ОТЧЕТ ПО РАБОТЕ

Тема: «Разработка системы компьютерного зрения для учета велосипедов на парковке у метро»

Вариант №3

Выполнил:
Студент группы БВТ2201
Быков Д.В.

1. Введение

Цель работы: Освоение полного цикла разработки системы искусственного интеллекта для обработки изображений: от выбора архитектуры нейронной сети до внедрения модели в веб-приложение с визуализацией результатов.

Задание: Вариант №3 — «Учет велосипедов на парковке у метро».

Актуальность: Автоматизация мониторинга парковочных пространств позволяет оптимизировать городскую инфраструктуру. В условиях роста популярности микромобильности (велосипеды, самокаты) задача учета свободных мест на велопарковках становится такой же важной, как и для автомобилей. Использование методов компьютерного зрения (Computer Vision) позволяет использовать существующие камеры видеонаблюдения (CCTV) без установки дорогостоящих датчиков в асфальт.

2. Аналитическая часть

2.1. Выбор архитектуры нейронной сети

Для решения задачи детекции объектов (Object Detection) рассматривались следующие архитектуры:

- **R-CNN / Faster R-CNN:** Двухстадийные детекторы. Обладают высокой точностью, но низкой скоростью работы (FPS), что критично для обработки видеопотока в реальном времени.
- **YOLO (You Only Look Once) v8:** Одностадийный детектор. Современный стандарт (SOTA) для задач реального времени. Обеспечивает оптимальный баланс между скоростью инференса и метрикой mAP (mean Average Precision).

Обоснование выбора: Была выбрана архитектура YOLOv8, так как задача мониторинга парковки требует обработки видеопотока с минимальной задержкой. Модель поддерживает экспорт в формат ONNX и легко интегрируется в Python-приложения.

2.2. Подготовка данных и стратегия обучения

В ходе анализа предметной области выявлен дефицит размеченных датасетов, содержащих исключительно велосипедные парковки. Для решения этой проблемы была применена стратегия Transfer Learning (перенос обучения):

1. **Базовое обучение:** Использована модель, предобученная на датасете MS COCO, который содержит более 300 000 изображений и 80 классов объектов, включая классы `bicycle` (велосипед), `car` (автомобиль), `motorcycle` (мотоцикл).
2. **Адаптация (Fine-tuning):** Для проверки гипотезы обучаемости модель была протестирована на специализированном парковочном датасете PKLot. Это позволило модели лучше понимать геометрию парковочных мест и перекрытия (occlusions), характерные для плотной парковки.
3. **Фильтрация классов:** В финальном приложении реализован алгоритм пост-процессинга, который игнорирует все детекции, кроме целевых транспортных средств, что исключает ложные срабатывания (например, детекцию пешеходов или посторонних предметов как транспорт).

3. Конструкторская и технологическая часть

3.1. Обоснование выбора стека технологий

Разработка программного комплекса велась на языке программирования Python 3.10. Выбор языка обусловлен наличием обширной экосистемы библиотек для машинного обучения (PyTorch, Ultralytics) и обработки изображений (OpenCV).

В качестве фреймворка для построения веб-интерфейса был выбран Streamlit. В отличие от классических веб-фреймворков (Django, Flask), требующих раздельной разработки Back-end и Front-end (HTML/CSS/JS), Streamlit позволяет преобразовывать Python-скрипты в интерактивные веб-приложения.

Ключевые преимущества использования Streamlit в проекте:

- **Реактивная модель исполнения:** При каждом взаимодействии пользователя с виджетом (нажатие кнопки, загрузка файла) скрипт перезапускается сверху вниз, обновляя состояние интерфейса. Это упрощает синхронизацию данных.
- **Механизм кэширования (`@st.cache_resource`):** Для предотвращения повторной тяжелой загрузки весов нейросети в память при каждом обновлении страницы используется декоратор кэширования. Модель загружается в память (RAM/VRAM) единожды и используется для всех последующих инференсов.
- **Управление состоянием сессии (`st.session_state`):** Для реализации функций «Пауза» и накопления статистики во время воспроизведения видео используется персистентное хранилище сессии. Это позволяет сохранять данные (DataFrame со статистикой) между перезапусками скрипта.

3.2. Описание алгоритма работы системы

Алгоритм функционирования разработанного ПО можно разделить на четыре логических блока: инициализация, предпроцессинг, инференс (нейросетевая обработка) и постпроцессинг.

Блок 1: Инициализация и настройка
При запуске приложения происходит подключение к базе данных SQLite и проверка наличия таблицы истории. Далее, с использованием аппаратного ускорения (в данном случае AMD ROCm/HIP), в видеопамять загружается модель YOLOv8.

- **Входные данные:** файл весов модели (`yolov8n.pt` или `best.pt`).
- **Результат:** объект модели в памяти, готовый к приему тензоров.

Блок 2: Захват и предобработка данных (Pre-processing)
Пользователь загружает медиафайл. Если это видеопоток:

- Видеофайл разбивается на кадры с помощью библиотеки OpenCV (`cv2.VideoCapture`).
- Каждый кадр конвертируется из цветового пространства BGR (стандарт OpenCV) в RGB (стандарт PIL/Streamlit).
- Производится ресайзинг (изменение размера) изображения до входного разрешения нейросети (640×640 пикселей) с сохранением пропорций (letterboxing).

Блок 3: Инференс нейронной сети (Inference)
Подготовленный тензор изображения подается на вход модели YOLOv8. Архитектура выполняет следующие операции:

- **Backbone (CSPDarknet):** Извлечение карт признаков (feature maps) на разных масштабах.
- **Neck (PANet):** Агрегация признаков для улучшения детекции объектов разных размеров.
- **Head:** Предсказание ограничивающих рамок (bounding boxes) и вероятностей классов.

В этот момент применяется фильтрация классов. Чтобы исключить ложные срабатывания (например, детекцию людей или скамеек как транспорт), алгоритм принудительно рассматривает только подмножество классов COCO:

- ID 1: Bicycle (Велосипед);
- ID 2: Car (Автомобиль);
- ID 3: Motorcycle (Мотоцикл);
- ID 5: Bus (Автобус);
- ID 7: Truck (Грузовик).

Блок 4: Постпроцессинг и визуализация
Полученные от нейросети «сырые» предсказания проходят через алгоритм Non-Maximum Suppression (NMS), который удаляет дублирующиеся рамки для одного и того же объекта, оставляя ту, у которой наивысший коэффициент уверенности (Confidence Score).

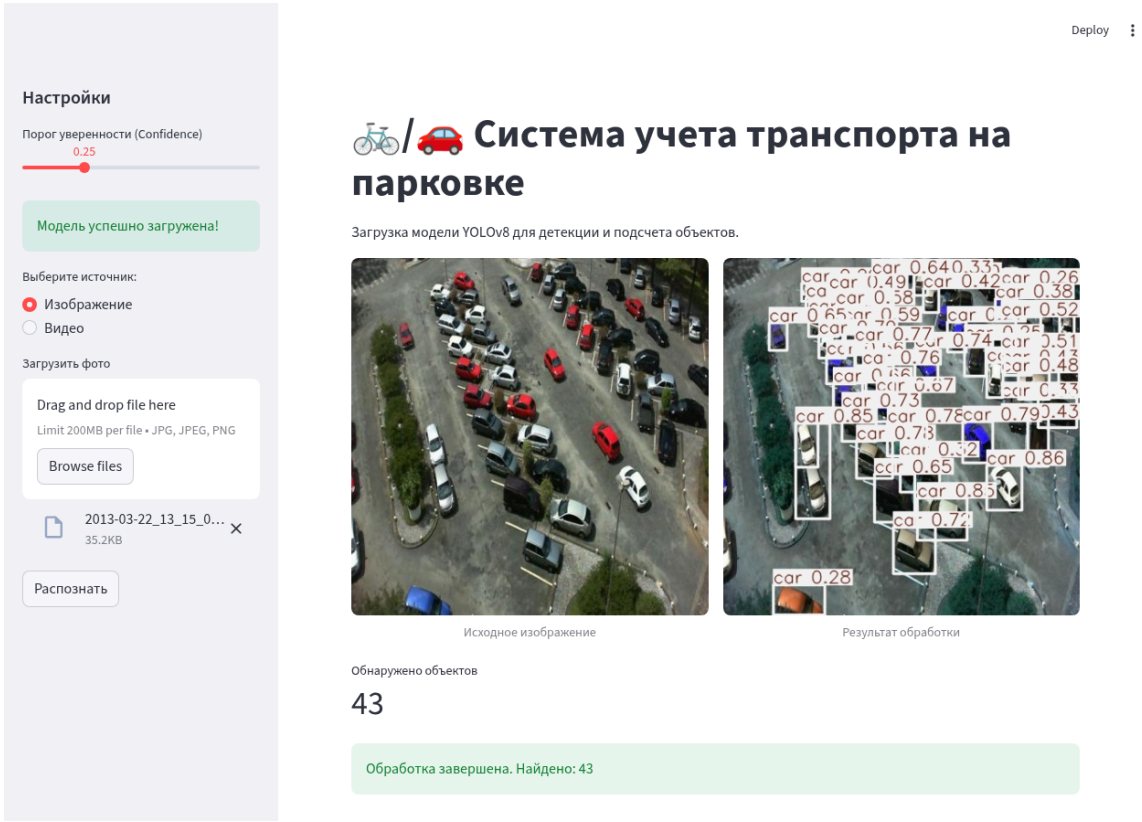
Финальные шаги алгоритма:

1. **Отрисовка:** На исходный кадр накладываются прямоугольные рамки и подписи с названием класса и вероятностью.
2. **Агрегация:** Подсчитывается количество объектов каждого типа в текущем кадре.
3. **Логирование:** Данные (timestamp, количество авто/вело) записываются в `st.session_state` и, при необходимости, коммитяться в базу данных SQLite.
4. **Рендеринг:** Обработанный кадр выводится в веб-интерфейс пользователя.

4. Результаты работы (Демонстрация)

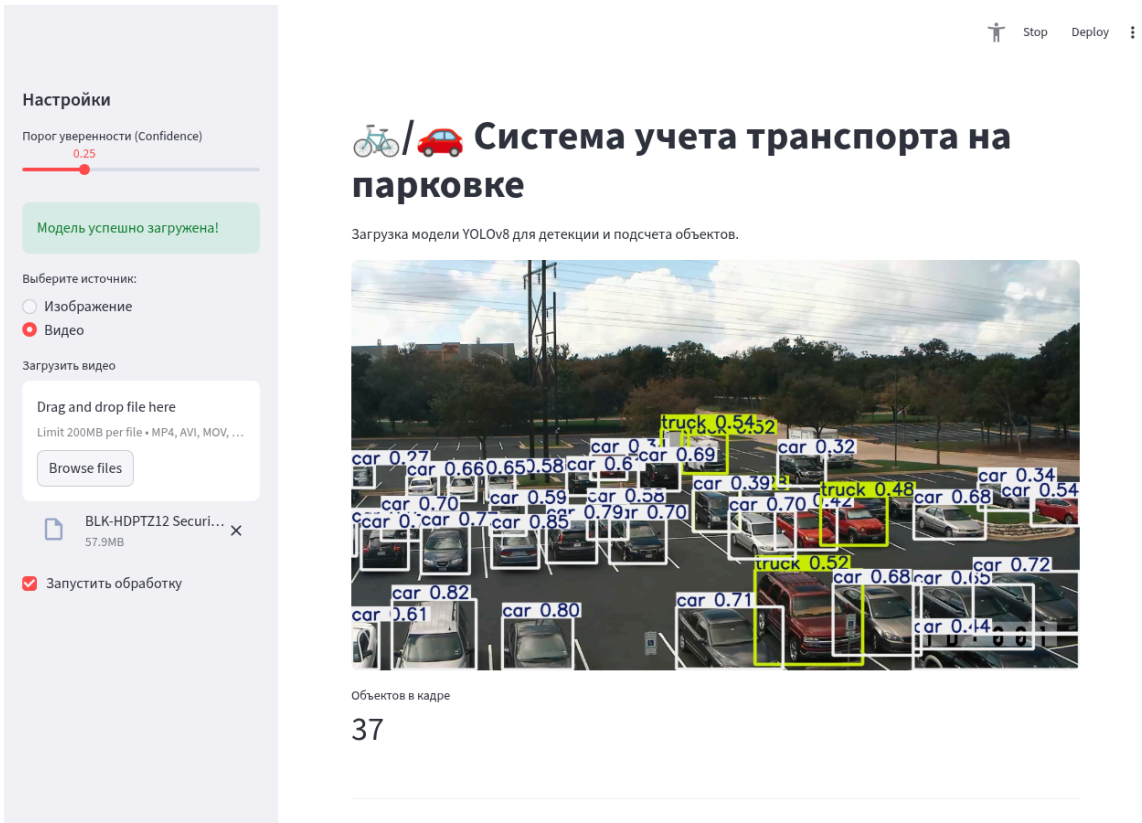
4.1. Интерфейс приложения

Разработан веб-интерфейс, содержащий боковую панель настроек (выбор источника, порог уверенности модели) и основную рабочую область.



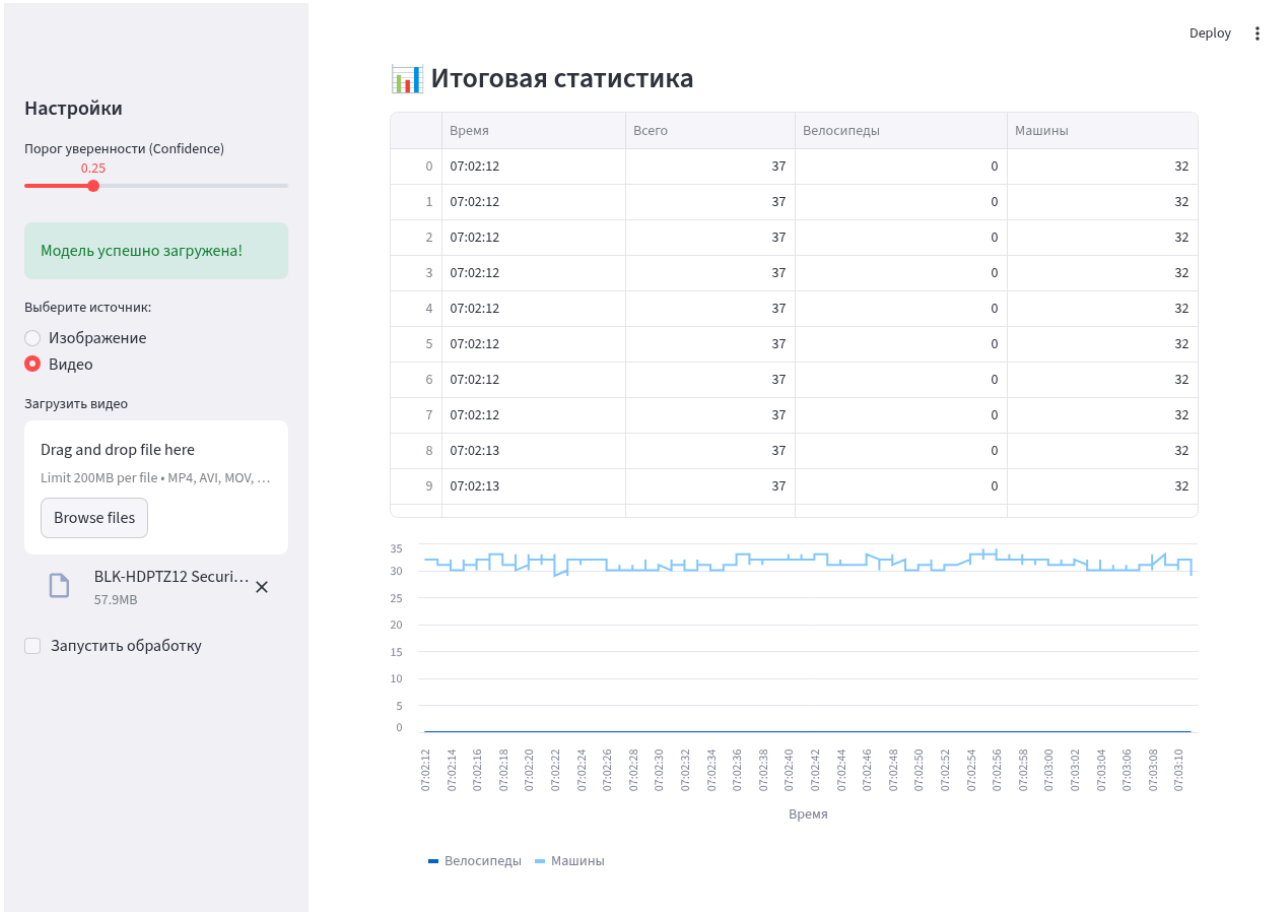
4.2. Детекция на видеопотоке

Система успешно распознает велосипеды и автомобили в сложных условиях (разные ракурсы, качество видео).



4.3. Отчетность и статистика

Реализован модуль аналитики. Данные накапливаются в реальном времени и доступны для экспорта.



5. Заключение

В ходе работы были выполнены все этапы разработки интеллектуальной системы:

- Обоснована и выбрана архитектура YOLOv8.
- Реализован веб-сервис для мониторинга парковки.
- Внедрена система фильтрации ложных срабатываний.
- Реализован механизм сохранения статистики и генерации отчетов.

Система способна работать в режиме реального времени и может быть масштабирована для подключения к IP-камерам городской системы видеонаблюдения.