайл с классифицированным автотранспортом.

3.2. Нефункциональные требования

Нефункциональные требования – это требования к характеру поведения системы, которые описывают, как должна работать система, свойства и ограничения, накладываемые на разрабатываемое программное обеспечение [34, 40].

В ходе обзора уже имеющихся решений для реализации приложения, представленного в первой главе, были найдены необходимые библиотеки, поэтому разрабатываемое приложение для определения и классификации автотранспорта на видеофайле должно удовлетворять следующим нефункциональным требованиям:

1) приложение должно быть реализовано на языке Python;

2) алгоритм классификации/сегментации автотранспортных средств должен выполняться с помощью искусственных нейронных сетей;

3) приложение должно быть единым и не требовать от пользователя дополнительных действий для работы, кроме указания входных данных и возможности запуска алгоритма.

3.3. Варианты использования приложения

Для проектирования приложения был использован язык графического описания для объектного моделирования UML. Была построена модель взаимодействия актера «Пользователь» с приложением определения автотранспорта. Диаграмма вариантов использования (use-case diagram) представлена на рис. 14.

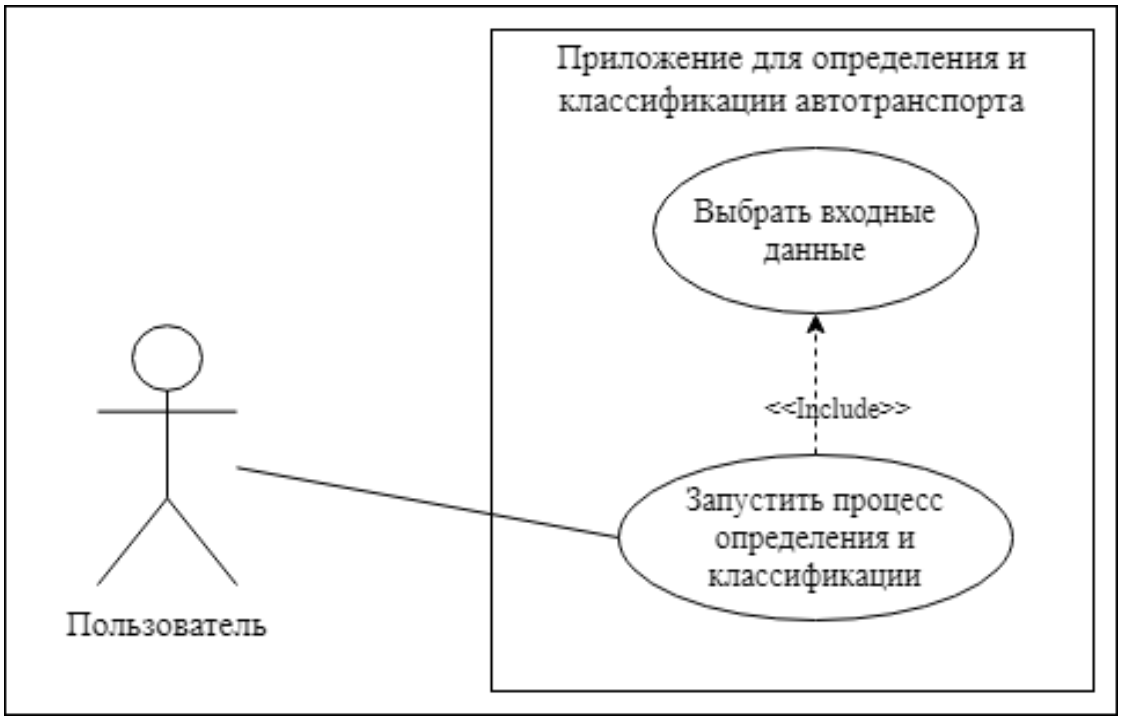

Рис. 14. Диаграмма вариантов использования

Актер *Пользователь* – человек, использующий приложение для определения и классификации различного автотранспорта на видеофайле.

Краткое описание вариантов использования

Пользователь может:

1) выбрать файл в файловой системе компьютера, который будет обрабатываться нейронной сетью;

2) запустить процесс определения и классификации автотранспорта на выбранном видеофайле.

Выводы по третьей главе

В третьей главе с учетом назначения приложения и обзора готовых решений для его реализации, проведенного в первой главе, были определены и сформулированы функциональные и нефункциональные требования, представлена UML диаграмма вариантов использования приложения.

4. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

4.1. Общее описание приложения

В ходе проектирования приложения была разработана диаграмма компонентов, которая показывает разбиение программной системы на структурные компоненты и связи между ними. Диаграмма представлена рис. 15.

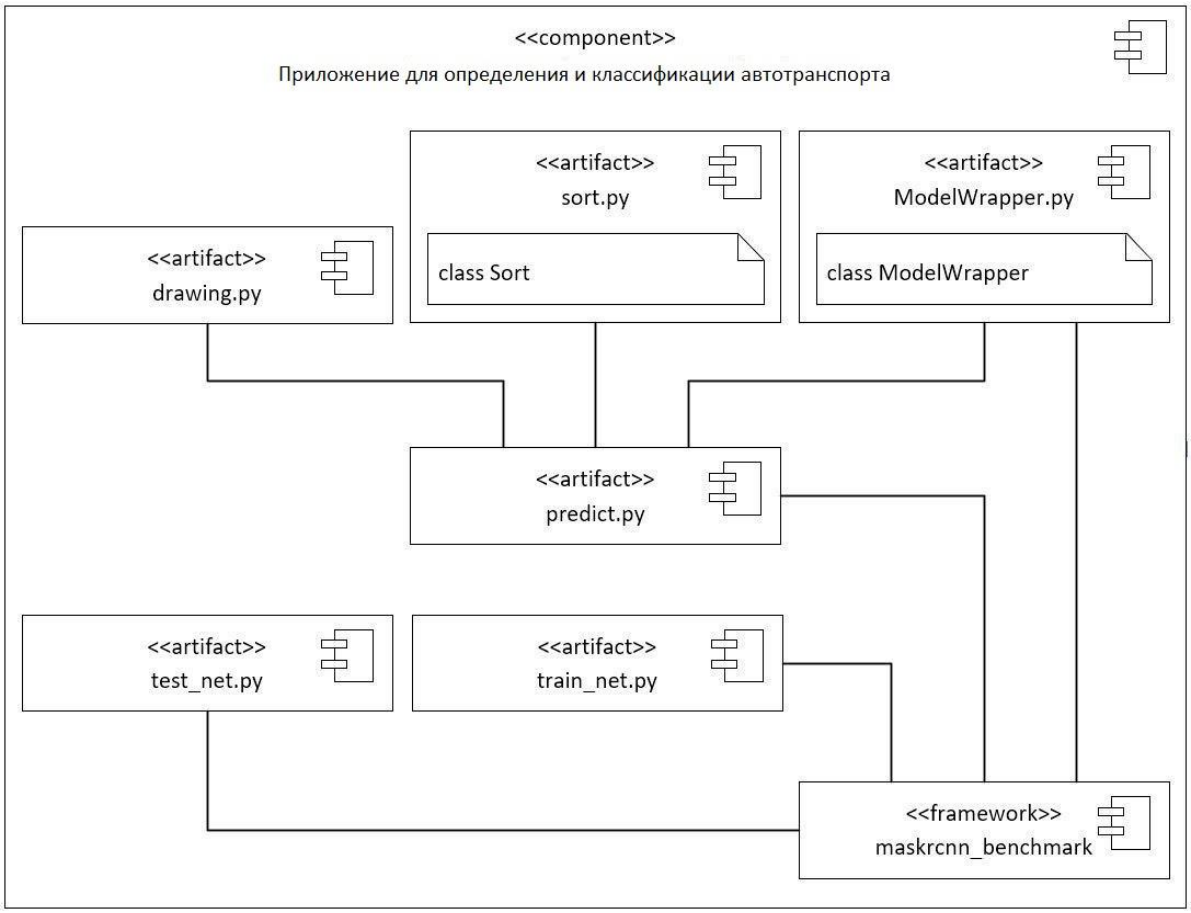

Рис. 15. Диаграмма компонентов

Представленная диаграмма компонентов состоит из следующих артефактов:

1) drawing.py – программный модуль, отвечающий за покадровую отрисовку ограничивающих рамок и цветное обозначение автомобилей;

2) sort.py – реализация алгоритма детекции автотранспорта на кадрах;

3) ModelWrapper.py – программный модуль, содержащий данные для работы модели нейронной сети;

4) predict.py – программный модуль, необходимый для запуска сегментации на видеофайле, выбранном пользователем;

5) test_net.py – программный модуль, отвечающий за тестирование модели нейронной сети;

6) train_net.py – программный модуль, содержащий необходимую информацию для обучения модели нейронной сети;

7) maskrcnn_benchmark – фреймворк, содержащий программную реализацию модели нейронной сети.

Все артефакты, представленные на диаграмме, взаимодействуют с фреймворком maskrcnn_benchmark.

В комментариях помечены классы, представленные в артефактах.

4.2. Диаграмма деятельности

На рис. 16 представлена диаграмма деятельности.

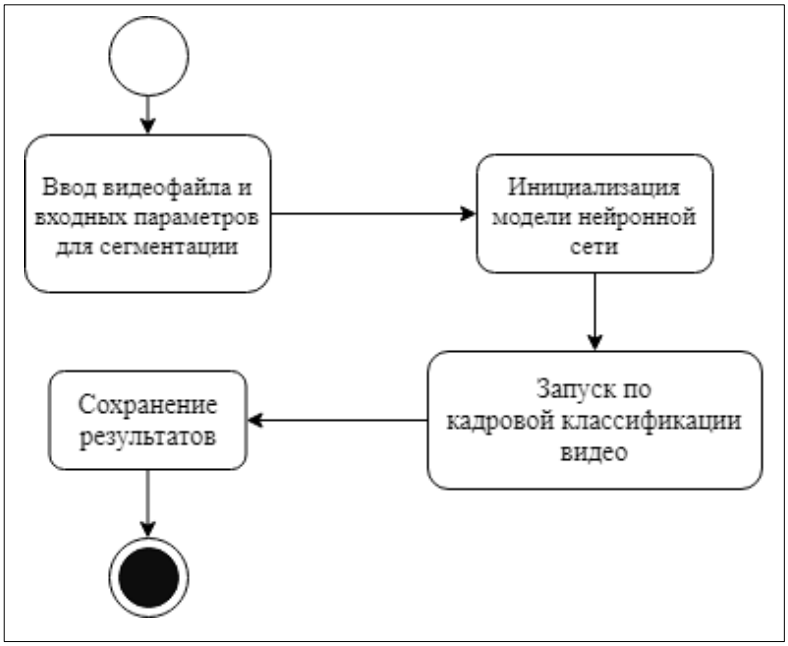

Рис. 16. Диаграмма деятельности

Выводы по четвертой главе

В четвертой главе с учетом определенных ранее функциональных и нефункциональных требований были выделены и спроектированы основные компоненты и классы и представлена диаграмма деятельности.

5. ПРАКТИЧЕСКАЯ ЧАСТЬ

5.1. Топология нейронной сети

За основу реализации приложения была взята нейронная сеть на основе библиотеки искусственного интеллекта с открытым кодом PyTorch 1.0 [27] Facebook Detectron [29]. Данный проект направлен на предоставление необходимых блоков для простого создания моделей обнаружения и сегментации. Реализация алгоритма Mask R-CNN в этом проекте эффективно использует память, а временные затраты примерно вдвое меньше, чем у других проектов на основе PyTorch.

Выбранная архитектура нейронной сети состоит из следующих компонентов (рис. 17):

1) backbone;

2) head.

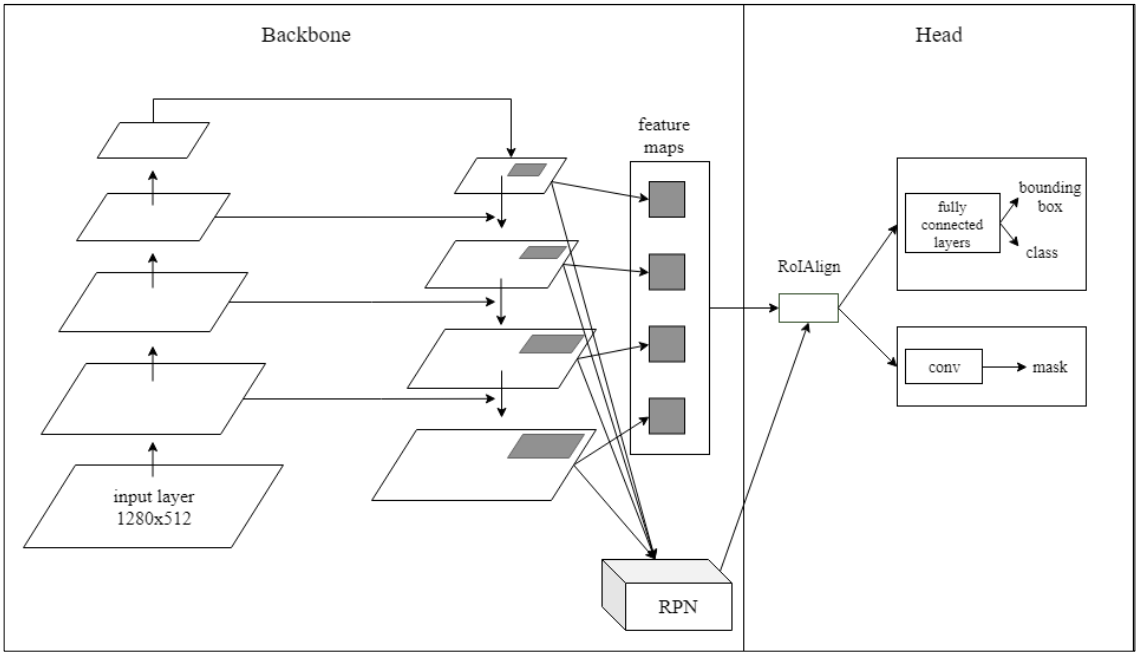

Рис. 17. Топология сети Mask R-CNN

Backbone

Стандартная сверточная нейронная сеть, служащая для формирования пространства признаков. В данном приложении использовалась Residual Network с 50 слоями (ResNet50) [5], топология данной нейронной сети представлена на рис. 18.

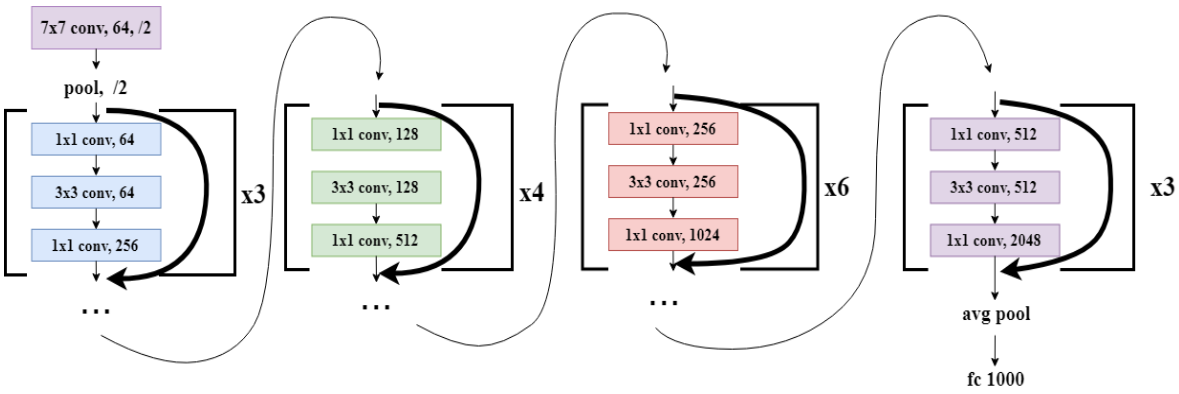

Рис. 18. Топология ResNet50

С увеличением глубины сети точность сначала увеличивается, а затем быстро ухудшается. Для решения этой проблемы в ResNet была введена глубокая «остаточная» структура, которая реализована с помощью соединений для быстрого доступа. На данном рисунке соединения для быстрого доступа изображены полужирными стрелками. Такие соединения пропускают один или несколько слоев и выполняют сопоставление идентификаторов. Их выходы добавляются к выходам предыдущего блока. Данная сеть построена по принципу Feature Pyramid Network (FPN) [7], который подразумевает создание путей снизу вверх (bottom-up pathway), сверху вниз (top-bottom pathway) и горизонтальных связей (lateral connections) между ними. В данном случае путем снизу вверх является ResNet, которая выделяет признаки из исходного изображения, а путь сверху вниз генерирует пирамидальную карту признаков, которая соответствует размерам пути снизу вверх. Горизонтальные соединения позволяют объединять признаки, выделенные на пути снизу вверх, с пирамидальными картами признаков (рис. 19). FPN превосходит другие одиночные сверточные сети преимущественно за счет их способности выделять сильные семантические признаки при различных разрешениях.

Следующим компонентом является Region Proposal Network (RPN), который представляет собой «легковесную» искусственную нейронную сеть, отвечающую за выделение регионов с потенциально возможными объектами. Данная нейронная сеть выделяет регионы на основе метода

скользящего окна. Данные регионы выделяют области на основе карты признаков, созданной предыдущим компонентом.

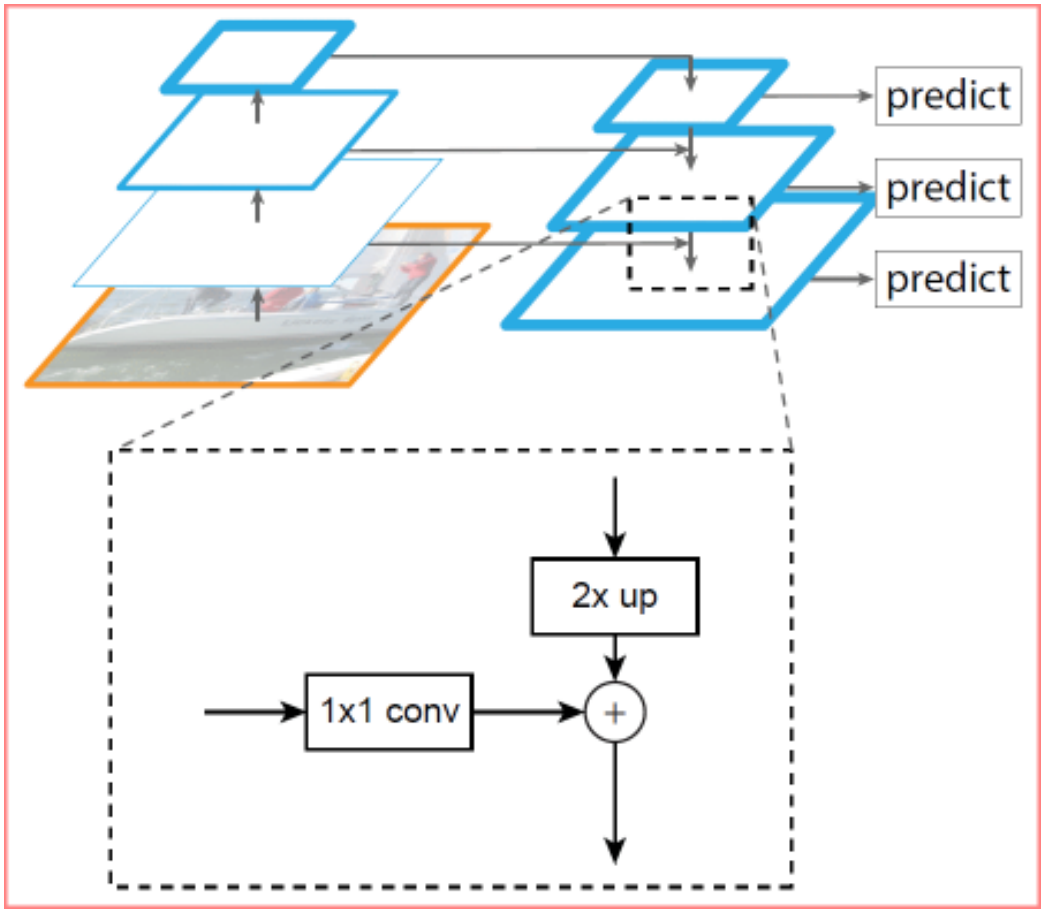

Рис. 19. Принцип построения архитектуры FPN

Head

Этот этап заключается в объединении частей, отвечающих за предсказание ограничивающей рамки, классификацию объекта и определение его маски.

Classification и bounding box – на этом этапе алгоритм принимает регионы, предложенные RPN, в качестве входных данных и выводит классификацию и ограничивающую рамку.

Mask – на этом этапе сверточная нейронная сеть берет области, выбранные классификатором ROI, и генерирует маски для них.

Оптимизация нейронной сети производится на основе функции ошибок (Loss function), которая суммируется по ошибкам на каждом из выходов сети: classification, bounding box, mask:

$$L = L_{cls} + L_{box} + L_{mask}.$$

На входной слой подается изображение, размером 1280x512 пикселей. На выходе – список координат и соответствующих им классов, равный количеству регионов, предложенных RPN, и бинарные маски объектов (если пиксель принадлежит классу маски, то 1, иначе 0).

5.2. Реализация нейронной сети

Для запуска сегментации на видео в форматах mp4 был реализован скрипт predicts.py, который имеет следующие входные параметры (рис. 20). Листинг программного кода обработки видеокадра представлен в приложении.

```
predict.py [-h] --config-file FILE --video-file FILE --save-to FOLDER
               --image-mask FILE
```

Рис. 20. Пример запуска скрипта для сегментации видео

1. config-file - конфигурация используемой нейронной сети.

2. video-file - видео файл в формате mp4 1920x1080 для выделения объектов.

3. save-to - путь до папки, в которую будет сохранен результат.

4. image-mask - маска участка дороги, которая необходима для сегментации.

Маска участка дороги (рис. 21, 22) используется для выделения только тех участков изображения, которые необходимы для корректной работы модели нейронной сети.

Результатом работы скрипта являются покадрово сегментированные изображения.

Для того чтобы создать видеофайл из данных изображений, используется набор свободных библиотек FFMPEG [23], которые позволяют записывать, конвертировать и передавать цифровые аудио- и видеозаписи в различных форматах. Следующая команда (рис. 23) позволит объединить обработанные изображения в видеофайл, где path соответствует значению

save-to в предыдущем скрипте, а out-path/out-name представляют собой путь, куда стоит сохранить видеофайл.

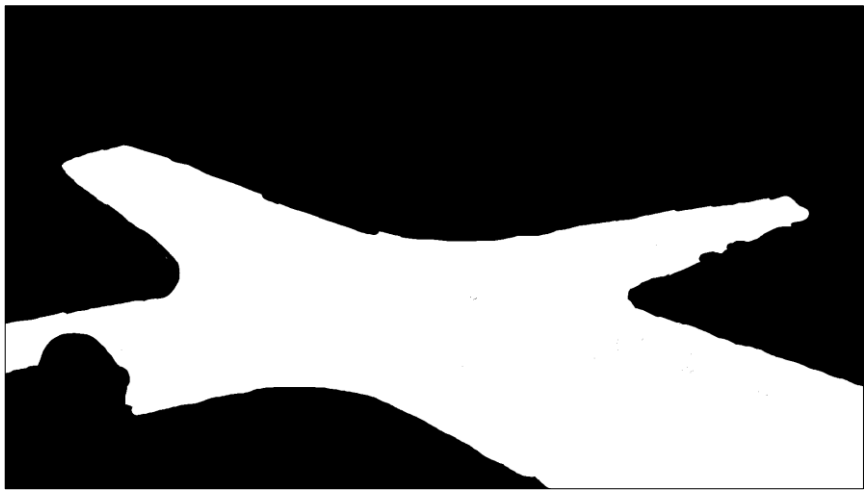

Рис. 21. Маска участка дороги

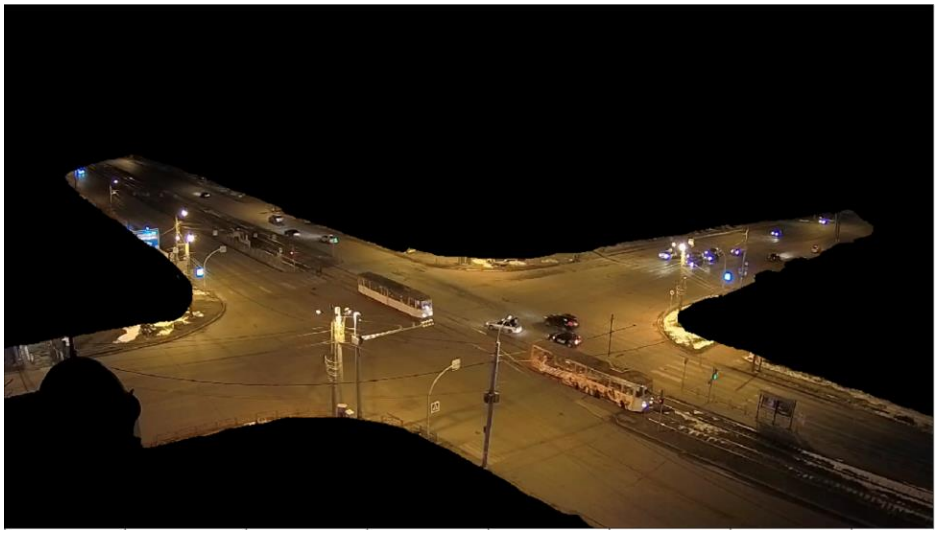

Рис. 22. Вид дороги с наложенной маской

```
ffmpeg -framerate 25 -i <path>/image-%d.png <out-path>/<out-name>.mp4
```

Рис. 23. Команда для создания видеофайла с сегментированными данными

5.3. Формирование обучающей выборки

Формирование множества обучающих данных имеет принципиально большое значение для успешной работы нейронных сетей. Из-за их активного развития проблема формирования обучающей выборки очень актуальна, поскольку во многих задачах глубокие нейронные сети демонстри-

руют качество, существенно превосходящее остальные алгоритмы машинного обучения [31].

Для реализации приложения была использована искусственная нейронная сеть, предобученная на данных из COCO dataset. COCO dataset– это большой набор изображений, предназначенный для обнаружения объектов, сегментации, определения ключевых точек и сегментации. Такой набор данных содержит 330 000 изображений, из них около 25 000 изображений с различным автотранспортом (рис. 24).

Рис. 24. Пример размеченных данных COCO dataset

Данные для обучения и тестирования искусственной нейронной сети были предоставлены ЗАО «Интерсвязь».

Формирование выборки для дальнейшего обучения нейронной сети было выполнено в веб-инструменте для аннотирования изображений

«COCO Annotator» [2], который делает процесс маркировки изображений простым и эффективным (рис. 25).

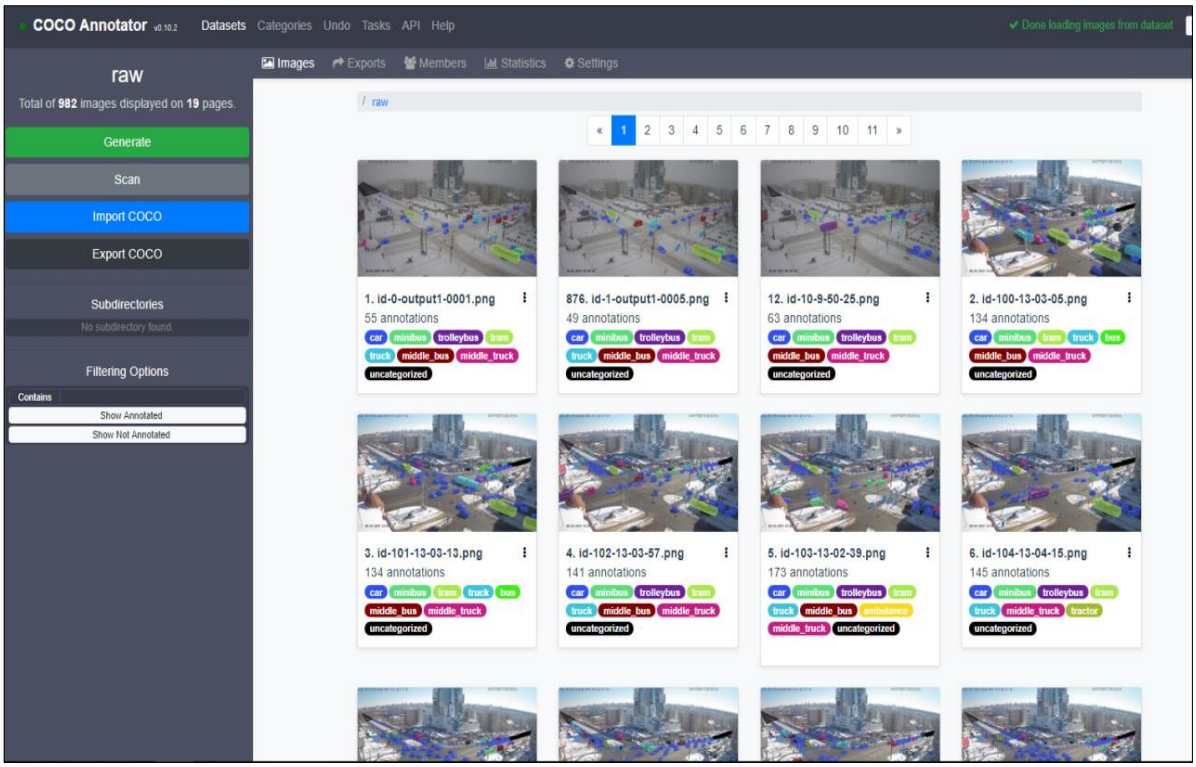

Рис. 25. Интерфейс COCO Annotator

Системы компьютерного зрения стали значительно более эффективными в таких задачах, как обнаружение объектов, их отслеживание и классификация. Многие из этих систем используют контролируемые методы обучения и обучаются на больших наборах аннотированных данных. По мере роста наборов таких данных аннотирование изображений становится все более сложной задачей, и эффективность используемого инструмента аннотирования становится все более важной.

Инструмент «COCO Annotator» предоставляет множество различных функций, таких как возможность маркировки сегмента изображения или отслеживания экземпляров объектов. Еще одной особенностью является возможность аннотирования объекта с визуально отсоединенными частями. Например, если автомобиль частично закрыт другим объектом, его видимые части могут быть аннотированы, как часть одного автомобиля без включения постороннего объекта.

Всего обучающая и тестовая выборка содержат 982 изображения (рис. 26, 27). Весь транспорт был разделен на 13 категорий, перечисленных ниже и представленных на рис. 28.

1. Легковые автомобили.
2. Автобусы.
3. Троллейбусы.
4. Трамваи.
5. Грузовики.
6. Маршрутки-газели.
7. Пазики.
8. Машины скорой помощи.
9. Пожарные машины.
10. Тракторы.
11. Большие фуры.
12. Грузовые газели.
13. Неопределенная категория.

Рис. 26. Пример входных данных для искусственной нейронной сети

Рис. 27. Маски различных видов автотранспорта

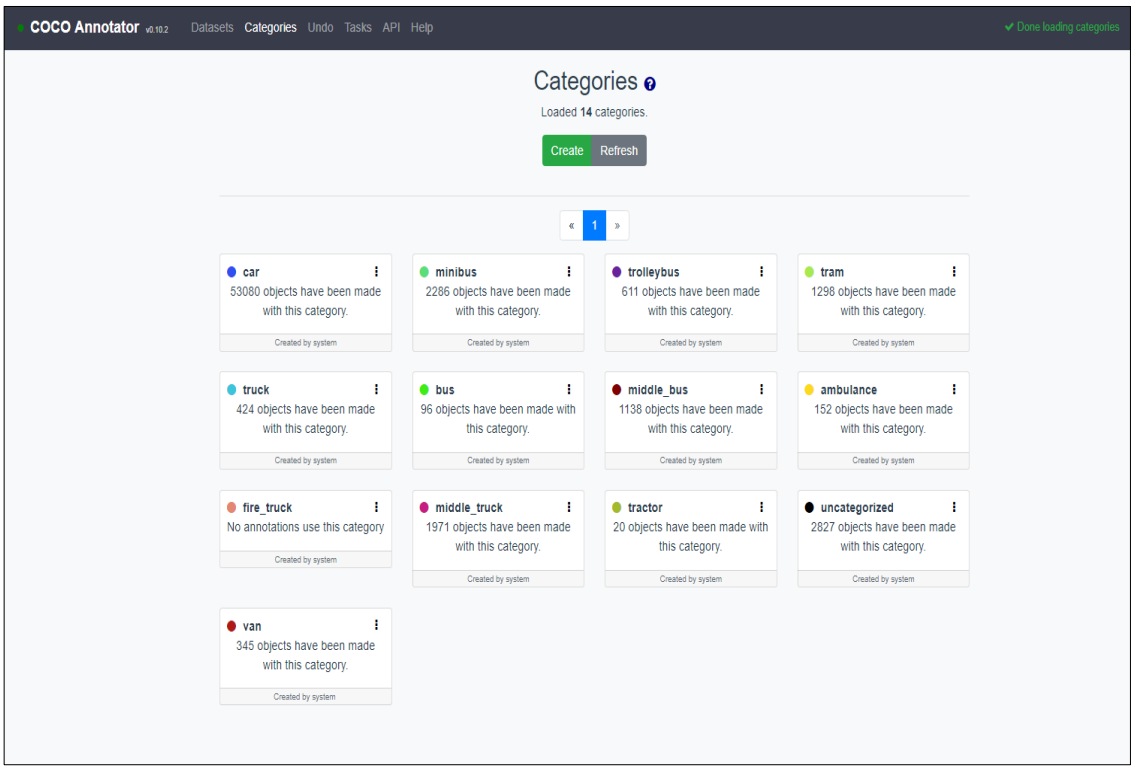

Рис. 28. Категории транспорта в COCO Annotator

5.4. Обучение нейронной сети

Обучение нейронной сети проводилось на компьютере, имеющем следующие характеристики:

1) графический ускоритель Nvidia RTX 2080 Ti (1545 MHz, 11GB GDDR6, 14 Gbps, 4352 CUDA CORE);

2) центральный процессор Intel® Core (TM) i7-8700 CPU (12 ядер @ 3,20 ГГц);

3) 32 ГБ оперативной памяти;

4) ОС Ubuntu 18.04.

Обучающая выборка содержала в себе 80 % от общей выборки, что составило 786 изображений. Обучение длилось 30000 итераций.

Для обучения искусственной нейронной сети был реализован скрипт train_net.py.

Для взаимодействия при обучении искусственной нейронной сети с фреймворком maskrcnn_benchmark необходим COCO-style dataset (рис. 29), что означает определенный способ аннотации изображений в обучающей выборке. Все необходимые данные для нейронной сети такие, как название изображения, его размеры, информация о категории автотранспорта и координаты ограничивающей рамки, содержатся в файле meta.json (рис. 30).

```
annotation{
   "id" : int,
   "image_id" : int,
   "category_id" : int,
   "segmentation" : RLE or [polygon],
   "area" : float,
   "bbox" : [x,y,width,height],
   "iscrowd" : 0 or 1,
}

categories[{
   "id" : int,
   "name" : str,
   "supercategory" : str,
}]
```

Рис. 29. Структура аннотации экземпляра объекта в COCO dataset

Для обучения модели нейронной сети помимо базовых параметров в файле meta.json были указаны данные о цвете маски.

Рис. 30. Содержание файла meta.json

5.5. Вид приложения

Было реализовано консольное приложение. После обработки загруженного в приложение видеофайла нейронной сетью, пользователю предоставляется тот же видеофайл с классифицированными категориями автотранспорта (рис. 31, 32).

Рис. 31. Результат работы алгоритма

Выводы по пятой главе

В соответствии с целью работы была реализована спроектированная топология искусственной нейронной сети, построена репрезентативная обучающая выборка, проведено обучение нейронной сети и было разработано приложение.

Разработанное приложение полностью соответствует предложенным требованиям.

Рис. 32. Классифицированные категории автотранспорта

6. ТЕСТИРОВАНИЕ

6.1. Тестирование нейронной сети

Тестовая выборка использовалась для определения эффективности обученной нейронной сети и содержит в себе 20 % от общей выборки, что равняется 196 изображениям.

Для тестирования искусственной нейронной сети был реализован скрипт test_net.py.

Обученная нейронная сеть показала эффективность 78 %.

6.2. Функциональное тестирование

Функциональное тестирование – это тестирование непосредственно функций и задач продукта [33].

Было проведено функциональное тестирование согласно требованиям, описанным в третьей главе. Результаты тестирования приведены ниже.

Тест № 1. Цель: проверка запуска приложения.

Действие: Приложение должно запускаться на компьютере пользователя.

Ожидаемый результат: На экране пользователя запустится консольное приложение.

Тест пройден? Да.

Тест № 2. Цель: проверка запуска процесса определения и классификации автотранспорта на указанном видеофайле.

Действие: После ввода команды о запуске пользователь нажимает кнопку «Enter» для загрузки видеофайла и запуска алгоритма.

Ожидаемый результат: Видеофайл загрузился в приложение, на экране отобразился процесс обработки изображений.

Тест пройден? Да.

Тест № 3. Цель: проверка возможности просмотра результатов работы приложения.

Действие: После запуска процесса классификации приложение загружает в указанную папку изображения с результатами работы.

Ожидаемый результат: Пользователь видит в указанной папке изображения с классифицированным автотранспортом.

Тест пройден? Да.

Выводы по шестой главе

Было произведено тестирование работы нейронной сети. Были подготовлены тесты для функционального тестирования, после чего было проведено функциональное тестирование реализованного приложения. Все подготовленные тесты были успешно пройдены.

ЗАКЛЮЧЕНИЕ

В соответствии с поставленной целью работы был проведен обзор аналогов приложения для определения и классификации автотранспорта и имеющихся решений для упрощения реализации проекта. Наличие аналогов реализованных по разным технологиям подтвердило актуальность задачи, а большой выбор алгоритмов работы для отслеживания движения и программных продуктов, предназначенных для работы с искусственными нейронными сетями, показал широкий спектр их применения.

Были приведены теоретические сведения об искусственных нейронных сетях и их разновидностях, а также определен этап предобработки видеофайлов для искусственной нейронной сети.

С учетом назначения приложения и обзора решений для его реализации были определены и сформулированы функциональные и нефункциональные требования, представлена UML диаграмма вариантов использования, спроектированы основные компоненты и классы и составлена диаграмма деятельности.

В соответствии с целью работы была составлена репрезентативная обучающая выборка, реализована спроектированная топология искусственной нейронной сети и проведено ее обучение.

Было проведено тестирование работы нейронной сети на тестовой выборке и тестирование приложения в целом. Все подготовленные тесты для функционального тестирования были успешно пройдены.

В рамках дипломного проекта было разработано приложение для определения и классификации автотранспорта на видео с применением нейростетевых технологий. Разработанное приложение полностью соответствует поставленным требованиям.

В ходе разработки приложения были решены следующие задачи:

1) проведен обзор аналогов и научной литературы;

2) подготовлена репрезентативная обучающая выборка;

3) спроектирована топология искусственной нейронной сети и проведено ее обучение;

4) разработано приложение для определения и классификации автотранспорта;

5) проведено тестирование приложения.

ЛИТЕРАТУРА

1. Bewley A. Simple Online and Realtime Tracking. / A. Bewley, Z. Ge, L. Ott, F. Ramos, B. Upcroft. // CoRR, abs/1602.00763, 2016, 5 p.

2. Coco Annotator. [Электронный ресурс] URL: https://github.com/jsbroks/coco-annotator/wiki (дата обращения: 23.04.2019).

3. Goodfellow I., Bengio Y., Courville A. Deep Learning, 2016. – 787 p.

4. Machine Learning. Glossary of Terms, Kluwer Academic Publishers, 1998. – 4 p.

5. He K., Gkioxari G., Doll P. Mask R-CNN. / K. He, G. Gkioxari, P. Doll. // CoRR, abs/1703.06870, 2017. – 12 p.

6. He K., Zhang X., Ren S. Deep Residual Learning for Image Recognition. / K. He, X. Zhang, S. Ren. // CoRR, abs/1512.03385, 2015, 12 p.

7. LeCun Y., Bottou L., Bengio Y. Gradient-Based Learning Applied to Document Recognition. / Y. LeCun, L. Bottou, Y. Bengio. // Proc. of the IEEE, November, 1998. – 46 p.

8. Lin T.–Y., Dollar P., Girshick R. Feature Pyramid Networks for Object Detection. / T.–Y. Lin, P. Dollar, R. Girshick. // CoRR, abs/1612.03144, 2016, 10 p.

9. Machine Learning. Glossary of Terms. [Электронный ресурс] URL: http://ai.stanford.edu/~ronnyk/glossary.html (дата обращения: 06.01.2019).

10. Number of vehicles in use worldwide 2006-2015. [Электронный ресурс] URL: https://www.statista.com/statistics/281134/number-of-vehicles-in-use-worldwide/ (дата обращения: 26.12.2018).

11. PureActiv, Vehicle Counting System. [Электронный ресурс] URL: https://www.puretechsystems.com/solutions-car-counting.html (дата обращения: 26.12.2018).

12. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement, 2018. – 6 p.

13. Ren S., He K., Girshick R.B. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. / S. Ren, K. He, R.B. Girshick. // CoRR, abs/1506.01497, 2015. – 14 p.

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

15. Vehicle Detection. [Электронный ресурс] URL: https://github.com/ahmetozlu/vehicle_counting_hog_svm (дата обращения: 26.12.2018).

16. Vehicle Detection, Tracking and Counting. [Электронный ресурс] URL: https://github.com/andrewssobral/simple_vehicle_counting (дата обращения: 26.12.2018).

17. VIERO, Vehicle Counting System. [Электронный ресурс] URL: http://www.issd.com.tr/en/22987/Vehicle-Counting-System-VIERO (дата обращения: 26.12.2018).

18. Zhang F., Li C., Yang F. Vehicle Detection in Urban Traffic Surveillance Images Based on Convolutional Neural Networks with Feature Concatenation, 2019.– 21 p.

19. Андросов К.Ю., Горбунов А.Н. Применение метода «Сдвига среднего» (Mean shift) для обработки металлографических изображений // Вестник образовательного консорциума среднерусский университет. Информационные технологии. №2 (6), 2015. 14 – 16 с.

20. Гончаров И.В., Потешкин А.С., Курулев А.П. Архитектура современной нейронной сети для сегментации объектов на изображении. Белорусский государственный университет информатики и радиоэлектроники, 2018. – 2 с.

21. Гуськова А.М. Математическое моделирование систем распознавания изображений, содержащих текстовую информацию, на основе нейронных сетей. // Молодой ученый. – 2015. – № 18. – С. 7–10. – URL: https://moluch.ru/archive/98/21912/ (дата обращения: 23.01.2019).

22. Документация Boost. [Электронный ресурс] URL: https://www.boost.org/ (дата обращения: 26.12.2018).

23. Документация cvBlob. [Электронный ресурс] URL: https://code.google.com/archive/p/cvblob/ (дата обращения: 26.12.2018).

24. Документация FFmpeg. [Электронный ресурс] URL: https://ffmpeg.org (дата обращения: 06.05.2019).

25. Документация Keras. [Электронный ресурс] URL: http://keras.io (дата обращения: 27.12.2018).

26. Документация Meanshift и Camshift. [Электронный ресурс] URL: https://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html (дата обращения: 27.12.2018).

27. Документация OpenCV. [Электронный ресурс] URL: https://opencv.org (дата обращения: 26.12.2018).

28. Документация PyTorch. [Электронный ресурс] URL: https://pytorch.org (дата обращения: 13.04.2019).

29. Документация TensorFlow. [Электронный ресурс] URL: https://www.tensorflow.org/api_docs/python/ (дата обращения: 27.12.2018).

30. Документация нейронной сети от Facebook. [Электронный ресурс] URL: https://github.com/facebookresearch/maskrcnn-benchmark (дата обращения: 13.04.2019).

31. Ивлев В.Ю., Титова А.А. Расчет интенсивности транспортного и пешеходного потоков. // Научное сообщество студентов XXI столетия. ТЕХНИЧЕСКИЕ НАУКИ: сб. ст. по мат. XXVII междунар. студ. науч.-практ. конф. № 12(26). [Электронный ресурс] URL: http://sibac.info/archive/technic/12(26).pdf (дата обращения: 26.12.2018).

32. Кафтанников И.Л., Парасич А.В. Проблемы формирования обучающей выборки в задачах машинного обучения. // Вестник ЮУрГУ, 2016. – 10 с.

33. Кореньков А.Н., Моногаров К.Е. Регистрация трехмерных облаков точек на основе метода максимального правдоподобия .//

Центральный научно-исследовательский и опытно-конструкторский институт робототехники и технической кибернетики. Робототехника и техническая кибернетика. №1 (6), 2015. 46 – 52 с.

34. Криспин Л. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд. – М. : Вильямс, 2010. – 464 с.

35. Макконнелл С. Совершенный код. – М.: Русская редакция, 2007. – 896 с.

36. Николашин А.А., Шубин М.А. Генетические алгоритмы и искусственные нейронные сети в интернете // Международная конференция по мягким вычислениям и измерениям. – Т.1, 2015. 400 – 403 с.

37. Романов В.П. Интеллектуальные информационные системы в экономике: Учебное пособие. / В.П. Романов; под ред. д.э.н., проф. Н.П. Тихомирова. – М.: Экзамен, 2003. – 496 с.

38. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество / Хабр. [Электронный ресурс] URL: https://habr.com/post/348000/ (дата обращения: 30.01.2019).

39. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика, 1992. – 184 с.

40. Хайкин С. Нейронные сети: полный курс, 2-е издание, 2006. – 1103 с.

41. Химонин Ю. Сбор и анализ требований к программному продукту. [Электронный ресурс] URL: https://pmi.ru/profes/Software_Requirements_Khimonin.pdf (дата обращения: 01.02.2019).

42. Шапиро Л., Стокман Дж. Компьютерное зрение, – БИНОМ. Лаборатория знаний, 2013. – 761 с.

ПРИЛОЖЕНИЕ

```
while stream.isOpened():
    ret, frame = stream.read()
    H, W = frame.shape[:2]
    det_frame = frame[min_y:max_y, min_x:max_x] * image_mask
    detections = model.predict([det_frame], project_mask=False)[0]
    detections = detec-
tions.convert('xyxy').clip_to_image(remove_empty=True)
    detections.size = (W, H)
    detections.bbox.add_(torch.tensor([min_x, min_y, min_x, min_y],
dtype=torch.float32))
    detections = tracker.update(detections)


    if detections.get_field('mask').dim() != 4:
        detections.get_field('mask').data = detec-
tions.get_field('mask').unsqueeze(1)


    detections = detections.resize((W,
H)).convert('xyxy').clip_to_image(remove_empty=True)
    detections = model.project_masks(detections)
    colors = [colors_generator[int(ind)] for ind in detec-
tions.get_field("index")]
    ax = plt.gca()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    frame = frame[:, :, [2, 1, 0]]
    display_instances(frame, detections, CLASS_NAMES, ax, colors=colors)
```

Рис. Листинг программного кода обработки видеокадра