Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Инженерная школа цифровых технологий

09.03.01 – Информатика и вычислительная техника

(код и наименование направления подготовки, специальности)

Информатика и вычислительная техника

(наименование профиля, специализации)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(БАКАЛАВРСКАЯ РОБОТА)

На тему Разработка программного модуля распознавания автомобилей экстренных служб на основе модели машинного обучения

|  |  |  |
| --- | --- | --- |
| Студент гр. 1192б  Проскуряков\_Богдан\_Григорьевич |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (личная подпись) |

|  |  |  |
| --- | --- | --- |
| Руководитель  Преподаватель  Усманов\_Руслан\_Талгатович |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (личная подпись) |

|  |  |  |
| --- | --- | --- |
| Нормоконтролер | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Фамилия имя отчество) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (личная подпись) |

Допустить к защите

|  |  |  |
| --- | --- | --- |
| Руководитель образовательной программы  Доцент ИШЦТ, к.т.н.  Самарин\_Валерий\_Анатольевич |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (личная подпись) |

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

г. Ханты-Мансийск

2023 год

РЕФЕРАТ

Бакалаврская работа содержит 40 с., 10 рисунков, 3 формулы, 2 приложения, 15 источников.

РАСПОЗНАВАНИЕ АВТОМОБИЛЕЙ, ЭКСТРЕННЫЕ СЛУЖБЫ, МАШИННОЕ ОБУЧЕНИЕ, ФУНКЦИОНАЛЬНАЯ СТРУКТУРА, БОЛЬШИЕ ДАННЫЕ, ОТКРЫТЫЙ ИСХОДНЫЙ КОД

Главной целью бакалаврской работы является разработка программного модуля «распознавания автомобилей экстренных служб на основе модели машинного обучения»

Объектом исследования данной выпускной квалификационной работы является процесс разработки программного модуля распознавания автомобилей экстренных служб на основе модели машинного обучения. Предметом исследования является разработка программного модуля, способного автоматически идентифицировать автомобили экстренных служб на основе изображений.

Выполнение работы включает в себя несколько этапов.

Первым этапом является анализ предметной области, обзор аналогов.

Следующим этапом является проектирование, он включает в себя анализ области применения разрабатываемого программного модуля, системный анализ и его архитектура, а также методы реализации.

Заключительным этапом является процесс сбора данных, их обработки и реализации программного продукта, согласно требованиям заказчика.

Модуль может иметь практическое значение для распознавания автомобилей экстренных служб на фото-видео файлах, а также послужит для дальнейшего развития темы машинного обучения своим открытым исходным кодом.

ОГЛАВЛЕНИЕ

[ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ 4](#_Toc138045469)

[ВВЕДЕНИЕ 5](#_Toc138045470)

[1 Обзорно-теоретическая часть 7](#_Toc138045471)

[1.1 Обзор объекта 7](#_Toc138045472)

[1.2 Обзор аналогов 7](#_Toc138045473)

[1.3 Анализ и заключение 9](#_Toc138045474)

[2 Проектирование 11](#_Toc138045475)

[2.1 Описание исследуемой предметной области 11](#_Toc138045476)

[2.2 Назначение разрабатываемого программного модуля 11](#_Toc138045477)

[2.3 Область применения разрабатываемого программного модуля 11](#_Toc138045478)

[2.4 Системный анализ и архитектура разрабатываемого модуля 13](#_Toc138045479)

[2.5 Анализ и заключение 14](#_Toc138045480)

[3 Реализация системы 15](#_Toc138045481)

[3.1 Описание реализации системы 15](#_Toc138045482)

[3.2 Подбор датасета 16](#_Toc138045483)

[3.3 Обучение модели 18](#_Toc138045484)

[3.4 Реализация отслеживания на видео 20](#_Toc138045485)

[3.5 Тестирование 22](#_Toc138045486)

[ЗАКЛЮЧЕНИЕ 24](#_Toc138045487)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc138045488)

[ПРИЛОЖЕНИЯ 27](#_Toc138045489)

[Приложение А. 27](#_Toc138045490)

[Приложение Б 34](#_Toc138045491)

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

В настоящей Выпускной квалификационной работе применяют следующие сокращения и обозначения:

API - Application Programming Interface, что значит программный интерфейс приложения

АИС - Автоматизированная информационная система

Open Source - исходный код программного обеспечения, который доступен для всех пользователей

Фреймворк - готовый набор инструментов

Датасет - механизм хранения информации, который предоставляет быстрый доступ к большим объемам данных

Эпоха - полная итерация всего обучающего набора данных за один цикл для обучения модели машинного обучения

Батч - небольшой пакет данных одной итерации

anchor box - набор предопределенных ограничивающих рамок определенной высоты и ширины

ВВЕДЕНИЕ

В современном мире технологии искусственного интеллекта и машинного обучения активно проникают в различные сферы нашей жизни. Они предлагают новые возможности для автоматизации процессов, улучшения безопасности и повышения эффективности работы различных организаций. Одной из таких сфер является сфера экстренных служб, где быстрая и точная идентификация автомобилей имеет решающее значение для оперативной реакции и предоставления помощи в критических ситуациях.

В условиях увеличения автомобильного трафика и роста числа аварийных ситуаций становится все более важным разработать эффективные технологии для распознавания автомобилей экстренных служб. Традиционные методы идентификации, основанные на визуальном восприятии оператора, могут быть недостаточно точными и подвержены человеческому фактору. В связи с этим, разработка программного модуля распознавания автомобилей экстренных служб на основе модели машинного обучения представляет большой интерес и актуальность.

Целью данной выпускной квалификационной работы является разработка программного модуля, способного автоматически распознавать автомобили экстренных служб на основе модели машинного обучения. Для достижения этой цели были поставлены следующие задачи:

1. Изучить существующие методы и технологии распознавания автомобилей экстренных служб.
2. Собрать и подготовить достаточный объем данных для обучения модели машинного обучения.
3. Разработать архитектуру программного модуля для распознавания автомобилей экстренных служб.
4. Реализовать и обучить модель машинного обучения на подготовленных данных.
5. Провести экспериментальное исследование и оценить точность и эффективность разработанного модуля.

Объектом исследования данной выпускной квалификационной работы является разработка программного модуля, способного автоматически идентифицировать автомобили экстренных служб на основе изображений.

Предметом исследования является процесс разработки программного модуля распознавания автомобилей экстренных служб на основе модели машинного обучения. Главной целью работы является создание эффективной модели машинного обучения, которая будет обучаться на предоставленных данных и способна распознавать и классифицировать автомобили экстренных служб на фото-видео объектах.

1. Обзорно-теоретическая часть
   1. Обзор объекта

Автомобили экстренных служб, такие как скорая помощь, пожарная служба и полиция, играют решающую роль в обеспечении безопасности и оказании помощи в критических ситуациях. Быстрая и точная идентификация таких автомобилей имеет важное значение для оперативной реакции и эффективной координации действий экстренных служб. Однако, визуальное распознавание автомобилей экстренных служб оператором может быть неполным, подвержено человеческому фактору и требует значительных усилий и времени. В связи с этим, разработка программного модуля на основе модели машинного обучения для автоматического распознавания автомобилей экстренных служб является актуальной задачей.

* 1. Обзор аналогов

На данный момент существуют различные аналоги и подходы к распознаванию автомобилей экстренных служб. Один из наиболее распространенных методов основан на классификации автомобилей по их визуальным признакам, таким как цвет, форма и наличие специфических маркеров или сигнальных устройств. Этот подход требует создания и поддержания базы данных изображений автомобилей экстренных служб, что может быть затруднительным и трудоемким.

Другой подход основан на использовании глубокого обучения и нейронных сетей для автоматического распознавания автомобилей экстренных служб. Этот метод позволяет создать модель машинного обучения, которая может обучаться на большом объеме данных и автоматически извлекать признаки для идентификации автомобилей экстренных служб. Такой подход обычно обеспечивает более высокую точность распознавания и может быть эффективно применен в реальном времени. [1, 4, 6]

На данный момент существует несколько готовых продуктов и решений, которые могут быть рассмотрены в качестве аналогов к данной работе по разработке программного модуля распознавания автомобилей экстренных служб на основе модели машинного обучения. Некоторые из них включают:

1. Amazon Rekognition: Amazon Rekognition предлагает облачные сервисы компьютерного зрения, включая функцию распознавания и классификации объектов на изображениях. Этот сервис может быть использован для разработки распознавания автомобилей экстренных служб на основе их визуальных признаков.
2. Microsoft Azure Computer Vision: Microsoft Azure Computer Vision предоставляет набор API и сервисов для обработки изображений и видео. С помощью этого инструмента можно реализовать модуль распознавания автомобилей экстренных служб на основе глубокого обучения и классификации.
3. Google Cloud Vision API: Google Cloud Vision API предлагает широкий спектр возможностей компьютерного зрения, включая обнаружение объектов, классификацию изображений и распознавание текста. Этот инструмент также может быть использован для создания модуля распознавания автомобилей экстренных служб.
4. OpenALPR: OpenALPR – это библиотека с открытым исходным кодом, специализирующаяся на распознавании номерных знаков автомобилей. Она может быть адаптирована для распознавания номеров автомобилей экстренных служб и использована в вашем программном модуле.

Указанные аналоги предоставляют готовые решения для распознавания объектов на изображениях и могут быть адаптированы для задачи распознавания автомобилей экстренных служб. Однако, все эти решения являются зарубежными и отечественных, более-менее популярных или используемых аналогов практически нет. На данный момент удалось найти лишь одну компанию “VideoNet”, которая предоставляет АИС для распознавания машин экстренных служб.

Недостатком сервиса “VideoNet” можно выделить приватность исходного кода, компания предоставляет уже готовый софт за фиксированную сумму (Рисунок 1)

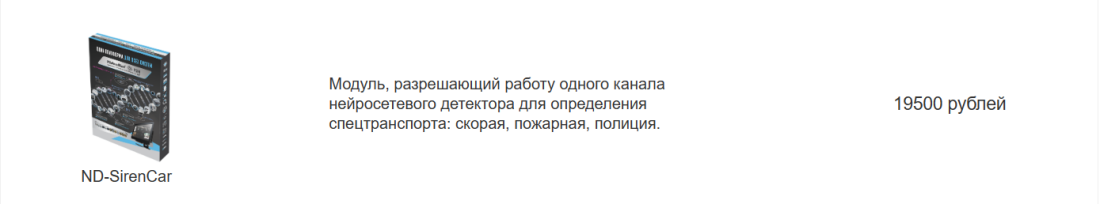


Рисунок 1 - Продукт компании VideoNet

Когда данная работа нацелена на Open Source разработку специализированного программного модуля, который будет основан на модели машинного обучения и может быть более гибким и точным в идентификации автомобилей экстренных служб, учитывая их специфические характеристики и признаки

* 1. Анализ и заключение

Анализ предметной области показывает, что разработка программного модуля распознавания автомобилей экстренных служб на основе модели машинного обучения имеет значительный потенциал и актуальность. Применение глубокого обучения и нейронных сетей позволяет создать модель, способную автоматически извлекать признаки и идентифицировать автомобили экстренных служб с высокой точностью. [7, 10, 14]

В ходе обзора аналогов были рассмотрены готовые продукты и решения, такие как Amazon Rekognition, Microsoft Azure Computer Vision, Google Cloud Vision API и OpenALPR, которые предлагают функциональность по распознаванию объектов на изображениях. Однако, данные аналоги не обладают специализированной моделью для распознавания автомобилей экстренных служб, что делает разрабатываемый программный модуль более привлекательным и уникальным. В свою очередь, решение VideoNet является отечественной разработкой, но распространяется по лицензии и не имеет открытого исходного кода, что в свою очередь снижает актуальность для научных работ и стартапов.

На основе проведенного анализа можно сделать вывод, что разработка программного модуля распознавания автомобилей экстренных служб на основе модели машинного обучения является перспективным направлением. Дальнейшая работа будет направлена на подготовку данных, выбор и разработку оптимальной архитектуры модели, а также проведение обучения и тестирования для достижения высокой точности распознавания и эффективной работы модуля.

1. Проектирование
   1. Описание исследуемой предметной области

Исследуемый предмет в данной работе - программный модуль распознавания автомобилей экстренных служб на основе модели машинного обучения. Модуль разрабатывается с использованием глубокого обучения и нейронных сетей для автоматического распознавания и идентификации автомобилей экстренных служб на изображениях и/или видео.

* 1. Назначение разрабатываемого программного модуля

Разрабатываемый программный модуль имеет следующие цели и назначение и будет разрабатываться по текущему ТЗ (ПРИЛОЖЕНИЕ А):

* Автоматическое распознавание и классификация автомобилей экстренных служб на изображениях или видео.
* Идентификация типов экстренных служб на основе особенностей автомобильной техники.
* Улучшение оперативности и эффективности процесса распознавания автомобилей экстренных служб.
* Создание Open Source базы данных с информацией о распознанных автомобилях экстренных служб.
  1. Область применения разрабатываемого программного модуля

Разрабатываемый программный модуль может найти применение в различных областях и сферах деятельности, включая:

* Системы видеонаблюдения: Модуль может быть интегрирован в системы видеонаблюдения для автоматического распознавания и идентификации автомобилей экстренных служб на записях с камер.
* Транспортная безопасность: Модуль может использоваться для мониторинга и контроля на дорогах с целью обеспечения безопасности и эффективности движения автомобилей экстренных служб.
* Оперативное реагирование: Модуль может помочь в оперативном реагировании на чрезвычайные ситуации, обеспечивая быстрое распознавание и идентификацию автомобилей экстренных служб. [11]

Чтобы лучше понять суть работы, была разработана DFD-диаграмма (Рисунок 2)

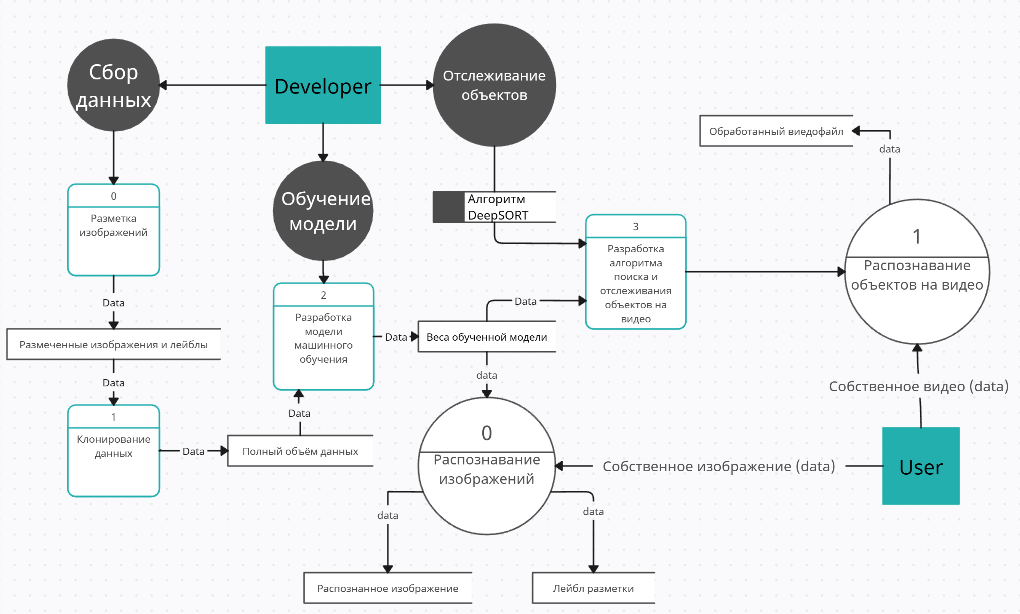


Рисунок 2 – DFD диаграмма

Для удобства взаимодействия с модулем и демонстрации его работоспособности была описана блок схема работы модуль обнаружения и отслеживания объектов на видеофайле. (Рисунок 3)

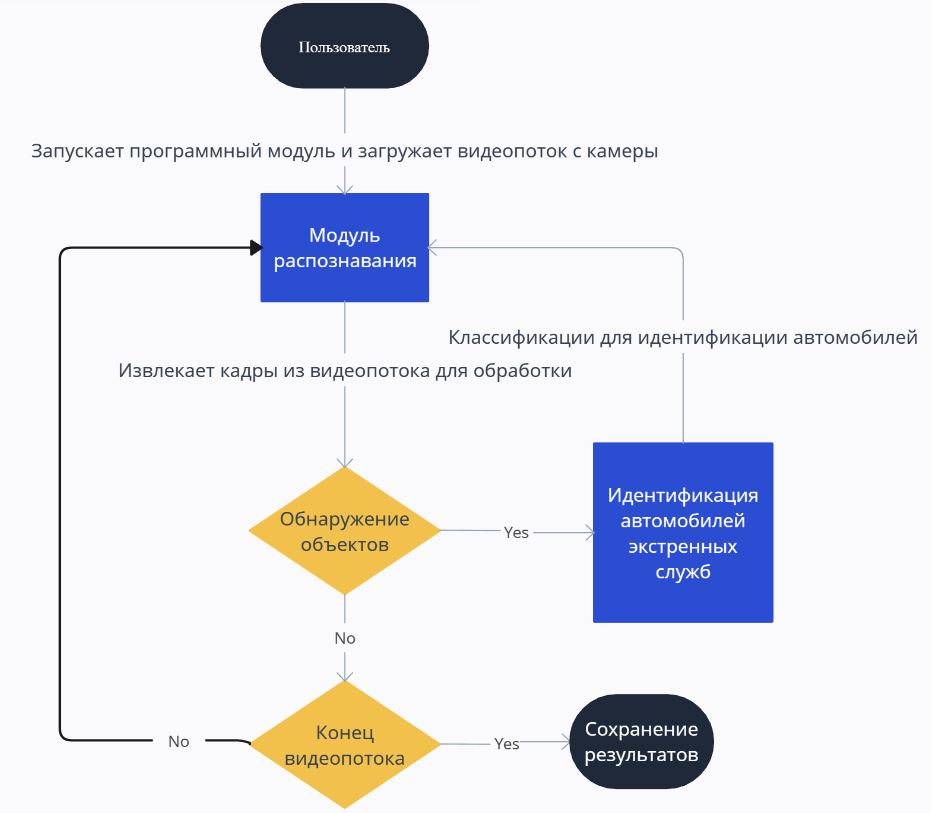


Рисунок 3 – Блок схема работы алгоритма отслеживания

* 1. Системный анализ и архитектура разрабатываемого модуля

В ходе системного анализа были рассмотрены различные варианты реализации разрабатываемого модуля распознавания автомобилей экстренных служб. Было проведено сравнение этих вариантов с учетом требований к точности распознавания, скорости обработки, работоспособности на слабом железе, адаптивности к различным условиям, а также возможности обновления и доработки модели.

Архитектура разрабатываемого модуля основана на использовании глубокого обучения и нейронных сетей YOLOv5 (You Only Look Once version 5) — это архитектура глубокой нейронной сети, предназначенная для обнаружения объектов на изображениях и в реальном времени. Она является последней версией серии алгоритмов YOLO, разработанных компанией Ultralytics. [1, 2, 4] А также для отслеживания объектов был выбран алгоритм DeepSORT (Deep Simple Online and Realtime Tracking) — это алгоритм, который был разработан командой исследователей компании DeepMind в 2017 году, предназначенный для множественного объектного отслеживания в режиме реального времени, который сочетает в себе глубокое обнаружение объектов и ассоциацию объектов во времени. Он является расширением алгоритма SORT (Simple Online and Realtime Tracking) и предназначен для устранения ложных срабатываний и обработки временных пропусков при отслеживании объектов. [5, 6, 8]

DeepSORT стал популярным в области компьютерного зрения и применяется в различных приложениях, включая видеонаблюдение, автономные автомобили, системы безопасности и другие задачи, где требуется отслеживание и идентификация объектов на видео. На данный момент работы, был выбран этот алгоритм с упором на его простоту и малую затрату ресурсов компьютера.

* 1. Анализ и заключение

Анализ предметной области и проектирование разрабатываемого программного модуля распознавания автомобилей экстренных служб позволяют сделать следующие выводы:

* Разработка программного модуля на основе глубокого обучения и нейронных сетей является актуальным и перспективным направлением.
* Разрабатываемый модуль имеет широкий спектр применения в области видеонаблюдения, транспортной безопасности и оперативного реагирования.

На основе проведенного анализа можно сделать вывод, что разрабатываемый программный модуль распознавания автомобилей экстренных служб на основе модели машинного обучения имеет большой потенциал для успешной реализации и практического применения.

1. Реализация системы
   1. Описание реализации системы

Языком программирования был выбран язык Python, так как он является лучшим языком программирования для машинного обучения на данный момент. Также средой разработки был выбран Jupyter Notebook - это интерактивная среда разработки, которая позволяет создавать и выполнять код, а также создавать документы, содержащие описательный текст, изображения, графики и результаты выполнения кода. Он широко используется для работы с языками программирования, такими как Python, R и Julia. [15]

Для дальнейшей работы необходимо было использовать следующие библиотеки:

* LabelImg - это инструмент для разметки изображений, который помогает создавать наборы данных для обучения моделей машинного обучения. [3]
* OpenCV (Open Source Computer Vision Library) — это библиотека компьютерного зрения и обработки изображений с открытым исходным кодом. Она предоставляет широкий спектр функций и алгоритмов для обработки изображений, видео, анализа и распознавания объектов, а также множество других задач в области компьютерного зрения. [5]
* Ultralytics - это набор инструментов и библиотек, разработанных одноименной компанией Ultralytics, для задач компьютерного зрения и глубокого обучения. Ultralytics предоставляет простой в использовании интерфейс для работы с некоторыми популярными моделями глубокого обучения, включая YOLO (You Only Look Once) [1, 2, 4]
* PyTorch - это фреймворк глубокого обучения, который предоставляет инструменты и библиотеки для разработки и обучения нейронных сетей. Он является одним из наиболее популярных фреймворков для работы с искусственными нейронными сетями и глубоким обучением. [13]
* Torchvision - это популярная библиотека для компьютерного зрения (computer vision), предназначенная для работы с изображениями и видео во фреймворке PyTorch. Она предоставляет различные удобные инструменты для загрузки, преобразования, аугментации и обработки данных в области компьютерного зрения. [13]
* TensorFlow - это фреймворк глубокого обучения с открытым исходным кодом, разработанный и поддерживаемый компанией Google. Он предоставляет инструменты и библиотеки для создания, обучения и развертывания искусственных нейронных сетей. [12]
  1. Подбор датасета

Для сбора изображений был проведен поиск готовых размеченных датасетов на сайте Roboflow (Рисунок 4)

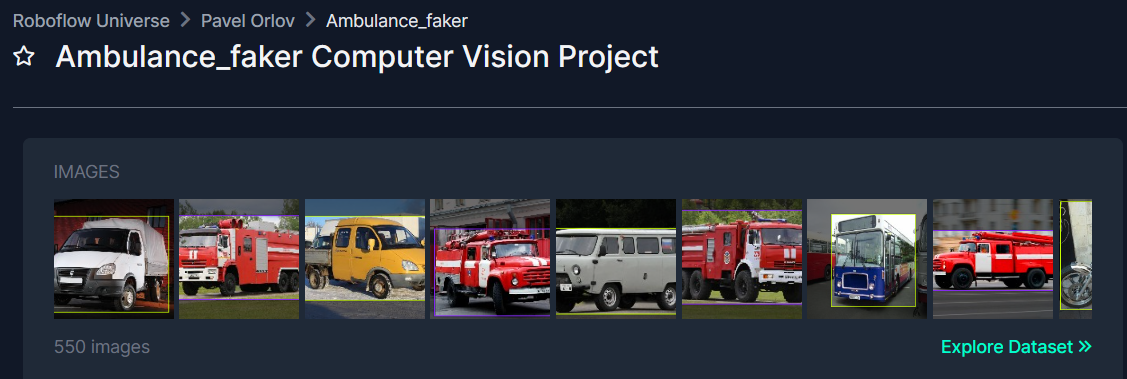


Рисунок 4 – Датасет на сайте Roboflow

Так удалось найти более 500 готовых для работы изображений. Так как такого количества было недостаточно, из интернета были выгружены еще более 200 изображений и размечены в ручном режиме с помощью инструмента LabelImg (Рисунок 5) [9]

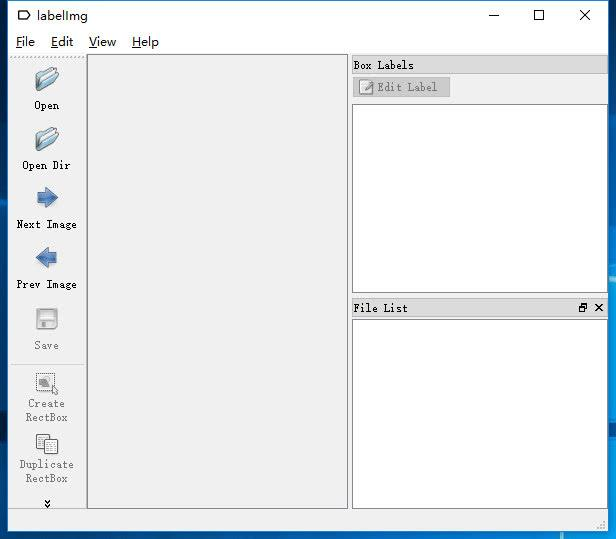


Рисунок 5 - Интерфейс LabelImg

Далее были получены более 700 размеченных изображений с лейблами, после чего, было необходимо сделать клонирование изображений для увеличения датасета.

Самым простым и эффективным способ являлось отзеркаливание изображений и их лейблов, для этого был написан скрипт, который на вход принимал изображение и лейбл, а на выходе выдавал полностью зеркальное изображение и также зеркальный лейбл с координатами для нового Bounding box`a (Рисунок 6) [10, 11]



Рисунок 6 – Зеркальное изображения и разметки

Bounding box (границы ограничивающей рамки) - это прямоугольная рамка, которая описывает положение и размер объекта или региона интереса (Region of Interest, ROI) на изображении или видео. Она определяется четырьмя параметрами: координаты верхнего левого угла (x, y) и ширина (width) и высота (height) рамки. [1]

Формула отзеркаливания достаточно проста и занимает пару строк (листинг 1), полный код находится в (ПРИЛОЖЕНИЕ Б - Листинг кода Б1)

Листинг кода 1 - отзеркаливание изображений и лейблов

flipped\_image = np.fliplr(image)

flipped\_x = 1-x

В конечном итоге получилось собрать 1647 размеченных изображений, которые дальше были разделены на 70% тренировочных, 20% валидационные и 10% тестовых данных.

* 1. Обучение модели

После сбора данных и их разделение было запущенно обучение на стандартных весах YOLOv5 с полностью стандартными настройками. В данных для обучения было представлено 4 класса: ['ambulance', 'fire', 'other', 'police'], запуск обучения первой модели был рассчитан на 96 эпох, 12 батчей с сохранением раз в 2 эпохи.

Обучение длилось более суток и на выходе достоверность предсказаний каждого типа было свыше 90%. Лучшим показателем стала машины скорой помощи - более 94%, а худшим - пожарная машина, с вероятностью 90.01%.

Немного о том, как YOLO понимает, что искать и как выделять объект на изображении: YOLO предсказывает 5 параметров (для каждого anchor box для определенной клетки)

Предсказание параметров для каждого anchor box выражены в формулах (1-3) [1]

 (1)

 (2)

 (3)

Значение переменных, используемых в формулах (1-3) приведены в (Рисунок 7) [1]

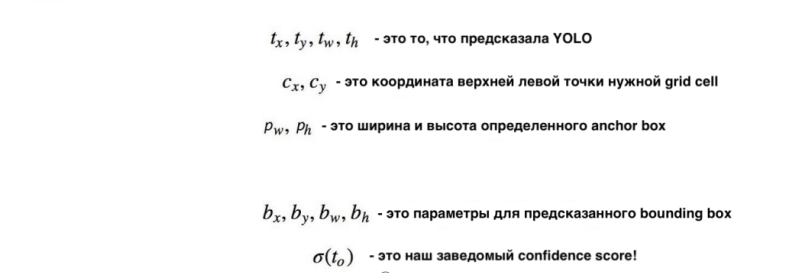


Рисунок 7 - переменные YOLO

Для более просто восприятия есть есть хорошая визуализация на эту тему (Рисунок 8) [1]

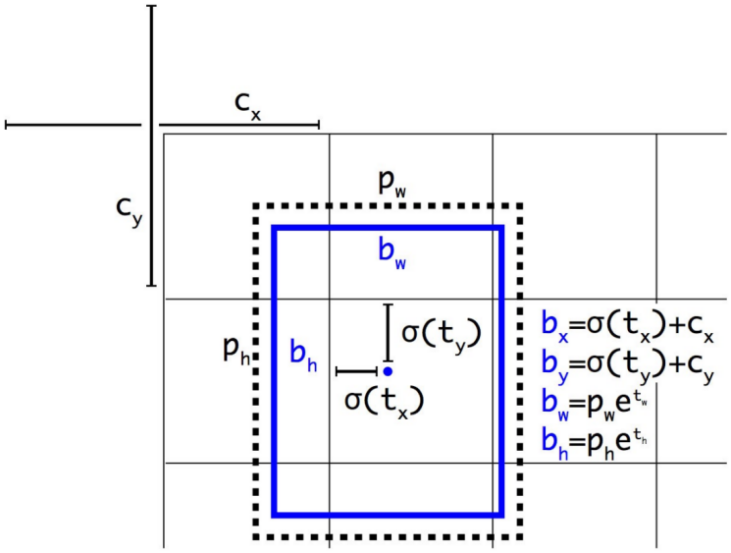


Рисунок 8 - Bounding box

* 1. Реализация отслеживания на видео

Для отслеживания объектов на видео использовался алгоритм DeepSORT с уже обученной моделью, сперва получаем ширину, высоту и частоту кадров видеопотока (листинг 2)

Листинг кода 2 - получение ширины, высоты и частоты кадров

frame\_width = int(video\_cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

frame\_height = int(video\_cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

fps = int(video\_cap.get(cv2.CAP\_PROP\_FPS))

Далее запускаем цикл по кадрам видео до момента обнаружения результата, после чего инициализируем списки ограничивающих рамок, уверенностей и идентификаторов классов, а дальше проходимся по обнаруженным объектам (листинг 3)

Листинг кода 3 - цикл по обнаруженным объектам

for data in result.boxes.data.tolist():

x1, y1, x2, y2, confidence, class\_id = data

x = int(x1)

y = int(y1)

w = int(x2) - int(x1)

h = int(y2) - int(y1)

class\_id = int(class\_id)

После чего производим фильтрацию слабых предсказаний, убеждаясь, что уверенность больше минимальной уверенности (листинг 4)

Листинг кода 4 - фильтрация слабых предсказаний

if confidence > conf\_threshold:

bboxes.append([x, y, w, h])

confidences.append(confidence)

class\_ids.append(class\_id)

Далее в работу вступает алгоритм DeepSORT, для этого преобразуем обнаружения в формат deepsort`a (листинг 5)

Листинг кода 5 - преобразуем обнаружения в формат DeepSort

dets = []

for bbox, conf, class\_name, feature in zip(bboxes, confidences, names, features):

dets.append(Detection(bbox, conf, class\_name, feature))

Вследствие чего попадаем в цикл по отслеживанию за объектами и после получаем ограничивающую рамку объекта, имя объекта и идентификатор трека (листинг 6)

Листинг кода 6 - цикл по отслеживаемым объектам

for track in tracker.tracks:

if not track.is\_confirmed() or track.time\_since\_update > 1:

Continue

bbox = track.to\_tlbr()

track\_id = track.track\_id

class\_name = track.get\_class()

В конце рисуем ограничивающую рамку объекта, имя предсказанного объекта и идентификатор трека (листинг 7)

Листинг кода 7 - рисуем ограничивающую рамку объекта

text = str(track\_id) + " - " + class\_name

cv2.rectangle(frame, (x1, y1), (x2, y2), (B, G, R), 2)

cv2.rectangle(frame, (x1 - 1, y1 - 20), (x1 + len(text) \* 12, y1), (B, G, R), -1)

cv2.putText(frame, text, (x1 + 5, y1 - 8), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2)

На выходе мы уже получаем кадр с разметкой (Рисунок 9)

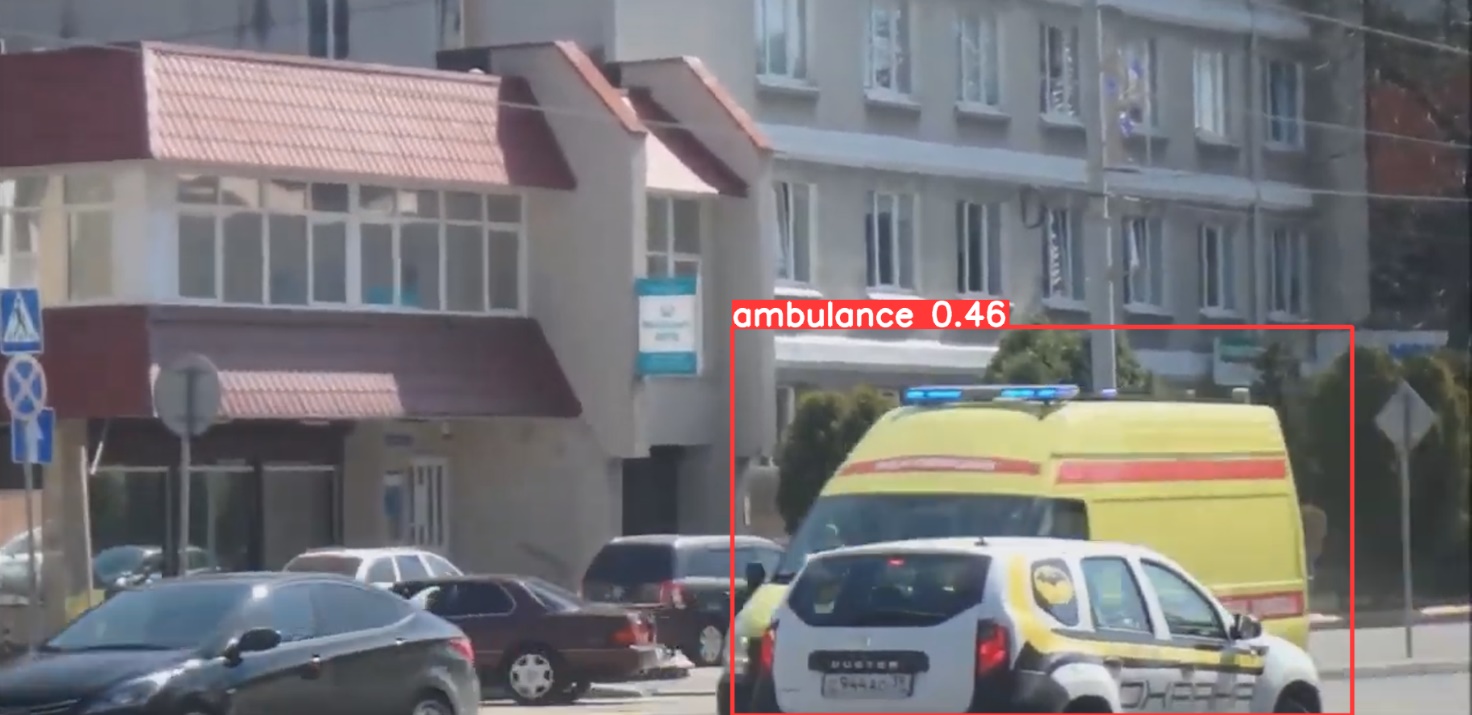


Рисунок 9 – Видеокадр с распознаванием

Процедура повторяется пока не закончится обработка видеофайла или не будет намеренно остановлено, полный код обработки видеофайла находится в (ПРИЛОЖЕНИЕ Б - Листинг кода Б2).

* 1. Тестирование

В ходе ручной и полуавтономной проверки, сама модель показала достойны результат, были лишь единичные случаи некорректного распознавания красных машин, модель принимала их за пожарные машины (Рисунок 10)



Рисунок 10 – Видеокадр с ложным распознаванием

В данном случаи, модель считает, что троллейбус бело-красного цвета и вытянутой формы на 66% похож на пожарную машину.

Далее было проведено ручное тестирование на нескольких видеофайлах, результат на которых был не столь однозначен, нежели на обычных изображениях. Участились некорректные срабатывания не только на красные машины, но и редкие случаи полицейских машин. После тестирования было решено дообучить существующую модель, добавив в нее больше изображений с красными машинами и расширить автопарк полицейских авто разного поколения.

После дообучения, фактический результат, по проверке на изображениях, улучшился, но на видеофайлах это не так сильно сказалось, поэтому на данном этапе разработке, было принято решение не продолжать наращивать датасет, а работать с существующим.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был разработан программный модуль распознавания автомобилей экстренных служб на основе модели машинного обучения. Были проведены анализ предметной области, обзор аналогов, разработка и обоснование архитектуры модуля, собраны и размечены изображения, выполнено обучение модели, а также проведение исследований и тестирование для проверки работоспособности и эффективности модели. Решения, представленные в данной выпускной квалификационной работе, достаточно полны и соответствуют поставленным задачам.

Данная работа имеет практическую значимость, так как разработанный программный модуль может быть применен в системах видеонаблюдения и обеспечения транспортной безопасности. Модуль позволяет автоматически распознавать и идентифицировать автомобили экстренных служб, что способствует более эффективному реагированию на аварийные ситуации и повышению безопасности на дорогах.

Таким образом, выполненная выпускная квалификационная работа успешно достигла своей цели, предоставив полные исследования, разработку и анализ программного модуля распознавания автомобилей экстренных служб. Результаты работы имеют практическую значимость и могут быть использованы в реальных системах для повышения эффективности и безопасности автомобильного движения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Как работает Object Tracking на YOLO и DeepSort // habr.com URL: https://habr.com/ru/articles/514450/ (дата обращения: 12.06.2023).
2. Large Scale Object Detection & Tracking with YOLOv5 Package // medium.com URL: https://medium.com/analytics-vidhya/large-scale-object-detection-tracking-with-yolov5-package-31e715d84121 (дата обращения: 12.06.2023).
3. О LabelImg // github.com URL: https://github.com/heartexlabs/labelImg (дата обращения: 12.06.2023).
4. Подробное руководство по ультралитикам YOLOv5 // ultralytics.com URL: https://docs.ultralytics.com/yolov5/ (дата обращения: 12.06.2023).
5. Документация openCV // opencv.org URL: https://docs.opencv.org/4.x/ (дата обращения: 12.06.2023).
6. Книга "Deep Learning for Computer Vision" by Adrian Rosebrock // (дата обращения: 11.06.2023).
7. Учебное пособие "Computer Vision: Algorithms and Applications" by Richard Szeliski // (дата обращения: 11.06.2023).
8. Сборники материалов конференции "Conference on Computer Vision and Pattern Recognition (CVPR)" // (дата обращения: 11.06.2023).
9. Разметка данных в машинном обучении: процесс, разновидности и рекомендации // habr.com URL: https://habr.com/ru/articles/678524/ (дата обращения: 10.06.2023).
10. Глубокое обучение. Погружение в мир нейронных сетей" (Deep Learning. Ian Goodfellow, Yoshua Bengio, Aaron Courville) Гудфеллоу Я. Бенджио И. Курвилль А. // (дата обращения: 10.06.2023).
11. Научный журнал "International Journal of Computer Vision" // springer.com URL: https://www.springer.com/journal/11263? (дата обращения: 08.06.2023).
12. API Documentation Tensorflow // tensorflow.org URL: https://www.tensorflow.org/api\_docs (дата обращения: 07.06.2023).
13. PyTorch documentation // pytorch.org URL: https://pytorch.org/docs/stable/index.html (дата обращения: 07.06.2023).
14. Книга «Python и машинное обучение» Себастьян Рашка. // (дата обращения: 04.06.2023).
15. Python Documentation // python.org URL: https://www.python.org/doc/ (дата обращения: 01.06.2023).

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А.

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**Инженерная школа цифровых технологий**

**Направление подготовки «Информатика и вычислительная техника»**

**На тему: «Разработка программного модуля распознавания автомобилей экстренных служб на основе модели машинного обучения»**

Студент: Проскуряков Богдан Григорьевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

Руководитель: Усманов Руслан Талгатович \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия имя отчество) (личная подпись)

г. Ханты-Мансийск

2023 год

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**НА РАЗРАБОТКУ ИНФОРМАЦИОННОЙ СИСТЕМЫ МАШИННОГО ОБУЧЕНИЯ**

1. **Общие сведения Модель машинного обучения с использованием компьютерного зрения**

Полное наименование: Программный модуля распознавания автомобилей экстренных служб на основе модели машинного обучения (далее Модель)

**Наименование организации – заказчика АС, наименование организации-разработчика (при наличии сведений о ней)**

Федеральное государственное бюджетное образовательное учреждение высшего образования «Югорский государственный университет» (ЮГУ)

**Перечень документов, на основании которых создается АС**

Перечень документов, на основании которых создаётся АИС:

1. Приказ ЮГУ об утверждении тем выпускных квалификационных работ от \_\_\_\_\_\_\_\_\_\_\_\_ №\_\_\_.
2. ГОСТ 34.602-2020 «Техническое задание на создание автоматизированной системы»
3. ГОСТ Р 59895-2021 Технологии искусственного интеллекта в образовании. Общие положения и терминология
4. ГОСТ 34.601-90 «Автоматизированные системы. Стадии создания».

**Плановые сроки начала и окончания работ по созданию АС**

Плановый срок начала работ по созданию АИС \_\_\_\_\_

Плановый срок окончания работ \_\_\_\_\_\_\_\_\_

(даты проставляются в соответствии с календарным графиком)

1. **Цели и назначение создания автоматизированной системы**

**Цели создания автоматизированной системы**

Цель создания АИС:

Улучшить оперативности работы экстренных служб путём разработки модуля модели машинного обучения для общедоступного использования в научных работах и интеграции в системы контрольно пропускных пунктов.

Критерием оценки достижения цели создания является способность модели распознавать машины экстренных служб с графических объектов с вероятностью более 90%.

**Назначение автоматизированной системы**

Модель предназначена для улучшить оперативности работы экстренных служб и последующим сокращение затрат на услуги работников КПП.

Назначение АИС:

Обработка видеоряда и отдельных изображений с поиском и выделением на них искомых объектов.

1. **Характеристика объектов автоматизации**

**Основные сведения об объекте автоматизации**

Система представляет собой алгоритм, который должен анализировать графические данные и учится на них для принятия решения.

Данные для обучения должны быть загружены единожды, с последующим (самостоятельным) дообучением. Сами данные должны быть собраны в произвольном режиме путём загрузки изображений из сети Интернет. После чего, данные, должны подлежать первичной обработке, после чего, предусмотреть возможность дублирования для расширения датасета, если собранных изображений будет недостаточно. Конечным этапом обработки будет разметка изображений для обучения модели.

1. **Требования к автоматизированной системе**

**Требования к структуре АС в целом**

**Требования к функциям (задачам), выполняемым АС**

**Перечень функций и составляющих их задач, реализуемых в АИС**

АИС реализует следующие функции:

1. модель должна быть обучена на различных типах машин экстренных служб (скорая помощь, пожарная машина, полицейский автомобиль и не относящиеся к экстренным службам машины), чтобы иметь возможность правильно идентифицировать их на фото-видео объектах.
2. Классификация машин экстренных служб по типу: модель должна быть обучена определять тип машины экстренной службы (скорая помощь, пожарная машина, полицейский автомобиль и т.д.).

Задачи, которые необходимо решить для реализации этих функций, включают в себя следующее:

1. Сбор и анализ датасета фото-видео объектов, содержащего изображения машин экстренных служб и других транспортных средств, а также аннотации для этого датасета.
2. Подготовка данных для обучения модели, включая обработку и фильтрацию изображений, выделение признаков и создание признаковых описаний объектов.
3. Выбор и настройка алгоритмов машинного обучения, которые будут использоваться для обучения модели, а также подбор гиперпараметров и архитектуры модели.
4. Обучение модели на обучающем наборе данных, оценка качества модели на валидационном наборе данных и настройка модели для повышения её точности и производительности
5. При необходимости, дообучении модели, путём догрузки изображений на уже существующую модель.

**Требования к видам обеспечения АС**

**Математическое обеспечение АС**

Обработка и анализ изображений: это задача обработки изображений и извлечения признаков из них, которые затем должны быть использованы для распознавания объектов. Для её решения должны быть использованы методы обработки изображений, такие как свертка, сегментация, дескрипторы особых точек и т.д.

Оценка качества модели: для оценки качества модели должны использовать различные метрики, такие как точность, полнота, F1-мера и т.д.

**Информационное обеспечение АС**

Информационное обеспечение должно содержать:

1. Выбор гиперпараметров модели: для определения оптимальных значений гиперпараметров модели могут быть использованы методы кросс-валидации, случайного поиска или оптимизации на основе градиентов.
2. Библиотеки для машинного обучения: для реализации модели машинного обучения можно использовать различные библиотеки, такие как TensorFlow, PyTorch, Keras, Scikit-learn и др.

**Лингвистическое обеспечение**

Разрабатываемая модель машинного обучения будет реализована на языке программирования Python

**Требования к языкам, используемым в АС, и возможности расширения набора языков (при необходимости)**

Языки программирования: для реализации модели машинного обучения будет использоваться язык программирования, Python. Нормативно-справочная документация разрабатываемой платформы, а также прикладное программное обеспечение взаимодействия с пользователем должны представляться на русском языке.

**Требования к разрабатываемому программному обеспечению**

Разрабатываемое программное обеспечение АИС должно предусматривать возможность дальнейшей функциональной доработки и развития.

**Требования к надежности**

Надёжной моделью машинного обучения считается модель, показатель вероятности которой более 90%, то есть, это не отменяет тот факт, что есть вероятность некорректного срабатывания системы.

1. **Состав и содержание работ по созданию автоматизированной системы**

Перечень этапов услуг по созданию АИС включает следующие стадии:

1. Предпроектная стадия, включающая этап «Разработка предпроектной документации на систему и её части».
2. Стадия «Сбор данных» включающая:
3. Сбор изображений машин экстренных служб из интернета.
4. Сбор изображений не относящихся к машинам экстренных служб.
5. Стадия «Обработка данных» включающая:
6. Дублирование данных.
7. Разметка изображений.
8. Стадия «Разработка классификатора», включающая:
9. Разработка алгоритма для классификации данных.
10. Стадия «Тестирование», включающая:
11. Тестирование модели на тестовых данных.
12. Тестирование модели на валидационных данных.
13. этап «Сдача и приёмка услуг».

Сроки оказания услуг по созданию робота определяются календарным планом графиком оказания услуг, приведённым в Таблице 1.

Таблица 1 — Календарный план график оказания услуг

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | **Наименование стадии, этапа** | **Документ, подтверждающий оказание услуг по этапу** | **Начало - окончание (месяц,**  **год)** |
| **1. Предпроектная стадия** | | | |
| 1.1 | Разработка предпроектной документации на систему и её части | Технический проект на модель машинного обучения для распознавания машин экстренных служб | 01.2023 04.2023 |
| **2. Сбор данных** | | | |
| 2.1 | Сбор изображений машин экстренных служб из интернета | Комплект рабочей документации на СИСТЕМУ и её части  Акт о разработке комплекта Рабочей документации на СИСТЕМУ и её части | 02.2023 04.2023 |
| 2.2 | Сбор изображений не относящихся к машинам экстренных служб | Комплект рабочей документации на СИСТЕМУ и её части  Акт о разработке комплекта Рабочей документации на СИСТЕМУ и её части | 02.2023 04.2023 |
| **3. Обработка данных** | | | |
| 3.1 | Дублирование данных | Комплект рабочей документации на СИСТЕМУ и её части  Акт о разработке комплекта Рабочей документации на СИСТЕМУ и её части | 03.2023 05.2023 |
| 3.2 | Разметка изображений | Комплект рабочей документации на СИСТЕМУ и её части  Акт о разработке комплекта Рабочей документации на СИСТЕМУ и её части | 03.2023 05.2023 |
| **4. Разработка классификатора** | | | |
| 4.1 | Разработка алгоритма для классификации данных | Комплект рабочей документации на СИСТЕМУ и её части  Акт о разработке комплекта Рабочей документации на СИСТЕМУ и её части | 04.2023 05.2023 |
| **5. Тестирование** | | | |
| 5.1 | Тестирование модели на тестовых данных | Отчёт о тестировании модели машинного обучения на тестовых данных | 05.2023 05.2023 |
| 5.2 | Тестирование модели на валидационных данных | Отчёт о тестировании модели машинного обучения на валидационных данных | 06.2023 06.2023 |
| **6. Сдача и приёмка услуг** | | | |
| 5.4 | Сдача и приёмка услуг | Программа и методика приёмочных испытаний робота, Отчётная документация. Лицензионные соглашения. Акты приёма-передачи прав использования ПО. Товарные накладные. Документ о приемке. | 06.2023 06.2023 |

ПРИЛОЖЕНИЕ Б

Листинг кода Б1 - алгоритм дублирование изображений методом отзеркаливания

import cv2

import numpy as np

import os

def flip\_image(image\_path, labels\_path):

# Загрузка изображения

image = cv2.imread(image\_path)

# Отражение изображения по горизонтали

flipped\_image = np.fliplr(image)

# Получение ширины и высоты изображения

height, width, \_ = image.shape

# Открытие файла с лейблами

with open(labels\_path, 'r') as file:

lines = file.readlines()

# Создание нового файла для отраженных лейблов

flipped\_labels\_path = os.path.splitext(labels\_path)[0] + '\_flipped.txt'

with open(flipped\_labels\_path, 'w') as file:

for line in lines:

# Разделение строк на отдельные значения

values = line.strip().split(' ')

class\_id = values[0]

x = float(values[1])

y = float(values[2])

w = float(values[3])

h = float(values[4])

# Изменение координат разметки для отраженного изображения

# flipped\_x = w/2+(1-(x+w/2)) # изменение координаты X и ориентации

flipped\_x = 1-x

flipped\_y = y

flipped\_w = w

flipped\_h = h

# Запись отраженных координат в новый файл

file.write(f'{class\_id} {flipped\_x} {flipped\_y} {flipped\_w} {flipped\_h}\n')

# Сохранение отраженного изображения

flipped\_image\_path = os.path.splitext(image\_path)[0] + '\_flipped.png'

cv2.imwrite(flipped\_image\_path, flipped\_image)

print(f'Отраженное изображение сохранено: {flipped\_image\_path}')

print(f'Отраженные лейблы сохранены: {flipped\_labels\_path}')

# Пример использования

image\_path = 'C:\\Users\\bogdan\\jupu\\dip\\ultratest\\image\\t18.png'

labels\_path = 'C:\\Users\\bogdan\\jupu\\dip\\ultratest\\label\\t18.txt'

flip\_image(image\_path, labels\_path)

Листинг кода Б2 - Алгоритм обработки видеофайла

import numpy as np

import datetime

import cv2

from ultralytics import YOLO

from deep\_sort.deep\_sort.tracker import Tracker

from deep\_sort.deep\_sort import nn\_matching

from deep\_sort.deep\_sort.detection import Detection

from deep\_sort.tools import generate\_detections as gdet

def create\_video\_writer(video\_cap, output\_filename):

# получаем ширину, высоту и частоту кадров видеопотока

frame\_width = int(video\_cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

frame\_height = int(video\_cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

fps = int(video\_cap.get(cv2.CAP\_PROP\_FPS))

# инициализируем объект FourCC и видеозаписи

fourcc = cv2.VideoWriter\_fourcc(\*'MP4V')

writer = cv2.VideoWriter(output\_filename, fourcc, fps,

(frame\_width, frame\_height))

return writer

# задаем некоторые параметры

conf\_threshold = 0.5

max\_cosine\_distance = 0.4

nn\_budget = None

# Инициализируем объекты захвата видео и записи видео

video\_cap = cv2.VideoCapture("C:\\Users\\bogdan\\jupu\\dip\\videotest1.mp4")

writer = create\_video\_writer(video\_cap, "outtest1.mp4")

# Инициализируем модель YOLOv8 с использованием стандартных весов

model = YOLO("config/best.pt")

# Инициализируем трекер DeepSort

model\_filename = "config/mars-small128.pb"

encoder = gdet.create\_box\_encoder(model\_filename, batch\_size=1)

metric = nn\_matching.NearestNeighborDistanceMetric(

"cosine", max\_cosine\_distance, nn\_budget)

tracker = Tracker(metric)

# загружаем классы COCO, на которых обучена модель YOLO

classes\_path = "config/coco.names"

with open(classes\_path, "r") as f:

class\_names = f.read().strip().split("\n")

# создаем список случайных цветов для каждого класса

np.random.seed(42) # чтобы получить одни и те же цвета

colors = np.random.randint(0, 255, size=(len(class\_names), 3)) # (80, 3)

# цикл по кадрам видео

while True:

# начальное время для вычисления FPS

start = datetime.datetime.now()

ret, frame = video\_cap.read()

# если кадр отсутствует, значит достигнут конец видео

if not ret:

print("Конец видеофайла...")

break

############################################################

### Обнаружение объектов на кадре с помощью модели YOLO ###

############################################################

# запускаем модель YOLO на кадре

results = model(frame)

# цикл по результатам

for result in results:

# инициализируем списки ограничивающих рамок, уверенностей и идентификаторов классов

bboxes = []

confidences = []

class\_ids = []

# цикл по обнаруженным объектам

for data in result.boxes.data.tolist():

x1, y1, x2, y2, confidence, class\_id = data

x = int(x1)

y = int(y1)

w = int(x2) - int(x1)

h = int(y2) - int(y1)

class\_id = int(class\_id)

# cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

# фильтрация слабых предсказаний, убеждаясь, что уверенность больше минимальной уверенности

if confidence > conf\_threshold:

bboxes.append([x, y, w, h])

confidences.append(confidence)

class\_ids.append(class\_id)

############################################################

### Отслеживание объектов на кадре с помощью DeepSort ###

############################################################

# получаем названия обнаруженных объектов

names = [class\_names[class\_id] for class\_id in class\_ids]

# получаем признаки обнаруженных объектов

features = encoder(frame, bboxes)

# преобразуем обнаружения в формат DeepSort

dets = []

for bbox, conf, class\_name, feature in zip(bboxes, confidences, names, features):

dets.append(Detection(bbox, conf, class\_name, feature))

# запускаем трекер на обнаружениях

tracker.predict()

tracker.update(dets)

# цикл по отслеживаемым объектам

for track in tracker.tracks:

if not track.is\_confirmed() or track.time\_since\_update > 1:

continue

# получаем ограничивающую рамку объекта, имя

# объекта и идентификатор трека

bbox = track.to\_tlbr()

track\_id = track.track\_id

class\_name = track.get\_class()

# преобразуем ограничивающую рамку в целые числа

x1, y1, x2, y2 = int(bbox[0]), int(bbox[1]), int(bbox[2]), int(bbox[3])

# получаем цвет, соответствующий имени класса

class\_id = class\_names.index(class\_name)

color = colors[class\_id]

B, G, R = int(color[0]), int(color[1]), int(color[2])

# рисуем ограничивающую рамку объекта, имя

# предсказанного объекта и идентификатор трека

text = str(track\_id) + " - " + class\_name

cv2.rectangle(frame, (x1, y1), (x2, y2), (B, G, R), 2)

cv2.rectangle(frame, (x1 - 1, y1 - 20),

(x1 + len(text) \* 12, y1), (B, G, R), -1)

cv2.putText(frame, text, (x1 + 5, y1 - 8),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2)

############################################################

### Некоторая постобработка для отображения результатов ###

############################################################

# конечное время для вычисления FPS

end = datetime.datetime.now()

# вычисляем кадры в секунду и выводим на кадр

fps = f"FPS: {1 / (end - start).total\_seconds():.2f}"

cv2.putText(frame, fps, (50, 50),

cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 0, 255), 8)

cv2.imshow("Output", frame)

# записываем кадр на диск

writer.write(frame)

if cv2.waitKey(1) == ord("q"):

break

# освобождаем ресурсы, связанные с захватом видео, записью видео и закрываем все окна

video\_cap.release()

writer.release()

cv2.destroyAllWindows()