Научная статья
УДК 004.81
https://doi.org/10.24143/2072-9502-2026-2-85-93
EDN PVMRWL

Оптимизация модели YOLO для работы на NPU

Максим Александрович Кукушкин*✉*, Петр Александрович Ухов

Московский авиационный институт (национальный исследовательский университет),
Москва, Россия, cnegbyj99@mail.ru✉

Аннотация. Решается задача оптимизации развертывания современных моделей компьютерного зрения на маломощных встраиваемых системах с использованием специализированных нейропроцессоров (NPU). В качестве целевой платформы выбран одноплатный компьютер Orange Pi 5 на базе системы на кристалле Rockchip RK3588S, оснащенной встроенным NPU производительностью до 6 TOPS. Исследование охватывает полный цикл подготовки архитектуры YOLOv11 к эффективному выполнению на встраиваемом оборудовании: от анализа совместимости операторов и структурной адаптации модели под ограничения целевой платформы до реализации высокопроизводительного конвейера обработки видеопотока в реальном времени. Разработана и детально описана методика преобразования модели из формата PyTorch в собственный формат RKNN с применением посттренировочной квантизации в 8-битное целочисленное представление (INT8), что обеспечивает значительное ускорение инференса и снижение потребления памяти без существенной потери точности. Для устранения блокирующего характера инференса на NPU предложена многопроцессорная архитектура обработки видео на основе параллельных рабочих процессов, с балансировкой загрузки между CPU и NPU. Экспериментально определены оптимальные конфигурации количества процессов для различных вариантов модели YOLOv11 (*n*, *s*, *m*). Достигнута производительность 54 FPS для YOLOv11-*n*, 48 FPS для YOLOv11-*s* и 27 FPS для YOLOv11-*m* при разрешении входного кадра 640 × 640. Показано, что превышение оптимального числа процессов приводит к насыщению пропускной способности шины памяти, росту температуры SoC и снижению энергоэффективности без прироста производительности. Полученные результаты подтверждают возможность создания доступных энергоэффективных и высокопроизводительных систем компьютерного зрения на базе массовых одноплатных компьютеров для широкого спектра приложений реального времени – от дронов и робототехники до умных камер видеонаблюдения.

Ключевые слова: компьютерное зрение, нейропроцессор, YOLOv11, Orange Pi 5, Rockchip RK3588S, квантизация, RKNN, многопроцессорная обработка, производительность, embedded-системы

Для цитирования: *Кукушкин М. А., Ухов П. А.* Оптимизация модели YOLO для работы на NPU // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2026. № 2. С. 85–93. https://doi.org/10.24143/2072-9502-2026-2-85-93. EDN PVMRWL.

Original article

Optimizing the YOLO model for NPU operation

Maksim A. Kukushkin*✉*, Peter A. Ukhov

Moscow Aviation Institute (National Research University),
Moscow, Russia, cnegbyj99@mail.ru✉

Abstract. The problem of optimizing the deployment of modern computer vision models on small-space embedded systems using specialized neuroprocessors (NPU) is being solved. The target platform is the Orange Pi 5 single-board computer based on the Rockchip RK3588S system-on-chip, which integrates a 6-TOPS NPU. The study encompasses the complete pipeline for adapting the YOLOv11 architecture to embedded execution from operator compatibility analysis and structural model modifications to meet hardware constraints, to the implementation of a real-time, high-throughput video processing pipeline. We present a detailed methodology for converting models from PyTorch to the vendor-specific RKNN format using post-training quantization to INT8 precision, which delivers substantial inference acceleration and memory footprint reduction with minimal accuracy loss. To overcome the inherently blocking nature of NPU inference, we propose a multiprocessing video processing architecture that employs parallel worker processes.

Through extensive experimentation, we identify the optimal number of concurrent processes for different YOLOv11 variants (*n*, *s*, *m*). Our implementation achieves 54 FPS for YOLOv11-*n*, 48 FPS for YOLOv11-*s*, and 27 FPS for YOLOv11-*m* at 640 × 640 input resolution. Crucially, we demonstrate that exceeding the optimal process count saturates memory bandwidth, increases SoC temperature, and reduces energy efficiency without improving throughput. These findings validate the feasibility of building cost-effective, energy-efficient, and high-performance computer vision systems using widely available single-board computers. The results are directly applicable to real-time use cases such as autonomous drones, robotics, smart surveillance, and edge AI applications where low latency and hardware accessibility are critical factors.

Keywords: computer vision, neural processing unit, YOLOv11, Orange Pi 5, Rockchip RK3588S, quantization, RKNN, multiprocess processing, performance, embedded systems

For citation: Kukushkin M. A., Ukhov P. A. Optimizing the YOLO model for NPU operation. *Vestnik of Astrakhan State Technical University. Series: Management, computer science and informatics*. 2026;2:85-93. (In Russ.). https://doi.org/10.24143/2072-9502-2026-2-85-93. EDN PVMRWL.

Введение

В последние годы задачи компьютерного зрения стали неотъемлемой частью современных приложений, где модели семейства YOLO (You Only Look Once) демонстрируют высокую эффективность для детекции объектов в реальном времени. Однако развертывание этих моделей на маломощных embedded-системах сталкивается с проблемой несоответствия между высокими вычислительными требованиями моделей и ограниченными ресурсами таких устройств.

Темой данного исследования является разработка методики развертывания нейросетевых моделей компьютерного зрения на встроенных системах с использованием нейропроцессоров NPU (Neural Processing Unit). В качестве целевой платформы выбран одноплатный компьютер Orange Pi 5 [1] с Rockchip RK3588S, а в качестве базовой модели – YOLOv11 [2].

Цель работы – достижение максимальной производительности (FPS – Frames Per Second) при детекции объектов в реальном времени с сохранением приемлемой точности (mAP). Для этого решены следующие задачи:

1. Анализ и оптимизация архитектуры YOLOv11 для адаптации к возможностям NPU.

2. Разработка конвейера конвертации модели в формат RKNN (Rockchip Neural Network) с применением квантизации.

3. Оптимизация этапов пред- и постобработки видео для минимизации нагрузки на CPU (Central Processing Unit).

4. Сравнительный анализ производительности различных конфигураций модели.

В отличие от традиционных подходов с использованием CPU-фреймворков в работе применяется специализированный аппаратный ускоритель – NPU, что позволяет достичь максимальной производительности на встроенных системах.

Научная новизна данной работы заключается:

– в комплексном подходе к оптимизации, охватывающем не только саму модель, но и этапы пред/постобработки данных;

– адаптации и всестороннем тестировании новейшей модели YOLOv11 для конкретной NPU-архитектуры Rockchip [3];

– разработке практических рекомендаций по выбору оптимальной конфигурации модели (точность vs. скорость) для развертывания на устройстве Orange Pi 5 [1].

Обзор модели нейросети

YOLOv11 представляет собой одну из наиболее современных архитектур для детекции объектов, оптимизированную для развертывания на ресурсоограниченных устройствах. В отличие от предыдущих версий данная модель предлагает целостный архитектурный подход с улучшенным бэкбоном на основе CSPNet и оптимизированной PAN-пирамидой для эффективного многомасштабного анализа.

Ключевыми преимуществами YOLOv11 являются более высокая точность, по сравнению с YOLOv8, при сопоставимой вычислительной сложности и, в отличие от YOLOv10, предсказуемость конвертации в формат NPU. Архитектура модели основана на стандартных операциях, хорошо поддерживаемых инструментарием RKNN-Toolkit2, что минимизирует риски при развертывании на NPU.

Выбор YOLOv11 в качестве базовой модели обусловлен оптимальным сочетанием современной архитектуры, высокой эффективности и хорошей совместимости с целевой аппаратной платформой Orange Pi 5.

Обоснованность выбора NPU

Развертывание современных моделей компьютерного зрения на центральных (CPU) и графических (GPU, Graphics Processing Unit) процессорах встраиваемых систем часто не позволяет достичь целевых показателей производительности в реальном времени. Это обусловлено архитектурным несоответствием: CPU оптимизированы для последовательных задач, в то время как нейросети требуют массового параллелизма. Кроме того, высокое

энергопотребление GPU и тепловыделение ограничивают их применение в автономных решениях.

Для преодоления этих ограничений были разработаны NPU – специализированные аппаратные ускорители с архитектурой, изначально спроектированной для эффективного выполнения операций нейронных сетей. Ключевые особенности NPU включают:

– массивно-параллельную обработку данных с использованием сотен вычислительных ядер;

– иерархическую организацию памяти для минимизации задержек доступа;

– аппаратную поддержку низкоразрядных вычислений (INT8/INT16);

– специализированные блоки для операций свертки и пулинга.

Для работы с NPU необходим специализированный программный стек, включающий компилятор для конвертации моделей (например, RKNN-Toolkit) и runtime-библиотеки для управления выполнением на аппаратном уровне.

Конвертация моделей для NPU представляет собой критически важный этап, поскольку не все операторы нейронных сетей могут быть эффективно отображены на аппаратные возможности конкретного NPU.

В качестве экспериментальной платформы выбран одноплатный компьютер Orange Pi 5 с системой на кристалле Rockchip RK3588S, содержащей специализированный NPU. Данная платформа обеспечивает оптимальный баланс между вычислительной мощностью, энергоэффективностью и стоимостью. Rockchip RK3588S реализует гетерогенную архитектуру с 8-ядерным процессором [4], оптимизированную для параллельного выполнения разнородных задач. Основные характеристики процессора представлены в табл. 1.

Таблица 1

Table 1

Основные характеристики Rockchip RK3588S

Main features of the Rockchip RK3588S

Компонент	Описание
Центральный процессор (CPU)	8 ядер ARM: • 4x Cortex-A76 @ до 2.4 ГГц • 4x Cortex-A55 @ до 1.8 ГГц
Нейропроцессор (NPU)	Производительность: до 6 TOPS (INT8) Поддержка форматов: INT4/INT8/INT16
Графический процессор (GPU)	ARM Mali-G610 MP4
Процессор обработки видеосигнала (VPU)	Поддержка кодирования и декодирования 8K@60fps (H.265/HEVC, H.264/AVC)

Плата Orange Pi 5 реализует потенциал RK3588S, предоставляя необходимые интерфейсы и ресурсы для создания законченных embedded-решений. Характеристики Orange Pi5 представлены в табл. 2.

Таблица 2

Table 2

Характеристики одноплатного компьютера Orange Pi 5

Characteristics of the Orange Pi 5 single-board computer

Параметр	Характеристика	Описание
Оперативная память	8 Гб LPDDR4	Достаточный объем высокоскоростной памяти
Память	Слот M.2 для NVMe SSD, слот для micro-SD	для загрузки модели, буферизации видеокадров
Интерфейсы камер	2x порта MIPI-CSI	NVMe-накопитель позволяет быстро загружать большие модели и datasets, уменьшая время разработки

Методика конвертации модели

Перевод обученной нейросетевой модели на специализированный аппаратный ускоритель – всегда нетривиальная задача. В нашем случае необходимо было адаптировать современную архитектуру YOLOv11 под особенности NPU платформы Rockchip, сохранив при этом точность детекции и достигнув максимального быстродействия. Разра-

ботанный процесс конвертации состоял из нескольких взаимосвязанных этапов.

Исходная модель YOLOv11 поставляется в формате PyTorch. Поскольку инструменты для работы с NPU требуют на вход модель в промежуточном формате, первым шагом стал экспорт в ONNX – открытый стандарт представления нейронных сетей.

На практике этот этап требовал настроить следующий набор параметров:

1. Задать размер входного тензора, соответствующий ожидаемому разрешению изображения. Это важно для последующей статической оптимизации графа вычислителей NPU.

2. Для повышения гибкости развертывания экспериментировали с указанием динамических размерностей, оставляя возможность изменять размер батча без пересборки модели.

3. Необходимо обеспечить совместимость операторов, используемых в YOLOv11, с поддерживаемым набором операций ONNX и, впоследствии, RKNN-Toolkit2. Некоторые специализированные активации или слои требовали замены на эквивалентные.

Результатом этого этапа становился файл модели в формате .onnx, готовый к загрузке в инструментарий Rockchip.

Следующим шагом была непосредственная трансформация ONNX-модели в собственный формат RKNN с помощью SDK RKNN-Toolkit2 [5]. Этот процесс включает в себя глубокую оптимизацию под конкретную архитектуру NPU RK3588S.

Ключевым аспектом здесь является квантизация – снижение разрядности весов и активаций модели с 32-битных чисел с плавающей запятой (FP32) до 8- или 16-битных целых чисел (INT8/INT16). Именно этот прием дает основной прирост в скорости и снижении энергопотребления на NPU [5].

Применялась посттренировочная статическая квантизация, для этого:

1. Формировался калибровочный датасет – набор из 100–200 изображений из валидационной выборки COCO [6], репрезентирующих типовые сцены.

2. Модель в режиме эмуляции прогонялась на этих данных, а RKNN-Toolkit2 собирал статистику распределения значений активаций для каждого слоя.

3. На основе этой статистики для каждого слоя автоматически подбирались параметры масштабирования (scale) и смещения (zero_point) для пересчета FP32 значений в INT8.

За квантизацией следовала обязательная проверка корректности конвертации. Выполнялось сравнение выходных тензоров оригинальной FP32-модели и полученной INT8-модели на идентичных входных данных. В качестве метрики использовалась косинусная близость. Это давало уверенность, что семантика выходных данных модели (расположение и класс объектов) не была искажена в процессе преобразований.

Финальным действием была компиляция модели в низкоуровневые инструкции NPU и экспорт в файл с расширением .rknn. Эта бинарная модель уже была готова к загрузке и исполнению на устройстве Orange Pi 5 с использованием Runtime-библиотеки Rockchip. Общий процесс конвертации модели представлен на рис. 1.

Рис. 1. Схема экспорта модели YOLO

Fig. 1. YOLO model export scheme

Разработанный конвейер позволил нам получить оптимизированные версии моделей YOLOv11 различных размеров (*n*, *s*, *m*) для проведения дальнейших экспериментов по оценке производительности каждой из моделей.

Методика оптимизации модели

Оптимизация исходной модели YOLOv11 проводилась с целью адаптации ее архитектуры к особенностям выполнения на NPU и снижения вычислительной нагрузки без существенной потери точности детекции. В результате был разработан и реализован метод реструктуризации выходных слоев модели, позволивший значительно повысить эффективность инференса.

Исходная модель YOLOv11 использует единый выходной тензор сложной структуры (выход размерностью [1, 84, 8 400]) [7], который содержит объединенные данные о координатах bounding boxes, оценках уверенности для каждого класса и общей уверенности детекции. Хотя такой подход эффективен для обучения и работы на CPU/GPU, он создает ряд проблем при выполнении на NPU:

– необходимость сложной постобработки на CPU для декомпозиции тензора;

– высокие требования к пропускной способности памяти;

– неоптимальное использование кэша NPU из-за разнородности данных.

Для устранения этих ограничений была предложена и реализована оптимизированная архитектура, в которой единый выходной тензор заменен на группу специализированных выходов. Вместо одного выхода размерностью [1, 84, 8 400] модель теперь производит 3 отдельных тензора для каждого масштабного уровня:

– координаты bounding boxes: [1, 64, 80, 80] – пространственные координаты детектируемых объектов;

– по классам уверенность: [1, 80, 80, 80] – оценки уверенности для 80 категорий COCO [6];

– общая уверенность: [1, 1, 80, 80] – суммарная уверенность детекции по всем классам.

На рис. 2 представлено сравнение выхода модели до преобразований и после, а в табл. 3 – общие характеристики до и после оптимизации модели YOLOv11.

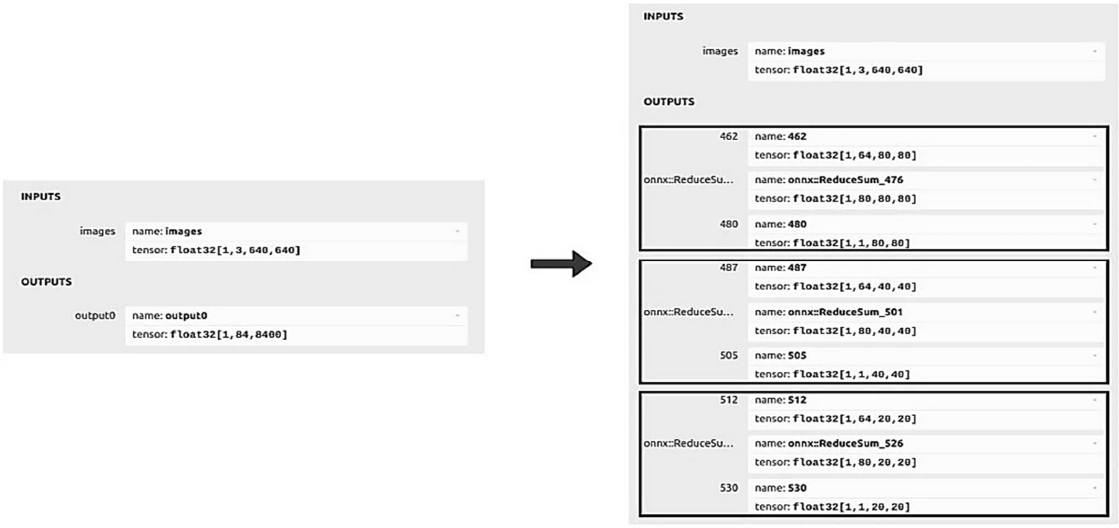

Рис. 2. Сравнение выходных структур оригинальной (слева) и оптимизированной (справа) моделей

Fig. 2. Comparison of the output structures of the original (left) and optimized (right) models

Таблица 3

Table 3

Сравнение выходной структуры оригинальной и оптимизированной моделей

Comparison of the output structure of the original and optimized models

Параметр	Оригинальная модель	Оптимизированная модель
Количество выходов	1 основной тензор	9 тензоров (3 группы по 3 масштаба)
Структура выхода	Объединенный: [1, 84, 8 400]	Разделенная: координаты, уверенность по классам, общая уверенность
Постобработка	Сложная декомпозиция на CPU	Параллельная обработка на NPU

Оптимизация была достигнута за счет реструктуризации выходных слоев модели:

1. Выделение специализированных ветвей: вместо объединения всех данных в единый тензор в архитектуре были сохранены выходы двух ключевых сверточных слоев и добавлена специализированная ветвь для вычисления суммарной уверенности.

2. Эффективный расчет общей уверенности: для вычисления общего показателя уверенности детекции ([1, 1, 80, 80]) применяется последовательность операций ReduceSum + Clip, что обеспечивает корректное агрегирование данных по всем 80 категориям с сохранением численной стабильности.

3. Масштабируемость под разные разрешения: предложенный подход применен ко всем трем масштабным уровням детекции (80 × 80, 40 × 40, 20 × 20), что обеспечивает согласованность обработки на всех уровнях feature pyramid.

Схематичное представление данной доработки приведено на рис. 3, общий упрощенный граф оптимизированной модели показан на рис. 4.

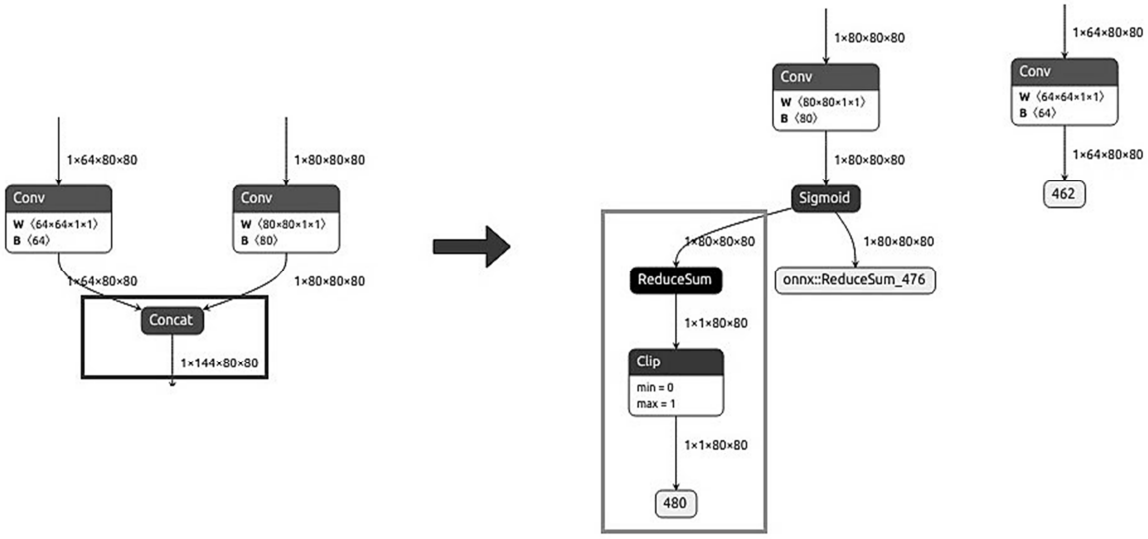

Рис. 3. Детализированная схема оптимизированного блока обработки

Fig. 3. Detailed diagram of the optimized processing unit

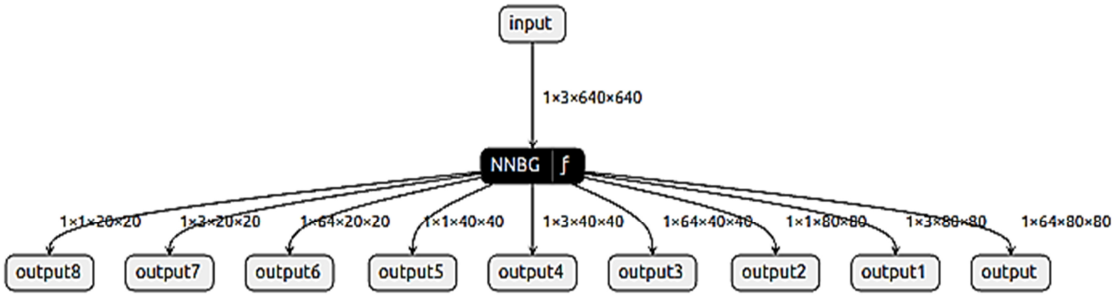

Рис. 4. Граф оптимизированной модели

Fig. 4. The optimized model graph

Разработанный подход обеспечил следующие преимущества:

– снижение нагрузки на CPU: упрощение постобработки за счет предварительного разделения данных на NPU;

– улучшение использования кэша NPU: специализированные тензоры лучше соответствуют архитектуре памяти NPU;

– повышение параллелизма: возможность параллельной обработки входных данных;

– сохранение точности: все операции оптимизации являются математически эквивалентными оригинальной модели.

Экспериментальная проверка показала, что предложенная оптимизация не оказывает негативного влияния на метрики точности (mAP), при этом обеспечивая значительное ускорение времени инференса на целевом оборудовании Orange Pi 5 с NPU Rockchip RK3588S.

Данная оптимизация архитектуры YOLOv11 представляет собой целенаправленную адаптацию модели под специфические требования embedded-систем с NPU и демонстрирует важность учета аппаратных особенностей при развертывании нейросетевых моделей на ресурсо-ограниченных устройствах [8].

Средства и методика тестирования

Эксперименты проводились на одноплатном компьютере Orange Pi 5 под управлением Ubuntu 22.04 LTS. Для работы с NPU использовались драйвер rockchip-npu-driver и библиотеки librknnrt, librga. Конвертация моделей YOLO выполнялась с помощью RKNN-Toolkit2 v1.5.2.

Основной проблемой являлась блокирующая природа операции инференса на NPU. Для ее решения разработана многопроцессорная архитектура обработки видео (рис. 5), состоящая:

– из главного процесса, осуществляющего чтение видео и предобработку кадров;

– буфера-очереди фиксированного размера;

– набора рабочих процессов с копиями модели RKNN.

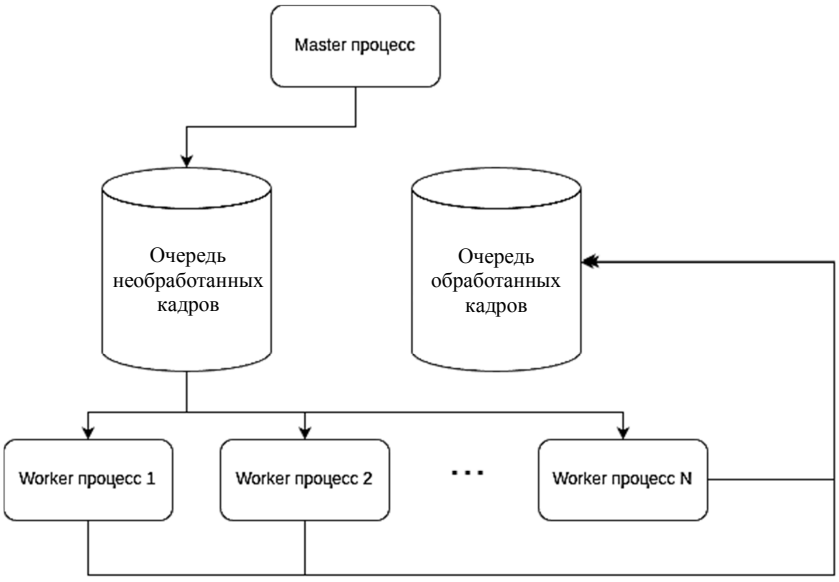

Рис. 5. Способ обработки кадров

Fig. 5. Frame processing method

Ключевое преимущество подхода – возможность параллельной обработки кадров: пока один рабочий процесс выполняет инференс, другие продолжают обработку следующих кадров. В ходе исследований количество рабочих процессов варьировалось от 1 до 6 для определения оптимальной конфигурации.

Для оценки точности моделей использовался стандартный датасет COCO [6] val2017, а для измерения производительности в реальном времени – видеопоток Full HD продолжительностью 5 минут. Все замеры производительности проводились на прогретой системе с минимизированной фоновой нагрузкой, чтобы исключить влияние сторонних факторов и теплового троттлинга. Каждый эксперимент повторялся трижды с последующим усреднением результатов, что обеспечивает статистическую значимость полученных данных.

Результаты

Проведенные эксперименты позволили всесторонне оценить влияние трех ключевых параметров – выбора модели (*n*, *s*, *m*), размера входного тензора (640, 896) и количества рабочих процессов – на итоговую производительность системы. Результаты представлены в табл. 4 и проанализированы ниже.

Таблица 4

Table 4

Сводные результаты тестирования производительности различных конфигураций

Summary results of performance testing of various configurations

Модель	Размер входного тензора	Количество процессов	Загрузка 1-го ядра NPU средняя, %	Загрузка 2-го ядра NPU средняя, %	Загрузка 3-го ядра NPU средняя, %	Пик температуры, °C	FPS
n	640	2	44	5	0	54	43
		3	52	18	2	63	52
		4	54	20	3	67	54
		5	60	24	8	69	54
		6	60	24	18	71	53
s	640	2	65	8	0	42	31
		3	72	33	3	51	43
		4	78	48	12	62	48
		5	78	53	20	66	48
		6	78	55	27	69	47
	896	2	70	22	0	48	21
		3	76	38	13	52	24
		4	77	55	29	62	27
		5	78	65	38	67	28
		6	78	65	49	71	27

Окончание табл. 4

Ending of Table 4

Сводные результаты тестирования производительности различных конфигураций

Summary results of performance testing of various configurations

Модель	Размер входного тензора	Количество процессов	Загрузка 1-го ядра NPU средняя, %	Загрузка 2-го ядра NPU средняя, %	Загрузка 3-го ядра NPU средняя, %	Пик температуры, °C	FPS
m	640	2	91	52	0	53	23
		3	92	83	28	61	27
		4	92	83	64	65	27
		5	92	84	71	67	27
		6	92	84	75	70	26

Мультипроцессорный подход показал высокую эффективность. Для всех моделей наблюдается четкая закономерность: увеличение числа рабочих процессов с 2 до 3–4 приводит к существенному росту FPS, т. к. позволяет эффективнее задействовать вычислительные ядра NPU.

Исходя из результатов эксперимента для каждой модели можно определить свой предел использования ресурсов системы:

1. Для модели YOLOv11-*n* максимальный FPS (54) достигается уже при 3 процессах, после чего производительность стабилизируется. Это указывает на то, что данная модель недостаточно сложна, чтобы полностью загрузить NPU большим количеством процессов.

2. Для модели YOLOv11-*s* оптимальным является использование 4–5 процессов, что позволяет достичь 48 FPS. Дальнейшее увеличение числа рабочих процессов не только не дает прироста, но и приводит к незначительному падению FPS из-за накладных расходов на переключение контекста и конкуренцию за ресурсы.

3. Для модели YOLOv11-*m* пиковая производительность (27 FPS) также достигается при 3 процессах. Высокая загрузка всех ядер NPU (до 92 % на первом и 83 % на втором) свидетельствует о том, что эта модель является вычислительно емкой и эффективно использует аппаратные ресурсы.

Данные о загрузке ядер NPU подтверждают эффективность многопроцессорного подхода. Распределение нагрузки определяется архитектурой NPU Rockchip RK3588S и механизмами планирования в RKNN-Toolkit2. При использовании двух процессов наблюдается недогрузка третьего ядра из-за недостаточного параллелизма задач инференса. Увеличение количества рабочих процессов позволяет планировщику RKNN равномернее распределять независимые задачи обработки кадров между ядрами, что проявляется в росте утилизации второго и третьего ядер и увеличении общего FPS.

Экспериментально установлена четкая зависимость теплового режима от вычислительной нагрузки. Для оптимальных конфигураций (3–4 процесса для моделей *n*, *m*, 4–5 для модели *s*) температура стабилизируется в допустимом диапазоне 62–67 °C. Дальнейшее увеличение числа процессов до 6 приводит к росту температуры до 69–71 °C без прироста производительности, что свидетельствует о достижении предела пропускной способности NPU и возрастании конкуренции за ресурсы.

Модель YOLOv11-*n* демонстрирует наивысшую производительность (54 FPS) и рекомендуется для задач, критичных к скорости отклика. YOLOv11-*s* представляет сбалансированное решение с производительностью 48 FPS при разрешении 640px, при этом увеличение размера входного тензора до 896px обеспечивает лучшую детализацию ценой снижения FPS до 27. YOLOv11-*m*, являясь наиболее точной, ограничивается 27 FPS due to высокой ресурсоемкости.

Заключение

В результате исследования разработана методикаоптимизации и развертывания моделей YOLOv11 на платформе Orange Pi 5, включающая структурную оптимизацию выходных слоев, конвертацию в формат RKNN и многопроцессорную обработку видео. Определены оптимальные конфигурации для различных моделей, подтверждающие возможность создания высокопроизводительных систем компьютерного зрения на базе доступных одноплатных компьютеров.

Список источников

1. Orange Pi. Orange Pi 5 User Manual. 2023. URL: http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-5.html (дата обращения: 15.11.2024).

2. Jocher G. YOLOv11: Next-Generation Real-Time Object Detection Model // Ultralytics GitHub Repository. 2024. URL: https://github.com/ultralytics/ultralytics (дата обращения: 15.11.2024).

3. Rockchip Electronics Co., Ltd. RKNN-Toolkit2 User Guide // Rockchip Developer. 2023. URL: https://github.com/rockchip-linux/rknn-toolkit2 (дата обращения: 15.11.2024).

4. Rockchip Electronics Co., Ltd. RK3588S Datasheet Version 1.0. 2022. URL: https://www.rock-chips.com/a/en/products/RK35_Series/ (дата обращения: 15.11.2024).

5. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). University of Washington∗, Allen Institute for AI, Facebook AI Research 2016. P. 779–788. URL: https://arxiv.org/pdf/1506.02640 (дата обращения: 15.11.2024).

6. Tsung-Yi Lin, Maire M., Belongie S., Bourdev L., Girshick R., Hays J., Perona P., Ramanan D., Zitnick C. L., Dollár P. Microsoft COCO: Common Objects in Context // Computer Vision – ECCV 2014. Cham: Springer International Publishing, 2014. P. 740–755. DOI 10.1007/978-3-319-10602-1_48.

7. Jocher G., Chaurasia A., Qiu J. YOLO by Ultralytics // Ultralytics Documentation. 2023. URL: https://docs.ultralytics.com/ (дата обращения: 15.11.2024).

8. Постобработка в YOLO: от Raw-тензоров к Bounding Boxам // Habr. 2022. URL: https://habr.com/ru/articles/676312/ (дата обращения: 15.11.2024).

References

1. *Orange Pi. Orange Pi 5 User Manual. 2023*. Available at: http://www.orangepi.org/html/hardWare/computer AndMicrocontrollers/details/Orange-Pi-5.html (accessed: 15.11.2024).

2. Jocher G. YOLOv11: Next-Generation Real-Time Object Detection Model. *Ultralytics GitHub Repository. 2024*. Available at: https://github.com/ultralytics/ultralytics (accessed: 15.11.2024).

3. Rockchip Electronics Co., Ltd. RKNN-Toolkit2 User Guide. *Rockchip Developer. 2023*. Available at: https://github.com/rock-chip-linux/rknn-toolkit2 (accessed: 15.11.2024).

4. *Rockchip Electronics Co., Ltd. RK3588S Datasheet Version 1.0. 2022*. Available at: https://www.rock-chips.com/a/en/products/RK35_Series/ (accessed: 15.11.2024).

5. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. University of Washington∗, Allen Institute for AI, Facebook AI Research. 2016. Pp. 779-788. Available at: https://arxiv.org/pdf/1506.02640 (accessed: 15.11.2024).

6. Tsung-Yi Lin, Maire M., Belongie S., Bourdev L., Girshick R., Hays J., Perona P., Ramanan D., Zitnick C. L., Dollár P. Microsoft COCO: Common Objects in Context. *Computer Vision – ECCV 2014*. Cham, Springer International Publishing, 2014. Pp. 740-755. DOI 10.1007/978-3-319-10602-1_48.

7. Jocher G., Chaurasia A., Qiu J. YOLO by Ultralytics. *Ultralytics Documentation. 2023*. Available at: https://docs.ultralytics.com/ (accessed: 15.11.2024).

8. Postobrabotka v YOLO: ot Raw-tenzorov k Bounding Boxam [Post-processing in YOLO: from Raw Tensors to Bounding Boxes]. *Habr. 2022*. Available at: https://habr.com/ru/articles/676312/ (accessed: 15.11.2024).

Статья поступила в редакцию 14.11.2025; одобрена после рецензирования 12.01.2026; принята к публикации 31.03.2026
The article was submitted 14.11.2025; approved after reviewing 12.01.2026; accepted for publication 31.03.2026

Информация об авторах / Information about the authors

Максим Александрович Кукушкин – аспирант кафедры вычислительной математики и программирования; Московский авиационный институт (национальный исследовательский университет); cnegbyj99@mail.ru

Петр Александрович Ухов – кандидат технических наук; доцент кафедры вычислительной математики и программирования; Московский авиационный институт (национальный исследовательский университет); ukhovpa@mai.ru

Maksim A. Kukushkin – Postgraduate Student of the Department of Computational Mathematics and Programming; Moscow Aviation Institute (National Research University); cnegbyj99@mail.ru

Peter A. Ukhov – Candidate of Technical Sciences; Assistant Professor of the Department of Computational Mathematics and Programming; Moscow Aviation Institute (National Research University); ukhovpa@mai.ru