МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

«Распознавание посторонних объектов на трамвайных путях в режиме реального времени»

Студент: _____ Салимли Айзек Мухтар Оглы

Преподаватель: _____ Востров Алексей Владимирович

«____»_____ 20__ г.

Санкт-Петербург, 2025

Содержание

Введение

Компьютерное зрение - представляет собой область информационных технологий, охватывающую методы автоматического анализа и интерпретации визуальных данных. Одной из ключевых задач данной области является распознавание объектов. Задача заключается в обнаружении, классификации и определении пространственных координат, объектов на изображениях или видеопотоке.

Современные подходы к решению задачи распознавания объектов, основаны на использовании сверточных нейронных сетей.

Существуют две основные архитектурные парадигмы:

- одноэтапные детекторы;
- двухэтапные детекторы.

Различие в перечисленных архитектурах обуславливают компромисс между выбором точности детекции и скорости распознавания объекта(-ов), что имеет значение при решении задач обнаружения объектов в режиме реального времени.

Актуальность исследования определяется необходимостью разработки эффективной системы обнаружения посторонних объектов на трамвайных путях, способной работать в условиях высокоскоростного движения трамваев Санкт-Петербурга.

Объектом исследования являются алгоритмы и модели машинного обучения для детектирования объектов в видеопотоке.

Предметом исследования являются архитектурные особенности и характеристики моделей семейств YOLO и R-CNN применительно к задаче обнаружения посторонних объектов на трамвайных путях.

Цель работы заключается в проведении сравнительного анализа моделей детектирования объектов и разработке критериев выбора оптимальной модели для системы мониторинга трамвайных путей в режиме реального времени.

1 Постановка задачи

Развитие высокоскоростного трамвайного движения в Санкт-Петербурге, включая внедрение новых моделей подвижного состава и создание специализированных маршрутов, предъявляет повышенные требования к системам безопасности, что обуславливает необходимость разработки эффективной системы обнаружения посторонних объектов для повышения безопасности пассажирских перевозок. Основные трамваи в городе имеют следующие характеристики:

- Высокоскоростные модели: Stadler B85600M (80 км/ч), 71-638-02 «Поларис» (75 км/ч), Alstom Citadis 301 CIS (70 км/ч)
- Среднескоростные модели: 71-931M «Витязь-М» (65 км/ч), 71-923M «Богатырь-М» (65 км/ч)
- Традиционные модели: ЛВС-86 (50 км/ч), ЛМ-68М2 (50 км/ч)

В рамках исследовательской работы требуется разработать подход к обнаружению посторонних объектов на трамвайных путях Санкт-Петербурга с учетом характеристик подвижного состава, включая высокоскоростные трамваи типа Stadler B85600M («Чижик») и перспективные модели для маршрута «Славянка».

Дано:

- Архитектуры нейронных сетей: YOLO (v5, v8, X) и R-CNN (Fast R-CNN, Faster R-CNN)
- Библиотека компьютерного зрения OpenCV с модулями opencv_ml и opencv_video
- Требования к системе обнаружения, обусловленные скоростными характеристиками трамваев:
  - Максимальная скорость: 80 км/ч (Stadler B85600M)
  - Средняя эксплуатационная скорость: 40-60 км/ч
- Метрики оценки: точность детекции (Precision, mAP), скорость обработки (FPS), потребление ресурсов

Требуется:

- Изучить теоретические основы и архитектурные особенности основных и современных моделей распознавания объектов;
- Провести сравнительный анализ классических, одноэтапных и двухэтапных детекторов;
- Исследовать производительность моделей при работе с видеопотоком в условиях высокоскоростного трамвайного движения;
- Разработать критерии выбора оптимальной модели для систем видеомониторинга трамвайных путей;
- Обосновать выбор модели для последующей реализации системы обнаружения посторонних объектов на трамвайных рельсах.

Ограничения:

- Время обработки кадра: $t_{processing} \leq 50$ мс (≥ 20 FPS) для обеспечения безопасности при максимальной скорости трамвая 80 км/ч;
- Точность детекции: $mAP \geq 70\%$;
- Исследование ограничено архитектурами семейств YOLO и R-CNN, а так же классическими алгоритмами распознавания объектов;
- Рассматриваются реализации, совместимые с OpenCV;

- Система должна обеспечивать надежное обнаружение объектов при различных погодных условиях.

2 Компьютерное зрение

Компьютерное зрение (Computer Vision) - это научная область искусственного интеллекта, целью которой является создание моделей и алгоритмов для автоматического извлечения, анализа и понимания полезной информации из цифровых изображений или видеопотоков[3]. Формально, это задача преобразования данных из исходного просттранства высокомерных пикселей (матрицы интенсивностей), в компактное пространство семантических описаний, пригодных для принятия решений. Формально можно задать отображение:

$$I(h, w) \to T_r\texttt{, где h - высота, w - ширина изображения.}$$

Основная цель - наделить вычислительную машину способностью интерпретировать визуальную информацию на подобии, как это делает человек. Ключевые задачи компьютерного зрения включают:

- **Обнаружение и распознавание объектов** Локализация и идентификация объектов на изображении или на видеопотоке;
- **Классификация изображений** Отнесение всего изображения к одному из заранее заданых классов;
- **Сегментация** Разделение изображения на семантически значемые области или отдельные экземпляры объектов;
- **Определение позы** Построение скелетной модели объекта в пространстве;
- **Трекинг** Слежение за перемещением объекта в последовательнсоти кадров видеопотока.

Система компьютерного зрения состоит из нескольких уровней:

1. **Аппаратный уровень:** Камеры (средства захвата), вычислительные блоки (ЦП, ГП, ПЛИС). Для повышения эффективности используются специализированные оптические системы (стереозрение, панорамные камеры) и графические ускорители (напимер, CUDA для GPU)[3,7].
2. **Алгоритмический уровень:** Программные платформы (OpenCV, MATLAB, PCL) и два фундаментальных подхода:
   - **Классическое компьютерное зрение:** основаный на формальных алгоритмах анализа изображения.
   - **Машинное обучение и глубокие нейронные сети:** основаный на обучении моделей на подготовленных данных.

В рамках работы, расмотрена задача обнаружения и распознавания объектов, а в качестве исследования, были взяты классические и современные алгоритмы и методы распознования объектов. Далее в главе 3, приведены формальные описания классических и современных алгоритмов распознания.

3 Алгоритмы распознавания объектов

3.1 Классические алгоритмы компьютерного зрения

В основе классических алгоритмах лежит применение заранее сконструированных признаков, создаваемых на основе экспертных знаний о предметной области. Они включают:

- **Выделение границ** - поиск резких изменений интенсивности пикселей. Например, фильтры Хаара для вертикальныхграниц в точке (i,j) вычисляет разность сумм интенсивностей в двух соседних областях:

$$H(i,j) = \sum I(right_{region}) - \sum I(left_{region}).$$

  Для ускорения вычислений используется интегральное изображение II, где II(i,j) равно сумме пикселей выше и левее (i,j). Это позволяет вычислить сумму в любом прямоугольнике за O(1):

- **Выделение ключевых точек** - алгоритмы на основе SIFT (Scale-Invariant Feature Transform) и SURF находят утойчивые к масштабу и повороту точки, вычисляя для каждой - вектор, характеризующий ее окрестность (дескриптор)[3]. Последующее сопоставление дескрипторов позволяет сравнивать изображения.

- **Метод Виолы-Джонса** - использует каскад слабых классификаторов на основе признаков Хаара(описановыше),что позволяет быстро отбрасывать области фона и концентрироваться на потенциальных объектах[7].

Ниже на рисунках 1 - 3, изображен процесс применения фильтра Хаара, для прямоугольной окрестности точки:

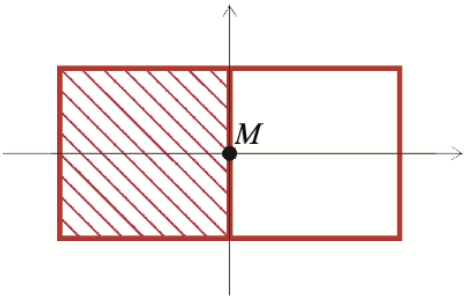

Рис. 1: Прямоугольная окрестность точки M

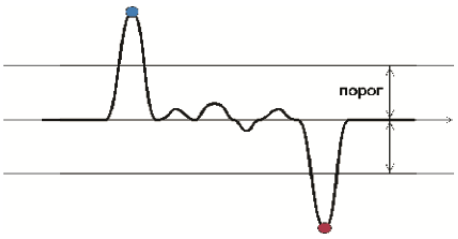

Рис. 2: Применение порога

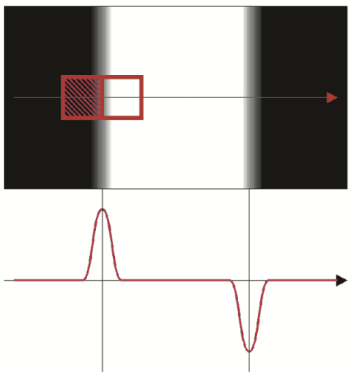

Рис. 3: Применение фильтра Хаара

3.2 Глубокое обучение в области компьютерного зрения

Глубокое обучение инициировало смену парадигмы в компьютерном зрении. В отличие от предыдущих методов, требующих ручного конструирования признаков, модели основываясь на обучении, автоматически извлекают признаки объектов непостредственно из исходных данных. Основные методы:

- **Сверточные нейронные сети (CNN)** - являются основной из современных подходов. Их архитектура схожа с принципом работы зрительной корой мозга:
  - **Сверточный слой (Convolutional Layer)** - применяет набор фильтров (ядер) K размером $k \times k$ к входному изображению I, выполняя операцию свертки:

$$(I \times K)(i,j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} I(i+m, j+n) \cdot K(n,m)$$

    Каждый фильтр учится реагировать на определенный низкоуровневый паттерн (градиенты, текстуры и др.). На рисунке 4, изображен пример работы сверточного слоя с матрицей I и фильтром K.
  - **Слой подвыборки (Pooling Layer)** - уменьшает размерность карт признаков, повышая инвариантность к малым смещениям. Наиболее распространена модификация Max-Pooling:

$$P(i,j) = max(I(region_{ij}))$$

  - **Функция активации** - в современных сетях (например в YOLO), часто используется функция Leaky ReLU: $f(x) = max(\alpha x, x)$, где α - малая константа $\approx (0.01)$, предотвращающая затухание нейронов[9].

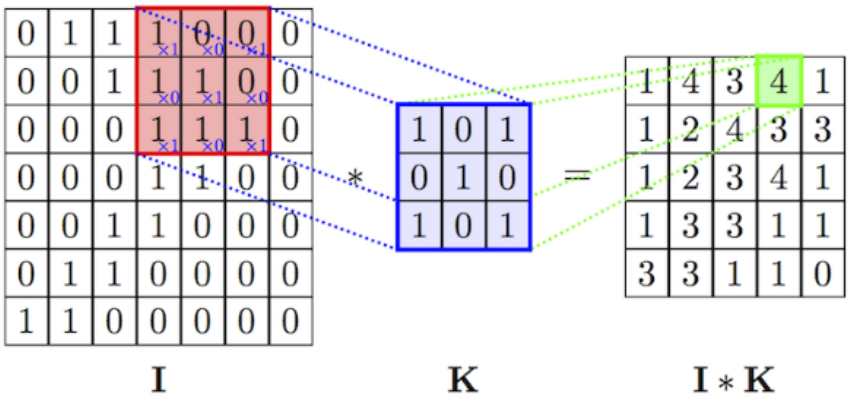

Рис. 4: Пример работы сверточного слоя с входной матрицей I и фильтром K

Обучение CNN происходит методом обратного распространения ошибки и стохастичекского градиентного спуска, минимизируя функцию потерь L между предсказанием модели $f(x_i, W)$ и истинной меткой y_i [3]:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i).$$

Распознавание объектов - это комбинированная задача локализации (где находится объект) и классификации (что это за объект). Существуют два основных подхода:

- **Двухэтапные детекторы на основе регионов** - такие методы сначала генерируют множество "предположительных"областей (Region Proposals), а затем классифицируют каж-

дую из них с помощью CNN. Такой подход обладает высокой точностью, однако их скорость решения задачи существенно ниже чем одноэтапные детекторы.

- **Одноэтапные детекторы (SSD)** - эти методы решают задачу локализации и классификации за один проход по сети, что делает их значительно быстрее чем двухэтапные подходы. В более раних одноэтапных моделей (например рание модели семейства YOLO), обладали точностью хуже чем двухэтапные модели (например из семейств R-CNN), на сегоднешний день, разница в точности минимальна.

Таким образом как одноэтапные (семейства YOLO) так и двухэтапные детекторы (семейства R-CNN) обладают своими характеристиками, находятся в состоянии постоянного развития и имеют различные компромиссы между точностью, скоростью и вычислительной сложностью. В связи с этим, ключевой задачей является выбор оптимального основного алгоритма для конкретной практичекой задачи. Последующие главы будут посвещены детальному рассмотрению различных модификаций обоих типов детекторов, и обоснованию выобора конкретного основного алгоритма для последующего решения задачи о "распознавании посторонних объектов на трамвайных рельсах в режиме реального времени".

3.3 Распознавание объектов в видеопотоке

Задача распознавания объектов в видеопотоке предполагает обработку последовательности кадров I_t с сохранением временной согласованности и учётом динамики сцены.

Существует несколько основных классов методов, применяемых для решения задачи детекции объектов в видеопотоке.

3.3.1 Методы фоновой субтракции

Методы фоновой субтракции используются для выделения движущихся объектов в видеопотоке путём отделения статического фона от динамического переднего плана. Данные методы широко применяются в системах видеонаблюдения, анализе транспортных потоков и робототехнике. Основная идея заключается в построении статистической модели фона и последующем сравнении каждого нового кадра с этой моделью для обнаружения изменений.

Алгоритм непрерывно обновляет модель фона, адаптируясь к постепенным изменениям освещения, погодных условий и других медленных изменений сцены. Для каждого пикселя оценивается вероятность его принадлежности к фону; пиксели, существенно отклоняющиеся от фоновой модели, классифицируются как элементы переднего плана.

Одним из наиболее распространённых подходов является использование модели гауссовой смеси (Gaussian Mixture Model, GMM), в которой распределение значений интенсивности каждого пикселя описывается как взвешенная сумма нескольких гауссовых распределений:

$$P(I_t(x,y)) = \sum_{k=1}^{K} \omega_{k,t}\, \mathcal{N}\big(I_t(x,y) \,|\, \mu_{k,t}, \Sigma_{k,t}\big),$$

где K - количество компонент гауссовой смеси (обычно $K \in [3,5]$), $\omega_{k,t}$ - вес k-й компоненты в момент времени t ($\sum_{k=1}^{K} \omega_{k,t} = 1$), $\mu_{k,t}$ - вектор математического ожидания, $\Sigma_{k,t}$ - ковариационная матрица k-й компоненты, а $\mathcal{N}(\cdot|\mu,\Sigma)$ - функция плотности многомерного нормального распределения.

Обновление параметров модели выполняется итеративно для каждого нового наблюдения. Веса компонент обновляются по правилу экспоненциального сглаживания:

$$\omega_{k,t} = (1-\alpha)\,\omega_{k,t-1} + \alpha\, M_{k,t},$$

где $\alpha \in (0, 1)$ - коэффициент обучения, определяющий скорость адаптации модели, а $M_{k,t}$ - индикаторная функция, принимающая значение 1, если текущее значение пикселя $I_t(x, y)$ соответствует k-й гауссовой компоненте (в пределах заданного порога), и 0 - в противном случае. После обновления веса нормализуются так, чтобы их сумма была равна единице.

Параметры $\mu_{k,t}$ и $\Sigma_{k,t}$ обновляются только для той компоненты, которая была сопоставлена текущему наблюдению, что позволяет модели эффективно адаптироваться к изменениям сцены при сохранении устойчивости фонового представления.

3.3.2 Метод оптического потока

Метод оптического потока используется для оценки видимого движения яркостных структур между последовательными кадрами видеопоследовательности. Алгоритмическая реализация метода позволяет вычислить поле векторов смещения (u, v), характеризующее движение точек изображения, что широко применяется в задачах трекинга, навигации и анализа динамических сцен.

Метод основан на предположении постоянства яркости, согласно которому интенсивность пикселя остаётся неизменной при его малом перемещении между соседними кадрами. Дополнительно вводится предположение о пространственной гладкости поля скоростей, что позволяет получить единственное решение задачи.

Предположение постоянства яркости формализуется следующим образом:

$$I(x + u,\, y + v,\, t + 1) = I(x,\, y,\, t).$$

При линейном разложении в ряд Тейлора и отбрасывании членов высшего порядка получаем дифференциальное уравнение оптического потока:

$$I_x\, u + I_y\, v + I_t = 0,$$

где u и v - компоненты вектора оптического потока по координатам x и y соответственно, $I_x = \frac{\partial I}{\partial x}$ и $I_y = \frac{\partial I}{\partial y}$ - частные производные интенсивности по каждой координате изображения, $I_t = \frac{\partial I}{\partial t}$ - частная производная по времени.

Следует отметить, что производная по времени I_t вводится в рамках непрерывной математической модели. В реальных видеосистемах время и изображение дискретны, поэтому I_t на практике представляет собой численную аппроксимацию временного изменения интенсивности, вычисляемую с помощью конечных разностей между соседними кадрами.

В дискретной форме временная производная аппроксимируется конечной разностью:

$$I_t(x, y, t) \approx \frac{I(x, y, t + 1) - I(x, y, t)}{\Delta t},$$

где Δt - интервал времени между соседними кадрами видеопоследовательности.

3.3.3 Многокадровое накопление и подтверждение объектов

В реальных условиях детекторы могут пропускать объекты на отдельных кадрах из-за помех, изменений освещения или сложного фона. Метод многокадрового накопления предназначен для увеличения надежности работы детектора в видеопотоке. Метод ведет учет всех объектов на детекцию и подсчитывает, сколько раз каждый из них был обнаружен в пределах временного окна (5-10 кадров). Объект считается подтвержденным, если количество его детекций превышает заданный порог. Объект считается исчезнувшим, если он не обнаружен в течение временного окна.

Вероятностная модель подтверждения нахождения объекта в кадре задается формулой:

$$P(valid|O_{1:T}) = \frac{\prod_{t=1}^{T} P(o_t|valid)P(valid)}{\prod_{t=1}^{T} P(o_t|valid)P(valid) + \prod_{t=1}^{T} P(o_t|invalid)P(invalid)}$$

Вероятностная модель основана на формуле Байеса, предполагая условную независимость наблюдений при фиксированной гипотезе V. $O_{1:t}$ - история наблюдений. Саму же гипотезу можно задать как:

$$H_V = \begin{cases} V: & \text{Последовательность } O_{1:t} \text{ вызвана присутствующим на кадрах объектом} \\ \neg V: & \text{Последовательность } O_{1:t} \text{ - несвязанные между собой ложные срабатывания (шум)} \end{cases}$$

Так как каждое новое обнаружение o_t, которое хорошо предсказуемо и похоже на объект, дает большое $P(o_t|V)$ и маленькое $P(o_t|\neg V)$. При превышении установленного порога, метод предсказывает нахождение объекта на временном окне.

3.4 Распознавание габаритов объекта

Задача распознавания габаритов объекта заключается в определении его положения на изображении и размеров в виде ограничивающего прямоугольника (bounding box). В современных одностадийных детекторах, таких как YOLO, параметры ограничивающего прямоугольника предсказываются непосредственно нейронной сетью в виде набора регрессионных коэффициентов.

Формально задача детекции может быть представлена как отображение:

$$I \to T_c, \quad I \in \mathbb{R}^{H \times W \times 3},$$

где входному изображению I сопоставляется набор предсказаний T_c, включающий параметры ограничивающих прямоугольников и вероятностные оценки.

Для каждого предсказанного прямоугольника сеть оценивает параметры t_x, t_y, t_w, t_h, ρ_{obj} и ρ_{class}, где t_x и t_y - параметры, определяющие смещение центра ограничивающего прямоугольника относительно левого верхнего угла соответствующей ячейки сетки, t_w и t_h - параметры масштабирования ширины и высоты прямоугольника относительно размеров априорного (anchor) прямоугольника, ρ_{obj} - вероятность наличия объекта в ячейке, ρ_{class} - условная вероятность принадлежности объекта к одному из классов.

Координаты центра и размеры ограничивающего прямоугольника вычисляются следующим образом:

$$x = \sigma(t_x) + c_x, \quad y = \sigma(t_y) + c_y,$$
$$w = a_w \cdot \exp(t_w), \quad h = a_h \cdot \exp(t_h),$$

где (c_x, c_y) - координаты ячейки сетки, (a_w, a_h) - размеры априорного ограничивающего прямоугольника (anchor), а $\sigma(\cdot)$ - сигмоидная функция активации.

3.4.1 Распознавание мелкогабаритных объектов

Задача обнаружения мелкогабаритных объектов представляет особую сложность, поскольку такие объекты занимают незначительную область изображения и представлены ограниченным числом пикселей. Классические алгоритмы компьютерного зрения, такие как SIFT

или SURF, а также базовые архитектуры нейронных сетей, как правило, оказываются недостаточно эффективными, поскольку не способны извлекать устойчивые и информативные признаки из малых областей изображения и слабо учитывают контекст сцены.

Современные подходы к детекции мелкогабаритных объектов основаны на использовании иерархического извлечения признаков и многомасштабного анализа, что позволяет сохранять информацию о мелких деталях при одновременном учёте глобального контекста.

Для повышения точности обнаружения мелкогабаритных объектов в современных архитектурах применяются следующие приёмы:

- увеличение пространственного разрешения карт признаков на ранних и промежуточных уровнях сети;
- использование модифицированных C2f-блоков с bottleneck-структурами и без них для эффективного повторного использования признаков;
- применение модулей пространственной пирамиды, таких как SPPF, для агрегирования признаков различных масштабов.

3.5 Распознавание материала объекта

Распознавание материала объекта заключается в классификации его поверхностных свойств на основе визуальных характеристик с использованием заранее подготовленной базы знаний. Пусть M - предсказанный материал объекта, $\mathcal{M}$ - ограниченное множество возможных материалов (база знаний), $F(I)$ - вектор признаков, извлеченных из изображения I. Тогда:

$$M = \arg\max_{m_i \in \mathcal{M}} P(m_i|F(I))$$

То есть вычисляется вероятность принадлежности объекта к каждому из возможных материалов из база знаний по признакам, затем выбирается тот материал, для которого условная вероятность оказывается наибольшей.

4 Библиотека OpenCV

OpenCV (Open Source Computer Vision Library) - представляет собой библиотеку алгоритмических реализаций методов компьютерного зрения с открытым исходным кодом. Библиотека включает в себя инструменты для обработки изображений, анализа видео, машинного обучения и распознавания объектов. Библиотека поддерживает кроссплатформенную реализацию на языках программирования C++, Python, Java и MATLAB с возможностью поддержки во всех видах операционных систем. Архитектура библиотеки организована в виде набора взаимосвязанных модулей:

Библиотека включает в себя классические алгоритмы для распознавания обхектов, такие как каскады Хаара, HOG - дескрипторы. Стоит отметить, что классические алгоритмы уступают в точности нейросетевым моедлям. Методы не способны обнаружить мелко-габаритные объекты, а набор предобученных моделей ограничен. Несмотря на это, библиотека OpenCV, способна обращатся к нейросетевым моделям посредством модуля opencv_ml и dnn.

Таблица 1: Основные модули библиотеки OpenCV

Модуль	**Назначение**	**Основные компоненты**
core	Базовые функции	Структуры данных, матричные операции, линейная алгебра
imgproc	Обработка изображений	Фильтрация, геометрические преобразования, морфологические операции
video	Анализ видео	Оптический поток, выделение фона, отслеживание объектов
ml	Машинное обучение	Классификаторы, регрессия, кластеризация
objdetect	Обнаружение объектов	Детекторы лиц, объектов, шаблонов
features2d	Работа с особенностями	Детекторы и дескрипторы ключевых точек

4.1 Модуль dnn

Модуль DNN (Deep Neural Networks) представляет собой высокоуровневый интерфейс для работы с нейронными сетями,обеспечивающий загрузку предобученных моделей из фреймворков глубокого обучения.

Таблица 2: Аспекты и описания модуля DNN

Аспект	**Описание**
Поддерживаемые фреймворки	TensorFlow, PyTorch, ONNX, Darknet, Caffe
Форматы моделей	.pb, .pth, .onnx, .cfg, .caffemodel
Аппаратное ускорение	CPU, NVIDIA GPU (CUDA), Intel (OpenVINO)
Оптимизации	Layer fusion, quantization, graph optimization

Процесс работы с моделями делится на четыре пункта:

- **Загрузка модели** - чтение архитектуры и весов из файлов;
- **Препроцессинг** - нормализация данных;
- **Инференс** - проход через сеть;
- **Постпроцессинг** - обработка выходных данных декодированием.

Данный модуль также может работать с моделями семейства YOLO и SSD. Возможна классификация изображений. Из минусов же стоит отметить, что модуль не предусматривает

возможность обучения самих моделей нейронных сетей и подходит более для чистых задач классификации, объекты которых точно принадлежат одному или нескольким из классов в базе знаний моделей.

4.2 Модуль opencv_video

Модуль `video` предоставляет алгоритмы для анализа видеопотоков в реальном времени. Основные направления включают оценку движения, выделение переднего плана и отслеживание объектов, что связано с поставленной в работе задачей. Модуль содержит методы, перечисленные в таблице 3.

Таблица 3: Методы модуля opencv_video

Метод	**Назначение**	**Принцип работы**
Optical Flow (Farneback)	Оценка плотного оптического потока	Поиск соответствий между кадрами на основе полиномиального разложения
Lucas–Kanade	Разреженный оптический поток	Вычисление смещения для характерных точек между последовательными кадрами
BackgroundSubtractorMOG2	Выделение движущихся объектов	Гауссова смесь для моделирования фона с адаптацией к изменению освещения
BackgroundSubtractorKNN	Выделение переднего плана	Метод k ближайших соседей для классификации пикселей на фон и передний план

Алгоритмы реализующие методы представленны в таблице 4:

Таблица 4: Алгоритмы отслеживания объектов в opencv_video

Алгоритм	**Тип отслеживания**	**Особенности**
MIL (Multiple Instance Learning)	Слежение за объектом	Обучение с несколькими экземплярами для устойчивости к окклюзии
KCF (Kernelized Correlation Filters)	Слежение на основе корреляции	Использование ядерных методов для эффективного вычисления корреляции
TLD (Tracking–Learning–Detection)	Долговременное слежение	Комбинация отслеживания, обучения и детектирования
GOTURN	Слежение на основе глубокого обучения	Использование предобученной сверточной нейросети

5 Семейство моделей YOLO

Семейство алгоритмов YOLO (You Only Look Once) представляет собой одноэтапные детекторы объектов, которые объединяют задачи локализации и классификации в единый процесс прямого распространения по сети[12]. Основная идея метода заключается в разделении входного изображения $I \in \mathbb{R}^{W \times H \times C}$ на сетку $S \times S$, где каждая ячейка отвечает за предсказание B ограничивающих рамок и соответствующих вероятностей классов объектов[12].

5.1 Математическая формализация

Пусть входное изображение имеет размерность $W \times H \times C$, где W и H - ширина и высота изображения, C - количество каналов (обычно 3 для RGB). Алгоритм YOLO осуществляет отображение:

$$f : \mathbb{R}^{W \times H \times C} \to \mathbb{R}^{S \times S \times (5B+K)} \tag{1}$$

где:

- S - размер сетки
- B - количество ограничивающих рамок на ячейку
- K - количество классов объектов

Для каждой ячейки сетки (i, j) модель предсказывает:

- Координаты центров ограничивающих рамок (x_{ij}, y_{ij}) относительно ячейки
- Размеры рамок (w_{ij}, h_{ij}) относительно размеров всего изображения
- Уверенность в наличии объекта $C_{ij} \in [0, 1]$
- Вероятности классов $p_{ij}^{(k)} \in [0, 1], k = 1, \ldots, K$

Финальные координаты ограничивающей рамки вычисляются как:

$$x = \sigma(t_x) + c_x \tag{2}$$
$$y = \sigma(t_y) + c_y \tag{3}$$
$$w = p_w e^{t_w} \tag{4}$$
$$h = p_h e^{t_h} \tag{5}$$

где (c_x, c_y) - координаты ячейки, (p_w, p_h) - размеры приоритетной рамки (anchor box), а σ - сигмоидная функция[11].

5.2 Функция потерь

Функция потерь в YOLO состоит из нескольких компонентов:

$$\mathcal{L} = \lambda_{coord}\mathcal{L}_{coord} + \lambda_{obj}\mathcal{L}_{obj} + \lambda_{noobj}\mathcal{L}_{noobj} + \lambda_{class}\mathcal{L}_{class} \tag{6}$$

где:

- $\mathcal{L}_{coord}$ - ошибка локализации (MSE по координатам рамок)
- $\mathcal{L}_{obj}$ и $\mathcal{L}_{noobj}$ - ошибки уверенности для ячеек с объектами и без
- $\mathcal{L}_{class}$ - ошибка классификации (бинарная кросс-энтропия)
- $\lambda_{coord}, \lambda_{obj}, \lambda_{noobj}, \lambda_{class}$ - весовые коэффициенты

В более поздних версиях YOLO используется функция потерь Complete IoU (CIoU):

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{7}$$

где ρ - евклидово расстояние, b и b^{gt} - центры предсказанной и истинной рамок, c - диагональ минимального ограничивающего прямоугольника, v - параметр, учитывающий соотношение сторон[11].

Архитектура моделей YOLO состоит из следующих основных компонентов:

5.2.1 Основная сверточная сеть (Backbone)

Сеть-основа использует модифицированную архитектуру CSPDarknet с блоками ELAN (Efficient Layer Aggregation Network). Каждый блок ELAN состоит из:

- Базового сверточного слоя с нормализацией по батчу и активацией SiLU:

$$\text{CBS}(x) = \text{SiLU}(\text{BN}(\text{Conv}(x))) \tag{8}$$

- Нескольких ветвей обработки с различными ядрами свертки
- Слоя конкатенации для объединения признаков

5.2.2 Блок агрегации признаков (Neck)

В YOLOv7 используется комбинация SPPCSPC (Spatial Pyramid Pooling Combined with CSPNet) и PANet (Path Aggregation Network). Модуль SPPCSPC выполняет:

$$\text{SPPCSPC}(x) = \text{Concat}[\text{MaxPool}(x, k_1), \text{MaxPool}(x, k_2), \text{MaxPool}(x, k_3), \text{MaxPool}(x, k_4)] \tag{9}$$

где k_i - размеры ядер пулинга: 1×1, 5×5, 9×9, 13×13[12].

5.2.3 Головная часть (Head)

Головная часть детектора YOLOv7 представляет собой выходной модуль нейронной сети, предназначенный для формирования предсказаний ограничивающих прямоугольников и вероятностных оценок классов объектов. В YOLOv7 применяется механизм многомасштабного предсказания, позволяющий осуществлять детекцию объектов различных размеров на нескольких уровнях пространственного разрешения.

В рамках данной архитектуры используются три уровня детекции:

- уровень P3 - высокий уровень пространственного разрешения, ориентированный на обнаружение мелких объектов;
- уровень P4 - промежуточный уровень, предназначенный для объектов среднего размера;
- уровень P5 - низкий уровень пространственного разрешения, используемый для детекции крупных объектов.

На каждом уровне формируется выходной тензор вида:

$$T \in \mathbb{R}^{S_l \times S_l \times (3 \cdot (5+K))}, \tag{10}$$

где S_l - размер сетки на уровне l, 3 - количество априорных ограничивающих прямоугольников (anchor boxes), используемых для каждой ячейки сетки, а K - число классов объектов.

5.2.4 Модуль C3K2

Блок C3K2 представляет собой оптимизированную версию CSP-архитектуры с ядрами свертки 3x3:

$$\text{C3K2}(x) = \text{Concat}[x_1, \text{Bottleneck}^N(x_2)] \tag{11}$$

где вход x разделяется на две части x_1 и x_2, а Bottleneck^N обозначает N последовательных Bottleneck-блоков[13].

Каждый Bottleneck-блок реализует операцию:

$$\text{Bottleneck}(x) = x + \text{Conv}_{3\times3}(\text{Conv}_{1\times1}(x)) \tag{12}$$

5.2.5 Модуль SPFF (Spatial Pyramid Pooling Fast)

SPFF использует параллельную обработку с различными размерами ядер пулинга:

$$\text{SPFF}(x) = \text{Concat}[\text{AdaptiveMaxPool}_{k_i}(x)]_{i=1}^{4} \tag{13}$$

где k_i адаптивные размеры пулинга, вычисляемые на основе входного разрешения[15].

5.2.6 Механизм внимания C2PSA

Блок C2PSA объединяет Cross Stage Partial архитектуру с пространственным вниманием:

$$\text{C2PSA}(x) = \text{CSP}(\text{SA}(x_1), x_2) \tag{14}$$

где SA - модуль пространственного внимания, вычисляемый как:

$$\text{SA}(x) = x \otimes \sigma(\text{Conv}_{7\times7}(x)) \tag{15}$$

где $\otimes$ - поэлементное умножение, σ - сигмоидная функция[15].

5.3 Оптимизации обучения

5.3.1 Аугментация данных

В обучении YOLO используются методы аугментации:

- Mosaic аугментация: объединение 4 изображений в одно
- MixUp: линейная интерполяция между изображениями и метками
- CutMix: вставка фрагментов одного изображения в другое

Математически MixUp определяется как:

$$\hat{x} = \lambda x_i + (1-\lambda)x_j, \quad \hat{y} = \lambda y_i + (1-\lambda)y_j \tag{16}$$

где $\lambda \sim \text{Beta}(\alpha, \alpha)$[13].

5.3.2 Оптимизация функции потерь

В YOLOv11 используется модифицированная функция потерь с фокусным фактором:

$$\mathcal{L}_{focal} = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{17}$$

где p_t - предсказанная вероятность правильного класса, α_t - балансирующий параметр, γ - фокусный параметр[13].

5.3.3 Регуляризация

Для предотвращения переобучения применяются:

- DropBlock: удаление смежных областей активаций

$$\text{DropBlock}(x) = M \odot x \tag{18}$$

где M - бинарная маска с блоками нулей

- Weight Decay: L_2-регуляризация весов
- Label Smoothing: сглаживание меток

$$y_{ls} = (1 - \epsilon)y + \frac{\epsilon}{K} \tag{19}$$

5.4 Вычислительная эффективность

Оценка сложности модели включает:

- Количество параметров: $P = \sum_{l=1}^{L}(C_{in}^{(l)} \cdot C_{out}^{(l)} \cdot K_w^{(l)} \cdot K_h^{(l)})$
- Вычислительная сложность (FLOPs): $F = \sum_{l=1}^{L}(2 \cdot C_{in}^{(l)} \cdot C_{out}^{(l)} \cdot K_w^{(l)} \cdot K_h^{(l)} \cdot W_{out}^{(l)} \cdot H_{out}^{(l)})$
- Объем памяти: $M = \sum_{l=1}^{L}(C_{out}^{(l)} \cdot W_{out}^{(l)} \cdot H_{out}^{(l)} \cdot 4)$ байт (float32)

Для оптимизации используются:

- Квантование весов до INT8/INT16
- Свертки с разделением по глубине (Depthwise Separable Convolutions)
- Призматические свертки для уменьшения размерности

5.5 YOLOv5

Модель YOLOv5 является реализацией семейства одноэтапных детекторов, разработанной Ultralytics. Её развитие привело к созданию модификации YOLOv5u, которая интегрирует разделённую головную часть, не использующую механизм якорей (anchor-free) и прогноз уверенности (objectness)[14]. Данная архитектурная модификация устраняет зависимость от параметров якорных рамок, характерную для первоначального дизайна, переходя к непосредственной регрессии центров и размеров ограничивающих рамок.

Архитектура модели YOLOv5, представлена на рисунке 5.

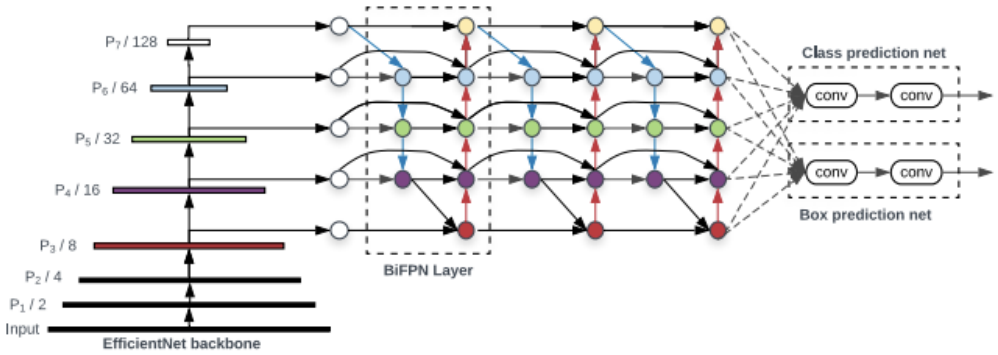

Рис. 5: Архитектура модели YOLOv5

Архитектурные особенности:

- Реализация безъякорной разделённой головы (anchor-free split head) в версии YOLOv5u.
- Наличие предобученных моделей различного масштаба (nano, small, medium, large, extra-large).
- Поддержка полного конвейера работы: обучение, валидация, инференс и экспорт[12].

Замечания:

- Модели, обученные с использованием оригинального репозитория yolov5, несовместимы с библиотекой ultralytics/ultralytics, что приводит к фрагментации экосистемы[12].
- Архитектура ориентирована на задачу детектирования объектов и не предоставляет встроенной поддержки других задач компьютерного зрения, таких как сегментация или оценка позы[12].

5.6 YOLOv8

Модель YOLOv8 представляет собой дальнейшее развитие архитектуры YOLO, представленное в 2023 году. Её ключевой особенностью является изначальная поддержка множества задач компьютерного зрения на основе единой архитектурной платформы. Архитектура включает усовершенствованную основную сеть (backbone), блок агрегации признаков (neck) и безъякорную головную часть (anchor-free head)[15].

Ниже на рисунке 6, показана архитектура модели.

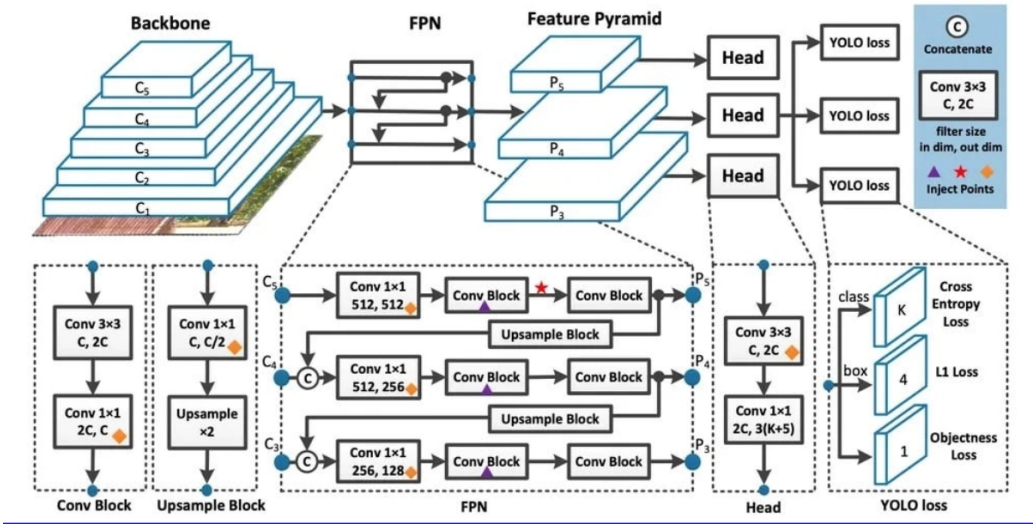

Рис. 6: Архитектура модели YOLOv8

Архитектурные особенности:

- Единая платформа для детектирования объектов (Detect), сегментации экземпляров (Segment), оценки позы (Pose), классификации изображений (Classify) и детектирования ориентированных объектов (OBB)[13].
- Использование безъякорной головной части (Anchor-Free Ultralytics Head).
- Предоставление ряда предобученных моделей, оптимизированных под каждую задачу (например, yolov8n.pt, yolov8n-seg.pt)[13].

Замечания:

- Реализация многозадачности в рамках единой экосистемы увеличивает общую сложность кодовой базы.
- Крупные варианты модели (YOLOv8x) обладают высокими требованиями к вычислительным ресурсам и памяти, особенно при выполнении задач сегментации или оценки позы в высоком разрешении[13].

5.7 YOLOX

Архитектура YOLOX, представленная в 2021 году, реализует парадигму детектирования без якорей (anchor-free) с использованием разделённой головной части (decoupled head). В данной конструкции задачи классификации объекта и регрессии ограничивающей рамки решаются раздельными ветвями сети. Для назначения позитивных примеров в процессе обучения применяется стратегия SimOTA (Simplified Optimal Transport Assignment)[11].

На рисунке 7, представлена архитектура модели YOLOX. Как видно, головная часть модели выделена в виде отдельного блока.

Архитектурные особенности:

- Разделённая головная часть (decoupled head) для независимой обработки задач классификации и регрессии.
- Полностью безъякорный дизайн (anchor-free), исключающий необходимость подбора параметров якорных рамок.
- Использование продвинутой стратегии назначения меток SimOTA для улучшения процесса обучения[11].

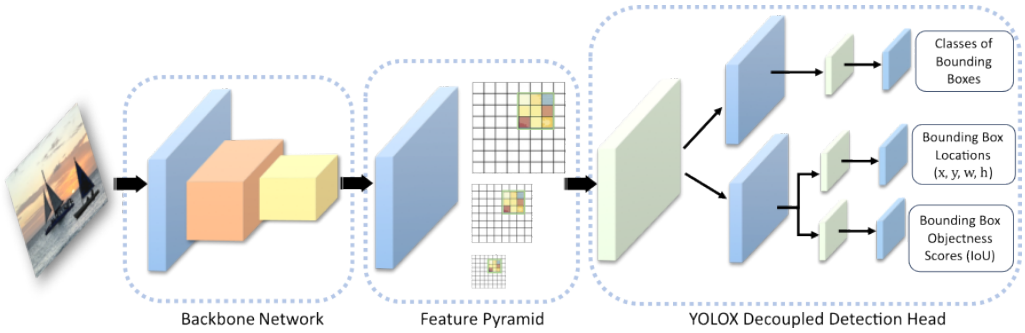

Рис. 7: Архитектура модели YOLOX

Замечания:

- Архитектура сфокусирована на задаче детектирования объектов и не включает встроенной поддержки других задач компьютерного зрения, таких как сегментация[11].
- Экосистема и инструментарий YOLOX менее развиты по сравнению с экосистемой Ultralytics, что может усложнять процесс развёртывания и интеграции[11].

5.8 SPP в моделях YOLO

Модуль Spatial Pyramid Pooling Fast (SPPF) является ключевым компонентом neck-части (промежуточного блока агрегации признаков) архитектуры YOLOv8, предназначенным для эффективного захвата контекстной информации на различных пространственных масштабах. Его функция критически важна для стабильного и точного определения габаритов (ограничивающих рамок, bounding boxes) объектов в видеопотоке в режиме реального времени.[13],[16]

5.8.1 Архитектура и математическая модель

На рисунке 8, представлена более подробная архитектура модели YOLOv8, которая включает в себя блок SPP.

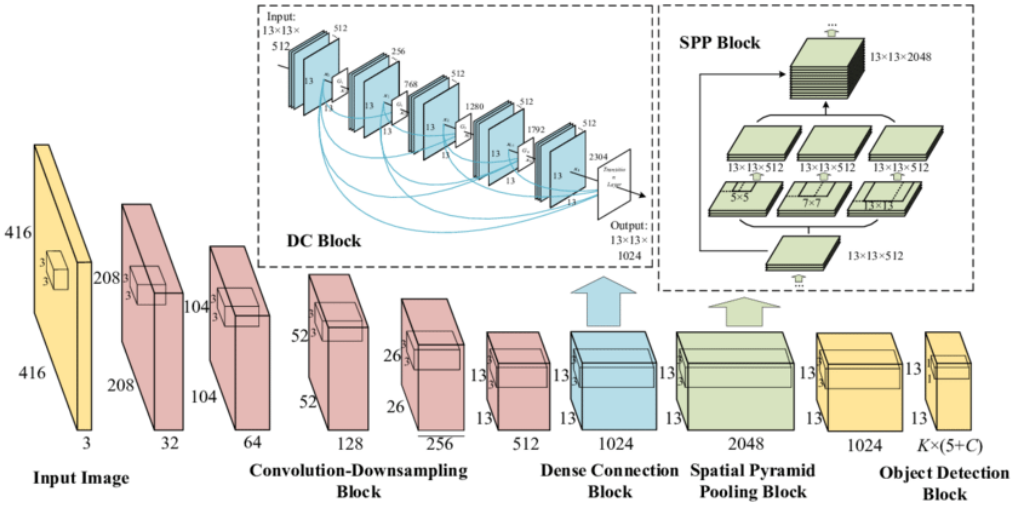

Рис. 8: Архитектура блока SPP

Модуль SPPF в YOLOv8 представляет собой оптимизированную и более быструю версию классического модуля Spatial Pyramid Pooling (SPP). Его архитектура следует последовательному паттерну, а не параллельному, что снижает вычислительные затраты[13],[16].

Алгоритм работы модуля может быть описан следующим образом:

1. **Входной сверточный слой:** Полученную от backbone-сети карту признаков $F_{in} \in \mathbb{R}^{C \times H \times W}$ пропускают через сверточный слой (Conv) для предварительной обработки и уменьшения размерности каналов: $F_{conv} = \text{Conv}(F_{in})$.

2. **Каскадное максимальное объединение (MaxPool):** Тензор F_{conv} последовательно пропускается через три идентичных слоя операции MaxPool с ядром фиксированного размера $k \times k$ (обычно 5×5) и шагом (stride) 1. Важной особенностью является использование паддинга (padding) для сохранения пространственных размеров. После каждого слоя пулинга сохраняется промежуточный тензор.

   - $P_1 = \text{MaxPool}_{k \times k, stride=1, pad}(F_{conv})$
   - $P_2 = \text{MaxPool}_{k \times k, stride=1, pad}(P_1)$
   - $P_3 = \text{MaxPool}_{k \times k, stride=1, pad}(P_2)$

   Каскадная структура эффективно имитирует рецептивные поля разного размера: P_1 захватывает признаки соседних пикселей, P_2 - более отдаленный контекст, а P_3 - глобальный контекст в пределах ядра.

3. **Конкатенация:** Исходный тензор F_{conv} и все три полученных тензора P_1, P_2, P_3 объединяются (concatenate) по измерению каналов (channel dimension):

$$F_{concat} = \text{Concat}(F_{conv}, P_1, P_2, P_3)$$

   В результате формируется обогащенная карта признаков F_{concat}, содержащая информацию об объекте и его окружении в разных масштабах.

4. **Выходной сверточный слой:** Объединенный тензор пропускается через финальный сверточный слой для смешивания (fusion) признаков из разных масштабов и снижения размерности до требуемого значения: $F_{out} = \text{Conv}(F_{concat})$.

Таблица 5: Этапы обработки данных в модуле SPPF YOLOv8

Этап	**Математическая операция / цель**	**Вклад в задачу детектирования**
Входной Convolution	$F_{conv} = \text{Conv}(F_{in})$	Предобработка признаков, уменьшение каналов.
Каскадный MaxPool (уровень 1)	$P_1 = \text{MaxPool}_{5\times5}(F_{conv})$	Извлечение локального контекста, инвариантность к малым смещениям.
Каскадный MaxPool (уровень 2)	$P_2 = \text{MaxPool}_{5\times5}(P_1)$	Извлечение контекста среднего масштаба, охват части объекта.
Каскадный MaxPool (уровень 3)	$P_3 = \text{MaxPool}_{5\times5}(P_2)$	Извлечение глобального контекста, понимание окружения объекта.
Конкатенация	$F_{concat} = \text{Concat}(F_{conv}, P_1, P_2, P_3)$	Агрегация мультимасштабных признаков в единый дескриптор.
Выходной Convolution	$F_{out} = \text{Conv}(F_{concat})$	Фьюжн признаков, адаптация для последующих слоев сети.

6 Альтернативные нейронные сети компьютерного зрения

6.1 R-CNN

R-CNN (Regions with Convolutional Neural Networks) - это двухэтапный метод детекции объектов, предложенный в 2014 году. На первом этапе алгоритм генерирует набор регионов-кандидатов (Region Proposals) с помощью алгоритма Selective Search. Затем каждый регион обрабатывается свёрточной нейронной сетью для извлечения признаков, после чего происходит классификация с помощью отдельного классификатора, например, SVM[10]. Несмотря на высокую точность, R-CNN отличается значительной вычислительной сложностью и медленной скоростью работы.

6.1.1 Fast R-CNN

Fast R-CNN является улучшением оригинальной модели R-CNN и решает проблему низкой скорости за счёт обработки всего изображения один раз свёрточной сетью с последующим извлечением признаков для регионов интереса (RoI Pooling). Далее регионы классифицируются и локализуются одновременно, что значительно ускоряет процесс по сравнению с R-CNN[10]. Однако Fast R-CNN остаётся двухэтапным детектором, что ограничивает его применимость в задачах с жёсткими требованиями к времени отклика.

6.2 Основание выбора семейства моделей YOLO

Выбор в пользу моделей семейства YOLO обусловлен их превосходством по ключевым критериям качества и эффективности, а также актуальностью для задач детекции в режиме реального времени. В частности, по сравнению с другими популярными архитектурами, такими как R-CNN и Fast R-CNN, модели YOLO предлагают интегрированный одноэтапный подход, объединяющий локализацию и классификацию объектов в единую сеть, что обеспечивает значительно более высокую скорость инференса при сопоставимом или лучшем уровне точности[10,11,12,13,15].

Среди различных версий YOLO, модели YOLOv5, YOLOX и YOLOv8 выделяются оптимальным балансом между точностью (mAP@50, mAP@50-95), скоростью обработки и размером модели (количеством параметров). Эти модели демонстрируют улучшенные показатели точности и производительности по сравнению с другими представителями семейства YOLO, что подтверждается рядом исследований и сравнений[10,11,12,13,15].

Отказ от использования других архитектур YOLO обоснован тем, что они либо уступают по mAP, либо имеют более низкую скорость инференса и/или существенно больший размер модели, что затрудняет их применение в системах с ограниченными вычислительными ресурсами и требованиями к быстродействию[10,11,12,13,15].

В свою очередь, традиционные двухэтапные детекторы, такие как R-CNN и Fast R-CNN, несмотря на высокую точность, обладают существенными недостатками с точки зрения вычислительной эффективности. Их необходимость выполнения отдельного этапа генерации регионов и последующего извлечения признаков приводит к значительному увеличению времени обработки одного изображения, что делает эти методы непригодными для задач, требующих работы в режиме реального времени[10].

Таким образом, использование одноэтапных моделей семейства YOLO, в частности YOLOv5, YOLOX и YOLOv8, является оптимальным выбором для задач объектной детекции, сочетающим высокую точность, вычислительную эффективность и гибкость применения в различных сценариях.

7 Сравнение моделей

7.1 Общее сравнение моделей YOLOv8, YOLOv5 и YOLOX

В данном разделе представлен детальный анализ и сравнение архитектурных особенностей, ключевых характеристик, производительности и экосистемы трёх популярных моделей детекции объектов: YOLOv5, YOLOX и последней версии YOLOv8. Сравнение основано на данных с датасета COCO и включает качественные и количественные показатели.

Критерии сравнения Для объективной оценки моделей были выбраны следующие основные критерии:

- **mAP@50** (mean Average Precision at IoU=0.5) - метрика, измеряющая среднюю точность обнаружения объектов при пороге перекрытия 50%. Это классический и широко используемый показатель, позволяющий оценить способность модели правильно обнаруживать объекты с достаточно свободным допуском по локализации.
- **mAP@50-95** - более строгая и информативная метрика, учитывающая среднюю точность при диапазоне порогов перекрытия от 50% до 95% с шагом 5%. Данный показатель отражает общую точность модели при более жёстких требованиях к качеству локализации и является стандартом для современных сравнений в задачах детекции.
- **Скорость инференса** (в миллисекундах) - время, необходимое модели для обработки одного изображения. Важный критерий для реального применения, особенно в системах с ограниченными ресурсами или требующих обработки в реальном времени.
- **Количество параметров** (в миллионах) - размер модели, напрямую влияющий на требования к памяти и вычислительным ресурсам. Компактные модели предпочтительнее для встраиваемых систем и устройств с ограниченными ресурсами.

Обоснование выбора критериев Выбор данных критериев обусловлен необходимостью комплексного анализа моделей с учётом как их точности, так и практических аспектов применения. Метрики mAP дают количественную оценку качества детекции, отражая способность моделей правильно классифицировать и локализовывать объекты. При этом mAP@50-95 обеспечивает более всестороннюю картину, нежели более простой mAP@50.

Скорость инференса и количество параметров характеризуют эффективность модели и её пригодность для внедрения в реальные системы. Высокая точность без приемлемой скорости и размера модели часто оказывается неприменимой в практических задачах, особенно в области мобильных и встроенных решений.

Таким образом, совокупность этих критериев позволяет сбалансированно оценить как качество работы модели, так и её технические и эксплуатационные характеристики.

7.1.1 Архитектурные компоненты

YOLOv5

- **Backbone:** Модифицированная версия CSPDarknet53, эффективно использующая Cross-Stage Partial connections (CSP) для снижения вычислительной сложности и улучшения градиентного потока.
- **Neck:** PANet (Path Aggregation Network) с усовершенствованной структурой для агрегации признаков разных масштабов.
- **Head:** Якорный (anchor-based) детектор с тремя масштабами выходных данных для обнаружения объектов разных размеров.

YOLOX

- **Безякорный (anchor-free) механизм:** Отказ от заранее заданных якорных рамок, что упрощает обучение и снижает вычислительную нагрузку.
- **Разделённая голова (Decoupled Head):** Отделение задач классификации и регрессии ограничивающих рамок на разные ветви, способствующее быстрой сходимости и улучшению точности.
- **Динамическое назначение меток (SimOTA):** Оптимизированный алгоритм для эффективного распределения меток объектам во время обучения.

YOLOv8

- **Backbone:** Усовершенствованная CSPDarknet с новым модулем C2f (Cross-Stage Partial Bottleneck с двумя свёртками), заменяющим C3 из YOLOv5, что улучшает градиентный поток и качество извлечения признаков.
- **Head:** Бесякорная (anchor-free) архитектура, исключающая необходимость определения размеров якорей.
- **Механизм детекции:** Центрированный подход к предсказанию bounding box вместо углового.
- **Task-Aligned Assigner:** Оптимизированный метод назначения меток, обеспечивающий лучшее согласование задач классификации и локализации.
- **Унифицированный фреймворк:** Поддержка не только обнаружения объектов, но и сегментации экземпляров, оценки позы и классификации изображений.

7.1.2 Ключевые особенности

YOLOv5

- Автоматическая генерация якорей (anchor boxes) через кластеризацию K-средних на обучающем наборе данных.
- Использование мозаичной аугментации данных (Mosaic augmentation) во время обучения.
- Гиперпараметрическая оптимизация для повышения производительности.
- Поддержка различных размеров моделей: n, s, m, l, x.

YOLOX

- Специализация исключительно на задаче обнаружения объектов без поддержки сегментации, оценки позы или классификации.
- Более сложная настройка и менее развитая экосистема и документация по сравнению с YOLOv8.

YOLOv8

- Новые функции потерь: VariFocal Loss для классификации вместо Binary Cross-Entropy; Distribution Focal Loss + CIOU Loss для регрессии bounding box.
- Улучшенные алгоритмы аугментации данных.
- Расширенные возможности обучения, включая поддержку трансферного обучения.

- Баланс между параметрами модели и производительностью.
- Комплексный фреймворк для различных задач компьютерного зрения.

Таблица 6: Архитектурные характеристики YOLOv8, YOLOv5 и YOLOX

Характеристика	**YOLOv5**	**YOLOX**	**YOLOv8**
Backbone	CSPDarknet53 (модификация)	CSPDarknet53 (модификация)	CSPDarknet с C2f модулем
Neck	PANet	PANet	PANet (обновленный)
Head	Якорный (anchor-based)	Безякорный (anchor-free), Decoupled Head	Безякорный (anchor-free), Task-Aligned Assigner
Механизм детекции	Прогноз по якорям	Безякорный, SimOTA	Центрированный, Task-Aligned Assigner
Поддержка задач	Обнаружение	Обнаружение	Обнаружение, сегментация, оценка позы, классификация
Тип потерь	BCE + CIOU	BCE + CIOU	VariFocal + Distribution Focal + CIOU
Аугментация	Mosaic и др.	Стандартная	Улучшенная аугментация
API и экосистема	Зрелая, широкая	Менее развитая	Современная, унифицированная

7.1.3 Сравнение производительности

Таблица 7 отражает сравнительные показатели средней точности (mAP), скорости инференса и количества параметров моделей на датасете COCO val2017.

Таблица 7: Сравнение производительности моделей YOLOv8, YOLOv5 и YOLOX на COCO val2017

Модель	**mAP@50-95**	**Скорость инференса (мс)**	**Параметры (млн)**	**Особенности**
YOLOv5n	28.0	6.3	1.9	Якорный
YOLOv8n	37.3	8.2	3.2	Безякорный, C2f
YOLOXs	40.5	2.56 (T4 TensorRT)	9.0	Безякорный, Decoupled Head
YOLOv5s	37.4	6.4	7.2	Якорный
YOLOv8s	44.9	8.2	11.2	Безякорный, новые функции потерь
YOLOXm	46.9	5.43 (T4 TensorRT)	25.3	Безякорный, SimOTA
YOLOv5m	45.4	8.2	21.2	Якорный
YOLOv8m	50.2	10.9	25.9	Унифицированный фреймворк
YOLOXl	49.7	9.04 (T4 TensorRT)	54.2	Безякорный
YOLOv5l	49.0	10.1	46.5	Якорный
YOLOv8l	52.9	13.4	43.7	Комплексный фреймворк
YOLOXx	51.1	16.1 (T4 TensorRT)	99.1	Безякорный
YOLOv5x	50.7	12.1	86.7	Якорный
YOLOv8x	53.9	15.4	68.2	Оптимизированный безякорный

Графики на рисунках 9 и 10 визуализируют производительность моделей YOLOv5 vs YOLOv8 и YOLOv8 vs YOLOX соответственно.

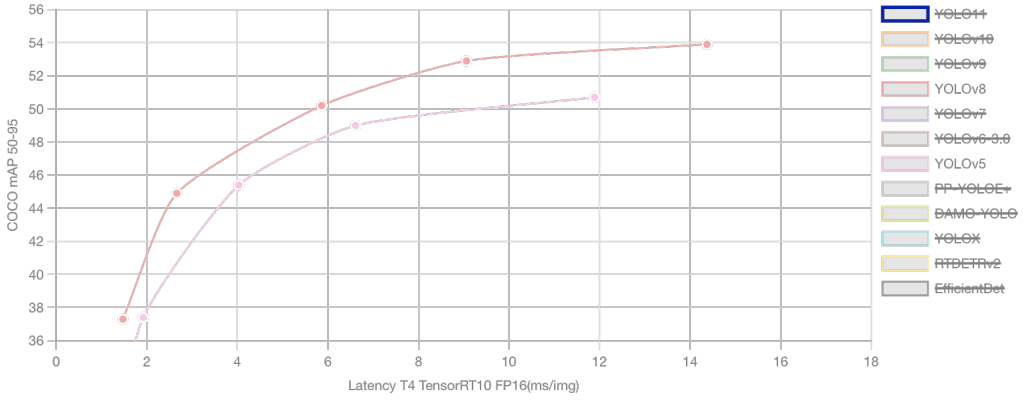

Рис. 9: Сравнение производительности YOLOv5 и YOLOv8

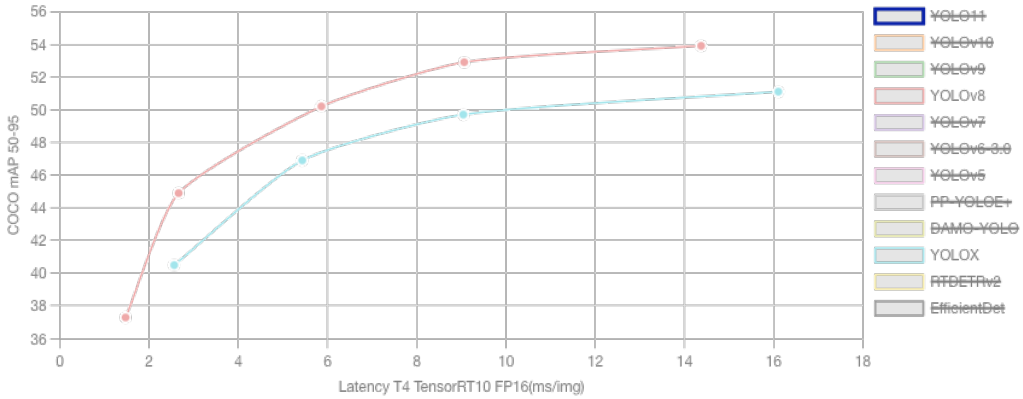

Рис. 10: Сравнение производительности YOLOv8 и YOLOX

7.1.4 Анализ и выводы

На основании представленных данных можно выделить следующие ключевые моменты:

1. **Преимущества YOLOv8:**
   - Последовательное превосходство по точности mAP@50–95 над YOLOv5 и YOLOX во всех сопоставимых категориях.
   - Эффективное использование параметров: например, YOLOv8x достигает более высокой точности, имея при этом меньше параметров, чем YOLOv5x и YOLOXx.
   - Унифицированный фреймворк для широкого спектра задач компьютерного зрения, включая сегментацию и оценку позы, в отличие от узконаправленных моделей YOLOv5 и YOLOX.
   - Улучшенные функции потерь и архитектурные модули (C2f), повышающие качество детекции и стабильность предсказаний.
2. **Сравнение скорости инференса:**
   - YOLOv5 обладает преимуществом в скорости инференса у моделей малого размера (n, s), что может быть критично в условиях жёстких вычислительных ограничений.
   - Для моделей среднего и крупного размера (m, l, x) разница во времени инференса несущественна и компенсируется значительным приростом точности YOLOv8.
   - YOLOX демонстрирует высокую производительность на GPU (T4 TensorRT), однако уступает YOLOv8 по точности и эффективности параметров.
3. **Ограничения и особенности экосистем:**
   - YOLOX ограничен задачей только обнаружения объектов и требует более глубокой настройки.
   - YOLOv5 не поддерживает задачи сегментации и оценки позы.
   - YOLOv8 обладает развитой экосистемой, удобным API, регулярными обновлениями и поддержкой промышленных форматов, что облегчает его использование и внедрение.

Итоговый вывод: С учётом требований задачи распознавания посторонних объектов на трамвайных путях, приоритетной является высокая точность детекции и универсальность модели. В этом контексте YOLOv8 представляется наиболее предпочтительной моделью благодаря более высокой точности, оптимальному балансу между скоростью и вычислительными ресурсами, а также широкой функциональности и поддержке.

8 Выбор модели

При выборе модели для задачи распознавания посторонних объектов на трамвайных путях в режиме реального времени основными критериями являются высокая точность детекции, скорость обработки кадров и экономия вычислительных ресурсов.

Классические алгоритмы и алгоритмы из библиотеки OpenCV не предоставляют возможностей для одновременного обеспечения высокой скорости, точности и комплексного анализа объектов, необходимых для задачи распознавания посторонних предметов на трамвайных путях в режиме реального времени. Такие алгоритмы обычно специализируются либо на выделении движущихся объектов, либо на простом отслеживании точек, но не способны эффективно объединять информацию о габаритах, типах и положении объектов в одном решении. Для получения полного набора признаков приходится комбинировать несколько алгоритмов, что значительно увеличивает вычислительную сложность и время обработки, а также усложняет разработку и поддержку системы. Кроме того, классические методы зачастую чувствительны к изменениям освещения, шумам и различным погодным условиям, что снижает надёжность детекции. Сложность алгоритмической реализации и недостаточная адаптивность таких методов делают их менее пригодными для практических систем видеомониторинга с жёсткими требованиями к быстродействию и точности.

В связи с этим для решения поставленной задачи более предпочтительно использовать нейронные сети, обучаемые на специализированных данных. Нейросетевые модели обеспечивают более высокую точность распознавания, обладают способностью учитывать сложные визуальные признаки, а также имеют унифицированный архитектурный подход, позволяющий одновременно выявлять объекты, их габариты и классы. Это существенно упрощает интеграцию и развитие системы, снижая алгоритмическую сложность и повышая стабильность работы в реальных условиях.

Сравнительный анализ современных моделей детектирования объектов показал, что модели семейства YOLO (YOLOv5, YOLOX, YOLOv8) имеют преимущество перед двухэтапными архитектурами R-CNN и Fast R-CNN в части скорости инференса, что критично для задач, связанных с высокоскоростным движением трамваев и требованиями к быстрому обнаружению объектов.

Из представленных одноэтапных моделей наибольший потенциал для практической реализации демонстрирует YOLOv8. Она сочетает в себе следующие достоинства:

- Высокую точность детекции (mAP@50-95) во всех категориях моделей, превосходя конкурентов;
- Оптимизированную архитектуру с современным backbone (CSPDarknet с C2f-модулем), обеспечивающую эффективное извлечение признаков;
- Безякорный механизм детекции (anchor-free), что упрощает обучение и повышает стабильность предсказаний;
- Унифицированный фреймворк, поддерживающий расширенные задачи компьютерного зрения, что открывает перспективы дальнейшего развития системы;
- Развитую и удобную экосистему с регулярными обновлениями, что облегчает интеграцию и сопровождение модели.

Несмотря на несколько более высокое время инференса у моделей YOLOv8 начального и среднего размера по сравнению с YOLOv5, превосходство по точности и функционалу делает её оптимальным выбором для системы видеомониторинга трамвайных путей. Модель обеспечивает баланс между качеством распознавания и скоростью работы, что позволяет выполнить требования к времени обработки кадров (не более 50 мс) и поддерживать необходимую точность (mAP $\geq 70\%$).

Таким образом, выбор падает на модель YOLOv8 для дальнейшей реализации системы обнаружения посторонних объектов на трамвайных путях Санкт-Петербурга.

Заключение

В ходе работы проведён сравнительный анализ современных моделей детектирования объектов, среди которых особое внимание уделено одноэтапным моделям семейства YOLO (YOLOv5, YOLOX, YOLOv8) и двухэтапным моделям R-CNN и Fast R-CNN. Анализ показал, что модели YOLO обладают преимуществом в скорости обработки и достаточной точности, что критично для задач видеомониторинга в условиях высокоскоростного движения трамваев.

На основании полученных результатов и требований к системе видеомониторинга трамвайных путей, оптимальной моделью для практической реализации выбрана YOLOv8. Эта модель обеспечивает необходимый баланс между высокой точностью детекции, скоростью инференса и удобством интеграции в существующую инфраструктуру.

Выбранная модель будет использоваться в практической реализации дипломной работы для создания системы распознавания посторонних объектов на трамвайных путях в режиме реального времени, что позволит повысить безопасность пассажирских перевозок в Санкт-Петербурге.

Список литературы

[1] YOLOv8 vs YOLO11: A Detailed Technical Comparison [Электронный ресурс]. - Ultralytics Documentation. - Режим доступа: https://docs.ultralytics.com/compare/yolov8-vs-yolo11/ (дата обращения: 04.11.2025).

[2] Алиев М. В., Бербенцев Д. А., Немыкин В. О., Алиева С. М. Алгоритмы распознавания объектов в видеопотоке и определение свойств их взаимного расположения // Вестник Адыгейского государственного университета. Сер. Естественно-математические и технические науки. - 2024. - № 2(341). - С. 27–34. DOI: 10.53598/2410-3225-2024-2-341-27-34.

[3] Горячкин Б. С., Китов М. А. Компьютерное зрение // Научная статья. - М.: МГТУ им. Н.Э. Баумана, 2023. - 29 с.

[4] Introduction to Support Vector Machines [Электронный ресурс] // OpenCV Documentation. - Режим доступа: https://docs.opencv.org/3.4/dc/dd6/ml_intro.html (дата обращения: 04.11.2025).

[5] Getting Started with Videos [Электронный ресурс] // OpenCV Documentation. - Режим доступа: https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html (дата обращения: 04.11.2025).

[6] Филичкин С. А., Вологдин С. В. Сравнение эффективности алгоритмов YOLOv5 и YOLOv8 для обнаружения средств индивидуальной защиты человека // Интеллектуальные системы в производстве. - 2023. - Т. 21, № 3. - С. 124–131. DOI: 10.22213/2410-9304-2023-3-124-131.

[7] Saha S. R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms [Электронный ресурс] // Towards Data Science. - 2018. - Режим доступа: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e/ (дата обращения: 04.11.2025).

[8] Алфимцев А. Н., Лычков И. И. Метод обнаружения объекта в видеопотоке в реальном времени // Вестник Тамбовского государственного технического университета. - 2011. - Т. 17, № 1. - С. 44–55.

[9] Зотов С. С., Яковлев А. А., Колчищев Д. А. Обнаружение объектов в реальном времени с помощью алгоритмов распознавания YOLO // Синергия наук. - 2018. - № 23. - С. 1245–1261.

[10] YOLOv5 vs YOLOv8: A Detailed Technical Comparison [Электронный ресурс]. - Ultralytics Documentation. - Режим доступа: https://docs.ultralytics.com/ru/compare/yolov5-vs-yolov8/ (дата обращения: 30.11.2025).

[11] YOLOX Documentation [Электронный ресурс]. - Режим доступа: https://yolox.readthedocs.io/en/latest/ (дата обращения: 30.11.2025).

[12] YOLOv5 Documentation [Электронный ресурс]. - Ultralytics. - Режим доступа: https://docs.ultralytics.com/ru/models/yolov5/ (дата обращения: 30.11.2025).

[13] YOLOv8 Documentation [Электронный ресурс]. - Ultralytics. - Режим доступа: https://docs.ultralytics.com/ru/models/yolov8/ (дата обращения: 12.12.2025).

[14] YOLOv8 vs YOLOX Comparison [Электронный ресурс]. - Ultralytics Documentation. - Режим доступа: https://docs.ultralytics.com/ru/compare/yolov8-vs-yolox/ (дата обращения: 12.12.2025).

[15] Ultralytics YOLOv5 Architecture [Электронный ресурс] // Ultralytics Documentation. - 2025-11-16. - URL: https://docs.ultralytics.com/yolov5/tutorials/architecture_description/#1-model-structure (дата обращения: 12.12.2025).

[16] Liu, Y., Yang, D., Song, T., Ye, Y., & Zhang, X. (2025). YOLO-SSP: An object detection model

based on pyramid spatial attention and improved downsampling strategy for remote sensing images. *The Visual Computer*, 41, 1467–1484.