Санников Михаил Александрович, бакалавр
Уфимский университет науки и технологий
Sannikov Mikhail Aleksandrovich
Ufa University of Science and Technology

ОПТИМИЗАЦИЯ НЕЙРОСЕТЕВОГО ВЫВОДА НА ВСТРОЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ ПЛАТФОРМАХ МОБИЛЬНЫХ РОБОТОВ
OPTIMIZATION OF NEURAL NETWORK INFERENCE ON EMBEDDED COMPUTING PLATFORMS OF MOBILE ROBOTS

Аннотация. Рассмотрены методы оптимизации нейросетевого вывода на встроенных вычислительных платформах мобильных роботов. Проанализированы выбор компактной архитектуры, оптимизация вычислительного графа, понижение точности вычислений, аппаратное ускорение и организация конвейера обработки видеоданных. Показаны основные компромиссы между задержкой, точностью и ресурсами.

Abstract. Methods for optimizing neural network inference on embedded computing platforms of mobile robots are reviewed. Compact model selection, graph optimization, reduced precision, hardware acceleration, and video processing pipelines are analyzed together with trade-offs between latency, accuracy, and resources.

Ключевые слова: Нейросетевой вывод, edge computing, мобильный робот, TensorRT, квантизация, компьютерное зрение.

Keywords: Neural inference, edge computing, mobile robot, TensorRT, quantization, computer vision.

Введение

Алгоритмы компьютерного зрения на мобильном роботе должны работать в условиях ограниченных вычислительных ресурсов, энергопотребления и теплового пакета. В отличие от серверной обработки, где можно масштабировать число ускорителей и использовать большие пакеты данных, бортовая система должна выдавать результат с предсказуемой задержкой для каждого поступающего кадра. Это особенно важно для детекции препятствий, распознавания объектов и других функций, связанных с движением.

Оптимизация нейросетевого вывода включает несколько уровней: выбор архитектуры модели, преобразование вычислительного графа, снижение точности представления чисел, эффективное использование аппаратного ускорителя и организацию конвейера ввода-вывода. TensorRT, например, предназначен для компиляции и оптимизации моделей под графические процессоры NVIDIA и поддерживает различные форматы пониженной точности [1]. ONNX Runtime также предоставляет графовые оптимизации и механизмы настройки исполнения [2]. Цель статьи – систематизировать основные методы ускорения вывода без привязки к одной конкретной модели.

Выбор модели и вычислительный бюджет

Наибольший эффект часто достигается до этапа низкоуровневой оптимизации – при выборе архитектуры. Модели семейства MobileNet создавались с ориентацией на мобильные устройства и компромисс между точностью и задержкой. MobileNetV3 сочетает аппаратно-ориентированный поиск архитектуры и компактные вычислительные блоки [3]. EfficientNet показывает другой подход: совместное масштабирование глубины, ширины и разрешения вместо независимого увеличения одного параметра [4].

Для робототехнической задачи следует оценивать не только точность на тестовом наборе, но и фактическое время обработки на целевой платформе. Две модели с близким числом операций могут иметь разную задержку из-за структуры слоёв, пропускной способности памяти и поддержки операций ускорителем. Поэтому выбор должен проводиться по метрикам, связанным с эксплуатацией: медианная и максимальная задержка, потребление памяти, стабильность частоты кадров и качество распознавания на собственных данных.

Разрешение входного изображения также является частью вычислительного бюджета. Его уменьшение снижает число операций во многих свёрточных сетях, но может ухудшать обнаружение мелких объектов. Практический вариант – использовать умеренное разрешение для постоянного анализа и переходить к дополнительной обработке области интереса только при необходимости. Это позволяет направлять ресурсы на содержательно важные участки кадра.

Оптимизация вычислительного графа

После обучения модель обычно содержит последовательность операций, которую можно упростить без изменения математического смысла. ONNX Runtime выполняет преобразования графа, включая свёртку констант, удаление избыточных узлов, объединение операций и оптимизацию расположения данных [2]. Предварительная оптимизация особенно полезна на встроенном устройстве, поскольку уменьшает работу, выполняемую при запуске, и позволяет заранее подготовить модель для целевой среды.

Компиляторы вывода могут объединять несколько операций в одно ядро, выбирать специализированные реализации и подбирать тактики исполнения для конкретного ускорителя. TensorRT строит оптимизированный engine, учитывая доступные форматы данных и свойства графа [1]. Однако результат зависит от поддерживаемых операторов: нестандартный слой может привести к менее эффективному пути или потребовать пользовательского расширения. Поэтому архитектура модели должна учитывать не только качество обучения, но и совместимость с целевой средой исполнения.

Важным источником задержки являются копирования между памятью центрального процессора и ускорителя. Если кадр после захвата несколько раз преобразуется и копируется между библиотеками, выигрыш от быстрого ядра нейросети может быть частично потерян. Желательно минимизировать число преобразований формата, повторно использовать выделенные буферы и, где возможно, связывать входные и выходные тензоры с уже существующей памятью. ONNX Runtime предоставляет механизмы I/O Binding для уменьшения лишних передач данных между устройствами [2].

Пониженная точность и квантизация

Использование чисел меньшей разрядности уменьшает объём памяти и может ускорять вычисления на поддерживаемом оборудовании. FP16 обычно является относительно мягким переходом по сравнению с FP32 и широко применяется на графических ускорителях. INT8 даёт более сильное сокращение объёма данных, но требует контроля масштабов и чувствительности модели. TensorRT поддерживает смешанную точность и квантизацию при наличии соответствующих возможностей аппаратуры [1].

Понижение точности нельзя оценивать только по скорости. После преобразования необходимо повторно измерить качество на репрезентативном наборе данных. Отдельные классы, малые объекты или сцены с плохим освещением могут оказаться чувствительнее к квантизации, чем средняя метрика. Если полная INT8-квантизация ухудшает результат, целесообразен смешанный режим, в котором критичные операции остаются в более высокой точности.

Для мобильного робота важна также повторяемость времени вывода. Максимальная производительность в идеальных условиях не гарантирует стабильной работы при длительной нагрузке. Частоты вычислителя могут снижаться из-за температурных ограничений, а параллельные процессы – конкурировать за память. Поэтому измерения следует проводить на длительных последовательностях, одновременно с реальными сервисами платформы, а не только на изолированном тесте одной модели.

Организация конвейера обработки

Нейросеть является только одной стадией видеоаналитического контура. Типичный конвейер включает захват кадра, изменение размера и цветового пространства, нормализацию, копирование на ускоритель, inference, постобработку и передачу результата другим модулям.

Последовательное выполнение всех стадий в одном потоке ограничивает производительность суммой их задержек. Разделение на независимые этапы с ограниченными очередями позволяет перекрывать захват следующего кадра с обработкой предыдущего.

При этом глубина очереди должна быть небольшой. Большой буфер увеличивает пропускную способность в офлайн-обработке, но в навигации приводит к накоплению устаревших кадров. Если система не успевает обрабатывать каждый кадр, предпочтительнее контролируемо пропускать часть входных данных, сохраняя работу с наиболее свежим изображением. Для задач реального времени batch обычно выбирается малым, часто равным единице, поскольку ожидание наполнения пакета увеличивает задержку реакции.

Полезно отделять нейросетевой вывод от принятия безопасного решения. Модель может временно не выдавать результат из-за перегрузки или ошибки, поэтому критический контур движения должен иметь тайм-ауты и резервные правила. Диагностика должна включать время каждой стадии, загрузку ускорителя, объём памяти и частоту пропусков кадров. Эти показатели позволяют определить, ограничена ли система вычислением модели, подготовкой входа, копированием данных или последующей обработкой.

Заключение

Оптимизация нейросетевого вывода на мобильном роботе должна выполняться на нескольких уровнях одновременно. Компактная архитектура и подходящее разрешение снижают исходный вычислительный бюджет; графовые преобразования устраняют избыточные операции; FP16 и INT8 уменьшают объём вычислений и памяти; специализированные runtimes используют возможности аппаратного ускорителя [1-4].

На практике важна не максимальная скорость отдельного запуска, а устойчивая сквозная задержка всего конвейера. Минимизация копирований, повторное использование буферов, ограниченные очереди и раздельное выполнение стадий помогают сохранить актуальность результатов. После каждой оптимизации необходимо проверять качество модели на целевых данных и измерять работу на реальной платформе под длительной нагрузкой. Такой подход позволяет получить предсказуемое сочетание точности, задержки и ресурсной эффективности

Список литературы:

1. NVIDIA. TensorRT Documentation: Inference and Optimization for NVIDIA GPUs. NVIDIA Developer Documentation.

2. Microsoft. ONNX Runtime Performance: Graph Optimizations, Performance Tuning and I/O Binding. ONNX Runtime Documentation.

3. Howard A., Sandler M., Chen B., Wang W., Chen L.-C., Tan M., Chu G., Vasudevan V., Zhu Y., Pang R., Adam H., Le Q. V. Searching for MobileNetV3 // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2019. P. 1314-1324.

4. Tan M., Le Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks // Proceedings of the 36th International Conference on Machine Learning (ICML). 2019. P. 6105-6114.

5. 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. P. 2704-2713