УДК 004.93

ПРИМЕНЕНИЕ ИИ В СИСТЕМАХ ВИДЕОНАБЛЮДЕНИЯ ДЛЯ ФИКСАЦИИ НАРУШЕНИЙ ПДД С УЧАСТИЕМ СРЕДСТВ ИНДИВИДУАЛЬНОЙ МОБИЛЬНОСТИ

Клецов М., студент гр. ПИб-221, IV курс,
Ярков А.В., студент гр. ПИб-221, IV курс
Научный руководитель: Глебова Е.А., старший преподаватель
Кузбасский государственный технический университет имени Т. Ф. Горбачёва,
г. Кемерово

Одним из ключевых направлений развития систем безопасности и видеонаблюдения является автоматический анализ видеопотока в режиме реального времени. Применение алгоритмов детекции объектов на базе глубокого обучения позволяет существенно расширить возможности подобных систем: автоматически выявлять присутствие людей и транспортных средств в кадре, фиксировать нештатные ситуации и вести непрерывный мониторинг без постоянного участия оператора. В настоящей статье рассматривается разработка настольного приложения с графическим интерфейсом, предназначенного для подключения к IP-камерам и выполнения детекции объектов в потоковом видео с использованием нейросетевой модели семейства YOLO.

Особую актуальность приобретает проблема нарушений правил дорожного движения (ПДД) с участием средств индивидуальной мобильности (СИМ) — прежде всего электросамокатов. Одним из наиболее распространённых нарушений является совместная езда двух и более человек на одном электросамокате. Ручная проверка видеозаписей операторами является трудоёмкой задачей, что обусловливает необходимость автоматизации данного процесса.

Задача, решаемая приложением, состоит в том, чтобы предоставить оператору удобный инструментарий: выбрать камеру из предварительно сформированного списка, подключиться к её видеопотоку и наблюдать результаты автоматической детекции непосредственно в интерфейсе программы. При этом необходимо минимизировать задержку отображения, а также корректно обрабатывать нестабильные HLS-потоки.

Архитектура приложения построена на трёхуровневом принципе. Первый уровень — слой представления — реализован с помощью фреймворка PySide6, представляющего собой официальные Python-биндинги к библиотеке Qt6, и отвечает за графический интерфейс пользователя [1]. Второй уровень — слой обработки — включает захват видеопотока средствами библиотеки OpenCV и выполнение нейросетевого инференса с помощью специализированной дообучен-

ной модели YOLOv8n, натренированной на датасете из более чем 700 изображений в соотношении ~63% / 22% / 15% (train / valid / test) для ситуации «два и более человека на электросамокате». Третий уровень — уровень данных — обеспечивает получение списка камер и потоковых адресов с внешних веб-ресурсов посредством HTTP-запросов. Такое разделение ответственности позволяет независимо развивать каждый из уровней и при необходимости подключать альтернативные источники данных или заменять детектор без изменения кода интерфейса. Пример нескольких фото для обучения представлен на рисунке 1.

Рис. 1. Некоторые изображения, участвовавшие в процессе дообучения модели

В качестве базовой модели была выбрана именно YOLOv8n (nano), поскольку она является наиболее компактной конфигурацией семейства YOLOv8, что обеспечивает высокую скорость инференса на ресурсоограниченном оборудовании. Дообучение проводилось в среде Conda на рабочей станции с GPU. Параметры обучения: 150 эпох, размер входного изображения 640×640 пикселей, оптимизатор и скорость обучения выбраны автоматически библиотекой Ultralytics [2]. В ходе обучения отслеживались метрики на валидационной выборке, что и показано на рисунке 2.

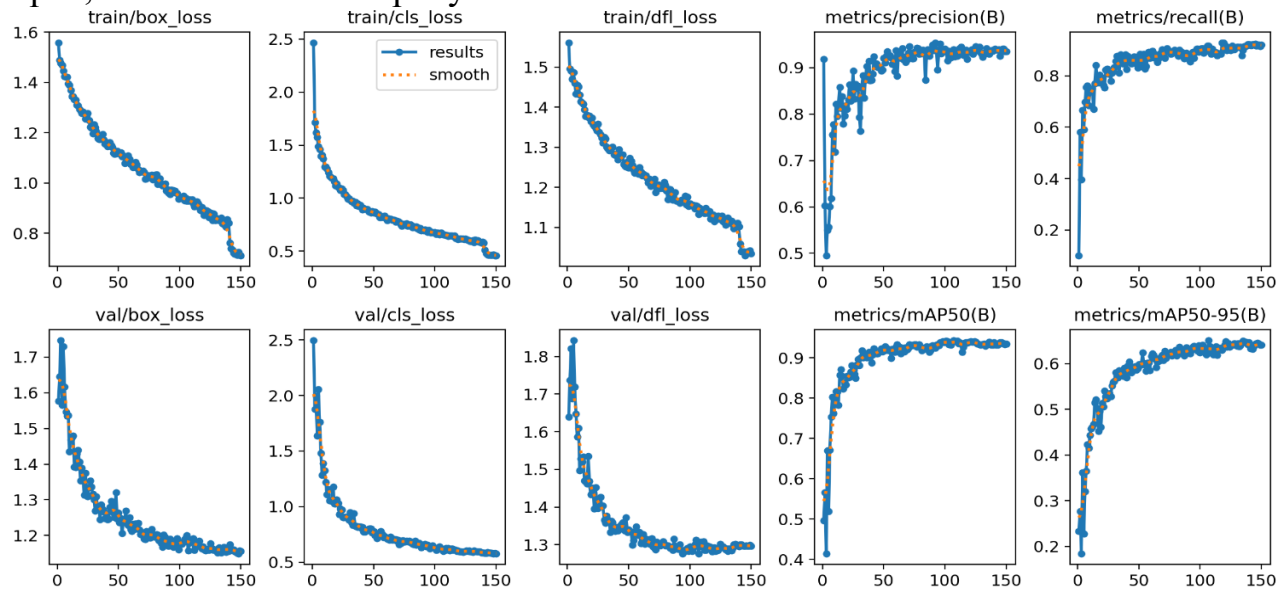

Рис. 2. Результаты дообучения модели, основанные на валидационной выборке

Были произведены тестовые запуски детекции в 2025 году. Использовались реальные данные с камер города Кемерово, это продемонстрировано на рисунках 3 и 4.

Рис. 3. Обработанный кадр: рядом с остановкой «Парк Ангелов»

Рис. 4. Обработанный кадр: рядом с Московской площадью

Основной частью оптимизации процесса распознавания в данном проекте является прореживание детекций: нейросеть обрабатывает не каждый кадр, а один из трёх, что позволяет поддерживать высокую частоту отображения видеопотока без существенной потери качества распознавания. Соответственно при воспроизведении потока частотой 24 кадра в секунду, детекция будет срабатывать 8 раз в секунду. В промежутках между прогонами модели результаты последнего распознавания — ограничивающие рамки и метки классов — отрисовываются поверх новых кадров средствами OpenCV. Порог уверенности для отсеивания ненадёжных детекций установлен в данный момент на уровне 0.5, что обеспечивает разумный компромисс между полнотой и точностью обнаружения.

Список доступных камер формируется автоматически при запуске приложения. Для этого выполняется HTTP-запрос к сайту, HTML-содержимое страницы разбирается с помощью библиотеки BeautifulSoup, и из якорных элементов извлекаются идентификаторы и названия камер. В случае недоступности ресур-

са или отсутствия необходимых зависимостей список формируется статически: создаются записи для 45 камер по умолчанию. При выборе камеры приложение дополнительно запрашивает страницу конкретной камеры и анализирует встроенный iframe-элемент, из которого извлекается имя потока и токен авторизации. Благодаря этому пользователю не требуется вводить технические параметры подключения вручную — все данные заполняются автоматически после простого нажатия на название камеры в списке.

Захват видеопотока осуществляется средствами OpenCV через интерфейс VideoCapture с использованием бэкенда CAP_FFMPEG, поддерживающего протокол HLS (HTTP Live Streaming) [3]. Потоковый адрес конструируется динамически на основе имени потока и токена авторизации. Работа с HLS-потоками сопряжена с рядом технических трудностей: сети видеонаблюдения нередко подвержены нестабильности, что выражается в кратковременных обрывах и накоплении устаревших кадров в буфере декодера. Для устранения этих проблем в приложении реализован механизм периодического сброса буфера VideoCapture каждые 100 прочитанных кадров, а также логика автоматического переподключения при превышении порогового числа последовательных ошибок чтения. Захват выполняется в выделенном системном потоке, что полностью исключает блокировку основного цикла обработки событий Qt и гарантирует отзывчивость интерфейса.

Поток захвата кадров непрерывно считывает кадры из VideoCapture и помещает их в очередь ёмкостью один элемент. Такое ограничение принципиально: при заполненной очереди новый кадр вытесняет предыдущий, что гарантирует минимальную задержку отображения — детектор всегда работает с самым свежим кадром, а не с накопившимися старыми. Поток детекции извлекает кадры из очереди и передаёт их модели YOLO. Для интеграции с Qt-интерфейсом рабочий объект DetectionWorker переносится в отдельный QThread посредством механизма moveToThread, а результаты обработки — аннотированные кадры и статистические данные — передаются в главный поток графического интерфейса через механизм сигналов и слотов (Signal/Slot), что обеспечивает потокобезопасное обновление виджетов.

Графический интерфейс приложения выполнен в тёмной цветовой схеме. Основное окно разделено на две части: левую панель управления и правую область отображения. В левой панели расположены поле для токена доступа, который заполняется автоматически при выборе камеры, прокручиваемый список камер с возможностью выбора одним нажатием и кнопка запуска и остановки детекции. Правая часть окна занята областью воспроизведения видеопотока с наложенными аннотациями детектора и строкой оперативной статистики в нижней части экрана. Пример интерфейса приложения продемонстрирован на рисунке 5.

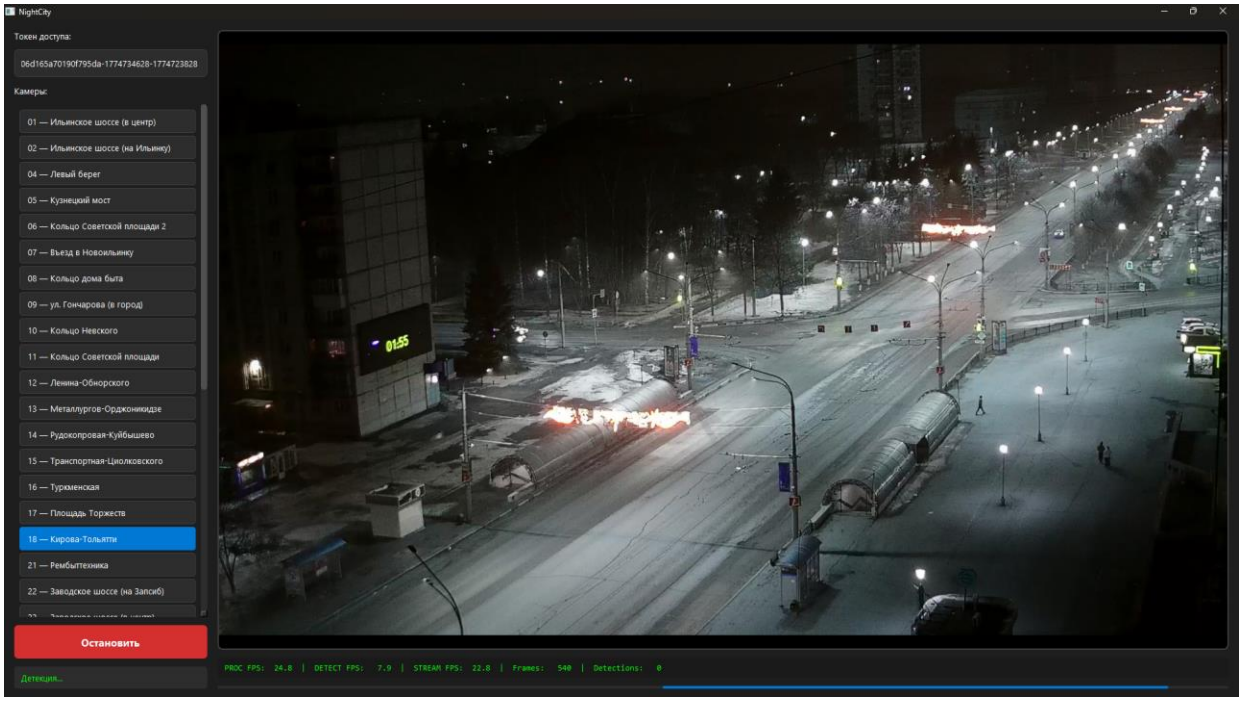

Рис. 5. Демонстрация интерфейса приложения

Разработанное приложение объединяет нейросетевой детектор YOLOv8, захват HLS-потоков от IP-камер и современный графический интерфейс в единый инструмент, доступный для развёртывания на обычном рабочем месте без специализированного серверного оборудования. Направлениями дальнейшего развития могут стать одновременный мониторинг нескольких камер, запись аннотированного видеопотока в файл, а также интеграция системы оповещений при обнаружении объектов заданных классов.

Список литературы

1. Gleb_Deryugin. Разработка Desktop приложений на Python и библиотеки PySide6/PyQt6. Часть 1. Установка и первое приложение на PySide6: [сайт]. URL: https://habr.com/ru/articles/799037/ (дата обращения: 02.02.2026).

2. dimanosov007. Обучите YOLOv8 на пользовательском наборе данных: [сайт]. URL: https://habr.com/ru/articles/714232/ (дата обращения: 06.10.2025).

3. RebelsOnTheMoon. HLS Video Streaming from OpenCV and FFmpeg : [сайт]. URL: https://medium.com/@vladakuc/hls-video-streaming-from-opencv-and-ffmpeg-828ca80b4124 (дата обращения: 10.03.2026).