## **МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

## **«Московский государственный технологический университет**

**«СТАНКИН»(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт информационных технологий** | **Кафедрауправления и информатики в**  **технических системах** |

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ ПО ДИСЦИПЛИНЕ

«МАШИННОЕ ОБУЧЕНИЕ И ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ»

Студентов 3 курса, Группы ИДБ-21-09

Власкина Сергея Алексеевича

Болдина Александра Владимировича

Комарова Владислава Сергеевича

НА ТЕМУ

**Видеоаналитика.**

|  |  |
| --- | --- |
| Направление: | 09.03.03 Прикладная информатика |
| Профиль подготовки: | Управление данными |

|  |  |  |
| --- | --- | --- |
| Отчет сдан « » 20 24 г. | | |
| Оценка | |  |
| Преподаватель | Ибатулин М.Ю., ст. преподаватель |  |
|  | *(Ф.И.О., должность, степень, звание.)* | *(подпись)* |

МОСКВА 2024

## **МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

## **«Московский государственный технологический университет «СТАНКИН» (ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт информационных технологий** | **Кафедра управления и информатики**  **в технических системах** |

**ЗАДАНИЕ**

На курсовую работу по дисциплине

**«**Машинное обучение и интеллектуальные системы**»**

Студенту: \_,

Группа ИДБ-21-

*Тема работы:* \_ Видеоаналитика.

*Исходные данные:*

1.

2.

3.

4.

*5.*

*Вопросы, подлежащие разработке:*

1.

2.

*Содержание графической части: (при необходимости)*

Задание выдано « » \_ 2024г.

Срок защиты работы « » \_ 2024г.

Руководитель /ст. преподаватель Ибатулин М.Ю./

Оглавление

[**МИНОБРНАУКИ РОССИИ** 1](#_Toc167487309)

[**«Московский государственный технологический университет** 1](#_Toc167487310)

[**МИНОБРНАУКИ РОССИИ** 2](#_Toc167487311)

[**«Московский государственный технологический университет «СТАНКИН» (ФГБОУ ВО «МГТУ «СТАНКИН»)** 2](#_Toc167487312)

[1.1 Цели и задачи курсового проекта 5](#_Toc167487313)

[1.2 Индивидуальные задания на курсовой проект 5](#_Toc167487314)

[1. Общие сведения 5](#_Toc167487315)

[1.1. Наименование проекта 5](#_Toc167487316)

[1.2. Список принятых сокращений 5](#_Toc167487317)

[2. Назначения, цели и сроки проекта 6](#_Toc167487318)

[2.1. Назначение проекта 6](#_Toc167487319)

[2.2. Цели проекта 6](#_Toc167487320)

[2.3. Задачи проекта 6](#_Toc167487321)

[2.4. Результаты проекта 7](#_Toc167487322)

[2.5. Требования к промежуточной отчетности 7](#_Toc167487323)

[2.6. Сроки проекта 8](#_Toc167487324)

[3. Требования к документированию 8](#_Toc167487325)

[2. Изучение предметной области и постановка задачи (регрессии, классификации, кластеризации); 8](#_Toc167487326)

[3. Сбор и обработка данных; 9](#_Toc167487327)

[4.  Разведочный анализ данных 9](#_Toc167487328)

[5.  Построение моделей и алгоритмов их машинного обучения, включая разработку искусственных нейронных сетей; 13](#_Toc167487329)

[**6.**  **Интерпретация и использование модели машинного обучения** 14](#_Toc167487330)

[**7. Прототип решения** 15](#_Toc167487331)

## 1.1 Цели и задачи курсового проекта

Курсовая работа является самостоятельной работой студента, завершающей изучение курса «Машинное обучение и интеллектуальные системы».

Цель курсовой работы – научить студента применять теоретические знания для решения практических задач в составе проекта анализа данных. Курсовая работа включает программные решения задач математико-статистического моделирования и машинного обучения моделей на языках программирования (например Pyton).

### 1.2 Индивидуальные задания на курсовой проект

Выбор темы курсовой работы осуществляется в соответствии с перечнем заданий и утверждается преподавателем.

Курсовая работа состоит из 5 разделов/**заданий** по разделам.

1. Подготовка технического задания на систему. Пример:

## 1. Общие сведения

### 1.1. Наименование проекта

«Модель распознавания действий с вагон-цистерной на эстакаде».

### 1.2. Список принятых сокращений

|  |  |
| --- | --- |
| Термин | Расшифровка/Определение |
| Компания | Команда студентов МГТУ «Станкин» |
| Заказчик | МГТУ «Станкин» |

## 2. Назначения, цели и сроки проекта

### 2.1. Назначение проекта

Целью этого проекта является создание нейронной сети, которая распознает происходящие на видеоролике действия. Модель определяет открытое и закрытое состояние люка, поднятое и опущенное состояние моста, человека, часть следующего вагона и сцеп. После определения объектов модель определяет происходящее на видеоролике событие. Модель может распознать такие события как: загрузка поезда, наличие человека и заезд или выезд вагон-цистерны. В конечном итоге определение всех этих событий позволит уменьшить количество несчастных случаев во время работы и повысить эффективность контроля жизненного цикла вагон-цистерны.

### 2.2. Цели проекта

|  |  |
| --- | --- |
| Цель проекта | Метрика |
| Реализовать и протестировать модель, распознающую объекты на видеоролике и анализирующую события, происходящие на основе распознанных объектов. | 1. Работоспособность решения подтверждена минимум на 3 видеороликах;  2. Качество моделей оценивается:  2.1 Экспертно;  2.2 процент уверенности модели в распознанном объекте более 80%. |

Для того, чтобы распознавать объекты с высоким процентом уверенности, вручную размечаются объекты на обучающей выборке и обучается модель, используя заранее размеченные объекты.

### 2.3. Задачи проекта

В рамках проекта требуется выполнить следующие задачи:

• Идентификация определяемых объектов;

• Обработка и разметка данных;

• Анализ возможных событий;

• Обзор и сравнительный анализ существующих решений по распознаванию объектов;

• Реализация и адаптация наилучшей модели или разработка собственной, в случае отсутствия подходящей под критерии модели;

• Тестирование модели на данных Заказчика, до этого никак не задействованных в создании алгоритма;

• Подготовка рекомендаций по дальнейшей доработке алгоритма;

• Подготовка итоговых материалов по проекту (отчет, презентация).

### 2.4. Результаты проекта

Результаты проекта:

1. Программный код модели

2. Чат-бот для определения событий

3. Итоговый отчёт, включающий в себя:

• Результаты анализа и предобработки данных;

• Метрики качества работы алгоритмов и необходимых эвристик;

• Сравнение алгоритмов и описание наилучшего с точки зрения метрик решения;

• Границы применимости алгоритма;

• Рекомендации по доработке алгоритма;

• Бэклог на следующий этап работ;

4. Демо-ролик работы алгоритма;

5. Презентация с выводами по итогу проекта, рекомендациями и заключением по дальнейшему развитию проекта.

### 2.5. Требования к промежуточной отчетности

В рамках проекта осуществляется взаимодействие между представителями Заказчика и Исполнителя путем оговоренной еженедельной конференцсвязи для демонстрации результатов еженедельного спринта в формате презентации. Вопросы, возникающие в ходе реализации проекта, уточняются между указанными лицами по мере возникновения.

### 2.6. Сроки проекта

Результаты по проекту необходимо передать и согласовать с Заказчиком в срок до **25.05.2024 г**.

## 3. Требования к документированию

По результатам реализации НИОКР-проекта необходимо предоставить:

1. Отчет, в формате .*doсx*;

2. Демо-ролик, в формате .*avi*;

3. Презентация об итогах реализации проекта, в формате .*pptx*.

### 2. Изучение предметной области и постановка задачи (регрессии, классификации, кластеризации);

В рамках задачи была обучена нейронная сеть YOLOv8, которая распознает действия, связанные с вагон-цистерной. Для этого было проведено исследование предметной области, чтобы изучить особенности и найти закономерности в выполняемых с вагон-цистернами действиями в депо.

В результате изучения предметной области, была поставлена задача классификации. Это значит, что были найдены объекты, которые нейронная сеть должна распознавать. К таким объектам относятся:

* Вагон-цистерна
* Поднятый мост
* Человек
* Опущенный мост
* Сцепка
* Открытый люк
* Закрытый люк

После распознавания объектов нейронная сеть должна научиться анализировать полученные данные и понимать какие действия происходят на видеоролике. К действиям, которые нейронная сеть должна уметь распознавать относятся:

* Заезд поезда в депо
* Выезд поезда из депо
* Наличие человека рядом с камерой
* Отсутствие действий

## 3. Сбор и обработка данных;

Данные для выполнения поставленной задачи были взяты с прошлогоднего хакатона: “Sibur Challenge 2023”.

Для обучения нейронной сети был взят датасет видеороликов с вагон-цистернами, на которых вагон-цистерны присутствуют в различных условиях. После этого видеоролики были разбиты на кадры и сгруппированы в разные места для создания выборок для обучения и тестирования. Для обучения модели необходимо произвести разметку изображений и создать обучающую выборку с размеченными данными. Для разметки изображений использовалась программа CVAT (Computer Vision Annotation Tool). Процедура разметки заключается в том, что на изображении каждый объект выделяется рамкой и помечается соответствующим классом.

После создания обучающей выборки модель была обучена на этой выборке с использованием предобученной нейронной сети YOLOv8 и алгоритма стохастического градиентного спуска (SGD). SGD - это эффективный оптимизационный алгоритм для машинного обучения. Работает он путём выбора случайным образом набора пачки тренировочных данных и вычисления градиента потерь. Затем алгоритм обновит веса в направлении обратном градиенту, с шагом который зависит от скорости обучения. В обучении модели Yolo SGD используется для оптимизации функции потерь, которая измеряет разницу между фактическими и предсказанными параметрами. Для проверки качества получившейся модели, она была была проверена на тестовой выборке.

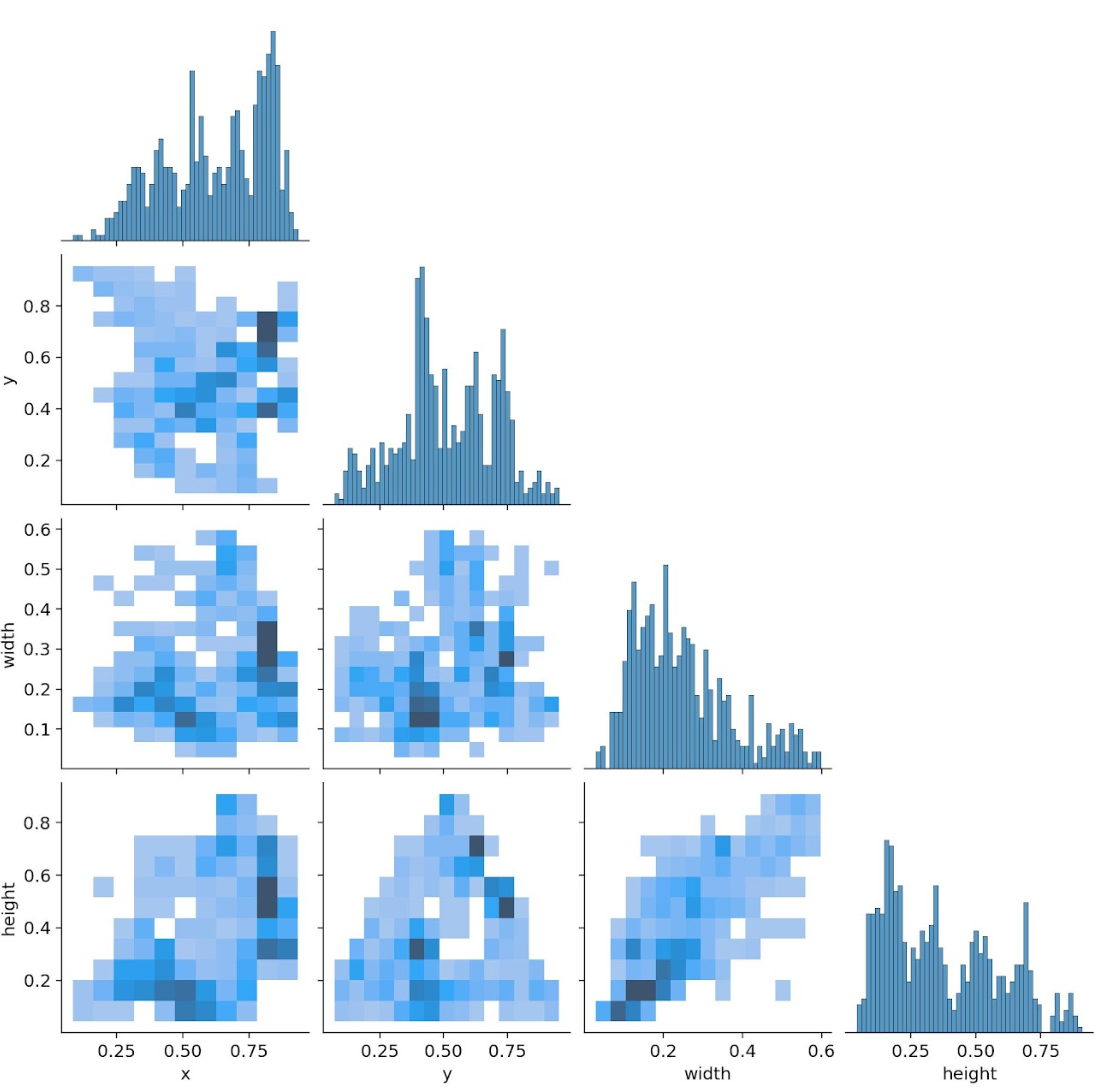
В конечном итоге, была создана обученная модель на базе нейронной сети YOLOv8, которая способна распознавать объекты и на основании полученных данных определять действия, связанные с вагон-цистернами, которые происходят на изображениях.

## 4.  Разведочный анализ данных

Далее проводится анализ графиков полученных при переобучении модели:

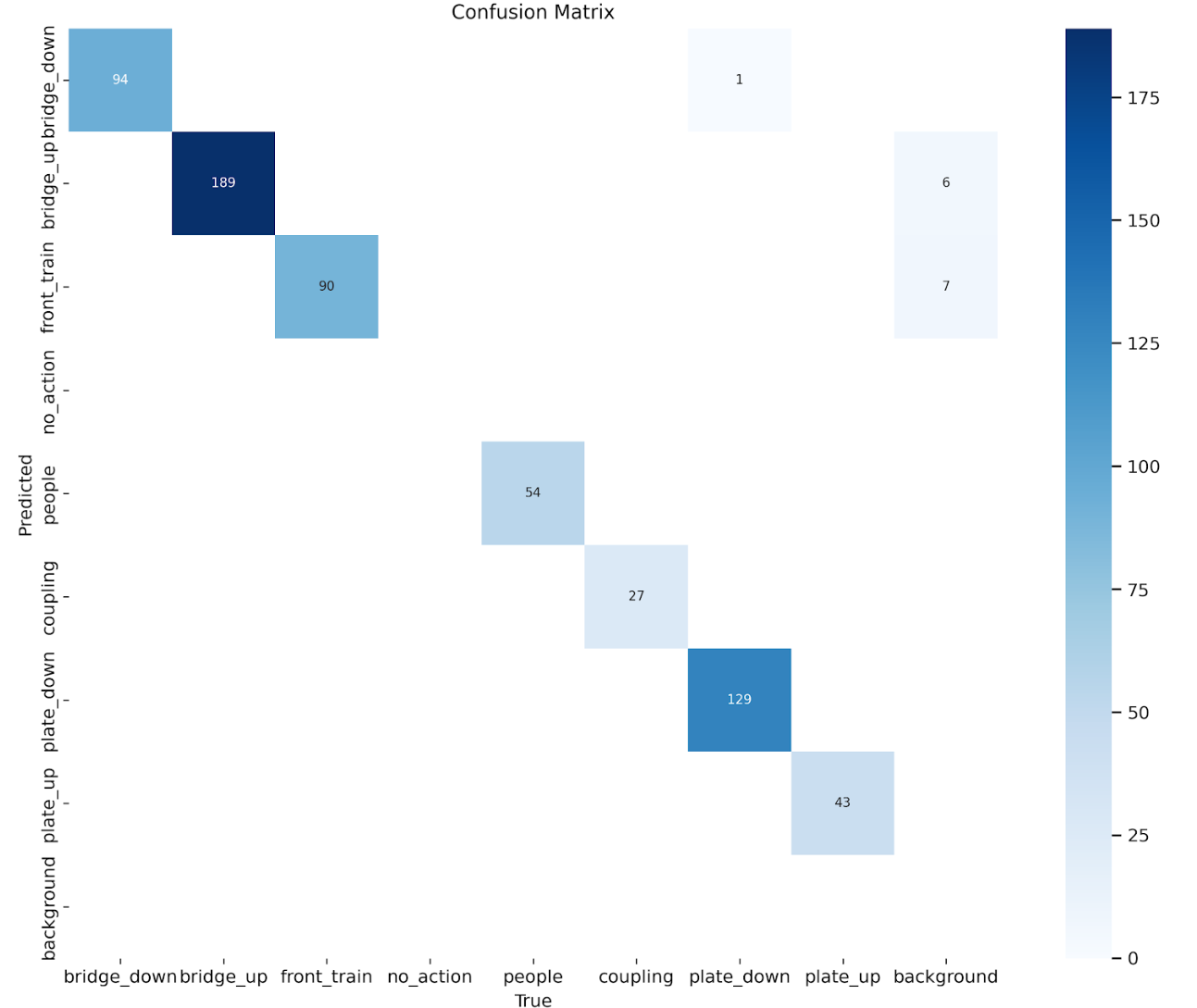
1. На графике с названием "labels\_correlogram" показана корреляция между классами объектов, распознаваемых моделью. Каждый класс, который отображен на графике, имеет свое обозначение (label), а корреляция между классами изображается в виде матрицы.

Каждый элемент матрицы может находиться в диапазоне от -1 до 1 и показывает корреляцию между двумя классами объектов. Значение, которое стремится к 1 означает, Что между классами существует высокая зависимость, то есть, если один класс присутствует на изображении, то с высокой долей вероятности и второй класс тоже будет присутствовать на изображении. Аналогичная логика относится и к значению близкому к -1, которое показывает, что наличие одного объекта практически всегда исключает наличие второго объекта на изображении. Значение близкое к 0 показывает, что в равной доле ситуаций один объект, как присутствовал со вторым объектом, так и отсутствовал, то есть корреляция между классами отсутствует.

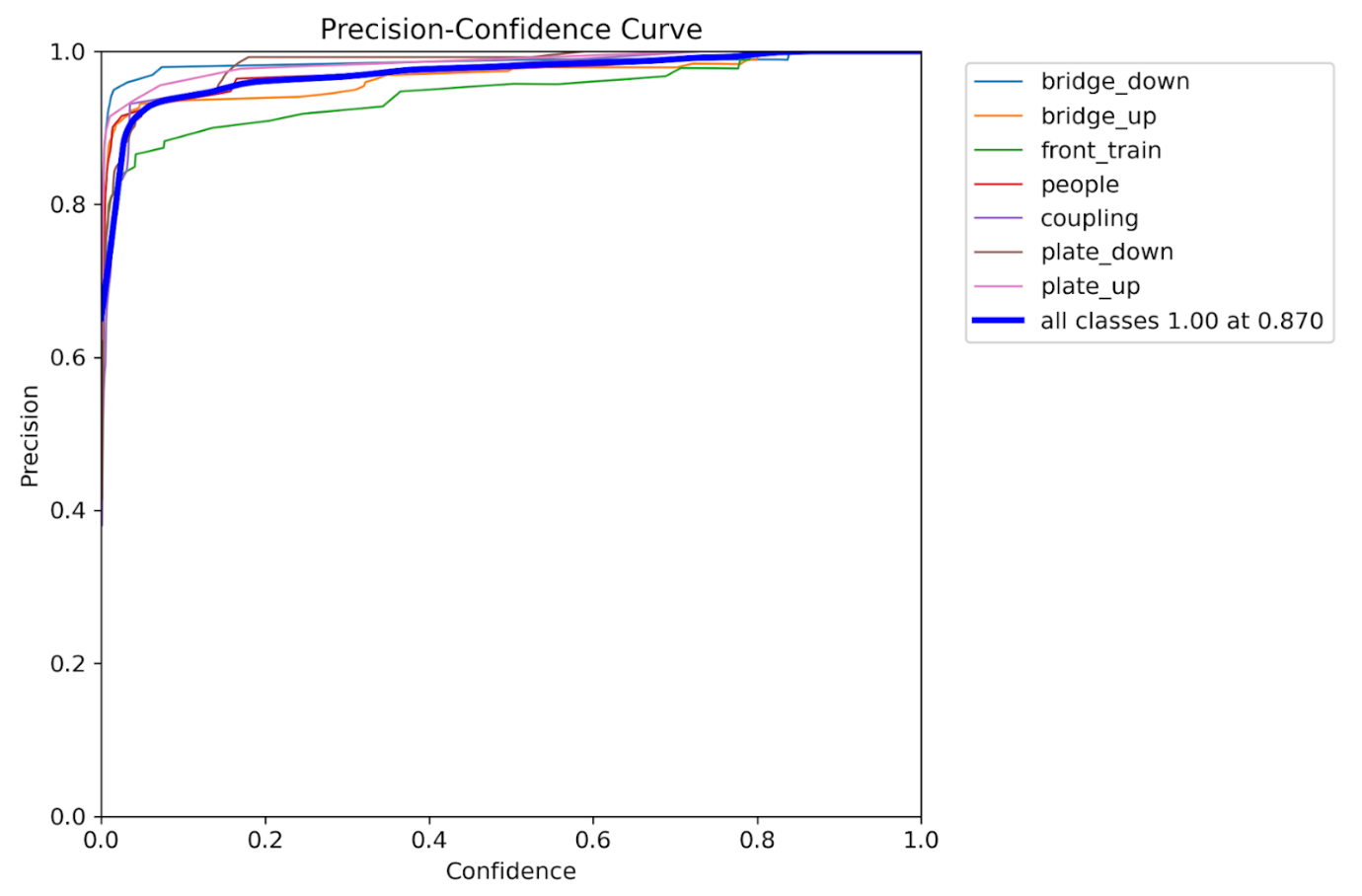


1. Матрица ошибок (confusion matrix) - это таблица, которая показывает качество созданной модели в задачах классификации.

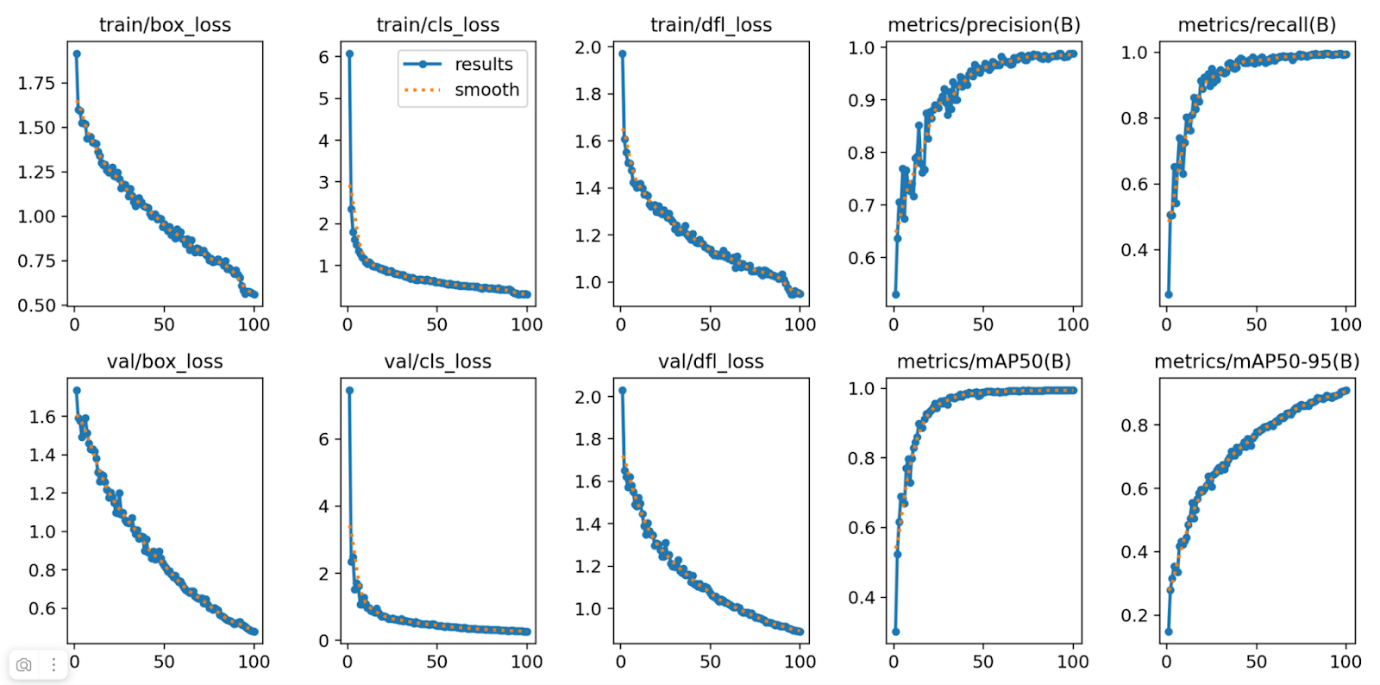
Матрица ошибок разделена на строки и столбцы, которые соответствуют классам объектов, которые участвуют в создании модели. В каждой ячейки хранится количество объектов, которые были классифицированы моделью в соответствующий класс.



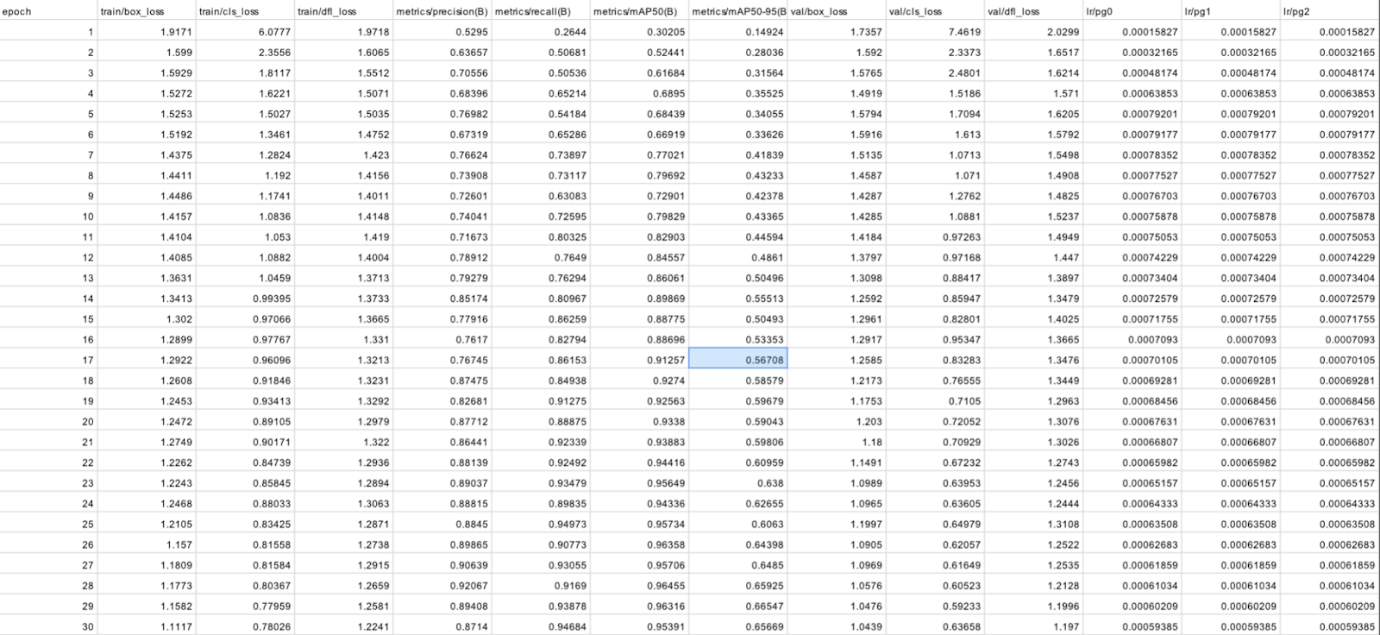
1. Precision-Recall curve (кривая точности-полноты) - это график, который показывает как меняется точность (precision) и полнота (recall) обнаружения объектов с различными порогами обнаружения. Порог обнаружения - это значение, которое используется для определения, является ли объект обнаруженным или нет. При низких значениях обнаружения, модель обнаруживает большее количество объектов, но повышается вероятность ложных срабатываний. При высоких порогах напротив, модель обнаруживает меньшее количество объектов, но при этом уменьшается вероятность ложного срабатывания. Кривая точности-полноты показывает работу модели при различных порогах обнаружения



1. Detection results (результаты обнаружения) - это набор изображений, которые показывают обнаруживаемые моделью объекты на тестовой выборке. Результаты обнаружения используются для визуального анализа работы модели и анализа достоинств и недостатков.



После осмотра результатов понятно, что к сотой эпохе модель начала хорошо распознавать данные в обучающей выборке. Процент распознавания увеличивался, а значение функции потерь стремилось к нулю.

Результаты обучения в первых 30ти эпохах: 

## 5.  Построение моделей и алгоритмов их машинного обучения, включая разработку искусственных нейронных сетей;

Архитектура YOLOv8s состоит из нескольких частей:

1. Backbone: это основная часть модели, которая отвечает за определение признаков на изображении. В модели YOLOv8s в качестве backbone используется модифицированная версия сети Darknet-53, которая состоит из 53 слоев, включая сверточные, пулинговые и нормализационные слои.
2. Neck: эта часть модели, которая отвечает за объединение признаков из различных слоев backbone. В модели YOLOv8s в качестве neck используется сеть FPN (Feature Pyramid Network), которая позволяет объединить признаки из различных слоев для более точного обнаружения объектов.
3. Head: эта часть модели, которая отвечает за “предсказание” ограничивающих рамок и классов объектов на основе извлеченных признаков. В модели YOLOv8s в качестве head используется сеть, которая предсказывает три ограничивающие рамки и объекты классов для каждой ячейки сетки, которая наложена на изображение.

С точки зрения математики YOLOv8s работает следующим образом:

1. Сначала изображение разбивается на сетку из ячеек размером N x N, где N - параметр модели, обычно равный 7 или 13.
2. После этого для каждой ячейки сетки модель “предсказывает” три ограничивающие рамки (bounding boxes) и вероятности отношения этих рамок к каждому из классов объектов, которые модель может обнаруживать.
3. Для каждой ячейки сетки вычисляется вероятность принадлежности ячейки к объекту (objectness score).
4. После этого модель объединяет все предсказания всех ячеек сетки и выбирает наиболее вероятные ограничивающие рамки и классы для каждого объекта на изображении.

Всего в YOLOv8s 106 слоев, включая сверточные, пулинговые, нормализационные и слои полносвязной сети. Модель обучается на большом количестве размеченных изображений, используя метод обратного распространения ошибки (backpropagation) и алгоритм стохастического градиентного спуска (SGD).

Основные преимущества YOLOv8s перед другими моделями заключаются в ее высокой скорости работы и способности обнаруживать объекты в реальном времени. Кроме того, YOLOv8s может обнаруживать несколько объектов на одном изображении и работает с изображениями различных размеров и с различными соотношениями сторон.

## **6. Интерпретация и использование модели машинного обучения**

В нашей работе мы используем обученную нейронную сеть YOLOv8 для распознавания объектов и действий, связанных с вагоном-цистерной, на видеозаписи. Для этого мы отправляем видеозапись через бота в Telegram, а на выходе получаем видео с распознаванием.

Далее, на основании полученных данных, бот делает вывод о происходящих событиях. Например, если на видеозаписи распознана сцепка и ее координаты соответствуют заезду или выезду вагона-цистерны, бот сообщает об этом событии. Если на видеозаписи распознан человек, бот сообщает о наличии человека в депо.

Кроме того, бот также делает выводы о происходящих процессах, связанных с вагоном-цистерной. Например, если на видеозаписи распознан опущенный крановый мост и наличие люка (открыт или закрыт), бот сообщает о процессе загрузки или разгрузки вагона-цистерны. Если на видеозаписи распознан поднятый крановый мост и наличие люка, бот сообщает, что в данный момент ничего не происходит.

Таким образом, мы используем обученную нейронную сеть YOLOv8 для автоматизации процессов мониторинга и контроля в депо, что позволяет увеличить эффективность и безопасность работы. Этот подход может быть также использован в других сферах, где необходимо автоматизировать процессы мониторинга и контроля на основе видеозаписи.

В дальнейшем, для автоматизации процессов мониторинга и контроля в депо, можно установить камеры видеонаблюдения и использовать обученную нейронную сеть YOLOv8 для распознавания объектов и действий, связанных с вагонами-цистернами, в режиме реального времени.

Например, камеры можно установить в местах, где происходят частые и опасные действия, такие как загрузка и разгрузка вагонов-цистерн, перемещение вагонов-цистерн, и т.д. Затем, с помощью нейронной сети YOLOv8, можно распознавать объекты и действия на видеопотоке с камер и делать выводы о происходящих событиях.

Это позволит оперативно реагировать на возникающие ситуации, предотвращать аварии и инциденты, а также увеличить эффективность работы в депо.

Однако, в нашей работе мы не имели доступа к камерам видеонаблюдения в депо, поэтому мы реализовали функционал распознавания объектов и действий, связанных с вагонами-цистернами, в Telegram-боте. Пользователи могут отправлять видеозаписи с вагонами-цистернами в депо в Telegram-бот, и бот, с помощью обученной нейронной сети YOLOv8, распознает объекты и действия на видеозаписи и делает выводы о происходящих событиях.

Этот подход также позволяет автоматизировать процессы мониторинга и контроля в депо, но, в отличие от использования камер видеонаблюдения, требует участия человека для отправки видеозаписей в Telegram-бот.

## **7. Прототип решения**

Описание нашей модели, ее возможностей

Модель связываясь с пользователем через Телеграмм бота получает видеофайл, в котором она обнаруживает объекты и на основе их наличия проводит анализ ситуации.

Пример реализации:

Загрузка поезда:



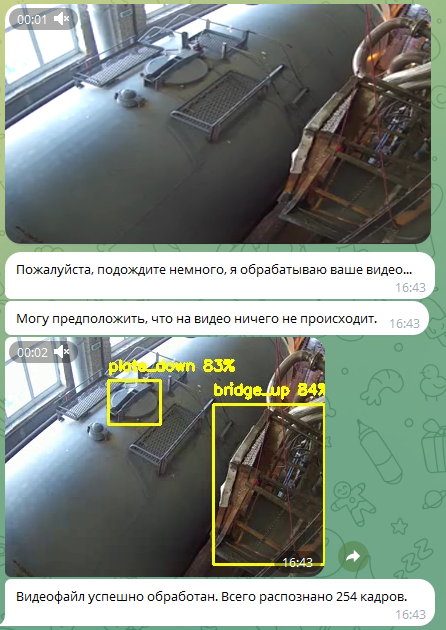
Загрузка поезда 2:



Въезд поезда:



Бездействие:



ссылка на видео: [output.mp4](https://drive.google.com/file/d/1Sf9UFnMCzVNOz7WcU_RpMwZq6uW7qp8I/view?usp=sharing)

Код приложения с описанием:

!pip install ultralytics

Установка библиотеки ultralytics которая используется для работы с моделями для обнаружения объектов.

from google.colab import drive

Импортирование модуля drive, который предоставляет функции для работы с Google drive в Google Collab.

drive.mount('/content/gdrive')

Подключение Google Drive к Google Collab для возможности работы с файлами внутри Collab.

!yolo task=detect mode=train model=yolov8s.pt data=/content/gdrive/MyDrive/ComputerVisionDeveloper/TrainYolov8CustomDataset/google\_colab\_config.yaml epochs=100 imgsz=640 project=/content/gdrive/MyDrive/ComputerVisionDeveloper/TrainYolov8CustomDataset/data name=TrainingTrainNew

Запуск обучения модели обнаружения объектов Yolov8s обозначение пути к конфиг-файлу, количество эпох равное 100, размер изображения равный 640px и путь к проекту для сохранения данных.

!yolo task=detect mode=predict model=/content/gdrive/MyDrive/ComputerVisionDeveloper/TrainYolov8CustomDataset/data/TrainingTrainNew2/weights/best.pt conf=0.60 source=/content/gdrive/MyDrive/ComputerVisionDeveloper/TrainYolov8CustomDataset/videos

Запуск обученной модели для обнаружения объектов в режиме предсказания, указание пути к модели с лучшими весами, порог уверенности равный 0.60, отвечает за вывод предсказания, если модель недостаточно уверена, тогда она не будет выводить найденный объект.

import io

Импорт библиотеки для работы с протоколами ввода-вывода.

import os

Импорт библиотеки для работы с операционной системой.

import cv2

Импорт библиотеки для работы с изображением и видео.

import ultralytics

Импорт модуля для работы с Yolo.

import requests

Импорт модуля для работы с Http-запросами.

import telebot

Импорт модуля для работы с телеграмм ботами.

from random import randint

Импорт функции получения рандомного числа из диапазона (а,б)

from google.colab import userdata

Импорт модуля для работы с секретными ключами.

bot\_token = userdata.get('BOT\_TOKEN')

Получение токена бота из секретного ключа.

model = ultralytics.YOLO('/content/gdrive/MyDrive/ComputerVisionDeveloper/TrainYolov8CustomDataset/data/TrainingTrainNew2/weights/best.pt')

загрузка модели YOLOv8 для дальнейшей работы.

bot = telebot.TeleBot(bot\_token) - создает экземпляр бота Telegram из пакета telebot и передает в него токен бота из переменной bot\_token.

@bot.message\_handler(commands=['start']) - определяет обработчик сообщений бота, который будет вызываться при получении команды /start.

def start(message) - определяет функцию-обработчик сообщений бота, которая будет вызываться при получении команды /start.

bot.send\_message(message.chat.id, 'Приветствую! Я бот, который может распознавать объекты на видео. Отправьте мне видеофайл, и я обработаю его.') - отправляет приветственное сообщение в чат, откуда было получено сообщение с командой /start.

@bot.message\_handler(content\_types=['video']) - определяет обработчик сообщений бота, который будет вызываться при получении видеофайлов.

def handle\_video(message) - определяет функцию-обработчик сообщений бота, которая будет вызываться при получении видеофайлов.

couplingX = 0 - инициализирует переменную couplingX, которая будет использоваться для хранения координаты X сцепки вагонов.

couplingY = 0 - инициализирует переменную couplingY, которая будет использоваться для хранения координаты Y сцепки вагонов.

XY=True - инициализирует переменную XY, которая будет использоваться для отслеживания того, была ли обнаружена сцепка вагонов.

FlagAssumption1 = True - инициализирует переменную FlagAssumption1, которая будет использоваться для отслеживания того, была ли сделана предположение 1.

FlagAssumption2 = True - инициализирует переменную FlagAssumption2, которая будет использоваться для отслеживания того, была ли сделана предположение 2.

FlagAssumption3 = True - инициализирует переменную FlagAssumption3, которая будет использоваться для отслеживания того, была ли сделана предположение 3.

FlagAssumption4 = True - инициализирует переменную FlagAssumption4, которая будет использоваться для отслеживания того, была ли сделана предположение 4.

FlagAssumption5 = True - инициализирует переменную FlagAssumption5, которая будет использоваться для отслеживания того, была ли сделана предположение 5.

bot.send\_message(message.chat.id, 'Пожалуйста, подождите немного, я обрабатываю ваше видео...') - отправляет сообщение в чат, откуда было получено видео, с просьбой подождать, пока видео будет обрабатываться.

file\_info = bot.get\_file(message.video.file\_id) - получает информацию о файле, который был отправлен боту.

file\_path = io.BytesIO(requests.get(f'https://api.telegram.org/file/bot{bot\_token}/{file\_info.file\_path}').content) - загружает файл, который был отправлен боту, из Telegram в память.

temp\_folder = '/tmp' - инициализирует переменную temp\_folder, которая будет использоваться для хранения временных файлов.

if not os.path.exists(temp\_folder): - проверяет, существует ли директория temp\_folder.

os.makedirs(temp\_folder) - создает директорию temp\_folder, если она не существует.

video\_path = os.path.join(temp\_folder, 'video.mp4') - создает путь к временному файлу video.mp4 в директории temp\_folder.

with open(video\_path, 'wb') as f: - открывает временный файл video.mp4 для записи в двоичном формате.

f.write(file\_path.getvalue()) - записывает содержимое файла, который был загружен в память, в временный файл video.mp4.

cap = cv2.VideoCapture(video\_path) - создает экземпляр VideoCapture из пакета cv2 и передает в него путь к временному файлу video.mp4.

frame\_width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)) - получает ширину кадра видео и сохраняет ее в переменной frame\_width.

frame\_height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)) - получает высоту кадра видео и сохраняет ее в переменной frame\_height.

fourcc = cv2.VideoWriter\_fourcc(\*'mp4v') - создает кодек mp4v для записи видеофайла.

out = cv2.VideoWriter('output.mp4', fourcc, 20.0, (frame\_width, frame\_height)) - создает экземпляр VideoWriter из пакета cv2 и передает в него путь к выходному файлу output.mp4, кодек fourcc, частоту кадров 20.0 и размеры кадра (frame\_width, frame\_height).

frame\_count = 0 - инициализирует переменную frame\_count, которая будет использоваться для отслеживания количества обработанных кадров.

while cap.isOpened(): - определяет цикл, который будет выполняться, пока видеофайл не будет полностью обработан.

ret, frame = cap.read() - читает следующий кадр из видеофайла и сохраняет его в переменной frame.

if not ret: break - прерывает цикл, если достигнут конец видеофайла.

results = model.predict(frame) - выполняет предсказание модели model на текущем кадре frame.

bridge\_down\_detected = False - инициализирует переменную bridge\_down\_detected, которая будет использоваться для отслеживания того, был ли обнаружен объект bridge\_down.

plate\_detected = False - инициализирует переменную plate\_detected, которая будет использоваться для отслеживания того, был ли обнаружен объект plate\_down или plate\_up.

bridge\_up\_detected = False - инициализирует переменную bridge\_up\_detected, которая будет использоваться для отслеживания того, был ли обнаружен объект bridge\_up.

for obj in results[0].boxes: - определяет цикл, который будет выполняться для каждого обнаруженного объекта obj на текущем кадре frame.

color = (0, 255, 255) - инициализирует переменную color, которая будет использоваться для рисования ограничивающего прямоугольника и текста.

x1, y1, x2, y2 = obj.xyxy[0] - получает координаты верхнего левого и нижнего правого углов ограничивающего прямоугольника obj и сохраняет их в переменных x1, y1, x2, y2.

x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2) - преобразует координаты x1, y1, x2, y2 в целые числа.

cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2) - рисует ограничивающий прямоугольник obj на текущем кадре frame.

obj\_name = model.names[int(obj.cls)] - получает имя obj\_name обнаруженного объекта obj.

obj\_confidence = int(obj.conf[0] \* 100) - получает уверенность obj\_confidence обнаруженного объекта obj в процентах.

cv2.putText(frame, f'{obj\_name} {obj\_confidence}%', (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2) - рисует имя obj\_name и уверенность obj\_confidence обнаруженного объекта obj на текущем кадре frame.

if obj\_name == 'coupling' and XY: - проверяет, был ли обнаружен объект coupling и была ли обнаружена сцепка вагонов.

couplingX = x1 - сохраняет координату X x1 сцепки вагонов в переменной couplingX.

couplingY = y1 - сохраняет координату Y y1 сцепки вагонов в переменной couplingY.

XY=False - сбрасывает флаг XY, отслеживающий то, была ли обнаружена сцепка вагонов.

if obj\_name == "coupling" and FlagAssumption4 and couplingX < x1-8 and couplingY > y1+15: - проверяет, был ли обнаружен объект coupling, был ли сделан предположение 4 и соответствуют ли координаты couplingX и couplingY сцепки вагонов условиям.

bot.send\_message(message.chat.id, 'Могу предположить, что вагон цистерна выезжает из депо.') - отправляет сообщение в чат, откуда было получено видео, с предположением 4.

FlagAssumption4 = False - сбрасывает флаг FlagAssumption4, отслеживающий то, было ли сделано предположение 4.

if obj\_name == "coupling" and FlagAssumption3 and couplingX > x1+8 and couplingY < y1-15: - проверяет, был ли обнаружен объект coupling, был ли сделан предположение 3 и соответствуют ли координаты couplingX и couplingY сцепки вагонов условиям.

bot.send\_message(message.chat.id, 'Могу предположить, что вагон цистерна заезжает в депо.') - отправляет сообщение в чат, откуда было получено видео, с предположением 3.

FlagAssumption3 = False - сбрасывает флаг FlagAssumption3, отслеживающий то, была ли сделана предположение 3.

if obj\_name == 'people' and FlagAssumption2: - проверяет, был ли обнаружен объект people и был ли сделан предположение 2.

bot.send\_message(message.chat.id, 'Могу предположить, что в кадре есть человек.') - отправляет сообщение в чат, откуда было получено видео, с предположением 2.

FlagAssumption2 = False - сбрасывает флаг FlagAssumption2, отслеживающий то, была ли сделана предположение 2.

if obj\_name == 'bridge\_down': - проверяет, был ли обнаружен объект bridge\_down.

bridge\_down\_detected = True - устанавливает флаг bridge\_down\_detected, отслеживающий то, был ли обнаружен объект bridge\_down.

elif obj\_name == 'plate\_down' or obj\_name == 'plate\_up': - проверяет, был ли обнаружен объект plate\_down или plate\_up.

plate\_detected = True - устанавливает флаг plate\_detected, отслеживающий то, был ли обнаружен объект plate\_down или plate\_up.

if obj\_name == 'bridge\_up': - проверяет, был ли обнаружен объект bridge\_up.

bridge\_up\_detected = True - устанавливает флаг bridge\_up\_detected, отслеживающий то, был ли обнаружен объект bridge\_up.

if bridge\_down\_detected and plate\_detected and FlagAssumption1: - проверяет, были ли обнаружены объекты bridge\_down и plate\_down или plate\_up и был ли сделан предположение 1.

bot.send\_message(message.chat.id, 'Могу предположить, что идет загрузка поезда.') - отправляет сообщение в чат, откуда было получено видео, с предположением 1.

FlagAssumption1 = False - сбрасывает флаг FlagAssumption1, отслеживающий то, была ли сделана предположение 1.

if bridge\_up\_detected and plate\_detected and FlagAssumption5: - проверяет, были ли обнаружены объекты bridge\_up и plate\_down или plate\_up и был ли сделан предположение 5.

bot.send\_message(message.chat.id, 'Могу предположить, что на видео ничего не происходит.') - отправляет сообщение в чат, откуда было получено видео, с предположением 5.

FlagAssumption5 = False - сбрасывает флаг FlagAssumption5, отслеживающий то, была ли сделана предположение 5.

out.write(frame) - записывает текущий кадр frame с наложенными объектами в выходной файл output.mp4.

frame\_count += 1 - увеличивает счетчик frame\_count обработанных кадров на 1.

cap.release() - освобождает ресурсы, занятые экземпляром VideoCapture.

out.release() - освобождает ресурсы, занятые экземпляром VideoWriter.

cv2.destroyAllWindows() - закрывает все окна, созданные пакетом cv2.

with open('output.mp4', 'rb') as f: - открывает выходной файл output.mp4 для чтения в двоичном формате.

bot.send\_video(message.chat.id, f) - отправляет выходной файл output.mp4 с наложенными объектами в чат, откуда было получено видео.

os.remove(video\_path) - удаляет временный файл video.mp4.

os.remove('output.mp4') - удаляет выходной файл output.mp4.

bot.send\_message(message.chat.id, f'Видеофайл успешно обработан. Всего распознано {frame\_count} кадров.') - отправляет сообщение в чат, откуда было получено видео, с информацией о количестве обработанных кадров frame\_count.

bot.polling() - запускает бота Telegram и начинает прослушивание входящих сообщений.