УДК 004.853.3
doi:10.31799/1684-8853-2026-3-23-34
EDN: ZZVLVG

Научные статьи
Articles

Проектирование адаптивной архитектуры сверточной нейронной сети с использованием глубинных сепарабельных сверток для работы в реальном времени на встраиваемых устройствах

А. В. Сацюк[a], *канд. техн. наук, заведующий лабораторией, orcid.org/0009-0006-7228-8279, alexandrsatsuk@gmail.com*
С. А. Радковский[a], *канд. техн. наук, доцент, orcid.org/0009-0007-9411-949X*
А. И. Шеховцов[a], *канд. техн. наук, доцент, orcid.org/0009-0003-6963-815X*
С. Д. Сонина[a], *старший преподаватель, orcid.org/0009-0009-6114-4386*
А. А. Воробьев[a], *старший преподаватель, orcid.org/0009-0000-0938-6139*
[a]*Донецкий институт железнодорожного транспорта, Горная ул., 6, Донецк, Донецкая Народная Республика, 283018, РФ*

Введение: *развертывание современных систем компьютерного зрения на встраиваемых устройствах (например, Raspberry Pi 5) сталкивается с проблемой достижения реального времени (целевой порог 20 кадров/с) при анализе потока с MIPI-камеры из-за жестких ограничений ресурсов. Базовые однопроходные детекторы, такие как YOLOv8n, обладают избыточной вычислительной сложностью для таких условий.* ***Цель:*** *разработать адаптированную, легковесную архитектуру сверточной нейронной сети путем модификации YOLOv8n, сфокусировав вычислительные ресурсы на объектах среднего и крупного масштаба.* ***Результаты:*** *основное внимание уделено радикальному снижению FLOPs и параметров через замену стандартных сверток на глубинные сепарабельные свертки и исключение избыточных уровней детекции (сокращение выходных голов). Исследование проводилось на платформе Raspberry Pi 5 с входным разрешением 1280 × 720. Архитектурная реконфигурация привела к созданию модели OptiYOLO с минимальными показателями: 1,98 млн параметров и 4,1 GFLOPs. После квантования до INT8 достигнута частота обработки 28,7 кадров/с, что значительно превосходит требование реального времени. Общая точность (mAP0.5) составила 73,4 %, что признано приемлемым компромиссом. Точность для критических классов («автомобиль» — 86,1 %, «человек» — 82,5 %) подтвердила успешное перераспределение ресурсов на более крупные объекты.* ***Практическая значимость:*** *разработанная OptiYOLO предоставляет сбалансированное решение для автономных, энергоэффективных систем видеонаблюдения, где приоритетом является скорость обработки, а не детекция мелких объектов.*

Ключевые слова *— YOLO, глубинные сепарабельные свертки, Raspberry Pi 5, оптимизация нейронных сетей, реальное время, сверточная нейронная сеть, компьютерное зрение.*

Для цитирования: Сацюк А. В., Радковский С. А., Шеховцов А. И., Сонина С. Д., Воробьев А. А. Проектирование адаптивной архитектуры сверточной нейронной сети с использованием глубинных сепарабельных сверток для работы в реальном времени на встраиваемых устройствах. *Информационно-управляющие системы*, 2026, № 3, с. 23–34. doi:10.31799/1684-8853-2026-3-23-34, EDN: ZZVLVG

For citation: Satsiuk A. V., Radkovsky S. A., Shekhovtsov A. I., Sonina S. D., Vorobyov A. A. Designing an adaptive convolutional neural network architecture using depthwise separable convolutions for real-time operation on embedded devices. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2026, no. 3, pp. 23–34 (In Russian). doi:10.31799/1684-8853-2026-3-23-34, EDN: ZZVLVG

Введение

Разработка эффективных систем компьютерного зрения для работы в режиме реального времени на устройствах с ограниченными вычислительными ресурсами представляет одну из наиболее актуальных задач современного искусственного интеллекта. Особую практическую значимость приобретают автономные системы видеомониторинга, способные детектировать и классифицировать потенциально опасные объекты (такие как несанкционированный транспорт, люди в запрещенных зонах или оставленные предметы) в охраняемом периметре [1–3]. Ключевым требованием к подобным системам является не только высокая точность распознавания, но и способность обрабатывать видеопоток с минимальной задержкой для своевременного оповещения оператора-диспетчера. Важным условием для данной задачи, подтвержденным предварительным анализом сцен с целевой MIPI-камеры, является тот факт, что целевые объекты детекции (например, человек, транспортное средство или крупная сумка) в среднем занимают в кадре область не менее 80 × 80 пикселей. Это условие позволяет проводить дальнейшую архитектурную оптимизацию, ориентируясь на детекцию объектов среднего и крупного масштаба.

Несмотря на выдающиеся успехи глубокого обучения, в частности архитектур сверточных нейронных сетей семейства YOLO (You Only Look Once), которые установили новые стандарты в задачах детекции объектов в реальном времени [4–8], их прямое применение на маломощных embedded-платформах, таких как Raspberry Pi, сопряжено с существенными трудностями. Эти модели часто требуют значительных вычислительных затрат и объема памяти, что делает их непригодными для длительной автономной работы в составе компактных систем, оснащенных ресурсоэффективными компонентами, например MIPI-камерами [9, 10]. В связи с этим значимыми направлениями исследований стали архитектурная оптимизация и сжатие нейронных сетей. Перспективным подходом является замена стандартных сверточных слоев на глубинные сепарабельные свертки (Depthwise Separable Convolutions, DSC), которые позволяют радикально снизить вычислительную сложность и количество параметров модели при незначительной потере точности, что детально исследовано в работах [11, 12]. Другим важным направлением является разработка и использование легковесных Backbone-архитектур (например, MobileNet, ShuffleNet), специально спроектированных для мобильных устройств [13–15]. Однако их интеграция в архитектуру однопроходной детекции, сохраняющую высокую точность локализации объектов, остается комплексной задачей.

Следовательно, существует острая необходимость в создании специализированных, оптимизированных архитектур нейронных сетей, наследующих принципы скоростной однопроходной детекции от моделей-доноров, но адаптированных под строгие аппаратные ограничения целевой платформы.

Основной задачей данного исследования является разработка архитектуры сверточной нейронной сети для системы мониторинга в реальном времени, способной детектировать и классифицировать ограниченный набор потенциально опасных объектов на видеопотоке с MIPI-камеры, установленной на платформе Raspberry Pi 5.

Предлагаемый подход основывается на модификации и адаптивном упрощении архитектурных принципов YOLO. Главное внимание уделяется оптимизации глубины и ширины сети, реконфигурации слоев и выбору эффективных операций, включая DSC и анализ чувствительности слоев к pruning-операциям [16], для достижения оптимального баланса между скоростью инференса и точностью.

Системные требования

Успешная реализация системы мониторинга в реальном времени на платформе Raspberry Pi 5 потребовала формулирования ключевых условий и целевых показателей. Выбор данной платформы обусловлен необходимостью найти оптимальный компромисс между доступной ценой и минимально необходимой производительностью для автономной обработки видео с требуемой скоростью [17–19], обеспечивая немедленное оповещение диспетчера при обнаружении угрозы.

Для корректного проектирования были определены следующие технические требования и характеристики:

1) аппаратная платформа: Raspberry Pi 5;

2) входные данные: видеопоток с MIPI-камеры с разрешением 1280 × 720 пикселей;

3) целевые объекты: четыре класса («автомобиль», «человек», «животное», «сумка») с минимальным размером объекта ~80 × 80 пикселей;

4) критерии эффективности: частота обработки не менее 20 кадров/с при сохранении среднего уровня точности (mAP0.5) не ниже 0,70;

5) оптимизационный базис: использование методологии адаптации модели-донора YOLOv8n с применением DSC для радикального снижения вычислительной сложности.

Необходимо было определить системные ограничения и характеристики целевой сцены для правильного формулирования требований к архитектуре нейронной сети. После проведенных предварительных экспериментов по балансировке между качеством изображения, вычислительной нагрузкой и пропускной способностью шины установлено оптимальное входное разрешение для нейронной сети 1280 × 720 пикселей. Данное разрешение является приемлемым с точки зрения информативности для задачи детекции и одновременно допустимым для обработки в реальном времени на целевом аппаратном обеспечении при использовании MIPI-камеры [20].

Целевыми объектами для детекции и классификации выбраны автомобили, люди, животные и сумки. Критическим системным допущением, основанным на планируемой физической конфигурации системы (высота и угол установки камеры, заданный периметр), является ограничение максимальной дальности наблюдения до 50 м.

При таких сценарии и разрешении кадра минимальный ожидаемый размер любого целевого объекта в пикселях составляет в среднем не менее 80 × 80 [1] (рис. 1). Это допущение позволяет существенно упростить задачу детекции, сфокусировав архитектурные оптимизации на работе с объектами среднего и крупного масштаба, и отказаться от детектирования крайне мелких объектов, что является одной из наиболее ресурсоемких подзадач [21–25].

■ ***Рис. 1.*** Размер объектов наблюдения на кадре с разрешением 1280 × 720 пикселей
■ ***Fig. 1.*** Observed object dimensions within the 1280 × 720 frame

Таким образом, задача формализуется как разработка легковесной однопроходной архитектуры детекции (one-stage detector), адаптированной под следующие ключевые условия:

1) аппаратная платформа: Raspberry Pi 5 (ограниченные ресурсы CPU/GPU, оперативной памяти);

2) входные данные: видеопоток с разрешением 1280 × 720 пикселей;

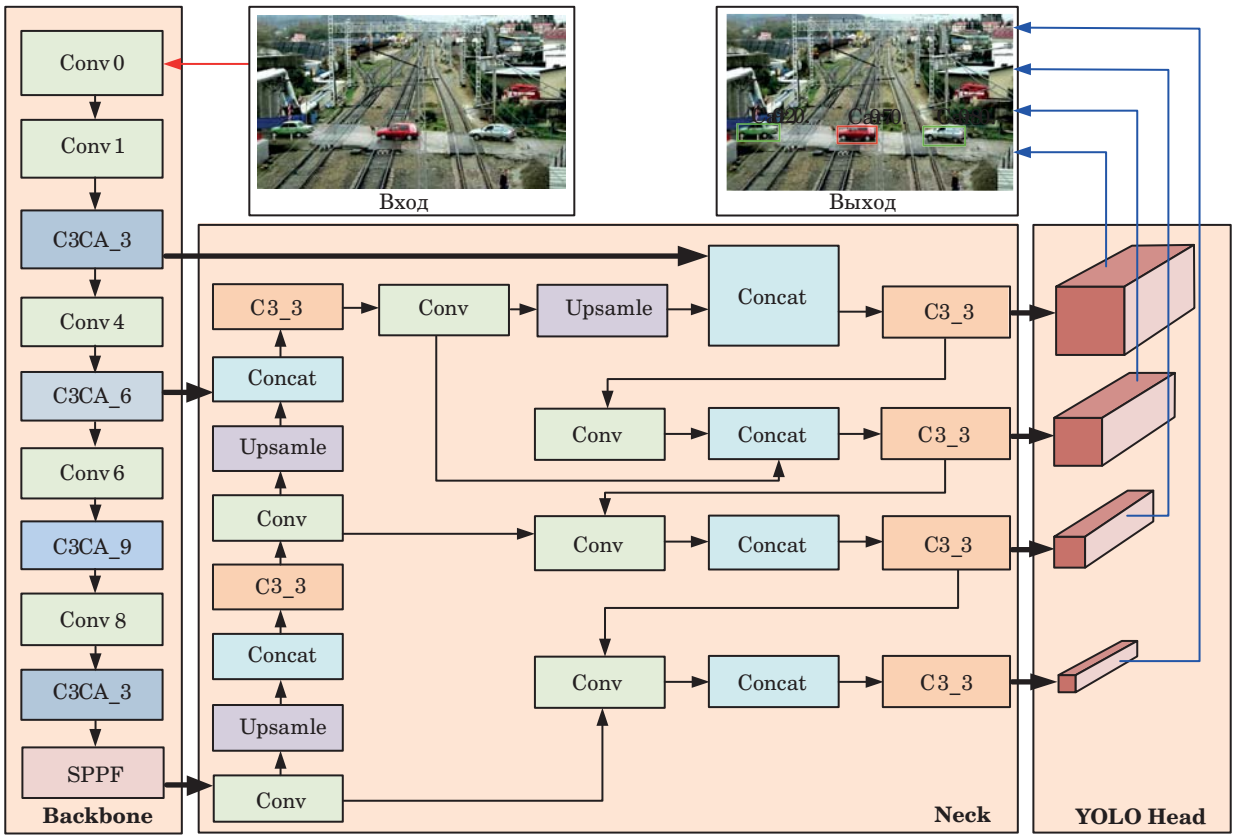
3) целевые объекты: четыре класса («автомобиль», «человек», «животное», «сумка») с минимальным ожидаемым размером ~80 × 80 пикселей;

4) критерий эффективности: частота обработки не менее 20 кадров/с для обеспечения работы в режиме, воспринимаемом как реальное время, при сохранении средней точности (mAP0.5) на уровне, достаточном для практического применения (не ниже 0,70);

5) методология: стратегическая адаптация и оптимизация модели-донора YOLOv8n, направленная на радикальное снижение вычислительной сложности при контролируемой потере точности, прежде всего на мелких объектах, исключенных из постановки задачи.

Общая концепция предлагаемой архитектуры

Предлагаемая архитектура нейронной сети, ориентированная на детекцию объектов в реальном времени на платформе Raspberry Pi 5, базируется на стратегии адаптации и оптимизации модели-донора YOLOv8n (рис. 2).



■ ***Рис. 2.*** Классическое представление архитектуры YOLO
■ ***Fig. 2.*** Standard representation of the YOLO architecture

Данная модель была выбрана в качестве исходного прототипа благодаря ее сбалансированному соотношению скорости и точности в рамках семейства YOLO, а также изначально заложенной ориентации на применение в ресурсоограниченных средах [26]. Однако, как показано выше, прямое использование базовой модели для целевого сценария классификации и детекции с разрешением 1280 × 720 и частотой не менее 20 кадров/с на Raspberry Pi 5 невозможно. Даже эта «легковесная» версия (YOLOv8n, YOLOv8s) не может быть напрямую развернута на целевом оборудовании без существенных доработок, направленных на достижение необходимой частоты кадров при работе с потоком данных от MIPI-камеры с сохранением точности на заданных классах объектов.

Принципом предлагаемой адаптации является учет специфики целевой сцены, в частности отсутствия необходимости детектировать объекты размером менее 80 × 80 пикселей. Это позволяет провести целенаправленную редукцию архитектуры, фокусируясь на эффективном извлечении признаков для объектов среднего и крупного масштаба.

Главная идея разработки заключается в последовательном и целенаправленном упрощении архитектуры-донора при сохранении ее ключевых функциональных преимуществ: однопроходной схемы детекции и эффективной многоуровневой пирамиды признаков. Упрощение реализуется по трем основным направлениям.

Во-первых, проводится модификация и редукция Backbone-сети (хребта архитектуры), ответственной за извлечение признаков. Стандартные сверточные блоки C2f в YOLOv8n, несмотря на их эффективность за счет обильных остаточных связей, содержат значительное количество стандартных 3 × 3 сверток и сложную ветвистую структуру, что делает их вычислительно дорогими для целевой платформы. Вместо них предлагается использовать оптимизированные блоки, основанные на DSC [11] и линейных bottleneck-структурах, что позволяет радикально сократить количество параметров и операций умножения-сложения. Глубина Backbone целенаправленно уменьшается для снижения латентности, при этом для компенсации потенциальной потери контекста усиливается роль механизмов агрегации признаков на разных уровнях. Учитывая, что входное изображение имеет фиксированное разрешение 1280 × 720, а целевые объекты не считаются экстремально мелкими, изначальная глубокая пирамида признаков донорской модели является избыточной. В этом случае сокращение числа этапов пространственного сжатия в Backbone позволяет сохранить более детальную информацию на финальных картах признаков, что критически важно для локализации объектов заданного размера.

Во-вторых, модификации подвергается секция агрегации признаков (Neck). Стандартная архитектура Neck в YOLOv8n, основанная на принципах FPN+PAN, включает сложные блоки C2f и множество операций upsampling, что создает значительную нагрузку на память и вычислительные ядра. В предлагаемой модели структура агрегации признаков оптимизирована: количество каналов в узлах feature fusion сокращено, а вычислительно затратные блоки заменены на облегченные skip-connections с конкатенацией признаков [27–29]. Это обеспечивает эффективную передачу семантической и пространственной информации при существенно меньших вычислительных затратах. Кроме того, на основе анализа размеров целевых объектов один из верхних уровней пирамиды, ответственный за детекцию самых мелких объектов, исключен, что упрощает топологию сети и сокращает количество предсказательных голов.

В-третьих, Head-часть (голова), выполняющая окончательное прогнозирование ограничивающих рамок, классов объектов и коэффициентов уверенности, подвергается двунаправленной структурной оптимизации:

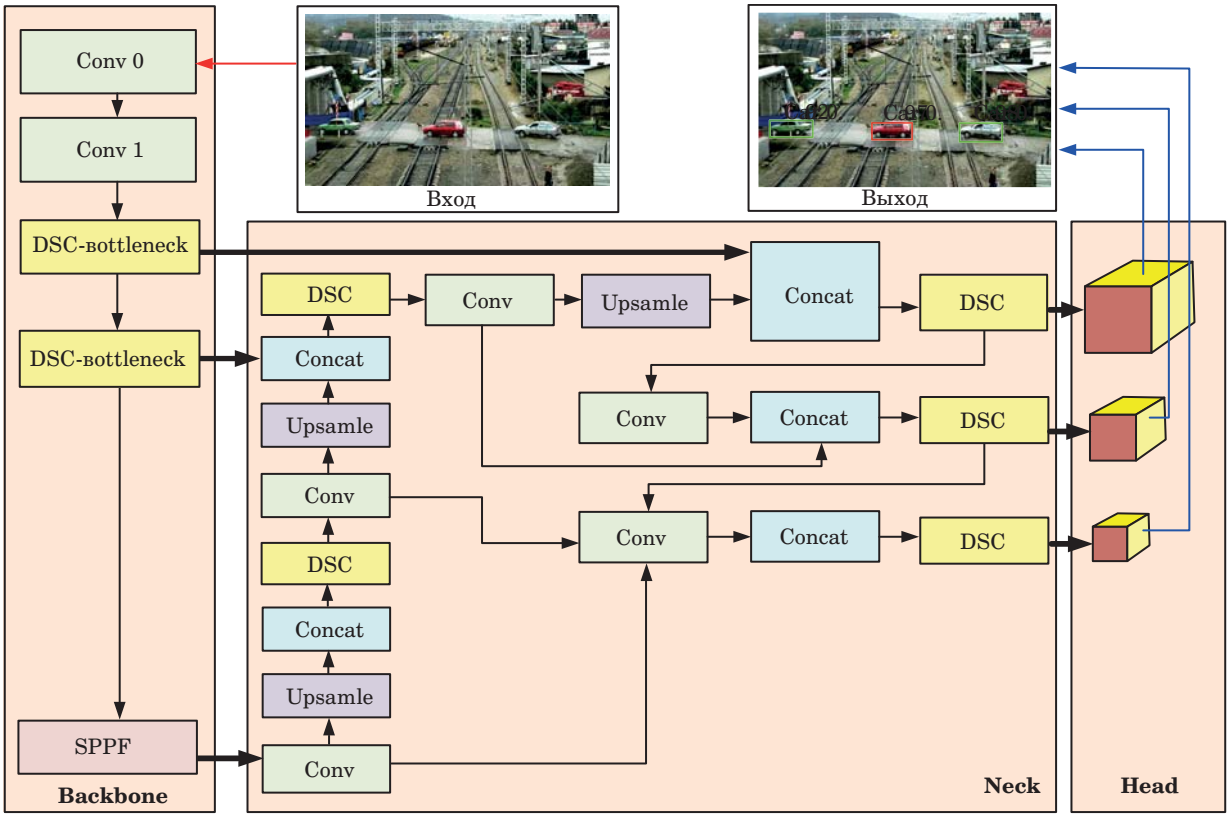
1) удаляется самый нижний уровень детекции (самая мелкая голова). Поскольку постановка задачи исключает необходимость детекции объектов размером менее 80 × 80 пикселей, сокращение числа голов с четырех до трех является обоснованным, обеспечивая максимальную экономию ресурсов на финальной стадии, где высокая точность для мелких объектов не является приоритетной;

2) финальный слой, проецирующий агрегированные признаки на выходные тензоры, также реализуется с использованием DSC, что гарантирует минимизацию потребления ресурсов Raspberry Pi 5 даже на стадии генерации предсказаний.

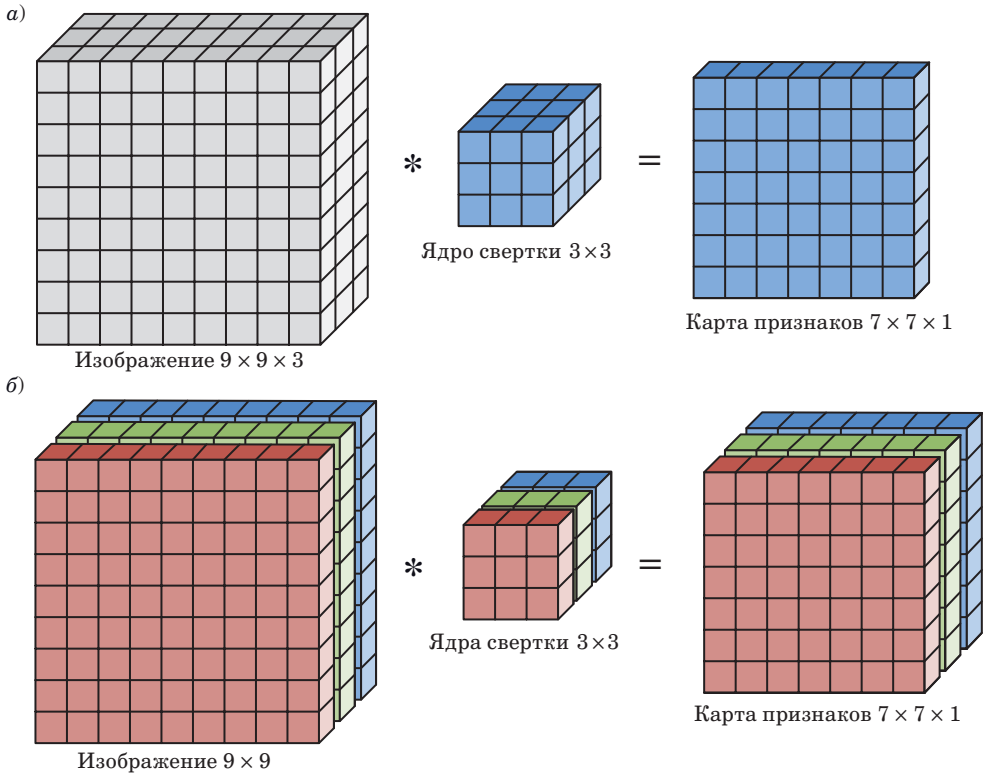
Общая схема предлагаемой архитектуры, наглядно демонстрирующая эволюцию от модели-донора к конечному дизайну, представлена на рис. 3. Желтым цветом выделены узлы структуры, которые были добавлены или заменены.

Визуальное сравнение двух архитектур (см. рис. 2 и 3) позволяет оценить сокращение сложности при сохранении ключевых модулей. Детализированная структура предложенного легковесного блока на основе сепарабельных сверток, который является строительным элементом модифицированных Backbone и Neck, показана на рис. 4.

Этот блок сочетает глубинные свертки для пространственного фильтрования и точечные свертки (1 × 1) для комбинирования каналов, что обеспечивает существенный выигрыш в производительности по сравнению со стандартными сверточными слоями.



■ ***Рис. 3.*** Предлагаемая архитектура сверточной нейронной сети
■ ***Fig. 3.*** Proposed CNN architecture



■ ***Рис. 4.*** Применения ядра свертки 3 × 3 с использованием классической сверточной обработки (*а*) и DSC-обработки (*б*)
■ ***Fig. 4.*** Application of a 3 × 3 convolution kernel using classical convolution processing (*а*) and DSC processing (*б*)

Разработка целевой архитектуры (см. рис. 3) сфокусирована на замене стандартных, вычислительно затратных слоев на их ресурсоэффективные аналоги, что обеспечивает достижение требуемой частоты кадров при обработке потока с разрешением 1280 × 780.

Блочная модификация архитектуры

Основной вклад в снижение вычислительной нагрузки заложен в оптимизации Backbone-сети. В донорской архитектуре (см. рис. 2) блоки C3CA_x и их аналоги (например, C2f) являются основным потребителем FLOPs. В предлагаемой архитектуре эти блоки полностью заменены на DSC-bottleneck.

Блок C3CA_x основан на стандартных свертках, которые выполняют пространственное сканирование и слияние каналов в одном шаге. Блок DSC-bottleneck реализует принцип DSC, разделяя эти две операции [11].

Рассмотрим стандартную свертку с входным и выходным количеством каналов C_{in} и C_{out} и ядром размера $K \times K$ на входе с размером карты признаков $H \times W$.

В этом случае сложность стандартной свертки (в блоке C3CA_x)

$$FLOPs_{std} = H \cdot W \cdot C_{in} \cdot C_{out} \cdot K^2,$$

а сложность DSC

$$FLOPs_{DSC} = H \cdot W \cdot (K^2 \cdot C_{in} + C_{in} \cdot C_{out}),$$

где первое слагаемое — глубинная свертка, второе — точечная свертка 1×1.

При условии, что K^2 значительно больше 1 (например, $3^2 = 9$), а C_{out} относительно велико, сокращение FLOPs по сравнению со стандартной сверткой приближается к сокращению вычислительной сложности в K^2 раз, сохраняя при этом эффективность комбинирования каналов благодаря точечной свертке [26, 30].

Так, например, для типичных параметров, встречающихся в блоках C3CA, когда $C_{in} = 64$, $C_{out} = 128$ и ядро $K = 3$, сложность стандартной свертки составляет порядка 737 280 операций. В то же время DSC требует всего ≈8768 операций с учетом точечной свертки. Таким образом, замена стандартных сверточных слоев на DSC-блоки обеспечивает локальное сокращение вычислительной сложности в ≈84,1 раза.

Для дальнейшей оптимизации вычислительной сложности предложенный блок (DSC-bottleneck) реализует архитектуру сжатия признаков [31]. Данный подход предполагает предварительное сокращение размерности (числа каналов) перед выполнением операции глубинной свертки. Это позволяет эффективно сжать данные и существенно снизить объем вычислительных операций при сохранении репрезентативности признаков.

Аналогичный принцип оптимизации применен и в секции Neck, отвечающей за мультимасштабную агрегацию (PANet-подобная структура), где в базовой модели использованы блоки C3_3 (сверточный блок с тремя сверхточными слоями) после операции слияния (Concat). Эти блоки выполняют как пространственную фильтрацию, так и смешивание каналов.

В модифицированной архитектуре (см. рис. 3) блоки C3_3 заменены на блоки, реализующие DSC (желтые блоки на выходе каждой ветви Neck). Эта замена направлена на снижение вычислительных требований в процессе обработки признаков, извлекаемых с разных уровней детализации. Использование одиночного DSC-слоя вместо полного блока C3_3 (который сам может содержать несколько стандартных сверток) обеспечивает линейное или сублинейное снижение FLOPs на данном этапе.

Ключевым архитектурным решением стало сокращение числа выходных голов в секции Head с четырех до трех (см. рис. 3). Удаление самой мелкой выходной головы, отвечающей за детекцию наименьших объектов, архитектурно обосновано требованием задачи исключать объекты размером менее 80 × 80 пикселей. Это делает соответствующую шкалу детекции избыточной и, как следствие, устраняет необходимость в самом нижнем пути агрегации в Neck. В результате удаляется целый каскад сверточных и объединяющих слоев, что обеспечивает значительный выигрыш в общей вычислительной сложности и позволяет сфокусировать вычислительные ресурсы на объектах среднего и крупного масштаба.

Теоретически сокращение числа голов с четырех до трех приводит к прямому снижению количества параметров и FLOPs в выходном блоке на величину, пропорциональную полному вкладу удаленной головы H_{small}:

$$K_{FLOPs} = \frac{FLOPs(H_{small})}{FLOPs(H_{all})} \cdot 100,$$

где K_{FLOPs} — выигрыш; $FLOPs(H_{small})$ — вычислительная сложность в канале малой головы; $FLOPs(H_{all})$ — вычислительная сложность всей модели.

По результатам расчетов в исходной архитектуре (с четырьмя головами) самая мелкая голова вносила вклад (при параметрах $H = 16$, $W = 16$, $C_{in} = 512$ для четырех классов) 0,85 GFLOPs при общем объеме 14,2 GFLOPs. Таким образом, исключение этого компонента обеспечило теоретическое снижение вычислительной сложности на 5,981 %, что в сочетании с заменой блоков на DSC позволило достичь роста итогового ускорения в 1,56 раза.

Дополнительно финальный слой в секции Head, проецирующий агрегированные признаки на выходные тензоры, был заменен на блоки, реализующие DSC, как и в других секциях. Это гарантирует, что даже на стадии постобработки и генерации предсказаний минимизируется потребление ресурсов целевой платформы Raspberry Pi 5.

Экспериментальная часть: проведение и анализ результатов

Для валидации разработанной легковесной архитектуры OptiYOLO были проведены сравнительные испытания на целевом оборудовании — одноплатном компьютере Raspberry Pi 5 с оперативной памятью 8 ГБ, на видеопотоке от MIPI-камеры с разрешением 1280 × 720.

Обучение и валидация проводились на модифицированном наборе данных, включающем расширенные подмножества COCO [32], дополненные сценами, релевантными для задач периметрального мониторинга (транспорт на железнодорожных путях, объекты в зонах ограниченного доступа). В датасет входят четыре целевых класса: «автомобиль», «человек», «животное», «сумка». В программном коде и при визуализации результатов детекции используются англоязычные обозначения (car, person, animal, bag). Общий объем датасета составил примерно 6500 изображений на каждый класс. Для повышения робастности моделей применялись методы аугментации, сфокусированные на изменении масштаба и освещенности. Обучение проводилось с использованием оптимизатора Adam (скорость обучения 0,001), размером батча 32 в течение 150 эпох. После обучения веса были подвергнуты квантованию до формата INT8 для оптимизации развертывания на встраиваемых системах [33–35].

В качестве базовых моделей-доноров использовались официальные, оптимизированные веса YOLOv8n, YOLOv8s, а также более крупная версия YOLOv8m для оценки того, насколько эффективно редукция обходит необходимость использования более мощных моделей.

Для достижения максимальной скорости на Raspberry Pi 5 все модели (базовые и OptiYOLO) были конвертированы в целочисленный формат INT8. Стоит заметить, что процесс квантования оказал минимальное негативное влияние на общую точность OptiYOLO, снизив mAP0.5 всего на 0,2 пункта (с 73,6 до 73,4 %), что значительно меньше, чем наблюдалось у более сложной YOLOv8s (потеря 1,1 пункта). Этот результат подтверждает большую устойчивость к сжатию архитектуры OptiYOLO, разработанной с акцентом на минимизацию числа каналов и параметров, что благоприятствует эффективному целочисленному инференсу на нейронных сопроцессорах (NPU) или оптимизированных ядрах RPi 5.

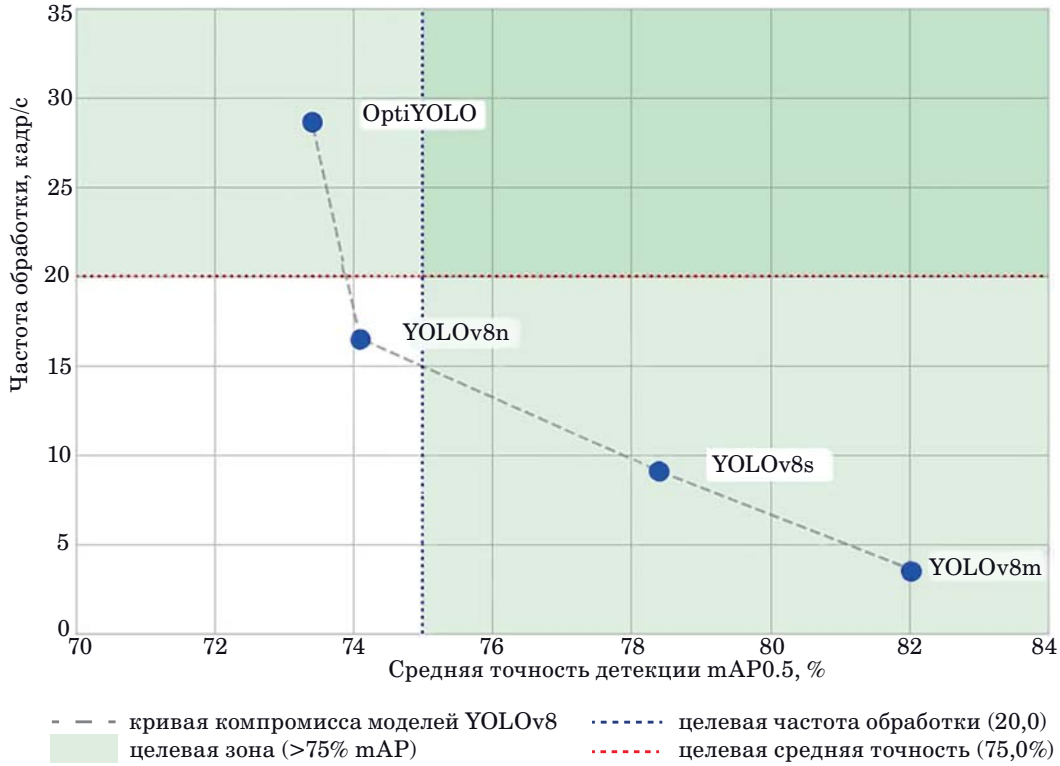
Оценка производительности проводилась по критериям, заданным при постановке задачи: точность детекции (mAP0.5) и частота обработки при инференсе на Raspberry Pi 5. Особое внимание уделялось количеству параметров как ключевому индикатору ресурсоемкости модели. Измерения частоты обработки проводились путем усреднения результатов 500 последовательных кадров.

Результаты тестирования OptiYOLO сопоставлены с тремя базовыми моделями YOLOv8 при инференсе в условиях, максимально приближенных к реальным. Компромисс между скоростью и точностью наглядно иллюстрирует рис. 5.

Разработанная архитектура OptiYOLO демонстрирует уникальное позиционирование: она достигает скорости, значительно превышающей требуемый порог 20 кадров/с, сохраняя при этом точность, сопоставимую с самой легковесной базовой моделью YOLOv8n. Основные количественные показатели производительности моделей, протестированных с входным разрешением 1280 × 720 и после квантования в формат INT8 [36], представлены в таблице.

■ Сравнительные характеристики моделей на Raspberry Pi 5
■ Comparative Model Characteristics on Raspberry Pi 5

Модель (INT8)	Параметры, млн	GFLOPs	Частота обработки, кадр/с	mAP0.5, %			
				общая	car	person	bag
YOLOv8m	25,9	88,6	3,5±0,1	82,1	91,5	87,8	65,4
YOLOv8s	7,22	15,8	9,1±0,3	78,4	88,9	85,1	61,2
YOLOv8n	3,25	8,7	16,5±0,5	74,1	85,2	81,0	55,9
OptiYOLO	1,98	4,1	28,7±0,7	73,4	86,1	82,5	56,1



■ ***Рис. 5.*** Зависимость частоты обработки от mAP0.5
■ ***Fig. 5.*** Comparative plot of processing frequency vs. mAP0.5

■ ***Рис. 6.*** Примеры детекции OptiYOLO на реальных кадрах
■ ***Fig. 6.*** OptiYOLO detection examples on real-world frames

Качественные результаты работы разработанной модели OptiYOLO на тестовых сценах, включая детекцию объектов на разных дистанциях, представлены на рис. 6. Как видно из рисунка, модель OptiYOLO устойчиво и с достаточной уверенностью классифицирует объекты (в данном примере класс «car») в зоне контроля.

Проведенный эксперимент убедительно демонстрирует преимущества архитектурного упрощения, основанного на специфике задачи.

1. Модель OptiYOLO достигла частоты 28,7 кадра/с, что обеспечивает работу в режиме реального времени с существенным запасом производительности на Raspberry Pi 5. При этом она является самой легковесной по числу параметров (1,98 млн против 3,2 млн у YOLOv8n) и вычислительной сложности (4,1 GFLOPs против 8,7 GFLOPs у YOLOv8n).

2. Несмотря на общее снижение mAP0.5 на 0,7 единицы относительно YOLOv8n, точность для классов «автомобиль» и «человек» демонстрирует улучшение (например, «автомобиль» — 86,1). Это подтверждает, что реконфигурация архитектуры, направленная на сохранение признаков среднего и крупного масштаба, привела к перераспределению точности в пользу наиболее критичных для безопасности классов.

3. Особо следует отметить, что при валидации на сценах с удалением объектов до 60 м (что соответствует границе целевого периметра) точность для классов «автомобиль» (86,1 %) и «человек» (82,5 %) остается на высоком уровне. Это доказывает, что прореживание в секции Backbone оказалось эффективным для сохранения пространственной информации, критичной для дальней детекции.

4. Модель YOLOv8m показывает наилучшую общую точность, однако требует значительно больших вычислительных ресурсов. По сравнению с OptiYOLO, она имеет в 13 раз больше параметров (25,9 млн против 1,98 млн) и в 21,6 раза выше вычислительную сложность (88,6 GFLOPs против 4,1 GFLOPs), что делает ее неприменимой для автономного использования на Raspberry Pi 5. Напротив, предлагаемая модель OptiYOLO предоставляет сбалансированное решение, недостижимое для стандартных версий YOLO.

Результаты подтверждают, что разработанная OptiYOLO успешно решает поставленную задачу, обеспечивая требуемую производительность при минимальных аппаратных затратах.

Заключение

Проведенное исследование позволило разработать и валидировать легковесную однопроходную архитектуру сверточной нейронной сети, получившую наименование OptiYOLO, специально адаптированную для задач детекции объектов в реальном времени на ресурсоограниченной платформе Raspberry Pi 5. Обоснованная стратегия оптимизации, базирующаяся на замене стандартных сверточных блоков на DSC и структурном упрощении пирамиды признаков, позволила радикально снизить вычислительную сложность модели.

Ключевым архитектурным решением стало удаление головы детекции, ориентированной на наименьшие объекты (менее 80 × 80 пикселей), что оказалось возможным благодаря специфике постановки задачи по периметральному мониторингу. Этот шаг в сочетании с заменой блоков C3CA_x и C3_3 на DSC-bottleneck привел к созданию самой ресурсоэффективной модели среди протестированных: OptiYOLO требует всего 1,98 млн параметров и 4,1 GFLOPs, что означает снижение числа параметров на 39 % и операций FLOPs на 53 % по сравнению с базовой YOLOv8n.

Экспериментальные результаты, полученные при инференсе на аппаратной платформе с использованием видеопотока 1280 × 720 после квантования до INT8, свидетельствуют о достижении поставленных целей. Модель OptiYOLO демонстрирует частоту обработки 28,7 кадров/с, что существенно превышает минимально требуемый порог 20 кадров/с, обеспечивая функционирование системы в режиме, воспринимаемом как реальное время. При этом наблюдаемое снижение общей точности (mAP0.5 на уровне 73,4 %) является приемлемым компромиссом. Следует отметить сохранение высокой точности для критически важных классов — «автомобиль» (86,1 %) и «человек» (82,5 %), что подтверждает эффективность сохранения пространственной информации на средних и крупных масштабах.

С разработанным подходом целенаправленная архитектурная модификация, сфокусированная на устранении избыточной сложности, связанной с детектированием мелких объектов, позволяет эффективно перенести передовые методы однопроходной детекции на бюджетные встраиваемые системы, открывая перспективы для их широкого практического применения в системах безопасности и автоматизации.

Финансовая поддержка

Исследование выполнено в рамках государственного задания EKTT-2026-0001 за счет средств федерального бюджета.

Литература

1. **Сацюк А. В.** Мониторинг инфраструктуры на основе искусственного интеллекта. *Автоматика, связь, информатика*, 2025, № 9, с. 32–34. doi:10.62994/AT.2025.9.9.005, EDN: OXBPFP
2. **Сацюк А. В., Воевода Е. Г., Каминский Р. Я.** Разработка алгоритма поиска заданного объекта в видеопотоке реального времени на основе метода корреляционного поиска. *Сборник научных трудов Донецкого института железнодорожного транспорта*, 2025, № 1(76), с. 15–23. EDN: VKNJZZ
3. **Сацюк А. В., Воевода Е. Г.** Система автоматического контроля безопасности на железнодорожных переездах. *Сборник научных трудов Донецкого института железнодорожного транспорта*, 2024, № 2(73), с. 39–45. EDN: LDLPWU
4. **Васильев М. Е., Шалимов А. С., Савина О. А.** Обзор версий YOLO: одноэтапная модель сверточной нейронной сети. *Universum: технические науки*, 2025, № 6-1(135), с. 36–46. EDN: HLDKKY
5. **Redmon J., Farhadi A**. YOLOv3: An incremental improvement. *arXiv preprint*. arXiv:1804.02767. 2018.
6. **Zeng Y., Guo D. J., He W. K., Zhang T., Liu Z. T.** ARF-YOLOv8: A novel real-time object detection model for UAV-captured images detection. *J Real-Time Image Proc*, 2024, vol. 21, Article 107. https://doi.org/10.1007/s11554-024-01483-z
7. **Bochkovskiy A., Wang C. Y., Liao H. Y. M.** YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint*. arXiv:2004.10934. 2020.
8. **Клековкин В. А., Марков Н. Г., Небаба С. Г.** Модели сверточных нейронных сетей YOLO с механизмом внимания для систем компьютерного зрения реального времени. *Вестник Томского государственного университета. Управление, вычислительная техника и информатика*, 2025, № 72, с. 39–50. doi:10.17223/19988605/72/4, EDN: KEWUSH
9. **Егунов В. А., Королева И. Ю., Типаев Д. В.** Алгоритмы движения мобильного робота с построением карты местности в реальном времени. *Инженерный вестник Дона*, 2022, № 4(88), с. 126–132. EDN: YPFPSM. http://www.ivdon.ru (дата обращения: 10.01.2026).
10. **Павленко Д. А., Ковалев В. А., Снежко Э. В., Левчук В. А., Печковский В. И.** Распознавание подстилающей поверхности Земли с помощью сверточной нейронной сети на одноплатном микрокомпьютере. *Информатика*, 2020, т. 17, № 3, с. 36–43. doi:10.37661/1816-0301-2020-17-3-36-43, EDN: DIUEIM
11. **Сацюк А. В., Володарец Н. В.** Модификация модели YOLO для гибридной системы детекции и трекинга в БПЛА с автоматическим наведением. *Информационно-управляющие системы*, 2025, № 4, с. 36–44. doi:10.31799/1684-8853-2025-4-36-44, EDN: YKQVJU
12. **Лимонова Е. Е., Шешкус А. В., Николаев Д. П., Иванова А. А., Ильин Д. А., Арлазаров В. Л.** Оптимизация быстродействия первых слоев глубоких сверточных нейронных сетей. *Вестник Российского фонда фундаментальных исследований*, 2016, № 4, с. 84–96. doi:10.22204/2410-4639-2016-092-04-84-96, EDN: YOSCDX
13. **Qin D., Leichner C., Delakis M., Fornoni M., Luo S., Yang F., Wang W., Banbury C., Ye C., Akin B., Aggarwal V., Zhu T., Moro D., Howard A.** MobileNetV4: Universal models for the mobile ecosystem. *European Conference on Computer Vision*, Cham, Springer Nature Switzerland, 2024, pp. 78–96. doi:10.1007/978-3-031-73661-2_5
14. **Zhang X., Zhou X., Lin M., Sun J.** Shufflenet: An extremely efficient convolutional neural network for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6848–6856. doi:10.1109/CVPR.2018.00716
15. **Tan M., Le Q.** Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning (PMLR)*, 2019, pp. 6105–6114. doi:10.48550/arXiv.1905.11946
16. **Han S., Pool J., Tran J., Dally W. J.** Learning both weights and connections for efficient neural network. arXiv. 2015. https://doi.org/10.48550/ARXIV.1506.02626
17. **Chen C., Wang T., Liu C., Liu Y., Cheng L.** Lightweight convolutional transformers enhanced meta-learning for compound fault diagnosis of industrial robot. *IEEE Transactions on Instrumentation and Measurement*, 2023, vol. 72, pp. 1–12. doi:10.1109/TIM.2023.3277956
18. **Фролов С. В., Коробов А. А., Савинова К. С., Потлов А. Ю.** Нейросетевая система управления микроклиматом и освещенностью в неонатальном инкубаторе. *Датчики и системы*, 2025, № 1(279), с. 62–68. doi:10.24412/1992-7185-2025-1-62-68, EDN: IIZLFV
19. **Аксенов Д. С., Жиляев В. А., Маркин Н. И., Титов И. А.** Система распознавания объектов на базе Raspberry Pi 4 и Intel Neural Computer Stick 2. *Информационные системы и технологии*, 2023, № 4(138), с. 10–16. EDN: FHOYQH
20. **Сацюк А. В., Белый Р. В., Ищенко А. Е.** Оценка эффективности алгоритмов YOLO для обнаружения объектов в реальном времени во встраиваемых системах беспилотных транспортных средств. *Сборник научных трудов Донецкого института железнодорожного транспорта*, 2024, № 4(75), с. 73–82. EDN:OCNLZZ
21. **Burggraaff O., Schmidt N., Zamorano J., Pauly K., Pascual S., Tapia C., Spyrakos E., Snik F.** Standardized spectral and radiometric calibration of consumer cameras. *Optics Express*, 2019, vol. 27, no. 14, pp. 19075–19101. doi:10.1364/OE.27.019075
22. **Lin T. Y., Dollar Pi., Girshick R., He K., Hariharan B., Belongie S.** Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125. doi:10.48550/arXiv.1612.03144
23. **Kisantal M., Wojna Z., Murawski J., Naruniecj J., Cho K.** Augmentation for small object detection. *arXiv preprint*. arXiv:1902.07296. 2019.
24. **Banakh T., Głąb S., Jabłońska E., Swaczyna J.** Haar I sets: Looking at small sets in Polish groups through compact glasses. *arXiv preprint*. arXiv:1803.06712. 2018.
25. **Wang A., Chen H., Liu L., Chen K., Lin Z., Han J., Ding G.** YOLOv10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, 2024, vol. 37, pp. 107984–108011. doi:10.48550/arXiv.2405.14458
26. **Howard A. G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H.** Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint*. arXiv:1704.04861. 2017.
27. **Liu S., Qi L., Qin H., Shi J., Jia J.** Path aggregation network for instance segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759-8768. doi:10.1109/CVPR.2018.00913

28. **Zhang Z., Li X., Yang Y.** Scaling YOLOv8 for high-precision real-time object detection. *arXiv preprint*. arXiv:2408.15857. 2024.
29. **He K., Zhang X., Ren S., Sun J.** Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. doi:10.1109/CVPR.2016.49
30. **Chollet F.** Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258. doi:10.1109/CVPR.2017.195
31. **Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.-C.** Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474
32. **Desai M., Mewada H., Pires I. M., Roy S.** Evaluating the performance of the YOLO object detection framework on COCO dataset and real-world scenarios. *Procedia Computer Science*, 2024, vol. 251, pp. 157–163. doi:10.1016/j.procs.2024.11.096
33. **Сацюк А. В.** Оптимизация архитектуры YOLOv8 для задач захвата объекта БПЛА: анализ компромисса между точностью, скоростью и вычислительными ресурсами. *Вестник Ростовского государственного университета путей сообщения*, 2025, № 2(98), с. 35–42. doi:10.46973/0201-727X_2025_2_35, EDN: LCAOOM
34. **Jacob B., Kligys S., Chen B., Zhu M., Tang M., Howard A., Adam H., Kalenichenko D.** Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713. doi:10.1109/CVPR.2018.00286
35. **Nagel M., Nagel M., Amjad R., Fournarakis M., Bondarenko Y., Baalen M., Blankevoort T.** A white paper on neural network quantization. *arXiv preprint*. arXiv:2106.08295. 2021.
36. **Cherepanov N. I., Stepina N. O., Nikiforov I. V.** Improving image analysis and processing performance on the RISC-V platform with Lichee Pi 4A. *Proceedings of the Institute for System Programming of the RAS*, 2025, vol. 37, no. 5, pp. 157–172. doi:10.15514/ISPRAS-2025-37(5)-12, EDN: UDVIEH

UDC 004.853.3
doi:10.31799/1684-8853-2026-3-23-34
EDN: ZZVLVG

Designing an adaptive convolutional neural network architecture using depthwise separable convolutions for real-time operation on embedded devices

A. V. Satsiuk[a], PhD, Tech., Acting Head of Labs, orcid.org/0009-0006-7228-8279, alexandrsatsuk@gmail.com
S. A. Radkovsky[a], PhD, Tech., Associate Professor, orcid.org/0009-0007-9411-949X
A. I. Shekhovtsov[a], PhD, Tech., Associate Professor, orcid.org/0009-0003-6963-815X
S. D. Sonina[a], Senior Lecturer, orcid.org/0009-0009-6114-4386
A. A. Vorobyov[a], Senior Lecturer, orcid.org/0009-0000-0938-6139
[a]Donetsk Institute of Railway Transport, 6, Gornaya St., 283018, Donetsk, Russian Federation

Introduction: Deploying modern computer vision systems on embedded devices (such as Raspberry Pi 5) faces the challenge of achieving real-time performance (target threshold of 20 FPS) when analyzing the stream from a MIPI camera, due to severe resource constraints. Baseline single-stage detectors, like YOLOv8n, possess excessive computational complexity for these conditions. **Purpose:** To develop an adapted, lightweight convolutional neural network architecture by modifying YOLOv8n, focusing computational resources primarily on medium and large-scale objects. **Results:** We place the main focus on the radical reduction of FLOPs and parameters through the replacement of standard convolutions with depthwise separable convolutions and the elimination of redundant detection levels (reduction of output heads). The study has been conducted on the Raspberry Pi 5 platform with an input resolution of 1280×720. This architectural reconfiguration has led to the OptiYOLO model achieving minimal metrics: 1.98 million parameters and 4.1 GFLOPs. After INT8 quantization, a processing speed of 28.7 FPS has been achieved, significantly exceeding the real-time requirement. The overall accuracy has reached 73.4% (mAP0.5), which is considered an acceptable trade-off. The precision for critical classes ("car" – 86.1%, "person" – 82.5%) confirms the successful redistribution of resources towards larger objects. **Practical relevance:** The developed OptiYOLO provides a balanced solution for autonomous, energy-efficient video surveillance systems where processing speed, rather than the detection of small objects, is the priority.

Keywords — YOLO, depthwise separable convolutions, Raspberry Pi 5, neural network optimization, real-time, convolutional neural network, computer vision.

For citation: Satsiuk A. V., Radkovsky S. A., Shekhovtsov A. I., Sonina S. D., Vorobyov A. A. Designing an adaptive convolutional neural network architecture using depthwise separable convolutions for real-time operation on embedded devices. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2026, no. 3, pp. 23–34 (In Russian). doi:10.31799/1684-8853-2026-3-23-34, EDN: ZZVLVG

Financial support
The study was carried out within the framework of state assignment EKTT-2026-0001 at the expense of the federal budget.

References

1. Satsiuk A. V. Automated monitoring system for railway infrastructure based on artificial intelligence. *Automation, Communications, Informatics*, 2025, no. 9, pp. 32–34 (In Russian). doi:10.62994/AT.2025.9.9.005, EDN: OXBPFP
2. Satsuk A. V., Voevoda E. G., Kaminskiy R. Ya. Development of an algorithm for searching a specified object in a real-time video stream based on the correlation search method. *Sbornik nauchnykh trudov Donetckogo instituta zheleznodorozhnogo transporta*, 2025, no. 1(76), pp. 15–23 (In Russian). EDN: VKNJZZ
3. Satsuk A. V., Voevoda E. G. Automated safety control system at railway crossings. *Sbornik nauchnykh trudov Donetckogo instituta zheleznodorozhnogo transporta*, 2024, no. 2(73), pp. 39–45 (In Russian). EDN: LDLPWU
4. Vasilyev M. E., Shalimov A. S., Savina O. A. Overview of YOLO versions: Single-stage convolutional neural network model. *Universum: tekhnicheskie nauki*, 2025, no. 6-1(135), pp. 36–46 (In Russian). EDN: HLDKKY
5. Redmon J., Farhadi A. YOLOv3: An incremental improvement. *arXiv preprint*. arXiv:1804.02767. 2018.
6. Zeng Y., Guo D. J., He W. K., Zhang T., Liu Z. T. ARF-YOLOv8: A novel real-time object detection model for UAV-captured images detection. *J Real-Time Image Proc*, 2024, vol. 21, Article 107. https://doi.org/10.1007/s11554-024-01483-z
7. Bochkovskiy A., Wang C. Y., Liao H. Y. M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint*. arXiv:2004.10934. 2020.
8. Klekovkin V. A., Markov N. G., Nebaba S. G. YOLO convolutional neural network models with attention mechanism for real-time computer vision systems. *Tomsk State University Journal of Control and Computer Science*, 2025, no. 72, pp. 39–50 (In Russian). doi:10.17223/19988605/72/4, EDN: KEWUSH
9. Egunov V. A., Koroleva I. Y., Tipaev D. V. Algorithms for the movement of a mobile robot with the construction of a real-time terrain map. *Engineering Journal of Don*, 2022, no. 4(88), pp. 126–132. EDN: YPFPSM. Available at: http://www.ivdon.ru (accessed 10 January 2026). (In Russian).
10. Paulenko D. A., Kovalev V. A., Snezhko E. V., Liauchuk V. A., Pechkovsky E. I. Recognition of underlying surface using a convolutional neural network on a single-board computer. *Informatics*, 2020, vol. 17, no. 3, pp. 36–43 (In Russian). doi:10.37661/1816-0301-2020-17-3-36-43, EDN: DIUEIM
11. Satsiuk A. V., Volodarets N. V. Modification of the YOLO model for a hybrid detection and trackingsystem in UAVs with an automatic guidance system. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2025, no. 4, pp. 36–44 (In Russian). doi:10.31799/1684-8853-2025-4-36-44, EDN: YKQVJU
12. Limonova E. E., Sheshkus A. V., Nikolaev D. P., Ivanova A. A., Ilin D. A., Arlazarov V. L. Performance optimization of the initial layers of deep convolutional neural networks. *Russian Foundation for Basic Research Journal*, 2016, no. 4, pp. 84–96 (In Russian). doi:10.22204/2410-4639-2016-092-04-84-96, EDN: YOSCDX
13. Qin D., Leichner C., Delakis M., Fornoni M., Luo S., Yang F., Wang W., Banbury C., Ye C., Akin B., Aggarwal V., Zhu T., Moro D., Howard A. MobileNetV4: Universal models for the mobile ecosystem. *European Conference on Computer Vision*, Cham, Springer Nature Switzerland, 2024, pp. 78–96. doi:10.1007/978-3-031-73661-2_5
14. Zhang X., Zhou X., Lin M., Sun J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6848–6856. doi:10.1109/CVPR.2018.00716
15. Tan M., Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning (PMLR)*, 2019, pp. 6105–6114. doi:10.48550/arXiv.1905.11946
16. Han S., Pool J., Tran J., Dally W. J. Learning both weights and connections for efficient neural network. arXiv. 2015. https://doi.org/10.48550/ARXIV.1506.02626
17. Chen C., Wang T., Liu C., Liu Y., Cheng L. Lightweight convolutional transformers enhanced meta-learning for compound fault diagnosis of industrial robot. *IEEE Transactions on Instrumentation and Measurement*, 2023, vol. 72, pp. 1–12. doi:10.1109/TIM.2023.3277956
18. Frolov S. V., Korobov A. A., Savinova K. S., Potlov A. Yu. Neural network based system for control of microclimate and lighting in a neonatal incubator. *Sensors & Systems*, 2025, no. 1(279), pp. 62–68 (In Russian). doi:10.24412/1992-7185-2025-1-62-68, EDN: IIZLFV
19. Aksyonov D. S., Zhilyaev V. A., Markin N. I., Titov I. A. Object recognition system based on Raspberry Pi 4 and Intel Neural Computer Stick 2. *Information Systems and Technologies*, 2023, no. 4(138), pp. 10–16 (In Russian). EDN: FHOYQH
20. Satsuk A. V., Belyy R. V., Ishchenko A. E. Performance evaluation of YOLO algorithms for real-time object detection in embedded systems for autonomous vehicles. *Sbornik nauchnykh trudov Donetckogo instituta zheleznodorozhnogo transporta*, 2024, no. 4(75), pp. 73–82 (In Russian). EDN: OCNLZZ
21. Burggraaff O., Schmidt N., Zamorano J., Pauly K., Pascual S., Tapia C., Spyrakos E., Snik F. Standardized spectral and radiometric calibration of consumer cameras. *Optics Express*, 2019, vol. 27, no. 14, pp. 19075–19101. doi:10.1364/OE.27.019075
22. Lin T. Y., Dollar Pi., Girshick R., He K., Hariharan B., Belongie S. Feature pyramid networks for object detection *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125. doi:10.48550/arXiv.1612.03144
23. Kisantal M., Wojna Z., Murawski J., Naruniecj J., Cho K. Augmentation for small object detection. *arXiv preprint*. arXiv:1902.07296. 2019.
24. Banakh T., Głąb S., Jabłońska E., Swaczyna J. Haar I sets: Looking at small sets in Polish groups through compact glasses. *arXiv preprint*. arXiv:1803.06712. 2018.
25. Wang A., Chen H., Liu L., Chen K., Lin Z., Han J., Ding G. YOLOv10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, 2024, vol. 37, pp. 107984–108011. doi:10.48550/arXiv.2405.14458
26. Howard A. G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint*. arXiv:1704.04861. 2017.
27. Liu S., Qi L., Qin H., Shi J., Jia J. Path aggregation network for instance segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759-8768. doi:10.1109/CVPR.2018.00913
28. Zhang Z., Li X., Yang Y. Scaling YOLOv8 for high-precision real-time object detection. *arXiv preprint*. arXiv:2408.15857. 2024.
29. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. doi:10.1109/CVPR.2016.49
30. Chollet F. Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258. doi:10.1109/CVPR.2017.195
31. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520. doi:10.1109/CVPR.2018.00474
32. Desai M., Mewada H., Pires I. M., Roy S. Evaluating the performance of the YOLO object detection framework on COCO dataset and real-world scenarios. *Procedia Computer Science,* 2024, vol. 251, pp. 157–163. doi:10.1016/j.procs.2024.11.096
33. Satsiuk A. V. Optimization of the YOLOv8 architecture for UAV object capture tasks: Analysis of the trade-off between accuracy, speed and computational resources. *Vestnik Rostovskogo gosudarstvennogo universiteta putey soobshcheniya*, 2025, no. 2(98), pp. 35–42 (In Russian). doi:10.46973/0201-727X_2025_2_35, EDN: LCAOOM
34. Jacob B., Kligys S., Chen B., Zhu M., Tang M., Howard A., Adam H., Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713. doi:10.1109/CVPR.2018.00286
35. Nagel M., Nagel M., Amjad R., Fournarakis M., Bondarenko Y., Baalen M., Blankevoort T. A white paper on neural network quantization. *arXiv preprint*. arXiv:2106.08295. 2021.
36. Cherepanov N. I., Stepina N. O., Nikiforov I. V. Improving image analysis and processing performance on the RISC-V platform with Lichee Pi 4A. *Proceedings of the Institute for System Programming of the RAS*, 2025, vol. 37, no. 5, p. 157–172. doi:10.15514/ISPRAS-2025-37(5)-12, EDN: UDVIEH