МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка системы компьютерного зрения для анализа автомобильного трафика»

Обучающийся В.И. Романов
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., В.С. Климов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Е.В. Косс
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Тема выпускной квалификационной работы: Разработка системы «умный светофор» с помощью компьютерного зрения для анализа автомобильного трафика.

Целью выпускной квалификационной работы (ВКР) является создание программного обеспечения (ПО) для технологии «умный светофор», использующей компьютерное зрение для анализа автомобильного трафика с получаемого изображения.

Объектом ВКР является распознавание объектов на изображении и их анализ.

Предметом ВКР является алгоритм распознавания автомобильного трафика на изображении с использованием методов компьютерного зрения.

Задачи работы:

- изучить существующие методы анализа трафика;
- изучить принцип работы нейронных сетей;
- разработать систему, выполняющая практическую задачу.

Структура ВКР состоит из введения, трех глав, заключения, списка литературы.

Во введении формулируется цель, ставятся задачи и описывается актуальность исследуемой области и практической задачи.

В первой главе рассматриваются методы анализа трафика, алгоритм построения нейронной сети, обзор действующих программ анализа.

Во второй главе разрабатываются основные программные модули, проектируется макет системы, реализуется программное решение.

В третьей главе проводится тестирование полученного программного обеспечения.

В работе использовано 41 рисунок, 25 источников используемой литературы. Общий объем составляет 57 страниц.

ABSTRACT

This graduation project is about the development of a "smart traffic light" system using computer vision for analyzing car traffic.

The graduation project consists of an explanatory note on 57 pages, introduction, including 41 figures, the list of 25 references including foreign sources.

The key issue of the graduation project is the possibility of creating software for the "smart traffic light" technology.

The aim of the work is to give some information on design work to provide traffic analysis for the "smart traffic light" technology using neural networks.

The object of the graduation project is the recognition of objects in the image and their analysis.

The subject of the graduation project is an algorithm for recognizing car traffic in an image using computer vision methods.

The graduation project may be divided into several logically connected parts which are the study of analysis systems, examples of the use of analyzing programs, justification of the need for traffic light regulation, technological and design solutions.

In conclusion we'd like to stress that this work is relevant isn't only for large cities where there is a high intensity of transport, but for small towns. It has such advantages as improving environmental safety and increasing the productivity of the population.

Содержание

Введение

Современный этап развития человечества характеризуется экспоненциальным ростом количества накопленной информации. Проблема объединения информации и их анализ очень важна в наше время, так как позволяет контролировать и направлять многие действия человека. А некоторые анализирующие системы позволяют прогнозировать возможные данные в будущем времени, и это дает большую вероятность определиться с дальнейшими действиями. Так с каждым годом добавляются все новые задачи для анализа, на которые уже не способен человек просто просматривая данные и делая выводы.

Во многих междисциплинарных исследованиях также встречается проблема анализа данных. Реализация методов и систем для анализа привлекает сегодня большое внимание. Анализ различных данных является проблемой в современной науке, так как невозможно выбрать один, подходящий ко всем случаям, метод.

В частности, анализ лежит в основе комплексных решений систем искусственного интеллекта. Автоматизация анализа данных особенно актуальна в областях жизни человека, где имеется большой поток поступающей информации, для этого уже сегодня часто используют системы на основе машинного обучения**.**

Целью выпускной квалификационной работы (ВКР) является разработка программного обеспечения для технологии «умный светофор» на основе компьютерного зрения для анализа трафика с камеры, установленной на светофор.

Актуальность данной программы заключается в возможности динамически изменять пропускную способность определенного светофора.

Объектом ВКР является распознавание объектов на изображении и их анализ.

Предметом ВКР является алгоритм распознавания автомобильного трафика на изображении с использованием методов компьютерного зрения.

Основные задачи ВКР**:**

- изучить существующие методы анализа трафика;
- изучить принцип работы нейронных сетей;

разработать алгоритм, решающий практико-ориентированную задачу и протестировать разработанное решение.

1 Методы анализа данных и алгоритм работы нейронных сетей

1.1 Задачи и методы анализа данных

Задачи анализа данных можно разделить на:

- задачи классификации: каждая переменная может принадлежать какому-то множеству в объеме данных, которому соответствует свой класс, то есть отделенная общность признаков;
- задачи регрессии подобны классификации. В ходе исследования (решения задачи) находиться форма зависимости переменных друг от друга;
- задачи прогнозирования: в том, что система выявляет новые значения на основе последовательностей полученных данных, в том числе ищет и выстраивает взаимосвязи;
- задачи кластеризации: похожа на классификацию, но отличается, тем что кластер создается входе исследования, в то же время класс создан заранее;
- анализ отклонений сообщает о событиях, которые несвойственны данному объему информации.

По состоянию на 2017 год самым большим источником открытых данных по Российской Федерации является «Портал открытых данных Российской Федерации», содержащий более 5,1 тыс. наборов данных. Принцип открытости получил отдельное название – «открытые данные» (Open Data). Считается, что предоставление свободного доступа к отдельным данным может способствовать повышению качества государственного, регионального и муниципального управления.

Анализ данных включает три основных этапа:

- сбор данных;
- подготовка данных;

- обработка данных.

Сбор данных – процесс формирования структурированного набора данных в цифровой форме. В некоторых случаях процесс сбора данных может включать также этап оцифровки. Многие инструменты на компьютере могут применять реализацию автоматизированного сбора данных, если значения приведены структурировано для использования в этих системах.

Для использования в системах анализа данные должны быть представлены в определенном виде. Однако возможны случаи когда набор данных имеет недостатки, к частым можно отнести, пропуски данных, некорректные значения данных при записи, текстовые данные в записях числовых. Перечисленные особенности могут либо привести к затруднениям в процессе дальнейшей обработки данных, либо сделать её невозможной. Для устранения отмеченных несоответствий могут быть применены следующие операции:

- структурирование – определения нужного исходного табличного вида и приведения всех значений к нему;
- отбор – отбрасывания данных, которые способны затруднить работу ;
- нормализация – представление значений в каком-то заданном диапазоне;
- кодирование – перевод значений в числовой вид, состоящий из «0» и «1» . Кодирование можно разбить на две крупных типа, бинарное кодирование: когда классы можно пронумеровать числами «0» и «1»; или множественное кодирование: когда каждый класс может быть записан в двоичной системе, как пример есть 8 классов и каждый кодирована, тогда третий класс будет закодирован как «010», а восьмой «111».

Рассмотрим алгоритмы анализа информации, применяемых в информационных и математических системах Первый тип это регрессионный анализ.

Предположим, что нам нужно узнать стоимость предмета за короткий срок в будущем. Сразу становиться понятно, что предмет обладает какими-то характеристиками, влияющими на стоимость, например использованные материалы и затраченное время. При обнаружение влияния текущих значений на предмет, можно сказать о создание ассоциативной зависимости, по которой можно предугадать дальнейшие возможные данные. Прогнозирование некоторого значения зависимой переменной с помощью независимых переменных является задачей регрессионного анализа.

Регрессионный анализ относится к моделям задач, где идет обучения с учителем, регрессионный анализ нуждается в первоначальных значения, так как не способен строить ассоциации без значений, следовательно, анализу нужен тренировочный материал, который и называется учителем. Регрессионный анализ, получая значения на вход, строит ассоциации между данными, это первый этап, следом идеи этап предсказания, где по полученным ассоциациям делается вывод о возможных поступлениях новых данных. Также в регрессионном анализе есть этап корректировки, это когда идет перестройка ассоциаций данных, если прогноз не сбылся.

Следующим типом является классификация. Классификация – это процесс определения принадлежности объектов к определенным множествам (классам). Так как данный тип анализа требует задачи первоначальных классов, то обучение данного типа тоже происходит с учителем. Предполагается, что имеется некоторые выбранные данные, в которых представлены признаки нескольких классов, которые помогут анализатору причислить объект к определенному классу.

Применение классификации производится в два этапа. Первый отвечает за обучения анализатора к распределению на классы, приписывает объектам признаки класса и их идентификаторы, а второй этап – работа анализатора уже на проверочных данных. Проверочные данные отличаются от тестовых, так как на тестовых данных кроме значения падают класс

значения, что при сравнение покажет, есть ли ошибка обучения или нет, а проверочный содержит только значение, которое надо внести в анализатор.

Также классификатор может давать оценку своим действиям по анализу классов, для этого правильно классифицированные объекты он помечает, как «true positives» или «true negatives», если первый говорит о правильности определения класса, то второй говорит, что классификатор сомневается в правильности, но тренировочные данных подтверждают данный класс. Такие же группы и у не правильно классифицированных данных, они называются «false positives» и «false negatives», первый указывает, на то что класс не выбран правильно для объекта, а второй говорит, что система не уверена, что объект не относится к классу, при этом поданные тренировочные данные утверждают, что объект не относится.

Существует несколько методов оценки качества классификации. Одним из методов является оценка с помощью F-критерия, которая и будет использоваться в тестирование программы.

Как было упомянуто, различают бинарную и множественную классификацию. Бинарная классификация предполагает наличие двух классов, множественная – трех и более классов. Задачей бинарной классификации является определение принадлежности некоего объекта к одному из двух возможных классов. Задачей множественной классификации является определение принадлежности некоего объекта к одному из нескольких (трех или более) возможных классов, например постановка диагноза пациенту. Наиболее известными методами множественной классификации являются:

- метод «один против всех» - присвоение класса объекту, после того как он не подошел ни одному из проверяемых тренировочными данными классов;
- нейронная сеть.

Именно нейронные сети и их возможности к множественной классификации будут рассматривать ниже.

Были рассмотрены задачи анализа данных, его возможности и проблемы, также был рассмотрены более подробно методы анализа данных, как регрессия и классификация.

1.2 Сверточные нейронные сети

Искусственная нейронная сеть – названа так потому, что в математической модели работает по биологическому принципу головного мозга, то есть идет возбуждение определенных слоев нейронов при выполнение определенных действий.

Ключевым принципом, который позволяет отнести нейронные сети к биологическим объектам, является возможность обучения на собственном примере. Каждая задача, решаемая нейронной сетью, требует своей конфигурации, так как распознание объектов приводит к проявлению разнообразных признаков, на которых и должна работать нейронная сеть. При этом создание большой и точной нейронной сети в принципе невозможно, так как на каждом измененном классе требуется все больше мощностей системы. На рисунке 1 представлена самая малая часть нейронной сети – искусственный нейрон.

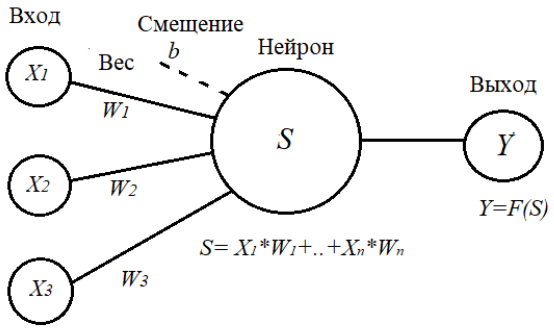

Рисунок 1 – Искусственный нейрон

Вес и входные данные проходят несколько итераций. В ходе прохождения входные данные перемножаются на вес для нейрона, в который проходит путь. Задача нейрона – просуммировать произведение входов на веса ото всех нейронов предыдущего слоя, то есть использует формула: $\sum_{i=1}^{n} W_i * X_i$. После просуммированные данные направляются на функцию преобразователь, в которой использует функцию активации, по результатам которой преобразуется число [1].

Когда в структурируемых данных возникают такие значения, которые могут сломать конструкцию, приходится прибегать к методам структурирования данных. Также работает и нейронная сеть. Чтобы получить правильный ответ нейронная сеть должна структурировать данные. Этим занимается функция нормализации данных или функция активации нейрона. Она определяет выходное значение нейрона, которое будет зависеть от суммарного значения входов и порогового значения [15]. Она проверяет произведенное нейроном значение выхода на предмет того, что должны ли внешние связи рассматривать этот нейрон как активированный, или его можно игнорировать [23]. Существуют несколько видов функции активации.

На рисунке 2 изображена ступенчатая функция активации. Для данного вида функции создается некий порог и когда значение выхода превышает некоторое порогового значение, считаем нейрон активированным. В противном случае говорим, что нейрон неактивен.

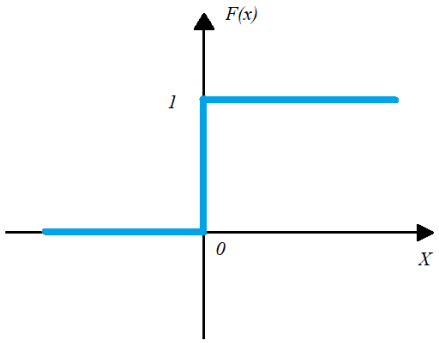

Рисунок 2 – Ступенчатая функция активации

К сожалению, данная функция неудобна, если хотим получить промежуточные значения, допустим надо получить значение больше чем на два класса, тогда можно рассмотреть линейную функцию активации. На рисунке 3 изображена линейная функция активации. Линейная функция представляет собой прямую линию и пропорциональна входу, то есть взвешенной сумме на этом нейроне. Такой выбор активационной функции позволяет получать спектр значений, а не только бинарный ответ.

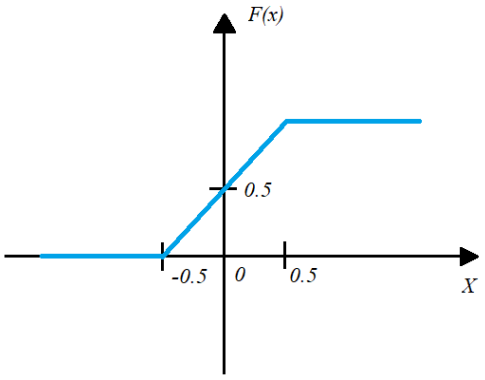

Рисунок 3 – Линейная функция активации

Следующая функция активации изображена на рисунке 4 – сигмоидальная функция. Сигмойда очень похожа на линейную функцию, но из-за того, что к своим краям имеет сглаживание предоставляет несколько плюсов.

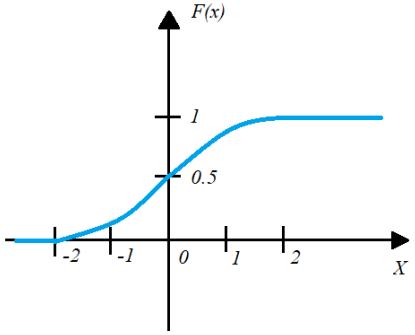

Рисунок 4 – Сигмоидальная функция активации

Преимущество сигмоиды нелинейность и небинарность по принципу построения. Для сигмоиды также характерен гладкий градиент. То есть в диапазоне значений *X* от -1 до 1 значения Y меняется значительно. Это означает, что любое малое изменение значения *X* в этой области влечет существенное изменение значения *Y*. Такое поведение функции указывает на то, что Y имеет тенденцию прижиматься к одному из краев кривой [5]. Сегодня сигмоида является одной из самых частых активационных функций в нейронных сетях.

Ещё одной функцией активации является изображенная на рисунке 5 функция усеченного линейного преобразования (ReLu), если использовать данное изоюражение, становится понятно, что ReLu возвращает значение *X*, если *X* положительно, и 0 в противном случае.

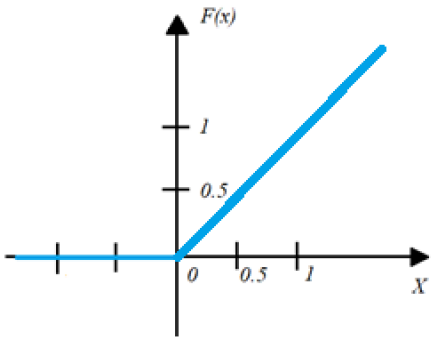

Рисунок 5 – Функция усеченного линейного преобразования (ReLu)

Функция усеченного линейного преобразования (ReLu) обладает рядом преимуществ:

- производная данной функции будет не линейна;
- нет большой нагрузки на систему;
- не происходит затухание или разрастание градиентов на конца или на местах сближение .

Искусственные нейронные сети имеет множество преимуществ, перед другими способами классификации, но главным является обучаемость. Основная цель обучение найти ошибку и устранить, но иногда такое не

может произойти по определению, тогда цель меняется на уменьшение ошибки до малого значения, на котором ошибка уже не влияет на исход решений нейронной сети.

Чтобы найти и откорректировать ошибку используют множество методов, но большинство уже готовых моделей нейронных сетей и более простое создание обеспечивает прямой метод распространения. Данный метод представляет процесс получения выходных данных нейронной сети на основе одного экземпляра тренировочных данных [10]. В данном методе ключевое значение имеет ошибка, если она будет устранена, минимизирована, так чтобы не влиять на действия нейронной сети, или станет меньше порогового значения, тогда можно сказать что нейронная сеть закончила своё обучение. Чтобы найти ошибку используют формулу (1):

$$Err = y' - y, \tag{1}$$

где y'– ожидаемое значение;

y – результат работы нейронной сети.

Существующие методы обучения нейронных сетей можно разделить на стохастические и детерминированные методы. Стохастические методы основаны на корректировании весовых коэффициентов сети случайно и сохраняют внесенные изменения, только при условии того что изменения приведет к снижению ошибки. Детерминированные методы работают более глобально, если в первом случаи происходит случайный отбор и изменения на каждом нейроне последовательно, то тут пытаются точно определять какие весовые коэффициенты при корректировки приведут к уменьшению ошибки [3].

Для обучения нейронных сетей в основном используется один из детерминированных методов, называемый градиентным спуском. Данный метод оптимизации позволяет минимизировать функцию потерь нейронной сети путем итеративного корректирования весовых коэффициентов [7]. На рисунке 6 изображен графический образ принципа градиентного спуска.

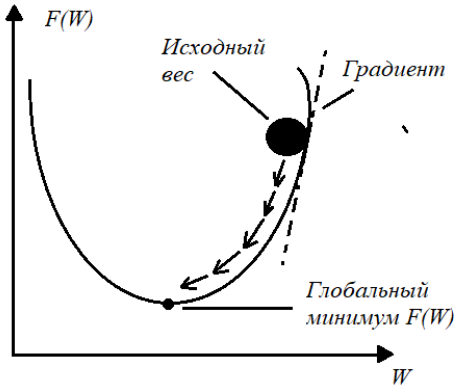

Рисунок 6 – Принцип градиентного спуска

Обучение по алгоритму градиентного спуска представляет из себя поиск градиента функции F в конкретной точке, являющейся текущей ошибкой нейронной сети. Данный градиент будет указывать направление возрастания функции, получается, чтобы минимизировать ошибку, необходимо сдвигать значение весового коэффициента в сторону противоположную направлению градиента. Корректирование весового коэффициента по методу градиентного спуска представлено в формуле(2)(3):

$$\omega := \omega + \Delta\omega, \tag{2}$$

$$\Delta\omega = -\mu\nabla F(\omega), \tag{3}$$

где $\Delta\omega$ – новое значение весового коэффициента;

ω – текущее значение весового коэффициента;

$\nabla F(\omega)$ – градиент ошибки;

μ – константа, характеризующая скорость обучения.

Для того чтобы минимизировать функцию потерь такой сети, необходимо найти градиенты для всех весовых коэффициентов нейронной сети, то есть рассчитывать их частные производные на каждом нейроне, и внести поправку на каждом весовом коэффициенте [18]. Метод, который позволит нам это сделать, называется обратным распространением ошибки. Принцип данного метода заключается в проходе в обратном направление от прямого распространения и нахождения дифферециалов по каждому

весовоуму коэффиценту, чтобы можно было определить влияние на нейрон предудущего слоя и вывод ошибки на нём.

Ошибка весовых коэффициентов последнего и всех остальных слоев вычисляется различными формулами [2]. Сначала, из-за того что мы идем в обратном порядке, находиться вклад весовых коэффициентов последнего слоя в общую ошибку по формуле (4).

$$\frac{\partial \mathrm{E}}{\partial \mathrm{w_{ji}}} = \mathrm{O_j} \times \frac{\partial \mathrm{F_i\,(S_i)}}{\partial \mathrm{S_i}} \times \delta_\mathrm{j}, \tag{4}$$

где $\frac{\partial \mathrm{E}}{\partial \mathrm{w_{ji}}}$ – вклад веса w_{ji} в общую ошибку;

$\mathrm{O_j}$ – выходной сигнал нейрона j;

$\frac{\partial \mathrm{F_i\,(S_i)}}{\partial \mathrm{S_i}}$ – производная функции активации нейрона j;

δ_j – ошибка выходного сигнала нейрона j, которую нашли по формулам (2) и (3).

Следующим этапом становиться нахождение вклада весовых коэффициентов всех слоев, кроме последнего в общую ошибку считается по формуле(5):

$$\frac{\partial \mathrm{E}}{\partial \mathrm{w_{ji}}} = \frac{\partial K_i}{\partial \mathrm{w_{ji}}} \times \frac{\partial O_i}{\partial K_i} \times \sum \frac{\delta_z}{\partial O_i}, \tag{5}$$

где $\frac{\partial \mathrm{E}}{\partial \mathrm{w_{ji}}}$ – вклад веса w_{ji} в общую ошибку;

$\frac{\partial K_i}{\partial \mathrm{w_{ji}}}$ – вклад веса $\mathrm{w_{ji}}$ во взвешенную сумму нейрона I;

$\frac{\partial O_i}{\partial K_i}$ – вклад взвешенной суммы в выходной сигнал нейрона i;

$\frac{\delta_z}{\partial O_i}$ – вклад выходного сигнала нейрона i в ошибку выходного сигнала нейрона z.

Обучение нейронной сети градиентным спуском происходит в цикличном прохождение нейронной сети в обратном порядке с изменением

весовых коэффициентов [17]. Одна итерация прохождения корректировки весов имеет несколько этапов:

– этап прямого распространения ошибки и нахождения ошибки набора тренировочных данных с помощью функции потерь;

– этап нахождения квадрата всех ошибок после подсчета функции потерь на всех тренировочных данных;

– этап корректировки весовых коэффициентов, используя метод обратного распространения.

В принципе система нейронной сети уже построена, но стоит рассмотреть несколько недостатков нашей сети.

Недостатком градиентного спуска является то, что проход по всем весовым коэффициентам может быть долгим, если задан большой тренировочный набор данных, что приводит к резкому ухудшению скорости обучения или ухудшению качества устранения ошибки на одной итерации. Данную проблему решает использование модифицированного алгоритма градиентного спуска [19]. В основе различия этих алгоритмов, можно видеть прохождения итерации корректировки данных, то есть обычный проходит весь список тренировочного материала, так как модифицированный разбивает на небольшие партии. Из-за того, что корректирование весовых коэффициентов на основе небольшой партии данных не может выдать особенно точные значения на всех этапах, функция потерь минимизируется не плавно, а рывками, то есть могу быть более не точные результаты на отдельной партии, зато к концу обучения происходит сходимость к локальному минимуму быстрее. На рисунке 7 изображено графическое отображения нахождения локального минимума на классическом методе градиентного спуска, а на рисунке 8 изображен модифицированный метод.

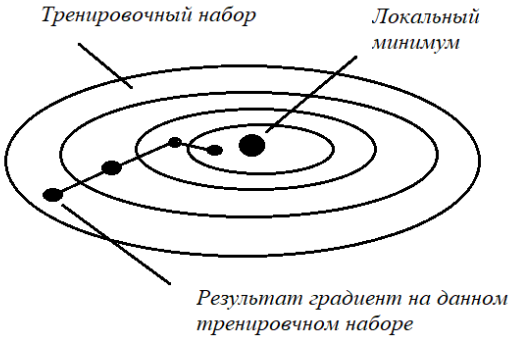

Рисунок 7 – Метод обучения градиентным спуском

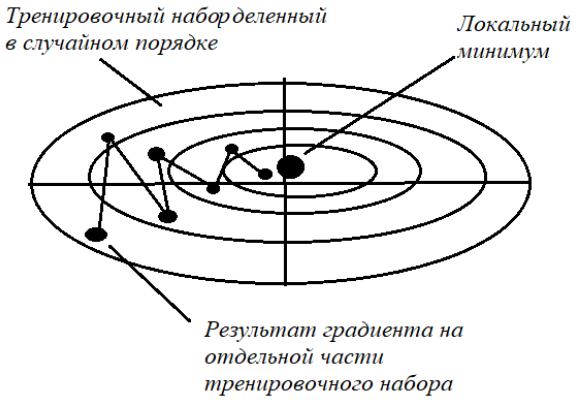

Рисунок 8 – Метод обучения модифицированным градиентом

Сверточная нейронная сеть является одной из наиболее популярных архитектур нейронных сетей, применяемая для анализа данных или изображения. В целом, архитектура сверточной нейронной сети позволяет выделять в данных различные признаки, начиная от очень простых и заканчивая более сложными [22].

Свёрточная нейронная сеть состоит из чередующихся слоев свертки и подвыборки, после которых следуют полносвязные слои. Сверточные нейронные сети имеют разное количество слоев и часто используют различные функции активации на разных слоях [22]. На рисунке 9 изображена графическое представление архитектуры строения сверточной сети.

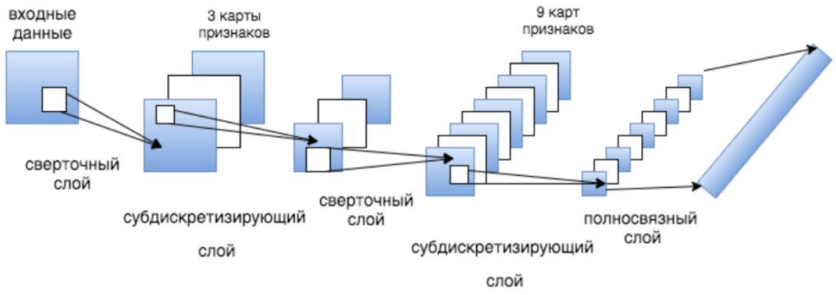

Рисунок 9 – Архитектура сверточной нейронной сети

На сверточном слое к исходному изображению применяется операция свертки. Использование алгоритма свертки позволяет найти объект на изображении независимо от его положения или углового смещения. На вход свёрточного слоя подается изображение в виде матрицы пикселей, размерность которой определяется размером изображения. Исходное изображение разбивается на несколько пересекающихся связанных фрагментов. Каждый полученный фрагмент по очереди подается на вход нейронной сети, которая их обрабатывает с использованием одинаковых весов, формирующих ядро свертки.

Операция свертки начинается с определения рецептивного поля, размер которого равен ядру свертки. Данное поле продвигает по исходному изображению с некоторым смещением и пиксели изображения, попавшие в него, поэлементно перемножаются с ядром свертки. Элементы новой матрицы, полученной в результате перемножения, суммируются и образуют новый выходной пиксель. Результатом работы алгоритма свертки является уменьшенное изображение, которое содержит наиболее «выдающиеся» участки родителя. На рисунке 10 графически изображен процесс работы слоя свертки.

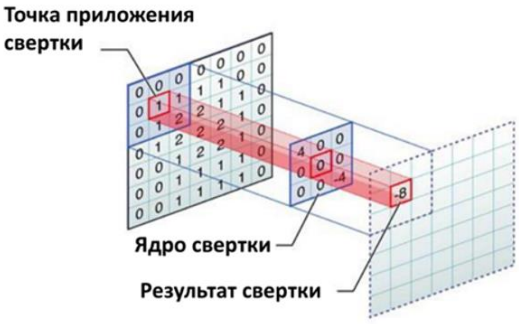

Рисунок 10 – Пример свертки изображения

Из-за того, что граничные пиксели исходного изображения не оказываются в центре рецептивного поля, матрица, полученная в результате свертки, имеет меньшую размерность, чем исходное изображение. Чтобы избежать постоянного сжатия изображения при выполнении свертки и сохранить информацию о его границах, вокруг изображения добавляется рамка из нулей, ширина которой зависит от размера ядра свертки. Данная операция называется заполнением [14].

Так как это уже многослойная нейронная сеть, то тут присутствует смещение – значение, изменяемое в процессе обучения, которое складывается с каждым элементом выходной матрицы. Для того чтобы продолжить свое прохождение полученная матрица должна пройти функцию активации. Так как берем нелинейную функцию активации, но при этом будем использовать функцию активации усеченного линейного преобразования (ReLu), то приводит к тому, что все отрицательные значения, если они есть, в матрице преобразуются в «0», а все остальные значения сохраняются.

Итак, в результате прохождения свертки получается матрица, называемая картой признаков, на которой выделен конкретный признаки, такие как: различные прямые и кривые линии, полуокружности или более сложные признаки, что именно выделяет тот или иной светорчный слой зависит от его местоположения в сети. Если первый сверточный слой выявляет базовые признаки, такие как линии, то последующие сверточные

слои выявляют признаки все более и более высокого уровня, постепенно обобщая информацию о объектах на изображении [14]. Сверточные слои чаще всего используют более одного ядра свертки, каждый из которых идентифицирует конкретный признак, поэтому в результате свертки получается набор различных числовых матриц, число которых характеризует глубину карты признаков. Чем больше количество используемых ядер свертки, тем более подробную информацию можно получить об исходном изображении.

Следующим слоем сети является пулинговый слой или слой подвыборки. На этом слое производится уменьшение размерности сформированных карт признаков. Данный слой имеет два типа: на одном мы ищем среднее значение в некоторой области, что показывает среднее значение интенсивности признака, или же по максимальному признаку в некоторой области, что дает точный показатель максимальной глубины в области. Чаще всего используют подвыборку по максимальному значению, так как она более четко выделяет ключевые признаки, подавляя лишние шумы на изображении [14]. На рисунке 11 графически изображен процесс работы слоя подвыборки.

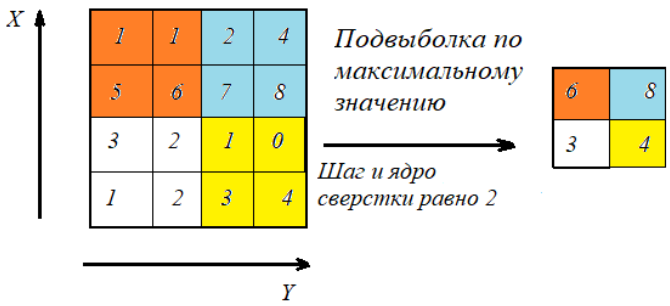

Рисунок 11 – Работа операции подвыборки

Для того, чтобы выявить сложные признаки на изображении, сверточные сети содержат множество чередующихся слоев свертки и подвыборки. Следовательно, размерность изображения уменьшается, а

глубина карты признаков растет, что позволяет более точно определить разницу признаков.

Конечными слоями сверточной нейронной сети являются полносвязные слои, с помощью которых происходит классификация объекта на изображении. Весовые коэффициенты данных слоев настраиваются в процессе обучения для определения метки класса на основе выявленных признаков. Перед подачей полученной карты признаков на полносвязный слой, происходит объединение слоев карты признаков в вектор, где каждый элемент показывает вероятность присутствия конкретного признака на изображении. Последний полносвязный слой содержит количество нейронов, равное количеству определяемых классов и с помощью выбранной функции активации выводит вероятности присутствия каждого класса на изображении [20].

Таким образом, были рассмотрены основные принципы работы искусственных нейронных сетей, особенности их построения и методы, с помощью которых происходит их обучение. Кроме того, был подробно разобран процесс применения сверточной нейронной сети для классификации объекта на изображении.

1.3 Обзор существующих решений для анализа трафика

Программы способные к распознанию, анализу и классификации трафика применяются во множестве областей и с огромным количеством данных. Рассмотрим системы, которые уже давно испытываются и работают в России.

Первая система, которая используется массово в России при помощи технологии анализа данных, это сервисы «Google Maps», с 2005 года разрабатываемые компанией «Google». Данная платформа позволяла сделать множество различных сервисов, таких как «прокладка маршрутов» или «просмотр улиц». Также на системе был представлен сервис

«GoogleПробки». Сервис работает через отслеживающие устройства водителей, такие как телефоны или навигаторы. Благодаря полученным данным и их анализу сервис способен с помощью собственной нейронной сети вывести за короткое время информацию о загруженности дорог на карту. На рисунке 12 изображен интерфейс рабочей программы «Google Maps».

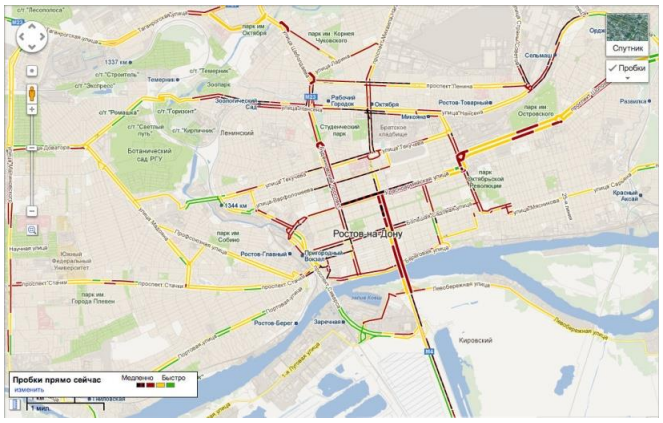

Рисунок 12 – Пример работы сервиса «Google Maps»

Вторая система, под названием «Angel Vision», созданная компанией «Angels IT» в 2009 году. Она работает как самообучающаяся система распознавания объектов и информации на изображении и в видеопотоке на основе искусственного интеллекта. Первым из проверочных заданий данной системы было создание механизма определение камерой цифр с номерного знака и запись данного номера в базу данных. На рисунке 13 изобрежен пример работы данной программы по считыванию номеров автомобиля с камеры в двиэжении.

Рисунок 13 – Пример работы первой версии программы «Angels Vision»

Как заявляет разработчик программного обеспечения, система работает с точностью локализации в 99%, при высоком потоке автомобилей, а также имеет такие плюсы как угол работы камеры до 50 градусов и большое количество разнообразных условий, такие как разные погодные явления, время суток и года.

Данная система получила дальнейшее развитие и дополнительные возможности, такие как контроль и анализ парковочных мест, разложение автомобильного трафика на дороге по типу машин и нахождение средней загруженности просматриваемого участка, локализация спецтранспорта в трафике.

В 2019 году система была установлена на 2500 камер города Воронежа, что позволило органам правопорядка увеличить раскрываемость преступлений по угонам автомобилей, а местным властям сконцентрировать силы для улучшения дорожной инфраструктуры в самых опасных участках города.

Следующая система анализа называется «TrafficData Land», разработанная компанией «TrafficData». Данная система разрабатывалась для практически любого анализа трафика, в первых версиях программа находила и классифицировала автомобильный поток, могла провести анализ парковочных мест на дороге, провести детекцию ДТП. На рисунке 14

изображена работа программы на перекрестке, а именно система определяет направления движения потока.

Рисунок 14 – Пример работы программы «TrafficData Land»

В дальнейшем разработчики усовершенствовали программу для использования её не только с камер, но и с квадракоптеров, что позволило увеличить возможности программы по размерам площади анализа, данная программ называется «TrafficData Air». Также были проведены тесты для объединения всех камер города Пермь в единую Интеллектуальную Транспортную Сеть (ИТС), включающую такие модули, как мониторинг дорожного движения, умная парковка, управление дорожным движением. В 2017 году количество камер ИТС было увеличено до 5000. На рисунке 15 изображена работа системы с рисунка 14, только программа была запущена с дрона и подключили блок записи автомобилей.

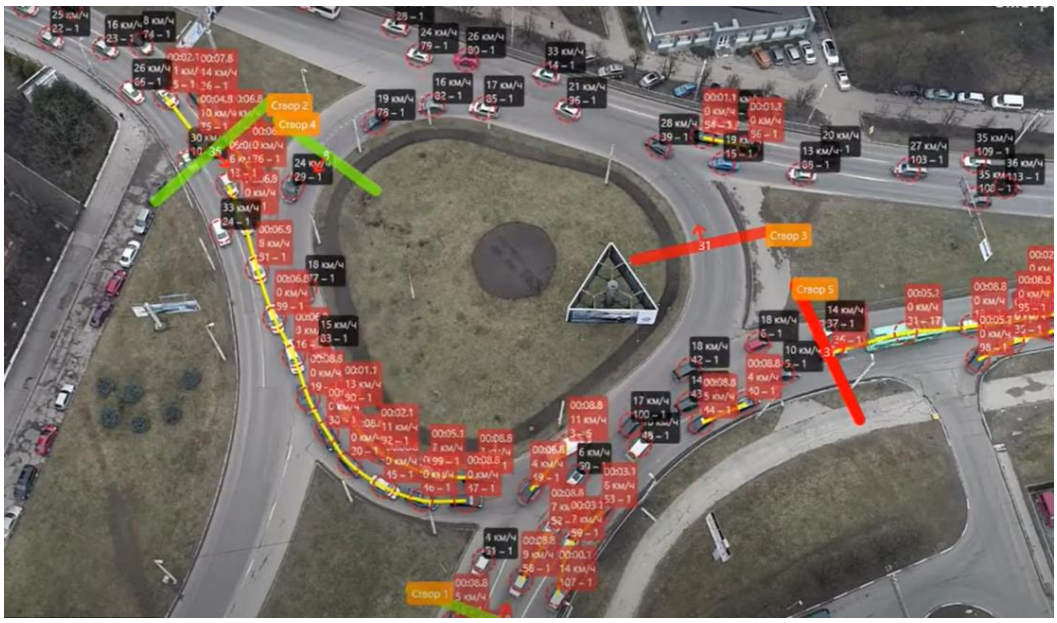

Рисунок 15 – Пример работы программы «TrafficData Air»

Данные системы работают в России уже более 10 лет и до сих пор поддерживаются и развиваются их разработчиками. На сайтах представленных систем выложена информация об этапах развития проектов и дальнейшие планы создателей этих проектов. Это позволяет предполагать, что данные системы не будут заброшены. А сами компании считают свои разработанные системы коммерческим продуктом, способным завоевать нишу на рынке.

Таким образом, были рассмотрены самые массово применяемые системы России, использующие нейронные сети для обработки данных. Была дана информация о структуре данных сетей и вехах их развития.

Выводы к главе 1

Были рассмотрены задачи анализа данных, дана информация о структуре поступающих значений, возможности их записи для анализа и исправления деструктурируемых данных. Была предоставлена информация о самых крупных методах анализа, их принципах работы. Рассмотрен алгоритм построения сверточной нейронной сети, принцип и проблема обучения сети, отдельные модули работы. Дана информация о системах анализа трафика в России, их путь развития и возможности для обработки данных.

2 Построение модели программного обеспечения и его реализация.

2.1 Обзор целей и задач светофоров на дороге

Рассмотрим случаи, когда устанавливают светофор на участке дороге, какие причины и цели несет установка светофора в данной местности. Первичной задачей светофорного дорожного регулирование является нормализация и придание иерархичности проезду транспорта на участке дороги, кроме того светофоры уже давно используются для подачи различных сигналов, такие как аварийные или опасные участки дорог. Чтобы светофор входил в систему регулирования, он должен быть расположен:

- на дорожных полосах, где возможно изменение направления движения на противоположное;
- на железнодорожных переездах, разводных мостах, причалах, паромах и переправах;
- в точках, где транспортные средства различных экстренных и специальных служб выезжают на дорогу с активным движением;
- на участках пути, где имеется несколько разных конфликтующих транспортных и пешеходных потоков (дорожные перекрёстки, пешеходные переходы);
- в местах, где необходим контроль над движением общественного транспорта.

Системы светофорного регулирования ― это крайне эффективный метод регулирования, так как не является дорогим, прост в обращение, но при этом используется большим потоком автомобилей. Светофоры выполняют ряд важнейших функций: повышают уровень безопасности на дорогах и улучшают качество движения, положительно влияют на экологические показатели. Рассмотрим задачи светофоров более подробно.

Задачи безопасности, выполняемые при светофорном регулирование движения, очень полезны и являются основополагающими для регулирования, их необходимо использовать в ситуациях, когда на том или ином участке пути прогнозируются либо уже происходят ДТП, которые можно было бы предотвратить, будь на данном отрезке установлен светофор. Как пример, к частным решением является расстановка приоритета проезда и регулирование при повороте транспорта налево с пересечением встречной полосы.

Задачи улучшения качества движения, как во всей транспортной сети, так и на отдельных её участках (перекрёстках, перегонах и т.п.) существенно возрастает при условии введения светофорного регулирования. То есть появляется возможность освобождения участков дороги от машин и предотвращения появления пробок.

На экологические показатели благоприятно влияет весь комплекс мер, такие как возможность обеспечение транспорта одинаковой скорости, сокращающих количество остановок, разделение основного потока на более мелкие части и способствующих равномерному движению через несколько пересечений. Это могут быть дополнительные светофоры, так называемая «зелёная волна» или же адаптивное светофорное регулирование. Благодаря этим мероприятиям сокращается расход топлива, снижается уровень шума от транспортных средств, уменьшаются объёмы выделяемых в окружающую среду выхлопных газов.

По полученным задачам светофоров можно точно сказать, что светофоры устанавливают на местах пересечения двух или более потоков автомобильного трафика. Теперь определим актуальность, задачи и принципы работы «умных светофоров».

Актуальностью данной системы является борьба больших городов с пробками, так как большая загруженность транспортной инфраструктуры города на участке могут приводить к ухудшению производительности труда, снижению уровня логистики и качества жизни города, увеличению затрат

для обслуживания служебного транспорта. Не в лучшую сторону меняется отношение людей к городской власти и идет повсеместное умножение количества вредных веществ в атмосфере.

Между тем проводят различные исследования на тему развития технологий способных улучшить работу светофоров в городской инфраструктуре. Как пример, специалисты американского Университета Карнеги-Меллона подсчитали, что если оснастить город численностью в 2 млн человек системами «умный светофор», то количество пробок в городской черте будут снижены от 25 до 40%, общие затраты на время проведенное в пробках будет уменьшено на 25 %, а количество негативного эффекта на окружающую среду будут сокращены на 15%. Данные исследования показывают высокую эффективность систем «умный светофор», если они будут интегрированы в существующие транспортные сети.

В задачу «умного светофора» входит управление световыми сигналами, в соответствии с анализом машин с ближайших камер или прогнозированием системы о том, что данный участок имеет высокую вероятность быть перегруженным в будущем времени.

Принцип работы системы состоит в том, что получив данные анализа, светофор посылает контролерам сообщение о смене сигнала или изменении времени выполнения данного сигнала. Сигналы включаются, так чтобы сократить время пребывания автомобиля на перекрестке и разделить автомобильный поток на меньшие доли. Данная система есть и в России. Из 45 тысяч светофоров города Москвы более 4 тысяч входят в единую систему контроля транспорта и способны динамически менять время переключения сигналов.

Систему «умный светофор» можно разделить на несколько различных типов светофоров. Самым распространенным является локальный светофор, то есть увеличение промежутка между сигналами на светофоре жестко выставлено на время суток, например как час пик. Светофоры данной модели

не могут проводить сами анализ транспортных средств и должны запускаться вручную.

Следующий тип динамические (адаптивные) светофоры, они способны проводить анализ транспортных средств и выставлять время сигналов.

Недостатком динамических (адаптивных) светофоров является невозможность установить их на маршруте интенсивного движения, так как существуют множество проблем с движением транспорта, например, многие датчики будут считать быстро движущийся автомобиль, как несколько автомобилей, или давать сбои в некоторых погодных условиях. Так что место работы динамических светофоров определено соединениями потоков с низкой интенсивностью движения.

Второй недостаток, что при работе светофоры очень сильно зависят от выполнения правил установки видеокамер, высоты подвеса, отсутствия препятствий на линии обзора. Для лучшего обзора, который будет обеспечивать осмотр большой площади, камеру можно установить в светофор. К такому выводу можно придти, так как по правилам «Национального стандарта Российской федерации», светофор должен быть установлен так, чтобы была обеспечена видимость их сигналов с расстояния не менее 100 метров с любой полосы движения, что позволяет камерам проводить контроль большей территории, так как ей не будут мешать разнообразные объекты. Но даже это не полная гарантия, что ничего не будет мешать установленной камере. Кроме того есть вероятность смещения как светофора так и камеры интегрированной в него.

Третий недостаток состоит в том, что для полноценной работы данного светофора на всем пути движения автомобилей для более точного контроля могут понадобиться несколько камер, что приводит к удорожанию всей системы.

Следующий тип «умного светофора» это динамический светофор, использующий системы нейронных сетей внутри себя. Данные системы самые сложные и одновременно самые эффективные, так нейронным сетям

не требуется высокое качество объекта, и они могут прекрасно работать при больших перекрытиях транспортных средств. Также какие-либо перемещения камеры не приводят к обрушению системы, так как нейронная сеть способна работать в условиях смещения углов обзора.

Таким образом, мы рассмотрели цели и причины установки светофора, актуальность создание и развития систем «умный светофор» и ознакомились с типами светофоров, входящие в данную систему.

2.2 Построение модели системы, принцип работы

Создадим модель программного обеспечения для технологии «умный светофор», зададим нужные цели и этапы работы программы. Построим модули работы системы.

Было определенно, что разрабатывать систему надо под самые распространенный метод скрещивания потоков, так можно не перестраивая сильно систему, оснастить большее количество светофоров. По полученным результатам, стало понятно, что под критерии подходит крестовый перекресток. Следовательно, будут задействованы четыре разных потока движения и на каждый поток должна быть установлена камера. Считать программа будет только поток, находящийся перед камерой, те транспортные средства, которые проедут камеру, будут исключены из счета.

Теперь рассмотрим, какие углы обзоров могут получить с камеры при интегрирование их в светофор. Светофор может быть установлен как сверху над проезжей частью дороги, так и сбоку от проезжей части. Так что построим схематические рисунки углов обзора камера, при разных условиях установки светофора. На рисунке 16 изображены углы обзора если камера будет установлена на светофор, который установлен сбоку от дороги, а на рисунке 17 если установлен на светофор, весящий над дорогой.

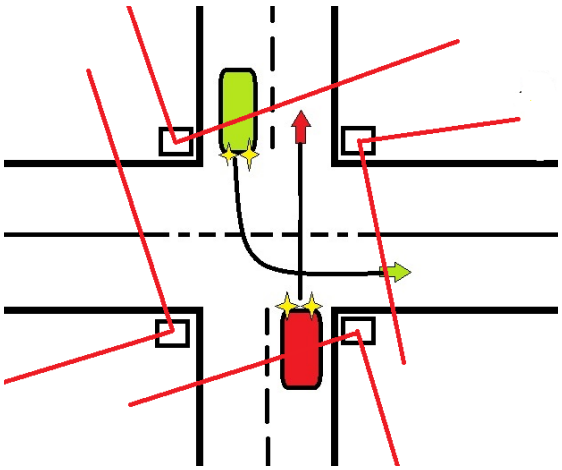

Рисунок 16 – Пример обзора камеры установленной сбоку

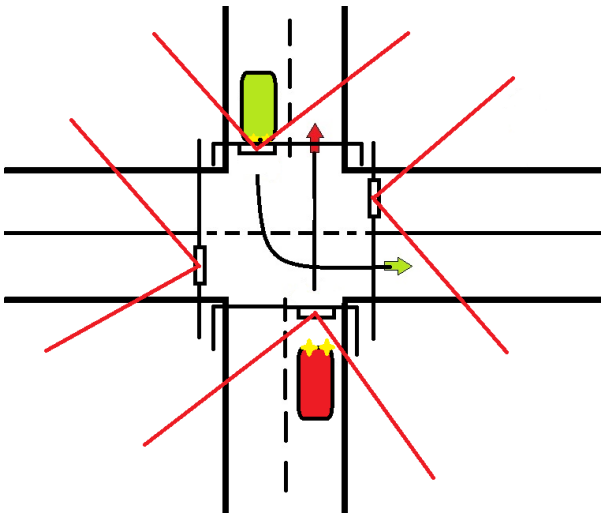

Рисунок 17 – Пример обзора камеры установленной сверху

В технологии построения динамического светофора, использующего нейронные сети, структурно выделяют 3 модуля работы:

- модуль получения информации;
- модуль анализа или принятия решения;
- модуль отправки.

Все модули жестко связаны в иерархии, пропуск модуля или не удачный результат работы приводит к неисправности системы и невозможности выполнять заложенные действия. Также система не способна работать, если произошел сбой, приведший к изменению порядка расположения модулей.

В модуле получения информации выполняется работа по вводу картинки, то есть проверка наличия, правильности работы камер, получения изображение с камер и передача на модуль анализа.

Анализом будет заниматься специально настроенная нейронная сеть. К задачам сети будет относиться локализация объектов на изображение, классификация в отведенные классы и выделение объектов рамками. После система считает количество объектов, выделенных на изображении у каждой камеры, и расположенные в противоположные стороны камеры складывает полученные значения количества объектов. Следующим шагом будет нахождение соотношение полученных данных с двух перпендикулярных потоков и проход по списку решений, который приведет к возможным изменениям графика движения на светофоре для более быстрого прохождения трафиком участка дороги.

Из этих шагов мы может разделить нейронную сеть на три этапа, следующие друг за другом:

- этап локализации и классификации;
- этап нахождения соотношения;
- этап принятия решения.

Для большей наглядности и понимания, того что должно быть вводными данными и результатом каждого этапа, построим логическую модель нейронной сети. Данная модель изображена виде блок-схемы на рисунке 18

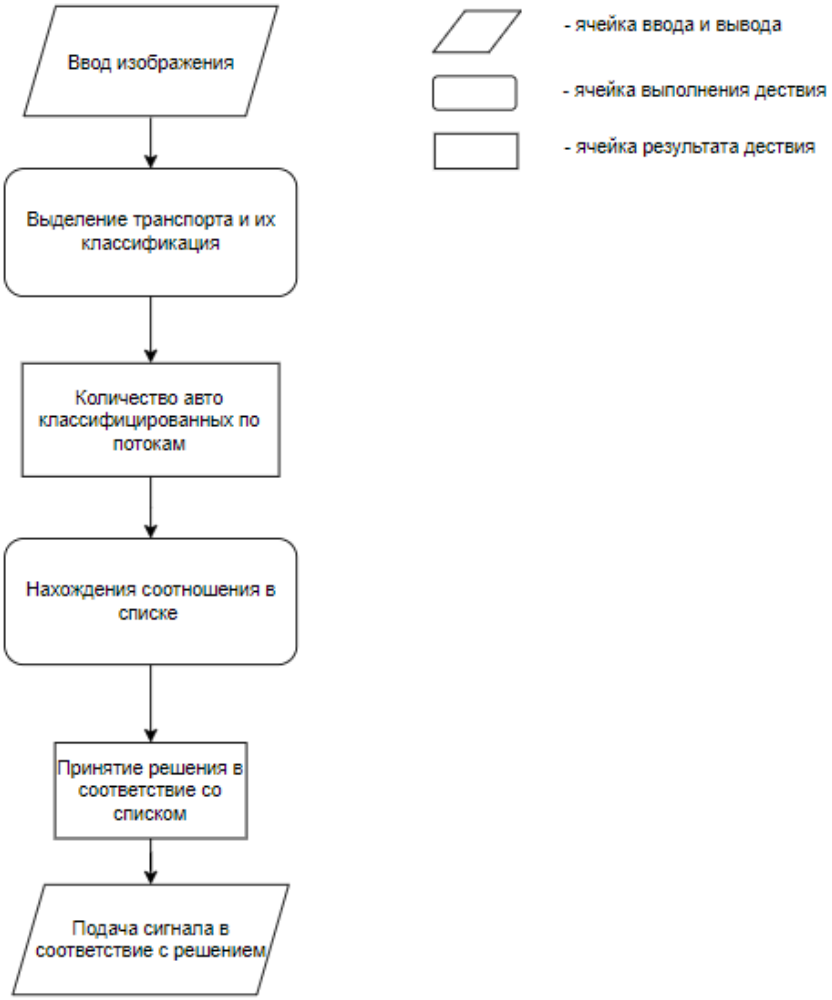

Рисунок 18 – Логическая схема модуля анализа

Процесс выделения будет происходить на нейронной сети, обученной к локализации и классификации транспорта. Деление на поток будет происходить по принципу, куда смотрит автомобиль, то есть если камера видит переднюю часть автомобиля, то он относиться к потоку, если видит заднюю часть, то не к одному из потоков не относится. Так нейронная сеть научится по углу обзора на автомобиль решать вносить ли его в класс или нет.

Противоположные потоки во время этапа соотношения будут складываться, так как светофор работает в двух фазах. Фаза пропуска одного

потока, следом перпендикулярного. Процесс нахождения соотношения будет сделан через простую формулу соотношения (6):

$$F = P : Q, \tag{6}$$

где F – результат соотношения;

P – класс имеющий меньшее значение;

Q – класс имеющий большее значение.

В программе будет сделан список принятия решений. По результату соотношения и тому, какой поток имеет большее значение, а какой меньшее, будет находиться один из вариантов решения из списка. Система сможет понять, какие команды должны быть переданы, чтобы эффективней выполнялся проезд транспортных средств через перекресток.

В последнем модуле выполняются несколько действий. Первоначально во время работы программы происходят постоянные проверки соединения светофора и системы, если соединение не найдено, то подается сигнал ошибки. Также после принятия решения именно данный модуль должен по соединению подавать сигнал принятого решения светофору.

Теперь нам не хватает построение списков. Для начала введем несколько констант. Время работы одного сигнала на светофоре 20 секунд, которое может быть изменено по решению светофора. На рисунке 19 представлено, как должны складываться противоположные потоки, результаты потока с камеры 1 и 3 сложатся и будут соотноситься с камерами 2 и 4:

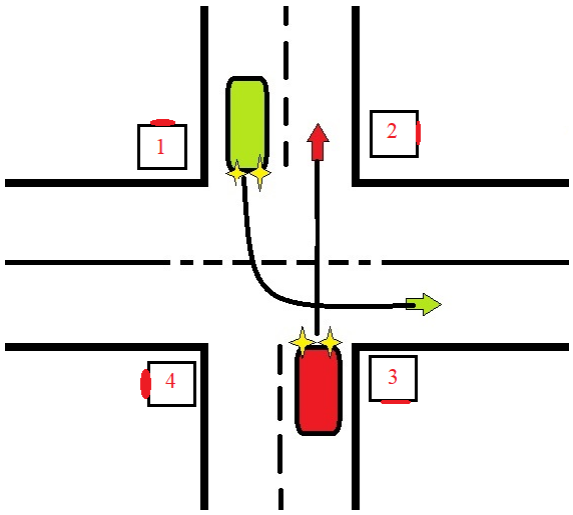

Рисунок 19 – Схема сложения потоков

Теперь мы должны выделить исключения, которые не позволят нам зайти в список, всего их будет два. Первое, если один из потоков не имеет ни одного транспортного средства, то второй, на котором есть автомобили, выполнит команду увеличение времени в два раза от зеленого цвета для данного потока, то есть время будет увеличено до 40 секунд. Второе исключение, когда вообще нет потока, тогда на светофор подается команда о сокращение обоих потоков на 10 секунд.

Если данные исключения не выполняются, то система идет в список. Весь список будет поделен на две части, данные части будут разделены тем, что полученным двум потокам будет присвоено значение большего и меньшего потока. Как пример, система решает идти в первую часть списка, так как количество автомобилей на объединенном 1 и 3 потоке машин больше, чем на объединенном потоке 2 и 4. В дальнейшем списки с двух сторон имеют одинаковые условия выполнения, приведенные на рисунке 20.

Если F = 1, то изменения не вводятся;
Если F >= 0.75 and F< 1, то светофор потока Q увеличивается 10 секунд, светофор потока P не изменяется;

Если F >= 0.5 and F< 0.75, то светофор потока Q увеличивается 20 секунд, светофор потока P не изменяется;

Если F >= 0.25 and F< 0.5, то светофор потока Q увеличивается 20 секунд, светофор потока P уменьшается на 10 секунд;

Если F > 0 and F < 0.25, то светофор потока Q увеличивается 30 секунд, светофор потока P уменьшается на 10 секунд;

Рисунок 20 – Условия выполнения команд

Теперь, когда были рассмотрены модули нашей программы и что каждый из них делает, можно приступать к реализации алгоритма

2.3 Реализация системы «умный светофор»

Для реализации системы, был выбран высокоуровневый язык Python, главными особенностями его является огромное количество библиотек и модулей для создания и обучения нейронных сетей. На данном языке написано множество руководств про созданию нейронных сетей, тут предложены огромные датасеты и уже присутствуют обученные модели нейронных сетей.

Для реализации и обучения нейронной сети была выбрана библиотека машинного обучения TensorFlow, включающую в себя множество решений и алгоритмов, интегрированная в сервис Google Collab, что позволяет не загружая систему компьютера, работать с нейронными сетями.

Сделаем нейронную сеть которая способна найти и классифицировать объект на изображение. Строить нейронную сеть самостоятельно не надо. Так как мы находим и классифицируем один объект на изображении, то построение слоёв модели выполняется модулем Keras, который ьыл сначаоа самостоятельным, но с 2014 года интегрирован разработчиками в библиотеку

TensorFlow 2 [4][12]. На рисунке 21 мы можем видеть часть кода ответсвенную за создание слоев нейронной сети модели Keras.

```
[49] base_layers = tf.keras.applications.MobileNetV2(input_shape=(SIZE, SIZE, 3), include_top=False)
     base_layers.trainable = False

[50] model = tf.keras.Sequential([
                                  base_layers,
                                  GlobalAveragePooling2D(),
                                  Dropout(0.2),
                                  Dense(1)
     ])
     model.compile(optimizer='adam', loss=tf.keras.losses.BinaryCrossentropy(from_logits=True), metrics=['accuracy'])
```

Рисунок 21 – Реализация модели через Keras

Смысл взять именно эту модель состоит в том, что данный образец уже обучен и нам нужно всего лишь дообучить его на нужные системе классы. Но также данная модель имеет существенный недостаток, так как для переобучения понадобиться установить значение размерности изображения 224X224, так как именно под эту размерность и написана данная модель [24]. При этом не стоит волноваться, потеря части объектов на изображение такой размерности происходит с 300 и более метров. На рисунке 22 показан код смены размерности, если будут поданы изображения не с видеопотока, а с фотографий.

```
def resize_image(img, label):
  img = tf.cast(img, tf.float32)
  img = tf.image.resize(img, (SIZE, SIZE))
  return img, label
train_resized = train[0].map(resize_image)
```

Рисунок 22 – Реализация смены размерности

Проверим работу получившейся программы для этого введем данные с датасета, в нашем случаи возьмем датасет «Beans», который интегрирован в Tensorflow. Данный датасет представляет собой набор изображений бобов,

сделанных в поле с помощью камер смартфонов. Состоит из 3 классов: 2 класса болезней и класса здоровых. Изображенные болезни включают угловую пятнистость листьев и бобовую ржавчину. На рисунке 23 показаны системные сообщения о загрузке датасета.

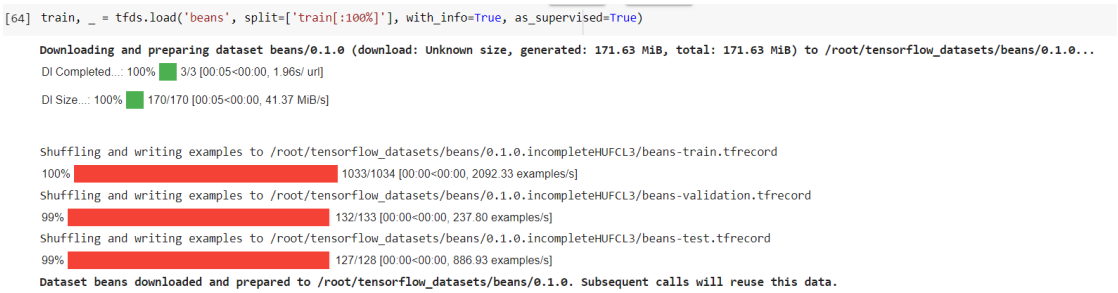

Рисунок 23 – Подключение и проверка датасета

Теперь с загруженным датасетом и построенной моделью Keras, попробуем обучить нейронную сеть классифицировать бобы на 10 эпохах обучения. На рисунке 24 изображены системные сообщения в программе «Google Collab» при обучение с загруженным датасетом.

```
model.fit(train_batches, epochs=10)
```

```
Epoch 1/10
65/65 [==============================] - 34s 477ms/step - loss: -2.3338 - accuracy: 0.3530
Epoch 2/10
65/65 [==============================] - 34s 491ms/step - loss: -2.9955 - accuracy: 0.3375
Epoch 3/10
65/65 [==============================] - 33s 483ms/step - loss: -3.6369 - accuracy: 0.3598
Epoch 4/10
65/65 [==============================] - 33s 481ms/step - loss: -4.2269 - accuracy: 0.3414
Epoch 5/10
65/65 [==============================] - 34s 499ms/step - loss: -4.9560 - accuracy: 0.3530
Epoch 6/10
65/65 [==============================] - 34s 487ms/step - loss: -5.5972 - accuracy: 0.3491
Epoch 7/10
65/65 [==============================] - 32s 466ms/step - loss: -6.2352 - accuracy: 0.3501
Epoch 8/10
65/65 [==============================] - 32s 457ms/step - loss: -7.0007 - accuracy: 0.3540
Epoch 9/10
65/65 [==============================] - 33s 488ms/step - loss: -7.6745 - accuracy: 0.3520
Epoch 10/10
65/65 [==============================] - 34s 490ms/step - loss: -8.3365 - accuracy: 0.3540
<keras.callbacks.History at 0x7fc1dd7175d0>
```

Рисунок 24 – Обучение на 10 эпохах датасета «Beans»

Из полученных результатов видно, что классификация на изображение держит в районе 33-35%, что является плохим результатом. Главной

причиной того, что нейронная сеть не может дать более точные ответы, состоит в том, что на изображении данного датасета находятся несколько признаков, а нейронная сеть модели Keras находит первый, ярко выраженный признак класса и сразу классифицирует объект, отбрасывая другие признаки. Так как всего признаков классов 3, то и правильных ответов должно быть примерно 33%.

Это близко к тому результату, что дает нам сеть. Но лучше всего мы научим нейронную сеть определять несколько признаков на изображение. Для этого существует несколько алгоритмов детектирования. С данной работой могут справиться алгоритмы, такие как R-CNN, YOLO, SSD [6]. Но проблема алгоритмов на архитектуре R-CNN в том, что требуют мощных вычислений и это приводит к увеличению времени обработки или алгоритмы SSD, которые не нагружают компьютерные системы, но не способны получать более точные результаты. Поэтому выбор пал на представителей средний машин, где самую высокую эффективность показывают различные версии алгоритма YOLO [8]. На рисунке 25 представлено сравнительные графики алгоритмов локализации и классификации по таким данным как точность работы и скорость работы.

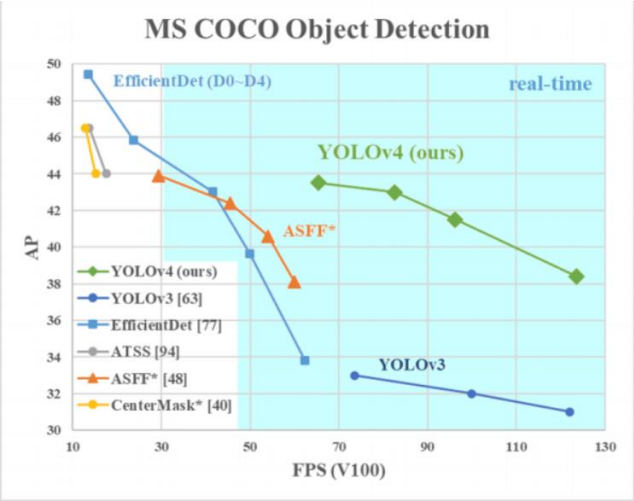

Рисунок 25 – Сравнительные тесты алгоритмов детектирования

Реализуем код, включающий нейронную сеть, которая будет находить, локализовать и классифицировать объекты. Для проверки правильности работы сначала сделаем программу, которая будет использовать изображения. В современном мире писать нейронную сеть уже не обязательно, в свободном доступе есть сразу обученные на множество данных нейронные сети. Так как будет использован алгоритм YOLO, то надо понимать, что данный алгоритм уже имеет свою обученную модель на датасете, который называется MS COCO. Данный датасет имеет больше 60000 тысяч изображений для обучения и 10000 для проверки, сам алгоритм выдает вероятность локализации и классификации в около 90%.

Для начала возьмем изображение, на котором надо детектировать объекты. На рисунке 26 изображена проезжая часть, которую и будет смотреть система.

Рисунок 26 – Изображение, подаваемое в нейронную сеть

```
# загружаем сеть YOLO
net = cv2.dnn.readNetFromDarknet(config_path, weights_path)

path_name = "images/horse.jpg"
image = cv2.imread(path_name)
file_name = os.path.basename(path_name)
filename, ext = file_name.split(".")

h, w = image.shape[:2]
blob = cv2.dnn.blobFromImage(image, 1/255.0, (416, 416), swapRB=True, crop=False

net.setInput(blob)
# получаем имена всех слоев
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

Рисунок 27 – Реализация подключения сети YOLO

На рисунке 27, виден алгоритм подключение обученной сети YOLO, также вводиться изображение рисунка 26, которые пережимается до размерности 416x416. YOLO делит изображение на матрицу 13x13 и в каждой ячейке матрицы нейронная сеть выделяет признаки объекта. Это могут быть линии, окружности, квадраты. После происходит проход по всей нейронной сети и выносится решение о том, какая классификация у объекта.

```
# перебираем каждый из выходов слоя
for output in layer_outputs:
    # перебираем все обнаруженные объекты
    for detection in output:
        # извлекаем идентификатор класса (метку) и достоверность (как вероятность)
        # обнаружение текущего объекта
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        # отбросим слабые прогнозы, убедившись, что обнаруженные
        # вероятность больше минимальной вероятности
        if confidence > CONFIDENCE:
            # масштабируем координаты ограничивающего прямоугольника относительно
            # размер изображения, учитывая, что YOLO на самом деле
            # возвращает центральные координаты (x, y) ограничивающего
            # поле, за которым следуют ширина и высота полей
            box = detection[:4] * np.array([w, h, w, h])
            (centerX, centerY, width, height) = box.astype("int")
            # используем центральные координаты (x, y) для получения вершины и
            # и левого угла ограничительной рамки
            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))
            # обновим наш список координат ограничивающего прямоугольника, достоверности,
            # и идентификаторы класса
            boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            class ids.append(class id)
```

Рисунок 28 – Реализация функции выделения объекта

Теперь для наглядности выведем рамки на объект. На рисунке 28 изображен алгоритм выделения всех рамок вокруг объекта. Но на каждый объект можно выделить огромное количество рамок одновременно, главное чтобы объект был внутри, но чтобы подавить сторонние рамки, нужно

выбрать наилучшие рамки из всех, этим займется алгоритм «подавления не-максимумов». На рисунке 29 изображен алгоритм подавления лишних рамок.

```
idxs = cv2.dnn.NMSBoxes(boxes, confidences, SCORE_THRESHOLD, IOU_THRESHOLD)
font_scale = 1
thickness = 1
# убедитесь, что существует хотя бы один обнаруженный объект
if len(idxs) > 0:
    # перебираем сохраняемые индексы
    for i in idxs.flatten():
        # извлекаем координаты ограничивающего прямоугольника
        x, y = boxes[i][0], boxes[i][1]
        w, h = boxes[i][2], boxes[i][3]
        # рисуем прямоугольник ограничивающей рамки и подписываем на изображении
        color = [int(c) for c in colors[class_ids[i]]]
        cv2.rectangle(image, (x, y), (x + w, y + h), color=color, thickness=thickness)
        text = f"{labels[class_ids[i]]}: {confidences[i]:.2f}"
        print(labels[class_ids[i]])
        count_object +=1
        # вычисляем ширину и высоту текста, чтобы рисовать прозрачные поля в качестве фона текста
        (text_width, text_height) = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, fontScale=font_scale, thickness=thickness)[0]
        text_offset_x = x
        text_offset_y = y - 5
        box_coords = ((text_offset_x, text_offset_y), (text_offset_x + text_width + 2, text_offset_y - text_height))
        overlay = image.copy()
        cv2.rectangle(overlay, box_coords[0], box_coords[1], color=color, thickness=cv2.FILLED)
        image = cv2.addWeighted(overlay, 0.6, image, 0.4, 0)
        # теперь поместите текст (метка: доверие%)
        cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,
            fontScale=font_scale, color=(0, 0, 0), thickness=thickness)
        print(count object)
```

Рисунок 29 – Реализация алгоритма подавления не-максимумов

Теперь можно посмотреть результат работы программы. На рисунке 30 показано изображение с выделением детектированных объектов.

Рисунок 30 – Результат выхода из нейронной сети.

Теперь реализуем функцию, которая будет считать количество детектированных объектов на изображение, как пример возьмем рисунок 30,

и затраченное время на детектирование изображения. На рисунке 31 показан результат данной информации выведенный в командную строку.

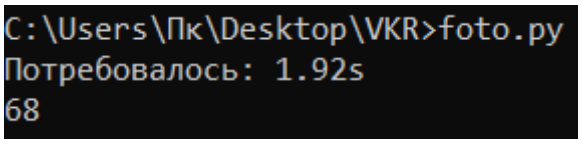

Рисунок 31 – Подсчет затраченного времени и количества объектов

Данный код создан для действий над изображениями. Нужна система способная работать с видеопотоком. Также стоит заметить, что время работы программы длится примерно 2 секунды, следовательно нужно сделать видеопоток со скоростью кадр раз в две секунды.

Начнем с первого модуля, ответственного за получения изображения. Для начала определимся с источником данных, для этого используем портал «https://weacom.ru/cams», так как тогда сможем с этого сервиса получить видеопоток, который преобразуем в изображения для программы. Для наглядности была выбрана камера со средней интенсивностью, а именно камера, находящиеся в городе Иркутск на Академическом мосту. Чтобы получить кадры с камеры, нам необходимо подключиться к потоку самой камеры. Для реализации кода будем использовать OpenCV.

```
import cv2
import time

cap = cv2.VideoCapture('https://cctv.baikal-telecom.net/Akademmost-2/index.m3u8'
cap.set(cv2.CAP_PROP_FPS, 5)
success, frame = cap.read()
prev = 0
print(cap.get(cv2.CAP_PROP_FPS))

while success:
    time_elapsed = time.time() - prev
    success, frame = cap.read()

    if time_elapsed > 1. / 5:
        prev = time.time()

    cv2.imshow('Weacom', cv2.resize(frame, (1280, 1080)))
    cv2.waitKey(20)
    key = cv2.waitKey(1)
    if key == ord('s'):
        cap.capture.release()
        cv2.destroyAllWindows()
        exit(1)
```

Рисунок 32 – Реализация передачи видеопотока OpenCV

На рисунке 32 изображен программный код, отвечающий за взятие изображения с видеопотока. Данный код позволит программе рассматривать по-фрагментно изображения видеопотока с камеры, то есть будет создан своеобразный датасет, в которой будут сохраняться изображения, а яерез 10 минут удаляться из датасета. Также для удобства работы изображение будет выведено на виджет, чтобы было более наглядно видно. На рисунке 33 представлено одно из изображений с камеры в виджете.

Рисунок 33 – Результат работы кода передачи

Так как есть код, ввода видеопотока, соединим его с нейронной сетью, способной детектровать объекты с полученных изображений. На рисунке 34 изображен код передающий изображения на нейронную сеть YOLO.

```
net = cv2.dnn.readNetFromDarknet(config_path, weights_path)
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
cap = cv2.VideoCapture("https://cctv.baikal-telecom.net/Akademmost-2/index.m3u8")
cap.set(cv2.CAP_PROP_FPS, 5)
print(cap.get(cv2.CAP_PROP_FPS))
_, image = cap.read()
h, w = image.shape[:2]
prev = 0
fourcc = cv2.VideoWriter_fourcc(*"XVID")
out = cv2.VideoWriter("output.avi", fourcc, 20.0, (w, h))
```

Рисунок 34 – Реализация кода, подающая видеопоток на нейронную сеть

Рассмотрим передачу сигнала на светофор, если все наши условия выполнены. Вся работа с устройством происходит благодаря библиотеке PySerial. Для начала узнаем, подключен ли к нам светофор. На рисунке 35 представлен код, отвечающий за нахождение порта к которому есть подключение.

```
def serial_ports():
    if sys.platform.startswith('win'):
        ports = ['COM%s' % (i + 1) for i in range(256)]
    elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
        ports = glob.glob('/dev/tty[A-Za-z]*')
    elif sys.platform.startswith('darwin'):
        ports = glob.glob('/dev/tty.*')
    else:
        raise EnvironmentError('Unsupported platform')
    result = []
    for port in ports:
        try:
            s = serial.Serial(port)
            s.close()
            result.append(port)
        except (OSError, serial.SerialException):
            pass
    return result
```

Рисунок 35 – Алгоритм поиска связи через порты

Теперь если система действительно находит соединение, то считаем, что это светофор. И программа к нему присоединяется и прописывает функцию отправки сигнала, который был принят по результатам анализа потоков.

```
def connect(self,result):
  for 1 in range(len(result))
    try:
        self.realport = serial.Serial(self.Port(),int(self.Speed(9600)))
    except Exception as e:
        print(e)

def send(self):
    if self.realport:
        self.realport.send("+")
```

Рисунок 36 – Алгоритм соединения со светофором и отправка сигнала

На данном этапе можно сказать, что система готова, теперь стоит проверить точность работы

Вывод к главе 2

В данной главе были описаны основные принципы работы системы, пояснены основные цели и задачи технологии «умный светофор». Дана краткая характеристика каждому из типов светофоров, был логически разобран план проекта и реализован код системы с помощью нейронной сети на алгоритме YOLO.

3 Тестирование и оценка точности реализованного алгоритма для распознавания

Сначала стоит дать информацию об обучение нейронной сети. Как и любой процесс он имеет несколько параметров, которые задаются при создание нейронной сети или же создаются вследствие ввода значений. Это эпоха, пакет тренировочных материалов и итерация.

Пакетом обычно называют небольшой набор примеров тренировочных данных, на основании которых вносятся коррективы в обучаемую модель. Весь тренировочный набор делиться на несколько пакетов и по очереди подается в нейронную сеть.

Итерация обозначает то, сколько таких пакетов должно пройти через модель, чтобы закончилась одна эпоха .

Эпохой называется процесс полного прохода тренировочной модели. Эпоха заканчивается, когда обучаемая модель не видит больше тренировочные данные [16].

Нейронная сеть останавливает свое обучение, если не наблюдает ошибки или когда любая ошибка нейронной сети перестала уменьшаться и при коррекции коэффициентов уже наблюдается рост ошибки увеличение ошибки.

Обучение обычно проводят в два этапа: сначала подают тренировочный материал, который позволяет пройти обучение нейронной сети. На втором этапе рассматривается обучаемость сети уже на проверочных материалах. Можно говорить, что обучение нейронной сети было пройдено успешно, если нет ошибок на тренировочных материалах, так и на проверочных [9].

Если нейронная сеть показывает нужную точность только на тренировочном материале, тогда обычно идет этап переобучения нейронной сети.

Переобучение – излишнее соответствие нейронной сети выбранному набору тренировочного материла, при котором теряется способность к обобщению, это когда нейронная сеть не способна найти одинаковые по классу, но разные по изображению объекты при использование сети [20].

Возникает проблема переобучение в нескольких случаях:

- неправильная архитектура нейронной сети;
- слишком много параметров, которые корректирует нейронная сеть;
- слишком мало тренировочных данных, нейронная сеть не способна найти ассоциации.

Как и везде у переобучения есть полная противоположность это недообучение. То есть нейронная сеть сообщает, что она смогла найти все решения, но при этом становиться понятно, что она может улучшить свой результат. Это может случиться по ряду причин: если модель недостаточно сильная, либо слишком сильно регуляризована, все данные похожи друг на друга, либо просто недостаточно долго тренирована. Это означает, что сеть недоучилась соответствующим паттернам в тренировочных данных [21].

Обучение модели, входящей в состав YOLO не требуется, но тогда благодаря системе «Google Collab», мы проведем проверочные задания для обученной сети, выведем время обучения и ошибку. Проверочный материал подается в наборе из 100 картинок взятых из датасета «cats vs dogs» , интегрированного в Tensorflow [4]. Данный датасет взят так как его можно легко настроить как проверочный материал, а также в самом датасете дана информация для всех классов, что позволяет легко проверить алгоритм детектирования. На рисунке 37 изображено системные сообщения прохождения проверочного датасета, с которого будут и искать сравнения в точности детектирования алгоритма YOLO.

```
Эпоха: 1; Время: 1e+54c; Ошибка на проверке : 3.5428
Эпоха: 2; Время: 1e+54c; Ошибка на проверке : 3.3773
Эпоха: 3; Время: 1e+54c; Ошибка на проверке : 3.3125
Эпоха: 4; Время: 1e+54c; Ошибка на проверке : 3.2442
Эпоха: 5; Время: 1e+54c; Ошибка на проверке : 3.1569
Эпоха: 6; Время: 1e+54c; Ошибка на проверке : 3.0190
Эпоха: 7; Время: 1e+54c; Ошибка на проверке : 2.9935
Эпоха: 8; Время: 1e+54c; Ошибка на проверке : 2.8972
Эпоха: 9; Время: 1e+54c; Ошибка на проверке : 2.8484
Эпоха: 10; Время: 1e+54c; Ошибка на проверке : 2.7721
Эпоха: 11; Время: 1e+54c; Ошибка на проверке : 2.7438
Эпоха: 12; Время: 1e+54c; Ошибка на проверке : 2.7157
Эпоха: 13; Время: 1e+54c; Ошибка на проверке : 2.5904
Эпоха: 14; Время: 1e+54c; Ошибка на проверке : 2.5473
Эпоха: 15; Время: 1e+54c; Ошибка на проверке : 2.4568
Эпоха: 16; Время: 1e+54c; Ошибка на проверке : 2.3371
Эпоха: 17; Время: 1e+54c; Ошибка на проверке : 2.1743
Эпоха: 18; Время: 1e+54c; Ошибка на проверке : 1.9541
Эпоха: 19; Время: 1e+54c; Ошибка на проверке : 1.8891
Эпоха: 20; Время: 1e+54c; Ошибка на проверке : 1.7958
```

Рисунок 37 – Результаты обучения на 20 эпохах.

По рисунку 37 можно сделать вывод, что нейронная сеть каждый раз проходя эпоху, делает ошибку все меньше и меньше. Более наглядно уменьшение ошибки можно увидеть на рисунке 38.

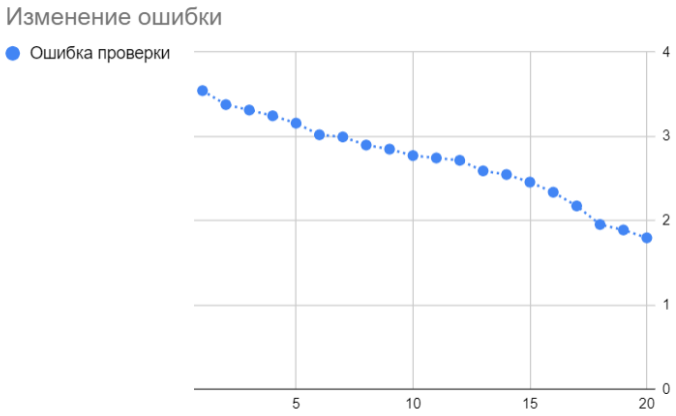

Рисунок 38 – Изменение ошибки на графике

Таким образом, можно наглядно видеть процент точности классификации локализованных объектов. Теперь перейдем к оценке точности самой локализации, найти можно по формуле метрики mAP(7):

$$class = \frac{\text{Количество корректного определения класса}}{\text{Общее количество объектов этого класса}} \quad (7)$$

Для вывода ошибки классификации на языке Python предусмотрен модуль sklearn.metrics, в которой есть функция precision_score(), которая рассчитывает перекрестную энтропию предсказанием нахождения класс и реальным объектом нахождения на изображении [13]. В ответе функции указывается количество изображений, на которых классифицирован хотя бы 1 объект деленный на все изображения датасета. А также указан процент правильных классификаций нейронной сети. Это прекрасно подойдет к работе данного датасета так как он представляет собой одиночные изображение котов и собак. На рисунке 39 можно видеть процентное соотношение нахождения объекта на изображении.

```
22906/23262
0.9846960708451552
```

Рисунок 39 – Результаты работы функции

Теперь возьмём данные с камеры и попробуем посмотреть какой процент классификации будет у программы считающей объекты с изображения в видеопотоке. Теперь просто берем полученным нами датасет с другой камеры и смотрим его точность, На рисунке 40 показан один из кадров из полученного датасета, а на рисунке 41 результаы работы метода sklearn.metrics. на нескольких различных классах.

Рисунок 40 – Пример изображения переданный на нейронную сеть

```
Car: err_class: 0.0331
Person: err_class: 0.0218
Ligths: err_class: 0.0336
Truck: err_class: 0.0258
Bus: err_class: 0.1421
Total: acc_class: 0.9487
```

Рисунок 41 – Результат нахождение классификации объекта

Таким образом были получены результаты локализации нейронной сети YOLO и качество классификации обнаруженных объектов. Оба показателя находяться в приделах 94-98 % детектирования объектов, что является хорошим результатом.

Вывод к главе 3.

Итак, в данном разделе был описан процесс создания проверочного набора данных с изображения в видеопотоке, с помощью которых было произведено обучение модели нейронной сети, входящей в состав алгоритма YOLO. И проведен анализ модели по метрике mAP. Наши тесты показали, что точность классификации объекта на изображение составляет 94.86 %, это можно назвать высоким уровнем классификации, так как средней возможностью распознавания объектов человеком составляет 98.84% [11].

Заключение

В данной выпускной квалификационной работе исследовался вопрос создание нейронной сети, которая могла осуществлять управление «умным светофором».

В процессе выполнения бакалаврской работы решены следующие задачи:

- изучены методы анализа данных, их задачи;
- изучен принцип работы и взаимодействия структуры нейронных сетей;
- разработан алгоритм, решающий практико-ориентированную задачу;
- протестировано разработанное решение;
- проведены тесты классификации объектов на изображении.

В первой части ВКР рассмотрены типы анализа данных, даны понятия структурированной информации и способы исправления деструктурированных значений. Рассмотрен алгоритм сверточных нейронных сетей метода обучения обратным распространением ошибки. Показаны крупные системы использующие нейронные сети для обработки информации.

Во второй главе дана информация о задачах светофорного регулирование проезжих частей, построено логическое описание нужной системы. Была реализована система, основанная на технологии «умный светофор». Для этого была использованная специальная модель нейронной сети типа YOLO.

В третей части были проведены проверочные тесты над модель нейронной сети типа YOLO, заранее обученной на датасете MS COCO, проведены тесты mAP, для нахождения точности в операции классификации.

Разработанное решение может получить дальнейшие развития, как пример, улучшение изображение применением систем R-CNN.

Список используемой литературы

1. Бурков А. Машинное обучение без лишних слов / Андрей Бурков - Питер СПб, 2020.
2. Вьюгин В. Математические основы машинного обучения и прогнозирования / Владимир Вьюгин. - МЦНМО, 2014.
3. Гелиг А., Матвеев А. Введение в математическую теорию обучаемых распознающих систем и нейронных сетей. Учебное пособие / Аркадий Гелиг, Алексей Матвеев - Издательство СПбГУ, 2014.
4. Документация по библиотеке машинного обучения TensorFlow 2.0 // TensorFlow [Электронный ресурс]: официальный сайт TensorFlow. URL: https://www.tensorflow.org/versions/r2.0/api_docs/python/.
5. Документация по языку программирования Python // Python [Электронный ресурс]: официальный сайт Python. URL: https://www.python.org/doc/.
6. Christian Szegedy. Scalable, High-Quality Object Detection, 2015 // arXiv [Электронный ресурс]: открытый архив научных статей. URL: https://arxiv.org/abs/1412.1441.
7. Ryo Takahashi. Data Augmentation using Random Image Cropping and Patching for Deep CNNs, 2019 // arXiv [Электронный ресурс]: открытый архив научных статей. URL: https://arxiv.org/abs/1811.09030 .
8. Siyuan Liang. Edge YOLO: Real-Time Intelligent Object Detection System Based on Edge-Cloud Cooperation in Autonomous Vehicles, 2022 // arXiv [Электронный ресурс]: открытый архив научных статей. URL: https://arxiv.org/abs/2205.14942.
9. Abu-Mostafa Y. Learning From Data / Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin – AMLBook. – 2012.-Jan.
10. Bishop C. Pattern Recognition and Machine Learning (Information Science and Statistics) / Christopher M. Bishop - Springer-Verlag New York Inc. - 2007.-Feb.

11. J. Stallkampa, M. Schlipsinga, J. Salmena C. Igelb Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition, 2012.

12. Eddison L. Python Machine Learning: A Technical Approach To Python Machine Learning For Beginners / Leonard Eddison - CreateSpace Independent Publishing Platform. – 2018.-Mar.

13. Geron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow / Aurelien Geron – O`Reilly Media. – 2017.-Mar.

14. Goodfellow I. Deep Learning (Adaptive Computation and Machine Learning series) / Ian Goodfellow, Yoshua Bengio, Aaron Courville - The MIT Press. -2016.-Nov.

15. Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists / Sarah Guido, August Mueller - O`Reilly Media. – 2016.-May.

16. Harrington P. Machine Learning in Action / Peter Harrington - Manning Publications. – 2012.-April.

17. Kelleher J. Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies / John D. Kelleher, Brian Mac Namee, Aoife D`Arcy - The MIT Press. – 2015.-July.

18. Mitchell T. Machine Learning / Tom Mitchell – Mc Graw Hill India. - 2017.– Mar.

19. Raschka S. Python Machine Learning / Sebastian Raschka - Packt Publishing. – 2015.-Sep.

20. Rashid T. Make Your Own Neural Network / Tariq Rashid - CreateSpace Independent Publishing Platform. – 2016.

21. Rojas R. Neural Networks: A Systematic Introduction / Raul Rojas, Peter Varga - Springer Berlin Heidelberg. – 1996.-Jul.

22. Russell S. Artificial Intelligence: Pearson New International Edition: A Modern Approach / Stuart Russel, Norvig Peter – Pearson. – 2013.-Aug.

23. Shalev-Shwarthz S. Understanding Machine Learning: From Theory to Algorithms / Shai Shalev-Shwartz, Shai Ben-David - Cambridge University Press. -2014.-May.

24. Shukla N. Machine Learning with TensorFlow / Nishant Shukla – Manning Publication. – 2018.-Jan.

25. Witten I. Data Mining: Practical Machine Learning Tools and Techniques / Ian H. Witten, Eibe Frank, Mark A. Hall - Morgan Kaufmann. – 2011.-Jan.