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

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.03.01 – Информатика и вычислительная техника | |
| **Профиль** | Организация и программирование вычислительных и информационных систем | |
| **Факультет** | КТИ | |
| **Кафедра** | ВТ | |
| *К защите допустить* |  | |
| **Зав. кафедрой**д. т. н., проф. |  | М. С. Куприянов |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

**Тема: Алгоритмы и программные модули для идентификации дорожных указателей с использованием искусственных нейронных сетей**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Д. Ш. Дашкин |
|  |  |  |  |
| Руководитель  к.т.н., доц. | |  | С. В. Костичев |
|  |  |  |  |
| Консультант | |  | Н. И. Олехова |
|  |  |  |  |
| Консультант | |  | М. Н. Гречухин |

Санкт-Петербург

2024

**Санкт-Петербургский государственный электротехнический университет**

**“ЛЭТИ” им. В. И. Ульянова (Ленина)**

**(СПбГЭТУ “ЛЭТИ”)**

|  |  |  |
| --- | --- | --- |
| **Направление** 09.03.01 – Информатика и  вычислительная техника  **Профиль** Организация и программирование  вычислительных и информационных систем  **Факультет** КТИ  **Кафедра** ВТ | **УТВЕРЖДАЮ**  Заведующий кафедрой ВТ  д. т. н., проф. | |
|  | М. С. Куприянов |
|  |
| « » 2024 г. | |

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент | Д. Ш. Дашкин | | |  | Группа | 0306 |
| **Тема работы**: Алгоритмы и программные модули для идентификации дорожных указателей с использованием искусственных нейронных сетей | | | | | | |
| **Место выполнения ВКР**: Кафедра ВТ | | | | | | |
| **Исходные данные**:  Исходными данными для разработки являются фотографии дорожных указателей, выложенные в открытый доступ, а также открытые наборы данных, предоставленные пользователями | | | | | | |
| **Содержание ВКР**:   1. Сравнение и анализ существующих методов для обнаружения объектов 2. Подготовка данных и модели к программной реализации 3. Программная реализация 4. Экономическое обоснование | | | | | | |
| **Дополнительные разделы**: Экономическое обоснование | | | | | | |
| **Перечень отчетных материалов**: Пояснительная записка, реферат, аннотация, презентация | | | | | | |
|  | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | |
| «25» февраля 2024 г. | | | «13» июня 2024 г. | | | |
|  | | |  | | | |
| Студент | |  | | | Д. Ш. Дашкин | | |
| Руководитель к.т.н., доц. | |  | | | С. В. Костичев | | |
|  | |  | | |  | | |

**КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ**

**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

|  |  |
| --- | --- |
| **Направление** 09.03.01 – Информатика и  вычислительная техника | Утверждаю |
| **Профиль** Организация и программирование  вычислительных и информационных систем | Зав. кафедрой ВТ  д.т.н., проф. |
| **Факультет** КТИ | \_\_\_\_\_\_\_\_\_\_\_\_М. С. Куприянов |
| **Кафедра** ВТ | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Д. Ш. Дашкин |  | Группа | 0306 |
| Тема работы: Алгоритмы и программные модули для идентификации дорожных указателей с использованием искусственных нейронных сетей | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Освоение материалов, необходимых для выполнения выпускной квалификационной работы | 01.03 – 26.03 |
| 2 | Обзор существующих методов распознавания объектов | 27.03 – 02.04 |
| 3 | Сбор данных | 03.04 – 26.04 |
| 4 | Обработка и разметка собранных данных | 27.04 – 02.05 |
| 3 | Выбор необходимой модели для программной реализации | 03.05 – 12.05 |
| 4 | Программная реализация для обучения выбранной модели нейронной сети | 13.05 – 20.05 |
| 5 | Анализ метрик, выявление особенностей применения | 21.05 – 23.05 |
| 6 | Проведение тестирования модели | 24.05 – 27.05 |
| 7 | Реализация дополнительного программного модуля | 28.05 – 31.05 |
| 8 | Оформление пояснительной записки | 01.06 – 12.06 |
| 9 | Представление работы к защите | 13.06 |

|  |  |  |
| --- | --- | --- |
| Студент |  | Д. Ш. Дашкин |
| Руководитель к.т.н., доц. |  | С. В. Костичев |

**РЕФЕРАТ**

Пояснительная записка к выпускной квалификационной работе включает … страниц текста, … рисунка, … таблицы и … использованных источников.

Цель данной выпускной квалификационной работы заключается в реализации алгоритмов и их дальнейшего внедрения в программные модули для последующего распознавания дорожных указателей при помощи искусственных нейронных сетей.

Объект работы – идентификация дорожных указателей

Для практической реализации программных модулей был выбран язык Python и интегрированная среда разработки Google Colab. Была выбрана определенная модель нейронной сети. Также был создан набор данных из … изображений для обучения модели. Разработан и протестирован программный модуль.

В результате работы была обучена модель искусственной нейронной сети, протестирована на наборе данных и внедрена в программный модуль. Разработанная программа в дальнейшем при ее развитии может быть внедрена в различные системы по управлению автотранспортными средствами и в системы помощи водителю.

**ABSTRACT**

The explanatory note to the graduation work includes ... pages of text, ... a drawing, ... a table and ... used sources.

The purpose of this final qualification work is to implement algorithms and their further implementation into software modules for subsequent recognition of road signs using artificial neural networks.

The object of the work is the identification of road signs

For the practical implementation of software modules, the Python language and the Google Colab integrated development environment were chosen. A specific neural network model was chosen. A dataset of ... images was also created to train the model. A software module has been developed and tested.

As a result of the work, an artificial neural network model was trained, tested on a dataset and implemented into a software module. The developed program in the future, with its development, can be implemented into various vehicle management systems and driver assistance systems.

**СОДЕРЖАНИЕ**

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ: 5](#_Toc167878661)

[ВВЕДЕНИЕ 6](#_Toc167878662)

[1 Обзор существующих методов для обнаружения объектов 9](#_Toc167878663)

[1.1 Сравнение существующих моделей детекции объектов 10](#_Toc167878664)

[1.1.1 Семейство сверточных нейронных сетей 12](#_Toc167878665)

[1.1.2 R-CNN 13](#_Toc167878666)

[1.1.3 FAST R-CNN 14](#_Toc167878667)

[1.1.4 Faster R-CNN 15](#_Toc167878668)

[1.1.5 Семейство YOLO 17](#_Toc167878669)

[2 Подготовка данных и модели к программной реализации 22](#_Toc167878670)

[2.1 Предварительная обработка собранных данных 22](#_Toc167878671)

[2.2 Выбор модели YOLOv8 27](#_Toc167878672)

[2.3 Выводы по разделу 28](#_Toc167878673)

[3 Программная реализация программы распознавания дорожных знаков 29](#_Toc167878674)

[3.1 Реализация программы идентификации дорожных указателей 30](#_Toc167878675)

[3.2 Анализ полученных результатов 31](#_Toc167878676)

[3.2.1 Метрики оценки эффективности работы нейронной сети 31](#_Toc167878677)

[3.2.2 Оценка эффективности работы нашей модели 41](#_Toc167878678)

[3.2.3 Создание программного модуля 43](#_Toc167878679)

[3.2.4 Проверка обученной нейронной сети на тестовом наборе 45](#_Toc167878680)

[4 Экономическое обоснование 49](#_Toc167878681)

[4.1 План выполнения разработки 49](#_Toc167878682)

[4.2 Расчет заработной платы разработчика 50](#_Toc167878683)

[4.3 Затраты на сырье и материалы 53](#_Toc167878684)

[4.4 Затраты на содержание и эксплуатацию оборудования 54](#_Toc167878685)

[4.5 Услуги сторонних организаций 55](#_Toc167878686)

[4.6 Амортизационные отчисления 55](#_Toc167878687)

[4.7 Накладные расходы 56](#_Toc167878688)

[4.8 Совокупная величина затрат 56](#_Toc167878689)

[4.9 Заключение 57](#_Toc167878690)

[ЗАКЛЮЧЕНИЕ 58](#_Toc167878691)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 59](#_Toc167878692)

[ПРИЛОЖЕНИЕ 61](#_Toc167878693)

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ:

**CNN** (сокр. от Convolutional Neural Network) – Сверточная нейронная сеть

**CPU** (сокр. от Central Processing Unit) – центральный процессор

**ML** – Машинное обучение

**Python –** Высокоуровневый язык программирования

**RoI** (сокр. от Region of Interest) – Область интереса – Часть основного изображения, выделяемая пользователем, как регион с повышенным интересом

**SoftMax** – функция, превращающая наборы чисел в вероятности, причем сумма последних равна единице

**SVM** (сокр. от Support Vector Machine) – Метод опорных векторов – набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа

**ИИ** – Искусственный интеллект

**НС** - Нейронная сеть

**СНС** - Сверточная нейронная сеть

**Фича –** Характеристика, атрибут, признаки, которые полезны для обучения и получения результатов ML-модели

# ВВЕДЕНИЕ

Надежная система обнаружения и распознавания дорожных знаков является важной задачей для автономных транспортных средств из-за повышения безопасности, эффективности, производительности в режиме реального времени и надежности: дымка, туман и другие атмосферные условия могут ухудшить видимость на дороге, затрудняя водителям и автономным транспортным средствам обнаружение препятствий и безопасное передвижение. Точное определение дорожных знаков повышает безопасность автомобиля. Точное определение дорожных знаков может помочь автономным транспортным средствам эффективно ориентироваться, предоставляя информацию об ограничении скорости, дорожных условиях и других важных деталях. Это может помочь сократить время в пути, оптимизировать маршруты и повысить топливную экономичность. Автономные транспортные средства работают в режиме реального времени; следовательно, алгоритмы обнаружения и распознавания дорожных знаков также должны работать в режиме реального времени. Модель машинного обучения может обеспечить производительность в режиме реального времени, используя оптимизированную облегченную архитектуру и эффективные алгоритмы. Автономные транспортные средства работают в различных условиях окружающей среды, включая освещение, погоду и видимость. Обнаружение дорожных знаков может улучшить работу транспортного средства в различных сценариях, что приводит к более безопасному и надежному автономному вождению. Существуют системы помощи водителю, например, ADAS (Advance Driver Assistance System), которая является помощником водителей в управлении транспортным средством путем предоставления информации, подсказок и напоминаний. В настоящее время система ADAS предлагает такие функции и цели, как:

- Система помощи при парковке ADAS уведомляет водителя о расстоянии до ближайшего объекта, что упрощает парковку.

- Технология удержания полосы движения ADAS не позволяет транспортному средству отклоняться от своей полосы движения, предупреждая или корректируя при необходимости направление движения.

- Система дистанционного управления ADAS позволяет водителю использовать компьютер и другие устройства для управления транспортным средством.

- Система ADAS следит за пешеходами и предупреждает водителя в случае опасности.

- Технология предупреждения о столкновении ADAS уведомляет водителя о возможности столкновения и при необходимости тормозит автомобиль, чтобы предотвратить его.

Однако, как системы помощи водителю, так и технологии, обеспечивающие работу автономных транспортных средств, зависят от того, как люди читают и понимают дорожные знаки. Одним из важнейших критериев безопасного и широкого использования этой технологии является алгоритм TSDR (Traffic Sign Detection and Recognition), который является точным и надежным в различных реальных ситуациях. Это одно из самых фундаментальных условий, которое должно быть выполнено.

Машинное обучение и глубокое обучение в последнее время привлекли большое внимание из-за высокой точности, которую они смогли достичь по сравнению с другими методами. В основе большинства методов машинного и глубокого обучения лежат нейронные сети. Возможности искусственного интеллекта решать самые сложные и сильно нелинейные задачи с высокой скоростью, умение приспосабливаться к изменениям в окружающей среде, их устойчивость перед шумами на входных данных – всё это несомненно выделяет их перед классическими алгоритмами.

Цель работы – автоматизировать процесс распознавания дорожных знаков. Для достижения цели необходимо решить следующие задачи:

1. провести обзор методов, которые используются при распознавании объектов и изучить их;
2. провести сравнительный анализ искусственных нейронный сетей для распознавания образов;
3. сформировать набор данных;
4. разметить сформированный набор данных;
5. выбрать модель НС;
6. обучить выбранную модель;
7. провести аналитический разбор полученных результатов тренировки НС;
8. провести тестирование разработанной НС;
9. разработать приложение с графическим интерфейсом пользователя;
10. протестировать разработанное приложение.

## 1 Обзор существующих методов для обнаружения объектов

Задача распознавания дорожных знаков достаточно известна. Это связано с тем, что, во-первых, дорожные знаки играют важнейшую роль в регулировании дорожного движения, обеспечении безопасности водителей и пешеходов, а также в задачах навигации и поиска маршрута; а во-вторых, дорожные знаки устроены таким образом, чтобы облегчить их идентификацию. Распознавание знаков обычно осуществляется в два этапа: обнаружение знака и его собственное распознавание. Дорожные знаки имеют определенные цвета и формы и обычно видны на дороге издалека. Как правило, известные алгоритмы и программы обнаружения признаков учитывают эти особенности. Входной сигнал (чаще всего видео с установленной на нем камеры) проходит ряд преобразований с целью выделения областей правильной формы и содержащих характерные для персонажа цвета.

Известно, что люди пытаются научить компьютер «видеть» еще с середин 20-го века. Но действительно серьезного прогресса получилось добиться лишь в последнее время. На это есть ряд причин: увеличение мощностей вычислительных устройств, повышение качества изображения с камер, развитие и стремительный прогресс в машинном и глубоком обучении, в искусственном интеллекте.

Количество научных статей и других публикаций в области обнаружения объектов на изображениях за период с 1998 по 2018 год увеличилось в 60 раз и к текущему моменту лишь продолжает свой незамедлительный рост. На рисунке 1 изображена диаграмма активности [1].



Рисунок 1 ­- Изучение активности научных публикаций

Таким образом, тема обнаружения и распознавания объектов на сегодняшний момент является более чем актуальной и занимает одну из ведущих ролей в IT-сфере.

1.1 Сравнение существующих моделей детекции объектов

Выше было сказано, что задача обнаружения дорожных знаков существует уже очень много лет. За это время были изобретены различные алгоритмы и инструменты, помогающие решить эту задачу.

Всего лишь около пары десятилетий назад распознавание скатывалось к ручному проектированию фичей, так как в то время не существовало совершенных и современных моделей, способных запросто решать данную задачу.

Самыми известными методами того времени принято считать:

1. Метод Виолы-Джонса
2. Метод HOG (гистограмма градиентов)

Метод Виолы-Джонса довольно известен как метод для распознавания лиц в режиме реального времени. Его основное преимущество в этих задачах заключалось в том, что этот метод работает на признаках Хаара. Но в то же время, это было и минусом, ведь применение метода Виолы-Джонса было ограничено только конкретными объектами.

Метод гистограммы градиентов в свою очередь был очень востребован в задачах обнаружения пешеходов. Метод HOG помогает описать форму объекта на изображении используя распределение градиентов интенсивности или направлением краев [1]. Однако данный метод показывает свою уязвимость при масштабировании количества объектов, например, дорожных знаков, как в нашем случае.

Хоть эти методы и были поистине революционны в то время, но их недостатки, такие как зависимость от фичей, которые приходилось создавать вручную, ограниченная масштабируемость, затрудняло их использование в реальной жизни.

Но прежде, чем продолжить, стоит отмотать еще немного назад к 80-ым годам 20-го века. В то время искусственный интеллект пытался решать различные вычислительные задачки, действуя подобно когнитивному механизму человеческого сознания. Искусственная нейронная сеть, одна из главных представительниц ИИ, представляла из себя целую сетевую систему, состоящую из огромного множества нейронных вершин, фактически “подражая” сетевой системе человеческого мозга. Однако, ИНС не всегда могла удовлетворить требованиям в области обработки изображений. По причине этого сверточная нейронная сеть и была изобретена. СНС, являющаяся по сути одной из составляющих ИНС, работает на механизме создания естественного зрения (извлечения признаков) и может за счет этого механизма легко признать правило распознавания непосредственно по исходным пикселям всего лишь с небольшой предварительной обработкой. В 90-ых годах прошлого века были окончательно внедрены современные структуры СНС с применением алгоритма обратного распространения ошибки, получившего свою популярность в 1986 за счет публикаций Дэвида Румельхарта, Джеффри Хинтона и Рональда Вильямса. Однако отсутствие возможности сбора больших тренировочных данных, а также ограниченные на то время вычислительные мощности компьютеров заставили ученых отложить изучение нейронных сетей вплоть до 2000-ых, когда вычислительные возможности компьютеров стремительно скакнули вверх.

В 2006 году Джеффри Хинтон предложил глубокое обучение. Его развитие было глотком свежего воздуха в задаче обнаружения объектов и стало действительно чем-то инновационным, поскольку помогало преодолеть ограничения, с которыми сталкивались традиционные методы.

Текущие подходы с использованием методов глубокого обучения для задач классификации и регрессии объектов можно разделить на две категории [2]. Поэтому прежде, чем мы сможем выбрать подходящую для нашей задачи архитектуру, мы должны рассмотреть обе категории более детально. Первая – это двухэтапные методы, которые представлены такими архитектурами как R-CNN, Fast R-CNN и Faster R-CNN. Данные методы выполняются в два этапа.

1.1.1 Семейство сверточных нейронных сетей

В данном разделе будем рассмотрено семейство сверточных нейронных сетей, которые относятся к первой категории методов. Все они являются последовательными модификациями друг друга. Их исходная модель СНС рассматриваться здесь не будет, так как в современном мире ее роль практически не актуальна.

1.1.2 R-CNN

Данная архитектура была предложена в 2014 году и стало первым улучшением существовавшего на тот момент метода СНС для распознавания изображений. R-CNN может выбрать большое количество ограничивающих рамок, которые могут быть конечными объектами, с помощью метода выборочного поиска, а затем разделить различные независимые области изображения, используя извлечение признаков CNN. Вообще процесс работы R-CNN состоит из четырех этапов:

* Ввод изображения
* Извлечение предложений по регионам (Region Proposals)
* Вычисление признаков СНС
* Классификация регионов

Шаблон алгоритма R-CNN может быть отнесен к нескольким процессам. Во-первых, CNN можно обучить, классифицируя изображения, а целевые области можно искать с помощью выборочного поиска. Кроме того, эти целевые области требуют изменения размера до размера по умолчанию. Во-вторых, для каждой области изображения генерируется вектор признаков посредством прямого распространения CNN, а затем этот вектор признаков необходимо ввести в двоичный SVM. Регрессионная модель используется для уменьшения ошибок позиционирования и исправления ограничительной области. Подводя итог, CNN, SVM и регрессионная модель включены в R-CNN.

R-CNN была поистине прорывной в задачах детектирования объектов, обладая рядом преимуществ, таких как меньшая частота ошибок за счет разделения изображений на отдельные области, возможность автоматического извлечения признаков изображения. Но при этом, этого всё равно было недостаточно для задачи детектирования объектов в режиме реального времени. Извлечение признаков данной архитектурой могло занимать огромное количество физической памяти, ведь для каждого регионального предложения приходилось создавать файлы с признаками. Но главный минус заключался в том, что одно и то же изображение приходилось прогонять по многу раз, ведь каждое региональное предложение приходилось запускать в классификаторе. Время работы R-CNN могло занимать сотни, а то и тысячи секунд, поэтому улучшения последовали незамедлительно.

1.1.3 FAST R-CNN

Данная архитектура была предложена в 2015, спустя год после предыдущей предложенной. Она решала основную проблему предыдущей архитектура, а именно запуск СНС для каждого предложенного выборочным поиском региона. Теперь независимые векторы признаков не задаются для каждого объекта, а наоборот каждое изображение проходит через СНС и на выходе получает общую матрицу признаков. Лишь потом запускается алгоритм выборочного поиска. Затем полученные ограничительные области приводятся к единому размеру и перебрасываются на финальный слой RoI (Region of Interest) Pooling. Fast R-CNN заменяют последний слой полного подключения и последний SoftMax k-го класса на слой полного подключения и SoftMax k+1-го класса. Наконец, Fast R-CNN выводит дискретное распределение вероятностей для каждого RoI и прогнозирует регрессионную модель с ограничивающим прямоугольником смещений относительно исходного RoI для каждого k-го класса.

Конечно, эта архитектура заметно увеличила скорость работы по сравнению с предыдущей версией R-CNN, за счет одиночного запуска СНС. Помимо этого, основными плюсами этой модели выступили многоступенчатое обучение, замена SVM-классификатора на SoftMax для задачи многоклассового прогнозирования, поддержка масштабных входных данных за счет распределения веса сверточной сети и использования объединения RoI. Однако, всё еще скорости модели не хватало для выполнения обнаружения объектов в режиме реального времени, ведь предложение по регионам с помощью регионального признака могло запускаться только в CPU, что значительно увеличивало время работы программы.

1.1.4 Faster R-CNN

Существенный прорыв случился в 2016 году, когда была представлена новая модель Faster R-CNN. В общем и целом, многие шаги в этой модели такие же, как и у ее предшественницы. Правда теперь для предложения ограничительных областей используется сквозная региональная сеть предположений (Region Proposal Network, RPN), использующая предварительно обученную классификацию изображений. Вообще, Fast R-CNN используется для инициализации обучения региональной сети предположений. При сохранении общего сверточного слоя происходит настройка только слоя RPN. На этом этапе слой RPN и слой сети обнаружения объединены. Архитектура представлена на рисунке 2.



Рисунок 2 - Архитектура модели Faster R-CNN [2]

Стоит отметить, что уникальность модели Faster R-CNN заключается в том, что она может создавать единую модель, состоящую из сети региональных предположений RPN, в которую интегрированы распределения региональных предположений и Fast R-CNN с общими сверточными слоями признаков. Общая архитектура Faster R-CNN представлена на рисунке 8.

Faster R-CNN, как наиболее совершенная модель в данном семействе, обладает рядом уникальных преимуществ. Как уже упоминалось ранее, благодаря оптимизации генерации области предположения, данная модель реализует сквозное обучение. Использование сети региональных предположений помогло добиться обнаружения объектов практически в режиме реального времени с высокой точностью, позволив сети изучать предложения по регионам. Однако предложения объектов требовало много времени, а производительность системы сильно зависела от производительности предыдущих систем. Избыточность на последующих этапах обнаружения вызывала беспокойство.

В 2017 году была предложена сверточная нейронная сеть Feature Pyramid Networks (FPN). Если детально разобраться в Faster-RCNN, то можно увидеть, что он в основном не способен улавливать мелкие объекты на изображении. Чтобы решить эту проблему, можно использовать простую графическую пирамиду для масштабирования изображения до разных размеров и отправки его в сеть.

Весь вышеперечисленный алгоритм распознает объекты только для локальной области изображения и не работает на полном изображении.

К другой категории относятся одноэтапные алгоритмы, представленные различными версиями YOLO и др. [2]. Данные алгоритмы не используют отдельную сеть для генерации регионов и основываются на методах регрессии, просматривая изображения целиком.

1.1.5 Семейство YOLO

Концептуально новой моделью нейронной сети для обнаружения объектов стала разработанная в 2015 году модель YOLO («You only look once»). Данная нейронная сеть представляет из себя одноступенчатый алгоритм, который использует сверточную нейронную сеть путем преобразования задачи обнаружения в задачу регрессии. Пример работы представлен на рисунке 3.

Первая версия YOLO, YOLOv1 – это система обнаружения объектов, основная идея которой заключалась в создании СНС для предсказания многомерных массивов (тензоров). Она может делить изображение на разные блоки для идентификации объектов для обработки изображения в режиме реального времени.

YOLOv1 обрабатывает логический вывод, чтобы получить положение всех объектов на одном изображении, их категории и соответствующую доверительную вероятность. Таким образом, он может лучше идентифицировать фон, имея полное представление о нем. Более того, каждая ячейка сетки предсказывает ограничивающие прямоугольники, так что каждая ячейка сетки имеет показатель достоверности для прямоугольника, который обнаруживает один объект независимо от количества прямоугольников и предсказывает вероятность условных классов. Каждая ограничивающая рамка включает в себя 5 значений: координата центра по оси абсцисс, координата центра по оси ординат, высота рамки, ширина рамки и значение достоверности [2].

Плюсы YOLOv1 по сравнению с семейством CNN, рассмотренном в предыдущем разделе существенно заметны. Во-первых, метод YOLOv1 может использоваться для сквозного обучения, так как предсказания, включая местоположение объектов и их классы, состоят из единой сети. Во-вторых, YOLOv1 более универсален. Он превосходит другие методы при переходе от естественных изображений к другим областям, таким как искусство. Наконец, метод определения региона ограничивает классификаторы конкретными областями. YOLOv1 получает доступ ко всему изображению, прогнозируя границы. Благодаря дополнительному контексту YOLOv1 показывает меньше ложных срабатываний (False Positive) в фоновой области.

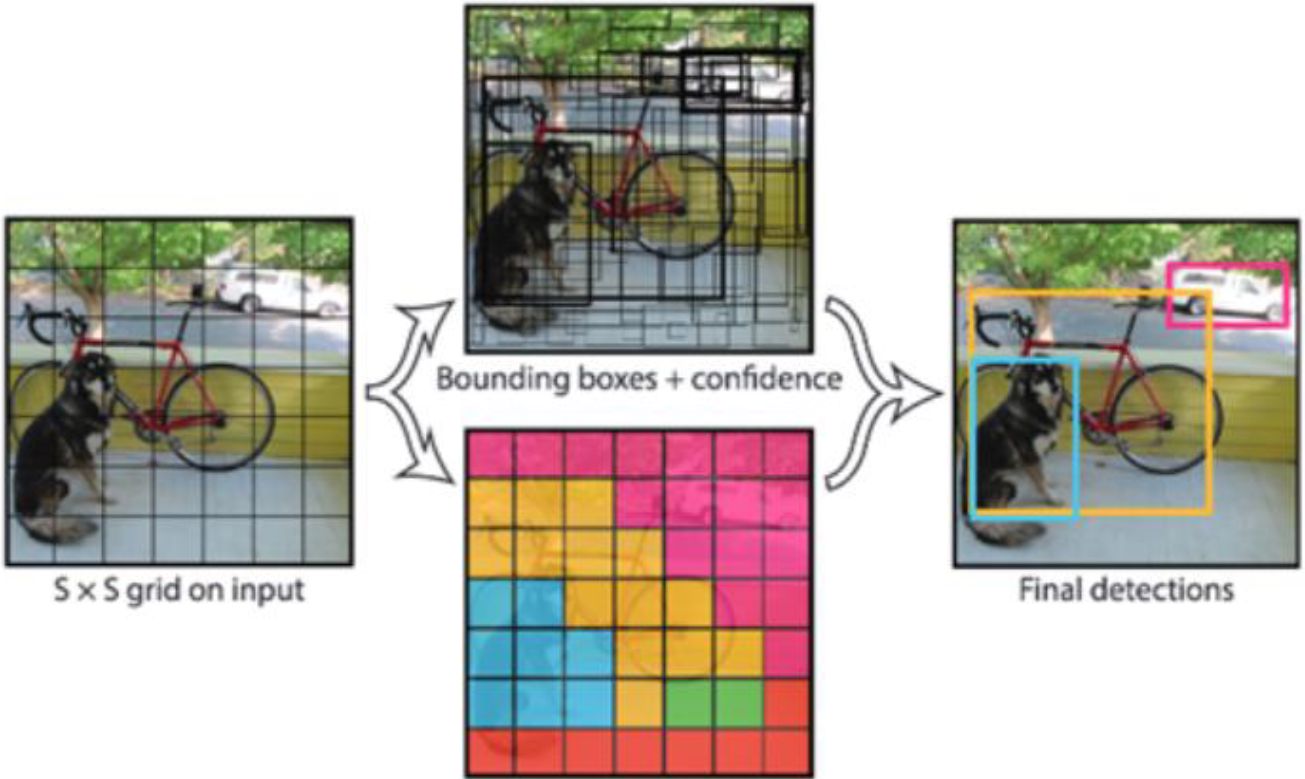


Рисунок 3 - Пример работы YOLO

Впоследствии для улучшения YOLOv1 был создан YOLOv2, который состоит из двух этапов (категории объектов и местоположения объектов). За счет того, что YOLOv2 использует изображения для тренировки классификаторов и использует изображения для обнаружения объектов, удалось увеличить среднюю точность и упростить тренировку детектора. Добавление в эту версию алгоритма пакетной нормализации и сверточных слоев заметно увеличило скорость и точность работы метода YOLO.

Затем была изобретена новая версия YOLOV2, YOLOv3, которая может использовать логистическую регрессию для прогнозирования оценки объективности каждого ограничивающего прямоугольника, а также изменяет способ расчета функций затрат. YOLOv3 использует отдельный логический классификатор вместо функции SoftMax для вычисления вероятности принадлежности входных данных к определенной метке. При расчете потери при классификации YOLOv3 не использует среднеквадратичную ошибку, а использует двоичную перекрестную потерю энтропии для каждой метки.

YOLOv5 – самая облегченная версия из предыдущих, которая использует фреймворк PyTorch вместо Darknet [2]. Также в нее был добавлен новый слой фокуса (англ. Focus layer), который заменил первые три слоя магистральной сети YOLOv3, что позволило увеличить скорость при минимальных потерях в точности.

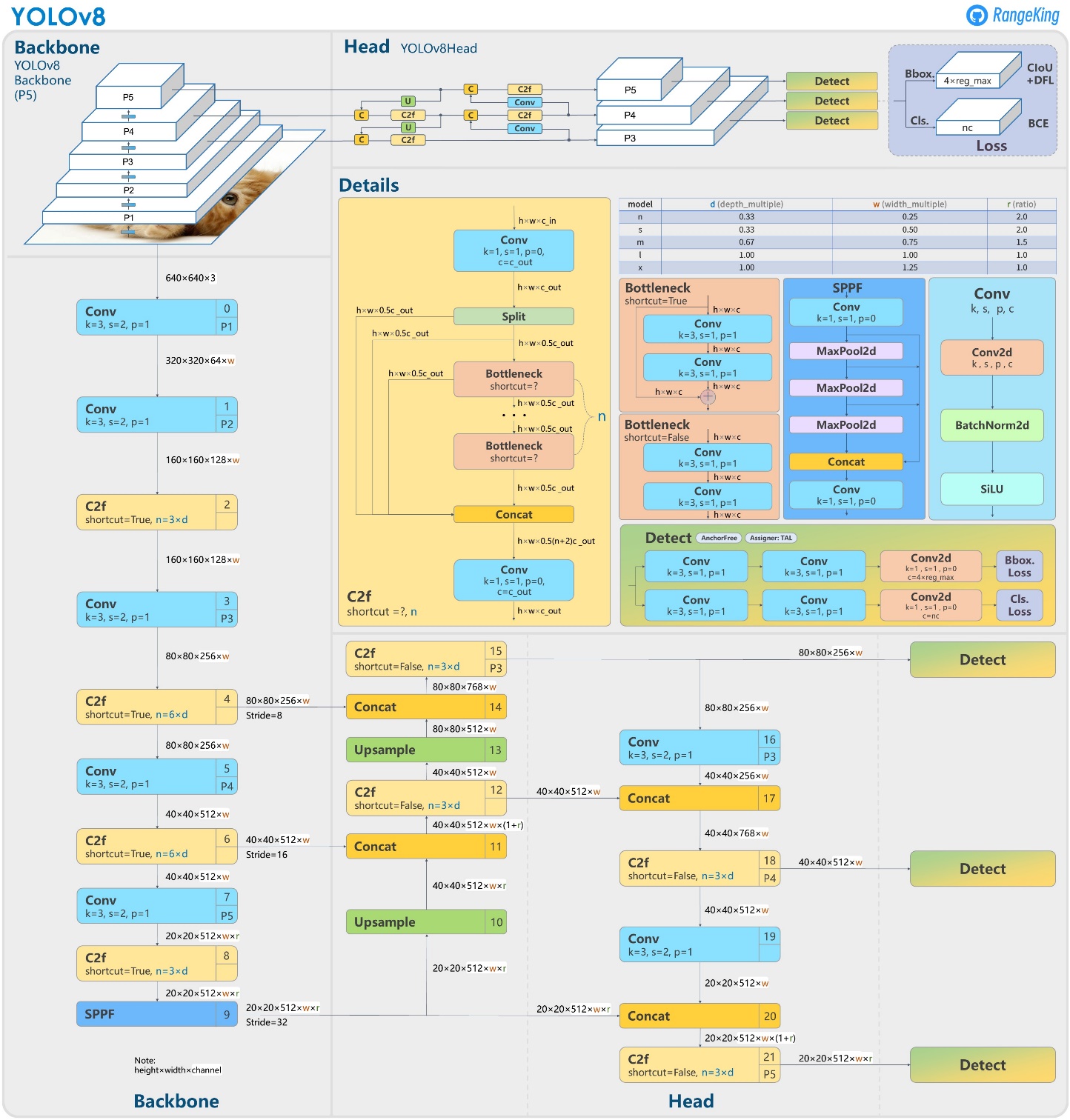
Наконец, вершиной творения семейства YOLO на сегодняшний день является YOLOv8 – наиболее усовершенствованная версия. Ее скорость и средняя точность по сравнению с предыдущими моделями показывают лучшие результаты [3][4]. Ее новыми особенностями стали:

* Новая магистральная сеть
* Конструкция для сравнения производительности с предыдущими

моделями YOLO

* Новая функция потерь
* Новый подход к обнаружению без привязки (anchor-free detection)

На рисунке 4 можно детально рассмотреть архитектуру YOLOv8, которая представлена в одном из github-репозиториев [5].

Рисунок 4 - Архитектура YOLOv8 [5]

Основное преимущество данной архитектуры является ее проход по изображению за один подход, в то время как зачастую в других моделях (R-CNN, Fast R-CNN, Faster R-CNN и др.) СНС применяются множество раз к разным ограничительным областям одного и того же изображения. За счет этого, скорость работы YOLO существенно выше, чем у таких моделей. В статье [2] приводится сравнение скорости работы модели YOLOv5 и последней версии семейства CNN Faster R-CNN. По результатам этого исследования, скорость YOLOv5 оказалась выше, чем у Faster R-CNN в задаче детектирования объектов. Исходя из упомянутого мною выше, YOLOv8, как самая последняя версия семейства YOLO, обладает наибольшей скоростью и точностью из всех ее предшественниц. Для задачи распознавания дорожных знаков точность и скорость работы системы невероятно важны, так как зачастую эта задача требует решения в реальном времени. Исходя из областей интегрирования распознавания дорожных знаков, о которых упоминалось в начале данной работы, точность работы модели играет не меньшую роль, ведь от этого напрямую зависит поведение автотранспортного средства и безопасность окружающих.

В данной работе мною не будет создана новая искусственная нейронная сеть. Готовая модель YOLOv8 служит готовым механизмом для распознавания дорожных указателей.

## 2 Подготовка данных и модели к программной реализации

Прежде чем переходить к непосредственно кодовой реализации нашей задачи, необходимо провести предварительную обработку данных и углубленно проанализировать выбранную архитектуру YOLOv8.

2.1 Предварительная обработка собранных данных

Собранный набор данных включает в себя 15 различных классов изображений дорожных указателей. Всего в наборе данных представлено 4969 изображений. Часть из этих изображений была собрана мной, а часть из них была взята из уже готовых датасетов. Ниже перечислим классы дорожных указателей, из которых и состоит датасет:

* Зеленый свет светофора
* Красный свет светофора
* Знак «Стоп»
* Знак «Ограничение максимальной скорости 10 км/ч»
* Знак «Ограничение максимальной скорости 100 км/ч»
* Знак «Ограничение максимальной скорости 110 км/ч»
* Знак «Ограничение максимальной скорости 120 км/ч»
* Знак «Ограничение максимальной скорости 20 км/ч»
* Знак «Ограничение максимальной скорости 30 км/ч»
* Знак «Ограничение максимальной скорости 40 км/ч»
* Знак «Ограничение максимальной скорости 50 км/ч»
* Знак «Ограничение максимальной скорости 60 км/ч»
* Знак «Ограничение максимальной скорости 70 км/ч»
* Знак «Ограничение максимальной скорости 80 км/ч»
* Знак «Ограничение максимальной скорости 90 км/ч»

Далее необходимо разметить наши данные. Разметка данных нужна для помощи в выявлении логических закономерностей, структурированию данных. Чем лучше будет произведена разметка данных, тем лучше пройдет обучение НС. Данный метод называется обучением с учителем, так как мы уже задаем правильные ответы, тем самым представляем нашей модели некий ориентир, по которому она может выявлять признаки и тем самым «обучаться».

Для разметки данных мною был выбран сайт Roboflow [6], который является одним из ведущих сайтов по предоставлению наборов данных для задач компьютерного зрения.

Для начала были созданы 15 классов, которыми потом мы будем давать метки нашим изображениям. Процесс создания метки класса представлен на рисунке 5.



Рисунок 5 - Создания меток класса

После создания классов мы непосредственно переходим к разметке набора данных. Процесс разметки запечатлен на рисунке 6. С помощью боковой панели инструментов справа и присвоения класса в графе слева происходит формирование ограничительной области и выдача ей соответствующей метки класса.



Рисунок 6 - Разметка изображения

Также все изображения стоит привести к единому размеру. Оптимальным размером было выбрано разрешение 416x416.

После разметки всех изображений мы можем экспортировать наш датасет. Большой плюс используемого сайта Roboflow заключается в предоставлении различных форматов экспорта данных. В современном мире два наиболее используемых формата хранения данных: COCO [7] и YOLO [8]. Процесс экспортирования датасета представлен на рисунке 7.

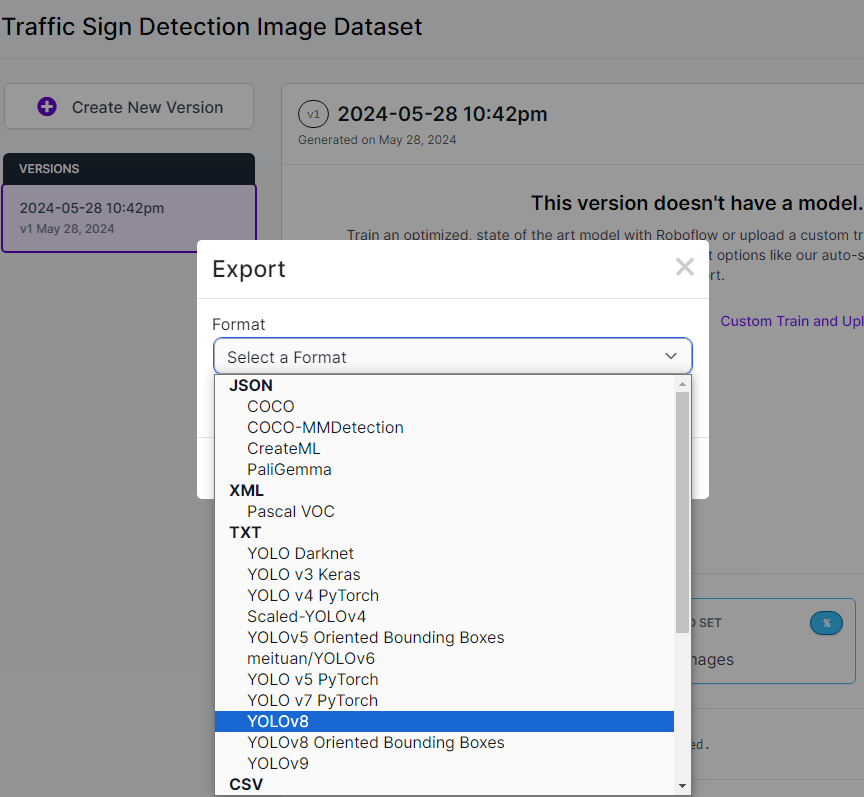


Рисунок 7 - Экспорт датасета

После эскпортирования данных в виде ZIP-архива и последующей его распаковки на компьютер мы получаем древовидную структуру каталога с датасетом (см. рисунок 8).

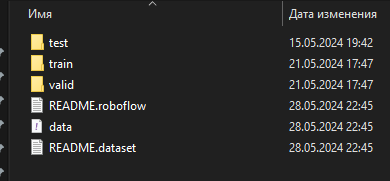


Рисунок 8 - Структура датасета

С сайта нам идет три папки для трех типов данных: тренировочных, валидационных и тестовых. Train, valid и test папки соответственно. Помимо папок в ZIP-архиве лежит файл data.yaml. В этом файле указываются пути до тренировочного, валидационного и тестового наборов данных, а также количество меток и их классы. Его структура представлена на рисунке 9.

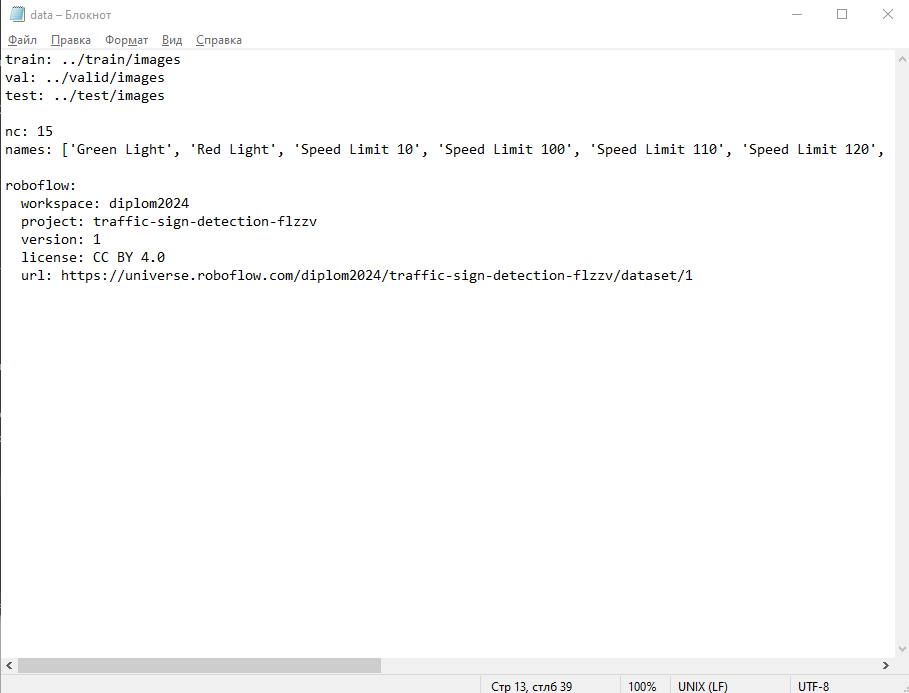


Рисунок 9 - Структура файла data.yaml

Подробнее рассмотрим папки в каталоге. Все они имеют одинаковую структуру, представленную на рисунке 10.

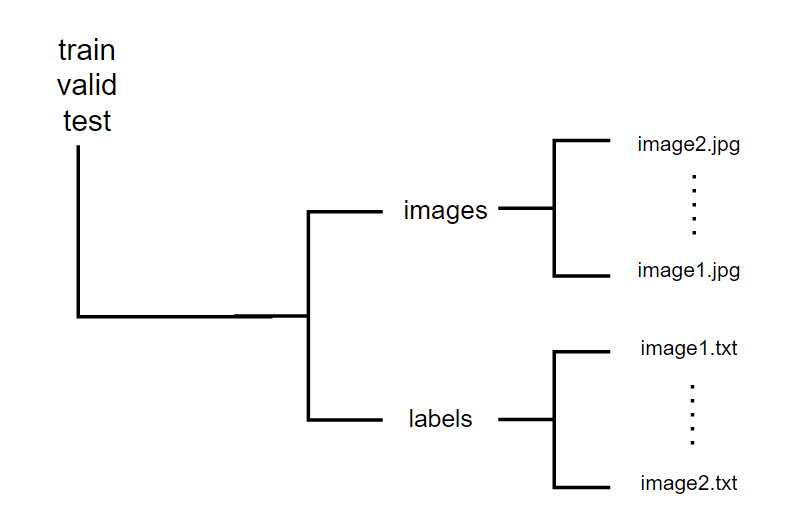


Рисунок 10 - Структура папок train, valid, test

В папке images лежат наши изображения. В папке labels лежат аннотации к нашим изображениям в виде файла в формате txt с таким же названием, как и у изображения в папке images, к которому он прилагается. Рассмотрим, что из себя этот файл представляет (см. рисунок 11):

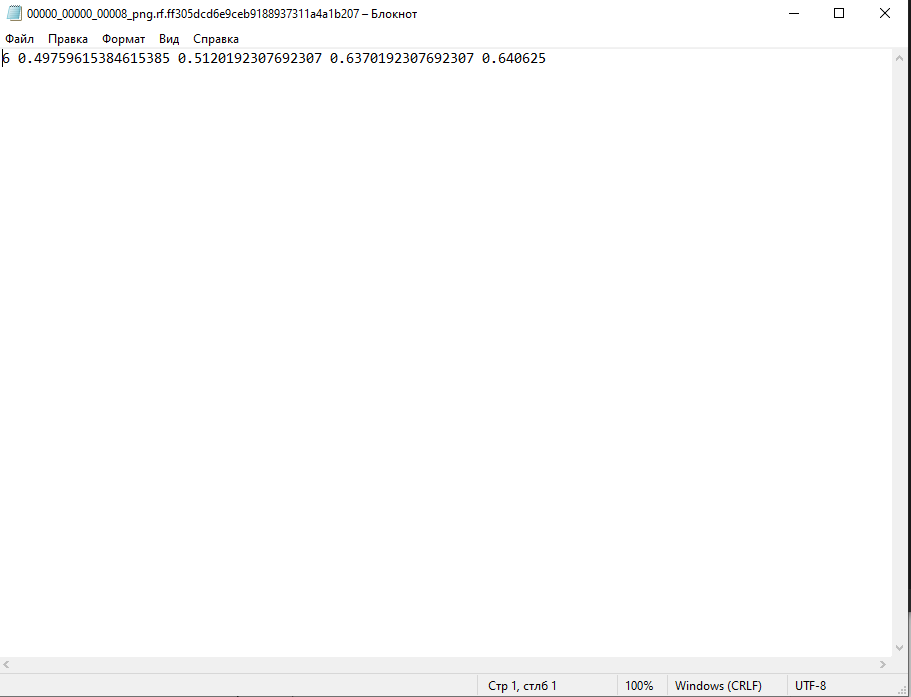


Рисунок 11 - Файл аннотации

Данный формат хранения аннотации называется аннотацией к детекции (bbox). Когда мы размечали датасет, мы выделяли дорожный указатель с помощью прямоугольника (ограничивающей области). Значения в данном текстовом файле по порядку слева-направо имеют следующий смысл:

* 1. Номер класса знака на соответствующем изображении
  2. Центр прямоугольника по оси x
  3. Центр прямоугольника по оси y
  4. Ширина прямоугольника
  5. Высота прямоугольника

2.2 Выбор модели YOLOv8

Выбранная модель нейронной сети YOLOv8 обладает разновидностями моделей. В каждой категории моделей YOLOv8 есть по пять моделей для обнаружения, сегментации и классификации. YOLOv8 Nano - самая быстрая и компактная, в то время как YOLOv8 Extra Large (YOLOv8x) - самая точная, но при этом самая медленная из них. В таблице 2.1 представлена сравнительная характеристика, взятая с официального сайта создателей YOLOv8 [9].

Таблица 2.1 - Параметры моделей YOLOv8 для обнаружения на COCO

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Модель | Размер (пикселей) | mAPval 50-95 | Скорость CPU ONNX(мс) | Скорость A100 TensorRT (мс) | Кол-во параметров (млн.) | FLOPs (млрд.) |
| YOLOv8n (Nano) | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |

Продолжение таблицы 2.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| YOLOv8s (Small) | 640 | 44.9 | 128.4 | 1.20 | 11.2 | 28.6 |
| YOLOv8m (Medium) | 640 | 50.2 | 234.7 | 1.83 | 25.9 | 78.9 |
| YOLOv8l (Large) | 640 | 52.9 | 375.2 | 2.39 | 43.7 | 165.2 |
| YOLOv8x (Xtra Large) | 640 | 53.9 | 479.1 | 3.53 | 68.2 | 257.8 |

В силу ограниченности имеющихся вычислительных мощностей, а также вследствие того факта, что в данной работе число классов не сильно велико (как упоминалось ранее, 15), а более «тяжеловесные» модели имеют больше параметров, как можно наблюдать из таблицы, это может повлечь за собой переобучение. Этим моделям требуется огромные тренировочные наборы данных для корректного обучения, даже наш набор, казалось бы, из почти 5 тысяч изображений для них является слишком простой задачей. Поэтому, в связи со всем вышеперечисленным, мною было принято решение выбрать модель YOLOv8n.

2.3 Выводы по разделу

В данном разделе мы решили задачу по предварительной подготовке набора данных, выполнили разметку изображений, а также выбрали необходимую модель выбранной ранее архитектуры YOLOv8 – YOLOv8n.

## 3 Программная реализация программы распознавания дорожных знаков

В данном разделе будет представлена программная реализация кода для распознавания дорожных указателей. В качестве языка программирования был выбран Python [10], так как библиотека Ultralytics [11], которая содержит используемую нами модель YOLOv8, представлена только для этого языка. Помимо этого, его основными преимуществами в сфере работы с нейронными сетями являются:

* обширная среда библиотек для машинного обучения и нейронных сетей, таких как TensorFlow, PyTorch и Keras
* простота и читаемость
* широкий спектр инструментов для работы с анализом данным и визуализации данных
* активное сообщество разработчиков и пользователей

В качестве среды разработки я выбрал сервис Google Colab. Это бесплатная платформа, предлагаемая Google, которая позволяет писать и запускать код на Python в браузере. Его основные преимущества в сфере ML и ИИ включают в себя:

* доступность (облачный хостинг, предустановка множества библиотек, доступ к вычислительным ресурсам, но ограниченный)
* простота совместной работы (хранилище на Google-диске, импорт и эскпорт файлов разработки, совместная работа в режиме реального времени, интеграция с Github’ом)
* гибкость в программировании (доступ к внешним данным, многообразие режимов выполнения, интеграция документации)

3.1 Реализация программы идентификации дорожных указателей

На данном этапе работы мы уже имеем собранный и размеченный датасет, а также определились с выбором модели, которую будем использовать. После всех необходимых операций импорта библиотек и проверки предварительно обученной модели мы запускаем обучение нашей модели на наш набор данных. Запуск данной команды производился на вычислительных ресурсах ПК с ОС Windows в консольной строке. Пример запуска можно увидеть ниже:

yolo detect train data=/content/drive/MyDrive/Kaggle dataset (Traffic sign detection)/data.yaml model=yolov8n.pt epochs=100 imgsz=416 batch=12 project=Diplom name=yolov8n\_416

Разберем его:

* yolo – вызов соответствующей библиотеки
* detect – выбор опции работы из предложенных вариантов (детекция – detect, сегментация – segment, классификация – classify, определение позы – pose, обнаружение объектов с ориентированными ограничивающими рамками – obb)
* train – режим выполнения задачи (тренировка – train, val – валидация, predict – прогнозирование, export – экспорт модели, track – отслеживание, benchmark – оценка эффективности)
* data – путь до файла data.yaml
* model – выбор модели (yolov8n, yolov8ns, yolov8m, yolov8l, yolov8x)
* epochs – количество эпох (сколько раз датасет прошел через нейронную сеть полноценно – в прямом и обратном направлении)
* imgsz – размер изображения
* batch – размер партии (датасет делят на маленькие партии – батчи, то есть количество картинок, на которых тренируется нейронная сеть одновременно)
* project – название проекта
* name – название эксперимента

Процесс тренировки представлен на рисунке 12.

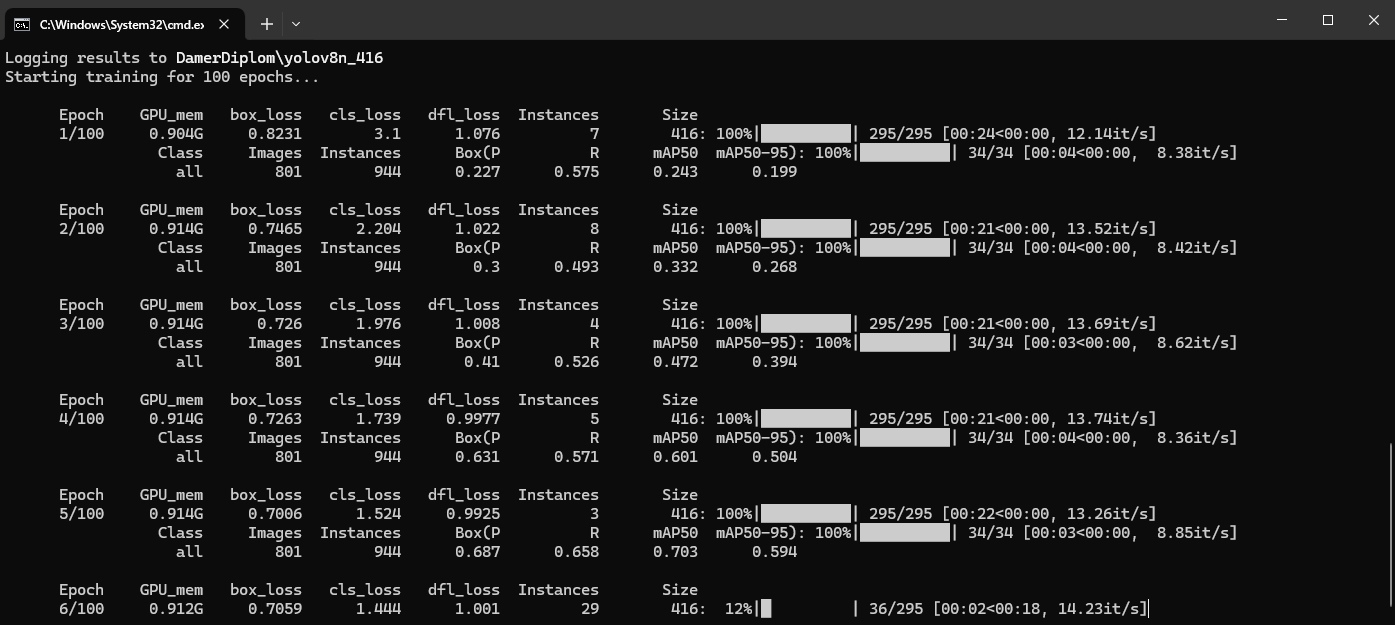


Рисунок 12 - Процесс обучения нейронной сети

Общее время тренировки нейронной сети составило около 30-35 минут.

3.2 Анализ полученных результатов

После обучения нейронной сети мы можем оценить ее эффективность, используя различные метрики для этого. Существует множество различных методов и метрик для оценки. Рассмотрим самые главные из них, по которым мы и будем оценивать работу нашей нейронной сети.

3.2.1 Метрики оценки эффективности работы нейронной сети

Так как зачастую на изображении мы можем встретить объекты различных классов, задача их обнаружения нередко превращается в задачу распознавания и определения нужного класса объекта. Благодаря этому, обнаружение объектов легко сводится к проверке гипотез *H*1 (о наличии объекта) и *H*0 (об отсутствии объекта).

Результат данной проверки может привести к ошибкам 1-го и 2-го рода. Ошибка 1-го рода или так называемая «ложноположительная» ошибка состоит в неверном отвержении нулевой гипотезы *H0* и ложном принятии гипотезы *H*1. Ошибка 2-го рода («ложноотрицательная») состоит в ошибочном принятии нулевой гипотезы *H*0.

Для решения задачи обнаружения есть возможность использовать метод полного перебора всех областей изображения. При этом необходимо уменьшить вероятность ошибки одного рода при заданном пороговом значении вероятности ошибки другого рода. Критериями эффективности такого подхода, в первую очередь, выступают доля правильных обнаружений и производительность алгоритма обнаружения. Последняя характеризуется числом стандартных кадров, обрабатываемых в единицу времени, либо временем, затрачиваемым на обработку стандартного кадра. Это приводит к необходимости оптимизации алгоритмов обнаружения с учетом задачи, которую необходимо решить [1].

Но помимо критериев корректного обнаружения, алгоритмы могут оцениваться по корректности определения самой области изображения, содержащей объект, например, метрикой loU (Intersection over Union – пересечение по объединению). Пересечение по объединению — это оценочная метрика, используемая для измерения точности детектора объектов в конкретном наборе данных. Мы часто видим, как эта метрика оценки используется в задачах обнаружения объектов. Любой алгоритм, который предоставляет прогнозируемые ограничивающие рамки в качестве выходных данных, может быть оценен с использованием IoU.

Более формально, чтобы применить IoU для оценки (произвольного) детектора объектов, нам нужны:

* Истинные ограничивающие рамки (т.е. помеченные вручную ограничивающие рамки из тестового набора, которые указывают, где на изображении находится наш объект).
* Предсказанные ограничивающие рамки из нашей модели.

Пока у нас есть эти два набора ограничивающих рамок, мы можем применять IoU.

Значение IoU является отношением площади, полученной в результате пересечения области, предсказанный алгоритмом, и реальной области с объектом, к площади, полученной в результате объединения этих областей. На рисунке 13 наглядно демонстрируется геометрический смысл метрики, где видно, если области совпадают, то значение IoU = 1, иначе IoU=0. На рисунке зона «А» - предсказанная алгоритмом зона расположения объекта, а зона «Б» - фактическое местоположение искомого объекта.

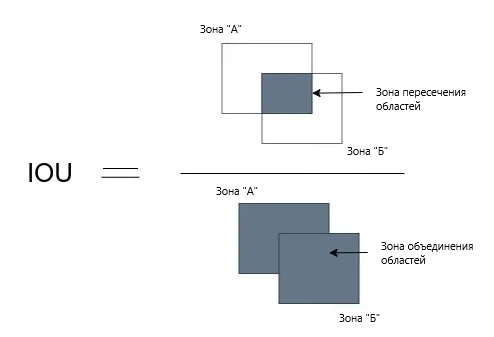


Рисунок 13 - Метрика IoU

Чтобы оценить производительность алгоритма обнаружения и локализации объектов, нам необходимо оценить, является ли прогнозируемый класс фактическим классом и насколько близок прогнозируемый ограничивающий прямоугольник к основной истине. Для этого и существует метрика под названием «Средняя точность» (AP) и «Интерполированная средняя точность» (mAP), которая позволяет как раз-таки оценить производительность нашего алгоритма. Но для ее понимания стоит разобраться с некоторыми другими метриками.

Прогноз считается правильным, если метка класса прогнозируемой ограничивающей рамки и основной истинной ограничивающей рамки одинакова, а IoU между ними превышает пороговое значение.

На основе IoU, порога и меток классов истинной истины и прогнозируемых ограничивающих рамок мы вычисляем следующие три метрики результата:

Истинно положительный (TP): модель предсказала, что ограничивающая рамка существует в определенной позиции (положительный результат), и это было правильно (истина).

Ложно положительный (FP): модель предсказала, что ограничивающая рамка существует в определенном положении (положительное), но это было неверно (ложное). Данный результат и называется ошибкой 1-го рода, о которой было сказано ранее.

Ложно отрицательный (FN): модель не предсказывала ограничивающую рамку в определенной позиции (отрицательно), и это было неправильно (ложно), т.е. в этой позиции существовала основная истинная ограничивающая рамка. Это является ошибкой 2-го рода. Подробная визуализация находится на рисунке 14.

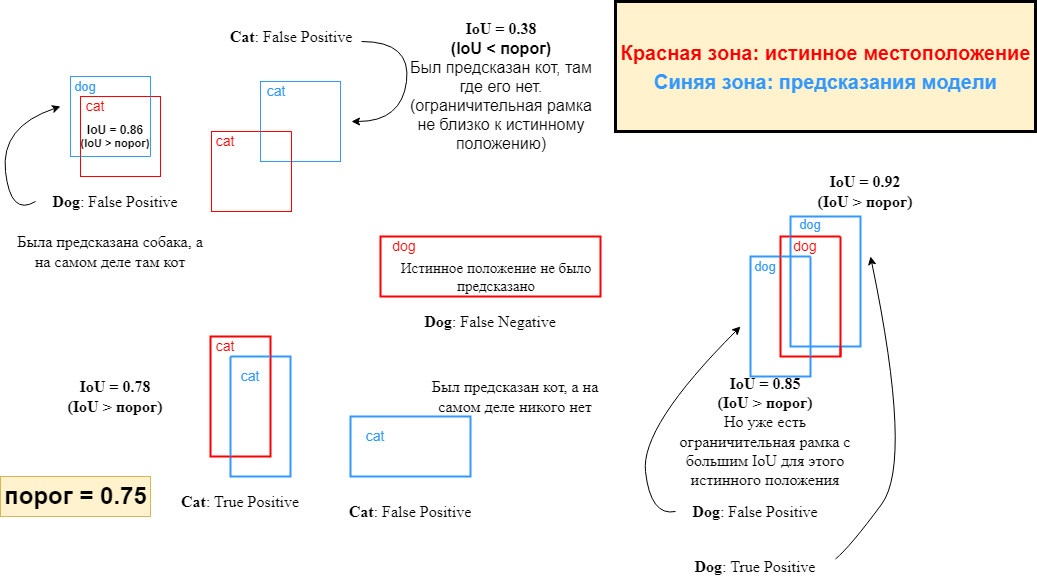


Рисунок 14 - Метрики TP, FP, FN

Истинно отрицательный результат (TF): модель не предсказала ограничивающую рамку (отрицательный результат), и это было правильно (верно). Это соответствует фону, области без ограничивающих рамок и не используется для расчета окончательных показателей.

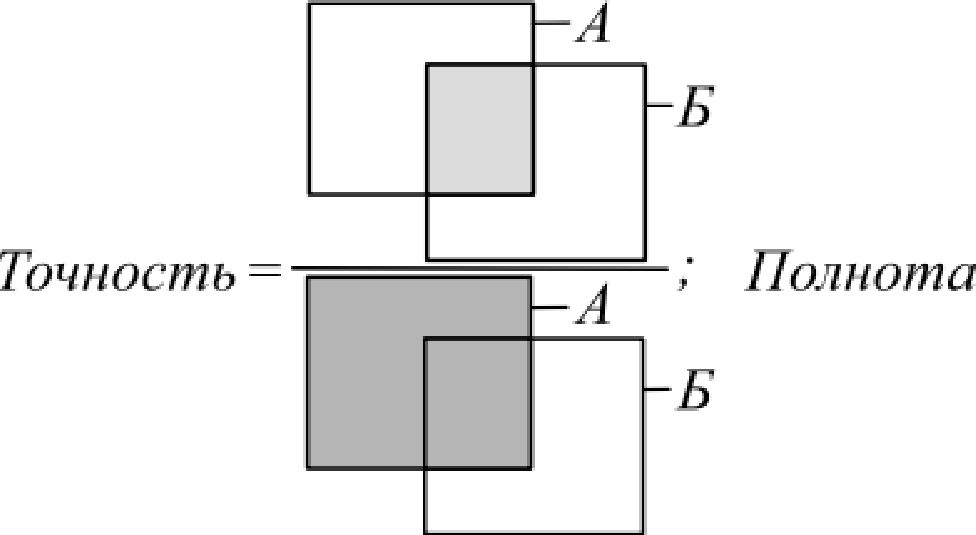
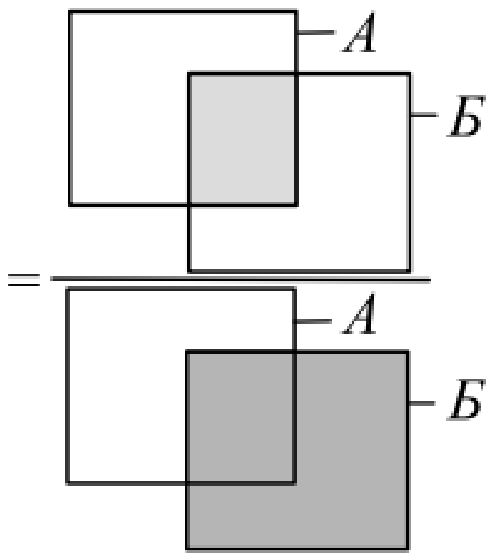
На основе TP, FP и FN для каждого помеченного класса мы рассчитываем два параметра: точность (precision) и полноту (recall).

Точность сообщает нам, насколько точна наша модель, т. е. насколько алгоритм способен обнаруживать объект, соответствующие требованиям. Следовательно, это соотношение между истинно положительными и общим количеством предсказаний (что эквивалентно сумме истинно положительных и ложно положительных результатов), сделанных моделью (1).

Полнота показывает нам, насколько хорошо модель «заполняет» классы из изображений, т. е. сколько из общего числа объектов класса модель смогла обнаружить. Следовательно, это соотношение между истинно положительными и общим количеством истинно положительных (эквивалентно сумме истинно положительных и ложно отрицательных), полученных с помощью модели (2).

где TP - число правильных решений о наличии объекта; FP - число ошибок первого рода; FN - число ошибок второго рода.

Геометрический смысл понятий полноты и точности демонстрируется на рисунке 15, где TN – количество правильных решений об отсутствии объекта [1].



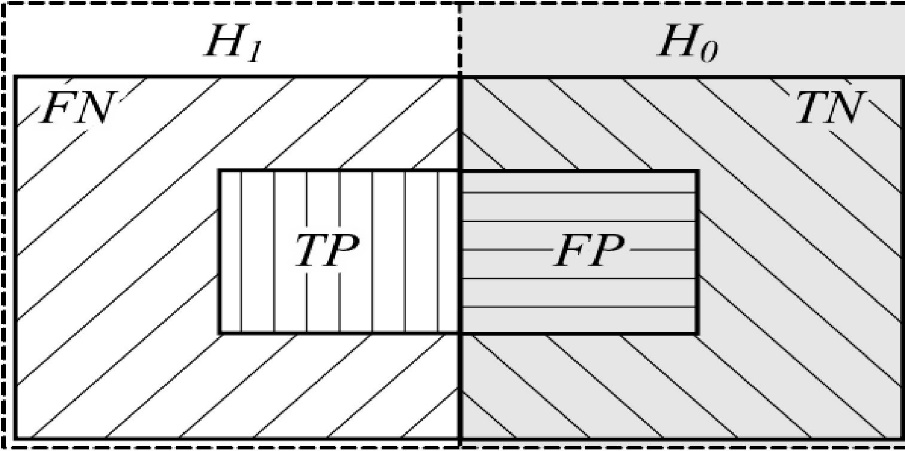


Рисунок 15 - Геометрический смысл precision и recall

Чтобы обратиться к средней точности (AP) нам необходимо понять, что В идеале мы хотим, чтобы и точность, и полнота были высокими, т. е. все, что обнаружено, является правильным, и модель может обнаружить все вхождения класса. Значение точности и полноты зависит от того, сколько истинных положительных результатов было обнаружено моделью. Назначение ограничивающей рамки TP, FP и FN зависит от следующих двух факторов:

* Прогнозируемая метка по сравнению с истинной меткой
* IoU между двумя зонами определения

Для задачи многоклассовой классификации модель выводит условную вероятность того, что ограничивающий прямоугольник принадлежит определенному классу. Чем больше вероятность класса, тем больше шансов, что ограничивающий прямоугольник будет содержать этот класс. Распределение вероятностей вместе с определяемым пользователем пороговым значением (от 0 до 1) используется для классификации ограничивающего прямоугольника.

Чем меньше этот порог достоверности вероятности, тем выше количество обнаружений, сделанных моделью, и тем ниже вероятность того, что метки основной истины были пропущены и, следовательно, выше полнота (recall) (как правило, но не всегда). С другой стороны, чем выше порог уверенности, тем более уверена модель в том, что она предсказывает, и, следовательно, выше точность (precision) (как правило, но не всегда). Мы хотим, чтобы и точность, и полнота были как можно выше, поэтому существует компромисс между точностью и полнотой, основанный на значении порога достоверности.

Кривая точности-полноты (PR) отображает зависимость точности от отзыва для различных пороговых значений достоверности.

С помощью кривой точности отзыва мы можем визуально увидеть, какой доверительный порог лучше всего подходит для нас (для нашего конкретного приложения). Чрезмерно упрощенный пример кривой PR можно увидеть на рисунке 16.

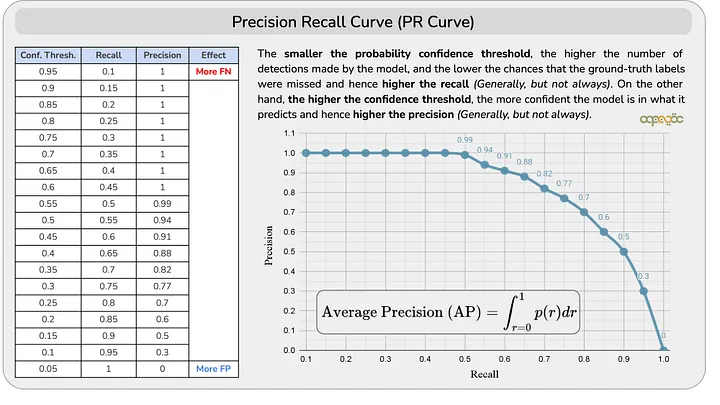


Рисунок 16 - PR-кривая

Выбор значения достоверности может быть трудным и субъективным. Средняя точность (AP) – это ключевой показатель производительности, который пытается устранить зависимость выбора одного порогового значения достоверности и определяется как область под кривой PR.

AP суммирует кривую PR до одного скалярного значения. Средняя точность высока, когда и точность, и полнота высоки, и низкая, когда любой из них низок в диапазоне значений порога достоверности. Диапазон значений AP составляет от 0 до 1.

Для определения площади под кривой PR обычно используются следующие два подхода:

1. Аппроксимируем кривую PR прямоугольниками

Для каждой пары точность-полнота (j=0,…, n-1) площадь под кривой PR можно найти путем аппроксимации кривой с помощью прямоугольников (см. рисунок 17).

Ширину таких прямоугольников можно найти, взяв разницу двух последовательных значений полноты (r(k), r(k-1)), а высоту можно найти, взяв максимальное значение точности для выбранных значений полноты. т. е

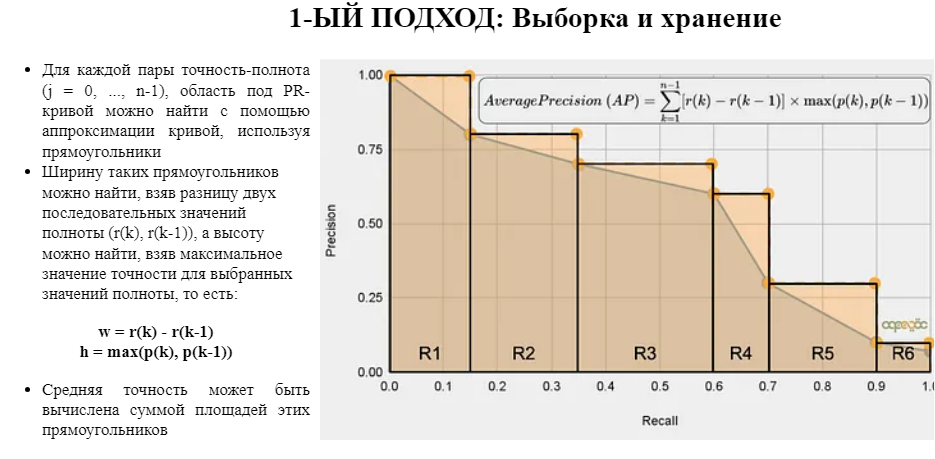
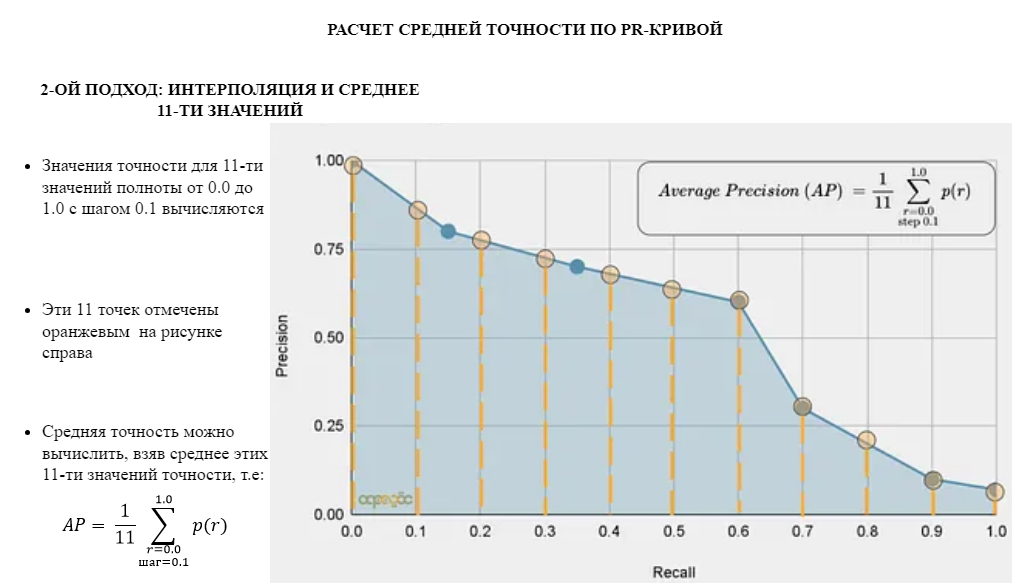


Рисунок 17 - 1-ый подход

1. Интерполяция и среднее значение по 11 точкам

Рассчитываются значения точности для 11 значений отзыва от 0,0 до 1,0 с шагом 0,1.

Эти 11 точек изображены как оранжевые образцы на рисунке 18. AP можно рассчитать, взяв среднее значение этих 11 значений точности.

Рисунок 18 - 2ой подход

Значение AP можно рассчитать для каждого класса. mAP же рассчитывается путем взятия среднего значения AP по всем рассматриваемым классам.

где k – количество классов

Но в некоторых задачах эти две характеристики означают то же самое. Например, для оценки проблемы COCO (сокращение от «Common Objects in Context» – «реальные объекты в контексте») нет разницы между AP и mAP.

Использование двух характеристик - точности и полноты позволяет более полно оценить качество работы алгоритма обнаружения по сравнению с расчетом доли верных обнаружений (accuracy):

Это особенно актуально для несбалансированных данных, где количество объектов в одном классе может значительно превышать количество объектов в другом классе. Использование формулы (3) не дает возможности адекватно оценить качество работы алгоритма [1].

Для интегральной характеристики точности и полноты используют F-меру. Данная метрика рассчитывается в соответствии с формулой (4):

где β- весовой коэффициент, определяющий вклад precision в значение F-меры.

Чаще всего весовой коэффициент β принимается равным единице и тогда F-мера превращается в F1-score.

F1-Score ­­– это мера, сочетающая точность и отзывчивость. Обычно его называют гармоническим средним из двух. Гармоническое среднее –это еще один способ вычисления «среднего» значений, который обычно описывается как более подходящий для соотношений (таких как точность и отзыв), чем традиционное среднее арифметическое. Формула, используемая для оценки F1 в этом случае:

Стремительно развивается множество метрик, которые могут быть использованы для оценки качества алгоритмов обнаружения. Таким образом, выбор метрик для конкретной задачи обосновывается необходимостью и должен быть специально подобран в каждом отдельном случае.

3.2.2 Оценка эффективности работы нашей модели

Удобство данной библиотеки заключается в том, что она автоматически генерирует различные графики, сводки, матрицы и заносит их в папку с обучением. Достаточно нужные результаты вывести на экран.

Представленные на рисунке 19 несколько строчек таблицы показывают нам как менялись значения точности, полноты и средней точности на последних эпохах тренировки в аналитическом виде.

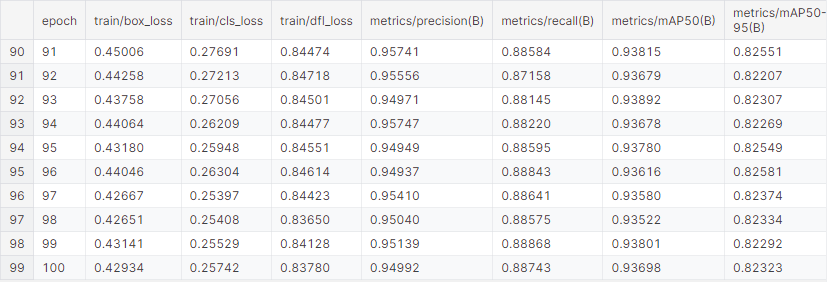


Рисунок 19 - Метрики оценки эффективности

Для наглядности эти же метрики представлены на рисунках 20-22 в виде визуализации.

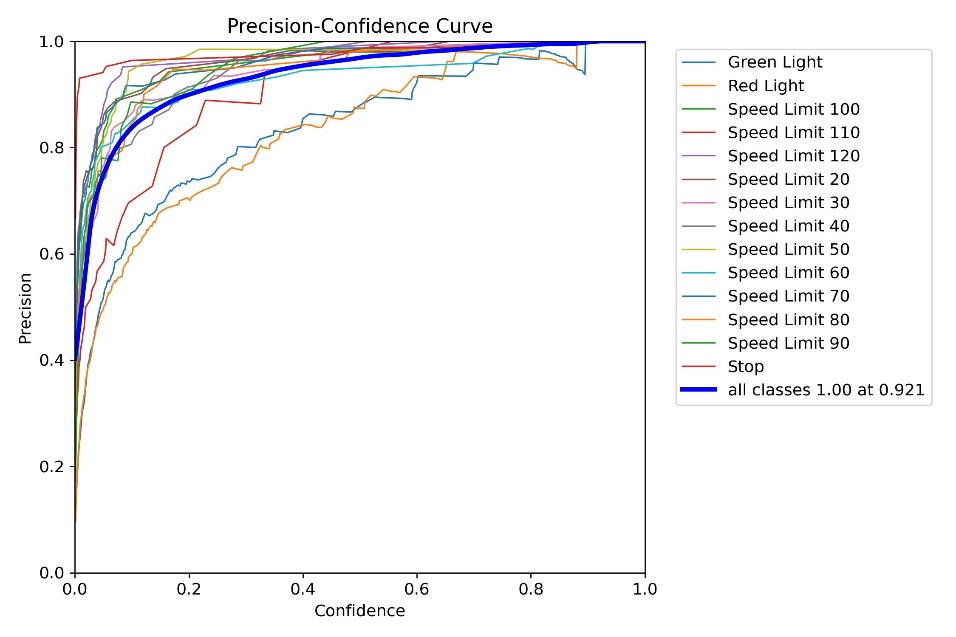


Рисунок 20 - Кривая точности

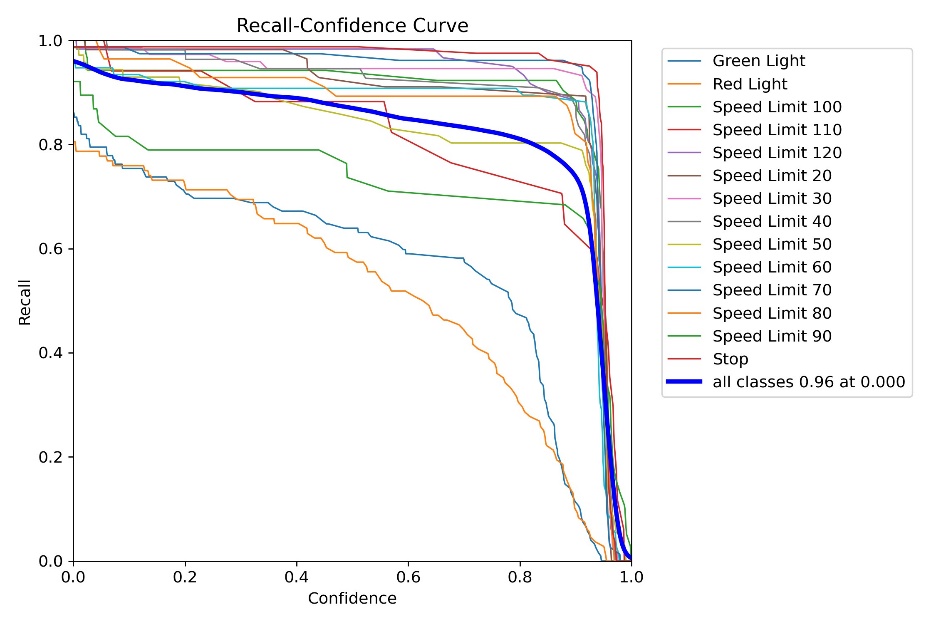


Рисунок 21- Кривая полноты



Рисунок 22 - PR-кривая

Из представленных данных можем видеть, что к концу обучения точность нашей модели составило 0.95, а полнота 0.88. Данные значения являются достаточно неплохими показателями. Средняя точность для случаев, когда пороговое значение для IoU варьировалось от 0.5 до 0.95, составила 0.82, а для случаев с пороговым значением IoU, равной 0.5, равна 0.94. Данные значения показывают неплохую эффективность тренировки нашей модели на сборном наборе данных.

Метрика F1-score представлена на рисунке 23.

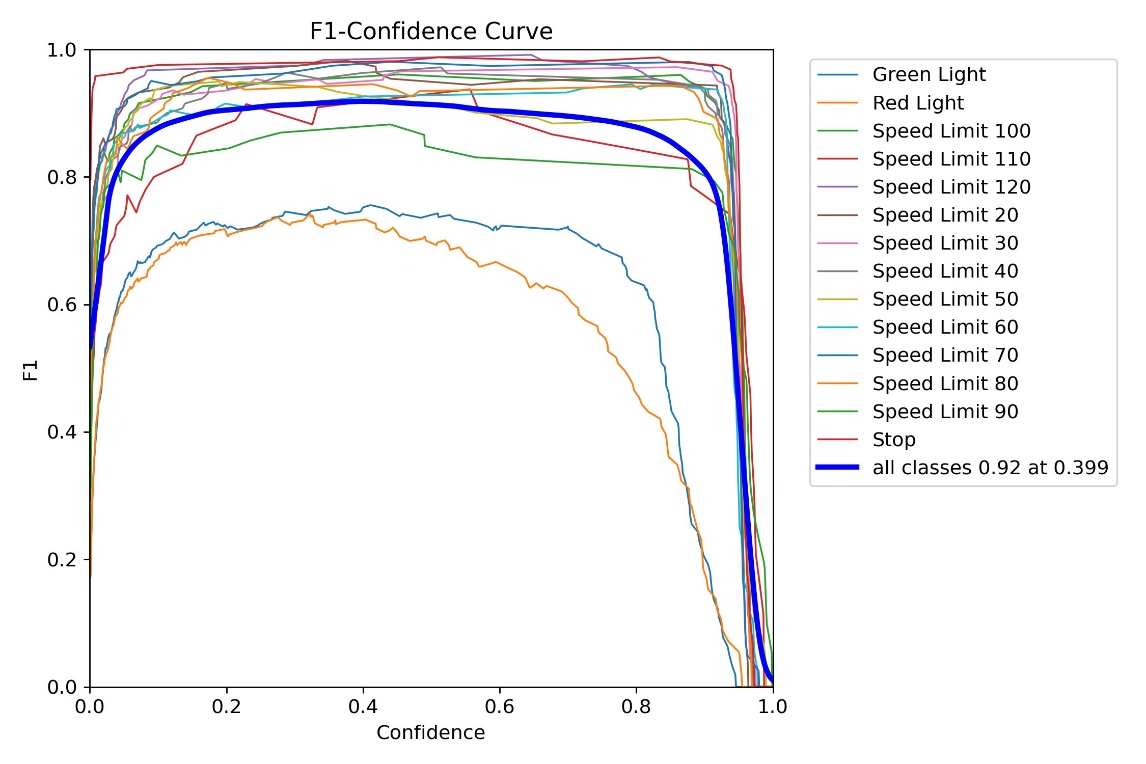


Рисунок 23 - F1-кривая

На данном графике мы можем видеть график метрики F1-score, являющейся средним гармоническим значением точности и полноты, при различных пороговых значениях. Самый высокий пик этой кривой – лучшая производительность. Стоит отметить, что пороговые значения достоверности не являются случайными; они обычно выбираются таким образом, чтобы сбалансировать точность и запоминаемость, и часто оцениваются на основе набора валидационных данных. В нашем случае, в среднем модель имеет точность 92% при пороговом значении 0.399 для всех классов.

3.2.3 Создание программного модуля

При создании интерфейса для PyQt5 был использован PyQt Designer, также было импортировано три модуля из библиотеки PyQt5: QtCore, QtGui, QtWidgets.

PyQt Designer - это визуальный редактор интерфейсов, которые работает вместе с PyQt5. Он позволяет создавать фреймы, надписи, кнопки, поля ввода и другие элементы пользовательского интерфейса, а затем использовать их на форме. После создания интерфейса в PyQt Designer, его можно сохранить в файл формата «.ui». Этот файл может быть загружен для использования в Python-приложений. Также, для создания графического интерфейса были импортированы такие классы, как: QtWidgets. QLabel и QtWidgets. QPushButton. Дизайн графического интерфейса изображен на рисунке 49.

QLabel и QPushButton - это классы виджетов из библиотеки PyQt5.

QLabel - используется для отображения текста и изображения в окне программы.

QPushButton - это кнопка для выполнения определенного действия. В нашей программе две кнопки данного класса - одна для выбора фото, а другая для запуска распознавания изображения.

Входными величинами будут изображения размером 416\*416 пикселей, выходными данными будут названия классов информационных знаков.

Наш программный код создает главное окно размером 432\*338 пикселей. Его пример представлен на рисунке 24.

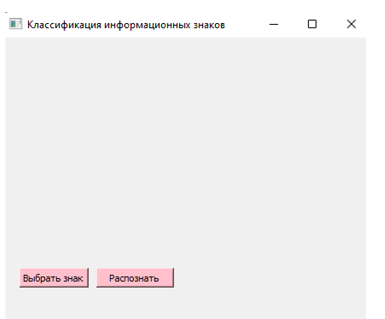


Рисунок 24 - Дизайн графического интерфейса

3.2.4 Проверка обученной нейронной сети на тестовом наборе

После обучения и анализа его результатов мы можем попробовать нашу модель на тестовом наборе данных. В нем содержится 638 различных изображений, что составляет 13% от всех изображений.

На рисунке 25 мы можем наблюдать метрики оценки эффективности работы нашей модели на тестовом наборе данных.

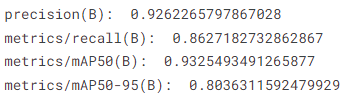


Рисунок 25 - Метрики теста нейронной сети

Основываясь на наблюдаемых результатах, очевидно, что точность модели как в наборах валидационных, так и тестовых данных демонстрирует высокую степень сходства. Этот результат служит показателем того, что модель была соответствующим образом обучена.

На рисунках 26-31 выведены некоторые результаты детекции дорожных указателей нашей моделью.

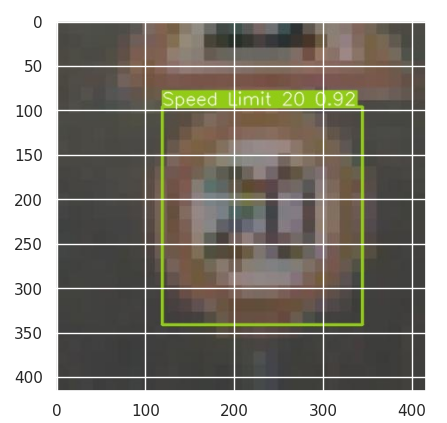


Рисунок 26 - Обнаружение знака «Ограничение максимальной скорости 20 км/ч»

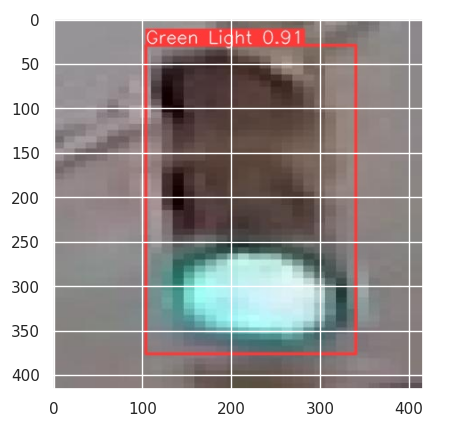


Рисунок 27 - Обнаружение указателя «Зеленый свет»

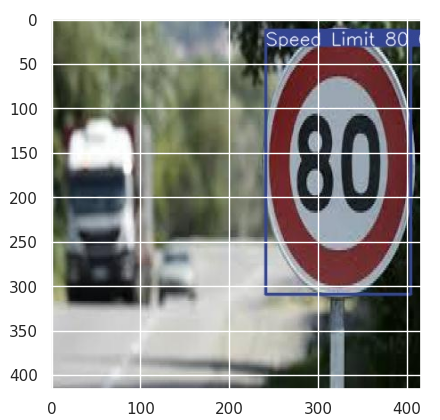


Рисунок 28 - Обнаружение знака «Ограничение максимальной скорости 80 км/ч»

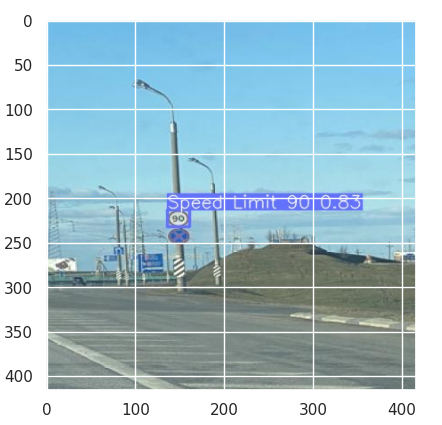


Рисунок 29 - Обнаружение знака «Ограничение максимальной скорости 90 км/ч»

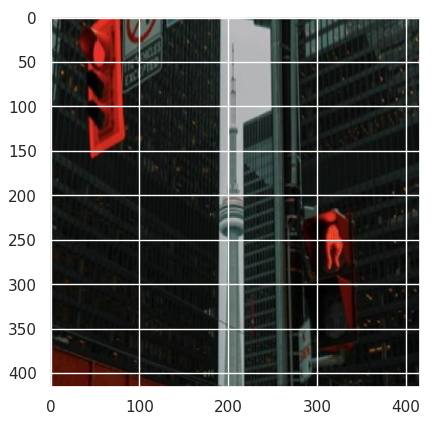


Рисунок 30 - Отсутствия распознавания указателя «Красный свет»

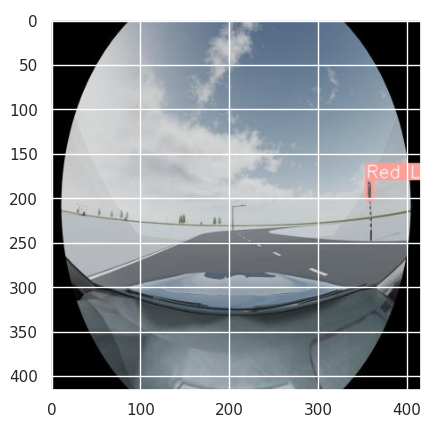


Рисунок 31 - Обнаружение указателя «Красный свет»

## 4 Экономическое обоснование

Целью данной выпускной квалификационной работы является создание программы для идентификации дорожных указателей, используя алгоритмы, основанные на искусственных нейронных сетях. Все расчеты были проведены в соответствии с методическими указаниями [12], рассчитаны затраты на исследования и разработку.

В данном разделе выпускной квалификационной работы была осуществлена оценка экономической эффективности разработки программы. Для оценки выполнены следующие расчеты:

1. Трудоемкость выполнения ВКР.
2. Заработная платы разработчика и руководителя.
3. Затраты на сырье и приобретение расходных материалов.
4. Затраты на эксплуатацию и содержания оборудования.
5. Амортизационные отчисления.
6. Затраты на услуги сторонних организаций.
7. Накладные расходы.
8. Общая величина затрат на ВКР.

Данные вычисления позволят рассчитать суммарные затраты необходимые для разработки программы распознавания дорожных указателей.

4.1 План выполнения разработки

Первым этапом необходимо составить план разработки для более точного и наглядного технико-экономического обоснования всей работы.

Необходимо рассчитать продолжительность каждого этапа работы. Продолжительность работ можно определить по следующей формуле (5): где - ожидаемая длительность j-й работы; tmin и tmax – наименьшая и наибольшая по мнению эксперта длительность работы.

Все значения и результаты расчетов занесены в таблицу 4.1.

Таблица 4.1 –Трудоемкость работ по разработке

| № | Наименование работы | Длительность работы, ед. t | | |
| --- | --- | --- | --- | --- |
| tmin | tmax | to |
| 1 | Подбор и изучение литературы | 17 | 24 | 20 |
| 2 | Анализ предметной области и методов реализации | 35 | 40 | 37 |
| 3 | Проектирование и разработка программы | 180 | 200 | 188 | |
| 4 | Оформление пояснительной записки | 20 | 48 | 31 | |
| ИТОГО | | 192 | 312 | 276 | |

Общая длительность работы составила 276 часов.

4.2 Расчет заработной платы разработчика

За основу заработной платы будет использоваться количество трудоемкости работы, связанное с созданием ВКР.

В разработке программы задействован один разработчик – студент, заработная плата которого считается равной окладу инженера. Средний оклад инженера в Санкт-Петербург на момент написания работы составляет 50 000 руб [13].

Для того, чтобы определить конечную заработную плату разработчика необходимо рассчитать часовую ставку. Определим часовую ставку по следующей формуле (6):

где s - оклад разработчика, t – количество рабочих часов в день, d – количество рабочих дней в месяце.

Будем исходить из того, что в среднем 21 рабочий день в месяце при 8 часовой дневной загруженности. Таким образом, часовая ставка разработчика:

q = 297.6 руб./час.

В моей разработке также принимает участие руководитель, необходимо рассчитать оплату труда и для него. Костичев С.В. является доцентом кафедры ВТ, кандидатом технических наук. Согласно Приказу ректора № ОД/0539 от 27.09.2019 **(«Об увеличении уровня оплаты труда работникам университета и. об изменении размеров минимальных должностных окладов и должностных окладов по профессионально-квалификационным группам»)** должностной оклад Костичева С.В. составляет 62500 руб. до уплаты налогов. Таким образом, получаем часовую ставку руководителя, по приведенной ранее формуле (6):

q = 372 руб./час.

Трудоемкость и ставка исполнителей занесены в таблицу 4.2.

Таблица 4.2 – Расчет ставки исполнителей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Этапы и содержание выполняемых работ | Исполнитель | Трудоемкость,  to, ед. t | Ставка,  руб./ед. t |
| 1 | Подбор и изучение литературы, постановка задачи | Д. Ш. Дашкин | 20 | 297.6 |

Продолжение таблицы 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Подбор и изучение литературы, постановка задачи | Д. Ш. Дашкин | 20 | 297.6 |
| 3 | Анализ предметной области и методов реализации | Д. Ш. Дашкин | 37 |
| 4 | Проектирование и разработка программы | Д. Ш. Дашкин | 188 |
| 5 | Оформление пояснительной записки | Д. Ш. Дашкин | 31 |  |
| 6 | Проверка ВКР | С. В. Костичев | 7 | 372 |

После получения данных по часовой ставке задействованных в ВКР исполнителей, можем рассчитать сумму заработной платы с учетом социальных отчислений, а именно, отчислений на следующие страховые взносы: пенсионные, медицинские, социальные.

Расходы на основную заработную плату исполнителей до уплаты налогов определяются по формуле (7):

где Зосн.з/пл – расходы на основную заработную плату исполнителей (руб.); k – количество исполнителей; T*i* – время, затраченное i-м исполнителем на проведение исследования в часах; C*i* – ставка i-го исполнителя, руб/час.

Рассчитаем расходы на дополнительную заработную плату исполнителей, которая определяются по формуле (8):

где Здоп.з/пл – расходы на дополнительную заработную плату исполнителей (руб.); Зосн.з/пл – расходы на основную заработную плату исполнителей (руб.); Hдоп. - норматив дополнительной заработной платы (%). Hдоп. принимаем за 8.3%.

Рассчитаем сумму дополнительной заработной платы, которая составляет 8.3% от ставки.

Страховые взносы отчисляются в следующие государственные внебюджетные фонды социального назначения:

* Пенсионный фонд России (ПФР)
* Федеральный фонд обязательного медицинского страхования (ФФОМС)
* Фонд социального страхования (ФСС)

Отчисления на социальные нужды: Пенсионные, медицинские и социальные страховые взносы в 2024, как и в 2023 году, работодатели будут перечислять одним платежом по единому тарифу 30%. В соответствии с этим рассчитываем отчисления на социальные нужды с помощью формулы (9):

Общие отчисления на социальные нужды составили 27 718 рублей.

4.3 Затраты на сырье и материалы

Для оформления данной ВКР были приобретены следующие товары: заправка картриджа для принтера, бумага для оргтехники А4, скрепки канцелярские, ручка шариковая. Данные собраны в таблицу 4.3.

Таблица 4.3 – Затраты на сырье и материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Материалы | Кол-во | Цена, руб | Сумма, руб |
| Бумага для оргтехники A4, пачка | 1 | 225 | 225 |
| Файлы для А4, пачка | 1 | 240 | 240 |
| Ручка шариковая | 1 | 78 | 78 |
| Папка для диплома | 1 | 414 | 414 |
| Скрепки канцелярские, пачка | 1 | 90 | 90 |
| ИТОГО | | | 1047 |

Общие затраты на сырье и материалы составили 1047 рублей.

4.4 Затраты на содержание и эксплуатацию оборудования

В ходе выполнения выпускной квалификационной работы был задействован стационарный ПК. За основную и единственную статью расходов, связанную с эксплуатацией оборудования, будут взяты затраты на электроэнергию при использовании компьютера от сети.

Формула (10) для расчета затрат на эксплуатацию оборудования:

где З­­эо – затраты на содержание и эксплуатацию оборудования (руб.); – расчетная себестоимость одного машино-часа работы оборудования на i-й технологической операции (руб./м-ч); – количество машино-часов, затрачиваемых на выполнение i-й технологической операции (м-ч).

Была подсчитана общая энергоемкость ПК. Она составляет 210 Вт. Текущий тариф на электроэнергию по Санкт-Петербургу составляет 5,7 руб./кВт\*ч.

По имеющимся данным рассчитаем итоговую стоимость машино-часа:

руб./м-ч

Количество часов работы на персональном компьютере соответствует количеству часов, затраченных на выполнение ВКР, а именно 276 часов.

Таким образом, итоговые затраты составили 330.4 рублей.

4.5 Услуги сторонних организаций

В течение процесса разработки использовались услуги провайдера домашнего интернета «Билайн».

Оплата данных услуг за один месяц составляет 500 рублей. С вычетом НДС затраты составили 416.7 рублей.

4.6 Амортизационные отчисления

Амортизационные отчисления по основному средству i за год рассчитываются по следующей формуле (11):

где A*i* – амортизационные отчисления за год по i-му основному средству (руб.); Цп.н.*i* – первоначальная стоимость i-го основного средства (руб.); H*ai* – годовая норма амортизации i-го основного средства (%).

Согласно постановлению Правительства РФ от 01.01.2002 №1 (ред. от 07.07.2016) «О классификации основных средств, включаемых в амортизационные группы», ПК относится ко второй амортизационной группе, срок полезного использования компьютера составляет 3 года, соответственно годовая норма амортизации данного ноутбука составит 33.3%.

Стоимость используемого ПК составляет 30 000 рублей. Отсюда, по формуле (11), амортизационные отчисления за год составят 9990 рублей.

Для расчета величины амортизационных отчислений по i-му основному средству, используемому во время разработки, воспользуемся данной формулой (12):

где A*i*ВКР – амортизационные отчисления по i-му основному средству, используемому студентом в работе над ВКР (руб.); A*i* – амортизационные отчисления за год по i-му основному средству (руб.); T*i*ВКР – время, в течение которого студент использует i-ое основное средство (мес.)

Процесс написания ВКР занимает 2 месяца.

Амортизационные отчисления по ПК составили 1665 рублей.

4.7 Накладные расходы

К накладным расходам относятся расходы на управление и хозяйственное обслуживание. Накладные расходы рассчитываются по формуле (13):

где Зн.р. – накладные расходы; Зосн.з/пл – расходы на основную заработную плату исполнителей (руб.); Здоп.з/пл – расходы на дополнительную заработную плату исполнителей (руб.); Hн.р. – норматив отчислений на накладные расходы (%).

Накладные расходы составляют 20% от суммы основной и дополнительной заработной платы

Накладные расходы составили 18356.2 рублей.

4.8 Совокупная величина затрат

Результаты расчета совокупной величины затрат, связанных с проведением разработки представлены в таблице 4.4.

Таблица 4.4 – Совокупная величина затрат на ВКР

|  |  |  |
| --- | --- | --- |
| № | Наименование статьи | Сумма, руб. |
| 1 | Расходы на основную заработную плату |  |
| 2 | Расходы на дополнительную заработную плату |  |
| 3 | Отчисления на социальные нужды |  |
| 4 | Затраты на сырье и материалы | 1 047 |
| 5 | Затраты на содержание и эксплуатацию оборудования |  |
| 6 | Услуги сторонних организаций | 416.7 |
| 7 | Амортизационных отчислений |  |
| 8 | Накладные расходы |  |
| Итого | | 141 314.3 |

Общая совокупность затрат составила 141 314.3 рублей.

4.9 Заключение

В данном разделе ВКР были рассчитаны затраты на создание программы для идентификации дорожных знаков по следующим статьям:

1. Расходы на основную заработную плату.
2. Расходы на дополнительную заработную плату.
3. Отчисления на социальные нужды.
4. Затраты на сырье и материалы.
5. Затраты на содержание и эксплуатацию оборудования.
6. Услуги сторонних организаций.
7. Амортизационных отчислений.
8. Накладные расходы.

Совокупные затраты на выполнение работы составили 141 314.3 рублей. Проводя аналитическое сравнение с существующими решениями на рынке, можно заявить, что подобные расходы являются достаточно небольшими. Таким образом, проведение разработки программы для идентификации дорожных знаков целесообразно и рационально, в связи со стремительным и продолжающимся ростом сферы искусственного интеллекта.

# ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы нами был проведен анализ существующих методов распознавания образов с использованием ИНС, где выявили самый удачный метод YOLOv8; сформирован набор данных из 4969 изображений, размечен и обработан. В этом наборе 3530 изображений мы используем для тренировки, 801 изображение для проверки работоспособности модели, а 638 изображений используются для тестов. Проанализировав представленные модели выбранной архитектуры YOLOv8, мы смогли подобрать оптимальный вариант модели под нашу задачу, обучить ее и протестировать. Также был создан графический пользовательский интерфейс «Traffic sign detection». Таким образом был реализован модуль «Идентификация дорожных указателей» с точностью выше 90%.

Модель выбранной нами нейронной сети, представленная в библиотеке Ultralytics на языке программирования Python, была обучена на вычислительных мощностях ПК с помощью консольного ввода на ОС Windows. Для написания основного кода программы и анализа результатов тренировки и тестов нами использовалась интегрированная среда разработки Google Colab, а разработка графического пользовательского интерфейса производилась в среде разработки Pycharm.

Применение НС в качестве готового инструмента предоставляет возможность быстрого и точного решения задачи обнаружения и распознавания объектов на видео и фотоматериалах. Выбор архитектуры СНС зависит от типа задачи, технических требований, а также от особенностей и характеристик собранных исходных данных.

В ходе работы я получил колоссальный опыт в работе с нейронными сетями в задаче идентификации дорожных указателей, их интеграции в графический пользовательский интерфейс. В дальнейшем при доработке программа может использоваться в технических решениях задачи распознавания дорожных знаков в режиме реального времени.

По итогам выпускной квалификационной работы все поставленные задачи были выполнены, а цель – реализация алгоритмов и их дальнейшего внедрения в программные модули для последующего распознавания дорожных указателей при помощи искусственных нейронных сетей – была достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Обнаружение объектов на изображении: от критериев Байеса и Неймана–Пирсона к детекторам на базе нейронных сетей EfficientDet // URL:https://computeroptics.ru/KO/PDF/KO46-1/460117.pdf (дата обращения: 20.04.2024).
2. Сравнение YOLOv5 и Faster R-CNN для обнаружения людей на изображении в потоковом режиме [Электронный ресурс]. -2022.- URL: https://cyberleninka.ru/article/n/sravnenie-yolo-v5-i-faster-r-cnn-dlya-obnaruzheniya-lyudey-na-izobrazhenii-v-potokovom-rezhime (дата обращения: 30.04.2024).
3. Performance Benchmark of YOLO v5, v7 and v8 [Электронный ресурс]. -2023.- URL: https://www.stereolabs.com/blog/performance-of-yolo-v5-v7-and-v8 (дата обращения: 10.05.2024).
4. YOLOv8 [Электронный ресурс]. -2023.- URL: https://roboflow.com/model/yolov8 (дата обращения 10.05.2024).
5. Brief summary of YOLOv8 model structure [Электронный ресурс]. -2023.- URL: https://github.com/ultralytics/ultralytics/issues/189 (дата 12.05.2024).
6. Roboflow. – URL: https://roboflow.com/(дата обращения 15.05.2024).
7. COCO. Common Objects in Context. – URL: https://cocodataset.org/(дата обращения 18.05.2024).
8. YOLO. Dataset Formats [Электронный ресурс]. -2024.- URL: https://docs.ultralytics.com/datasets/detect/#supported-datasets (дата обращения 18.05.2024).
9. YOLOv8 [Электронный ресурс]. -2024.- URL: https://docs.ultralytics.com/ru/models/yolov8/#performance-metrics (дата обращения 20.05.2024).
10. Python 3.12.3 documentation. [Электронный ресурс]. -2024.- URL: https://docs.python.org/3/ (дата обращения 25.05.2024).
11. Ultralytics YOLO docs. – URL: https://docs.ultralytics.com/ru(дата обращения 25.05.2024).
12. Алексеева О.Г. Методические указания по экономическому обоснованию выпускных квалификационных работ бакалавров:// Алексеева

О.Г. // Метод. Указания – СПБ.: СПБГЭТУ «ЛЭТИ» – 2013. с.

1. Город работ.ру. Статистика рынка труда в Санкт-Петербурге — «Инженер». https://clck.ru/3Az2f9

# ПРИЛОЖЕНИЕ