

КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
ИМЕНИ АЛЬ-ФАРАБИ

ЛЕКЦИЯ № 8

Особенности интеграции нейронных ускорителей Intel Neural Compute Stick и Google Coral с мини-компьютерами Raspberry Pi

Бельгибаев Бауржан Абдрахимович
Д.т.н., доцент · Кафедра «Искусственный интеллект и big data»
«Технологии в интеллектуальных системах управления в IoT» · 4 курс, бакалавриат

1 Цель лекции

Сформировать системное представление об аппаратно-программной интеграции нейронных ускорителей с Raspberry Pi для локального выполнения моделей машинного обучения, обосновать выбор платформы и освоить методы оценки задержки, точности, энергопотребления и надежности интеллектуального IoT-узла.

2 План лекции

1. Вычисления на периферии сети (Edge AI) и роль нейронных ускорителей
2. Архитектура Intel Neural Compute Stick 2 и Google Coral Edge TPU
3. Аппаратная интеграция с Raspberry Pi: питание, USB, камера и сеть
4. Подготовка моделей: OpenVINO, TensorFlow Lite, компиляция и квантование
5. Конвейер компьютерного зрения и программная реализация
6. Математические модели производительности и выбор оборудования
7. Экспериментальная верификация, безопасность и ограничения

1 Edge AI в интеллектуальных системах управления

Интеллектуальная IoT-система состоит из измерительного уровня, вычислительного узла, коммуникационной подсистемы и исполнительных механизмов. При традиционной облачной обработке данные камеры передаются на удаленный сервер, который возвращает результат классификации или детекции. В Edge AI модель исполняется непосредственно на локальном узле. Это уменьшает зависимость от сети, сокращает объем передаваемых данных и позволяет реализовать локальные ограничения безопасности.

Нейронный ускоритель — специализированный сопроцессор для операций, типичных для нейронных сетей (свертки, умножения матриц, нелинейные преобразования). В отличие от самостоятельного микроконтроллера, USB-ускоритель не выполняет функции универсального управляющего устройства: он работает совместно с хостом Raspberry Pi, который отвечает за ОС, драйверы, подготовку входных данных и взаимодействие с датчиками.

Пример структуры: камера → Raspberry Pi (захват и предобработка) → нейроускоритель (инференс) → Raspberry Pi (постобработка и логика управления) → GPIO/промышленный интерфейс → исполнительное устройство. Для критически важных функций необходимо предусматривать независимые аппаратные блокировки.

2 Аппаратные платформы и их особенности

2.1. Intel Neural Compute Stick 2

Intel Neural Compute Stick 2 (NCS2) — USB-устройство на базе процессора Intel Movidius Myriad X, исторически применявшееся для ускорения инференса компьютерного зрения. Типовой программный стек связан с OpenVINO. Поддержка конкретных версий Raspberry Pi OS, архитектуры процессора, OpenVINO и USB-устройства зависит от поколения ПО. Перед установкой необходимо проверить совместимость и доступность архивных пакетов: NCS2 является устаревшей платформой, поэтому его целесообразно рассматривать как учебно-исследовательский объект и как пример эволюции Edge AI.

2.2. Google Coral USB Accelerator

Google Coral USB Accelerator использует Edge TPU — специализированный ASIC для выполнения совместимых квантованных моделей TensorFlow Lite. Для развертывания требуется модель с подходящими операциями и типами данных, а также компилятор Edge TPU. Если часть операций не поддерживается, они могут исполняться на CPU, что увеличивает задержку. Поддержка устройств и пакетов также зависит от версии ОС и архитектуры.

2 Аппаратные платформы и их особенности (продолжение)

Критерий	Intel NCS2	Google Coral USB
Вычислительный блок	Movidius Myriad X VPU	Edge TPU ASIC
Типовой стек	OpenVINO (совместимость версий обязательна)	TensorFlow Lite + Edge TPU runtime
Формат модели	Поддерживаемое представление OpenVINO	Совместимая TFLite-модель, обычно INT8
Сильная сторона	Учебные эксперименты с OpenVINO	Энергоэффективный инференс поддерживаемых моделей
Ограничение	Устаревшая поддержка современных ОС/SDK	Ограничения на операции и квантование

2 Аппаратные платформы и их особенности (продолжение)

Важно: заявленные TOPS, FPS и ватт нельзя напрямую сравнивать без единого набора моделей, разрешения изображений, точности вычислений, условий охлаждения и методики измерения.

3 Аппаратная интеграция с Raspberry Pi

Выбор Raspberry Pi определяется требуемой пропускной способностью USB, производительностью CPU при пред- и постобработке, доступной оперативной памятью и энергобюджетом. Для камер применяются интерфейсы CSI или USB. При одновременной работе камеры, ускорителя и сетевого адаптера следует учитывать конкуренцию за шину и стабильность питания.

Типовая последовательность интеграции: (1) обновить ОС и зафиксировать версии пакетов; (2) подключить ускоритель к совместимому USB-порту; (3) проверить обнаружение устройства; (4) установить поддерживаемый runtime; (5) запустить контрольную модель; (6) проверить работу при длительной нагрузке; (7) подключить управляющий контур и журналирование.

```
uname -m
cat /etc/os-release
lsusb
```

Команда lsusb показывает факт обнаружения устройства, но не гарантирует корректной работы драйвера и выполнения модели. Следует отдельно проверять наличие runtime и результат тестового инференса.

5 Аппаратное подключение

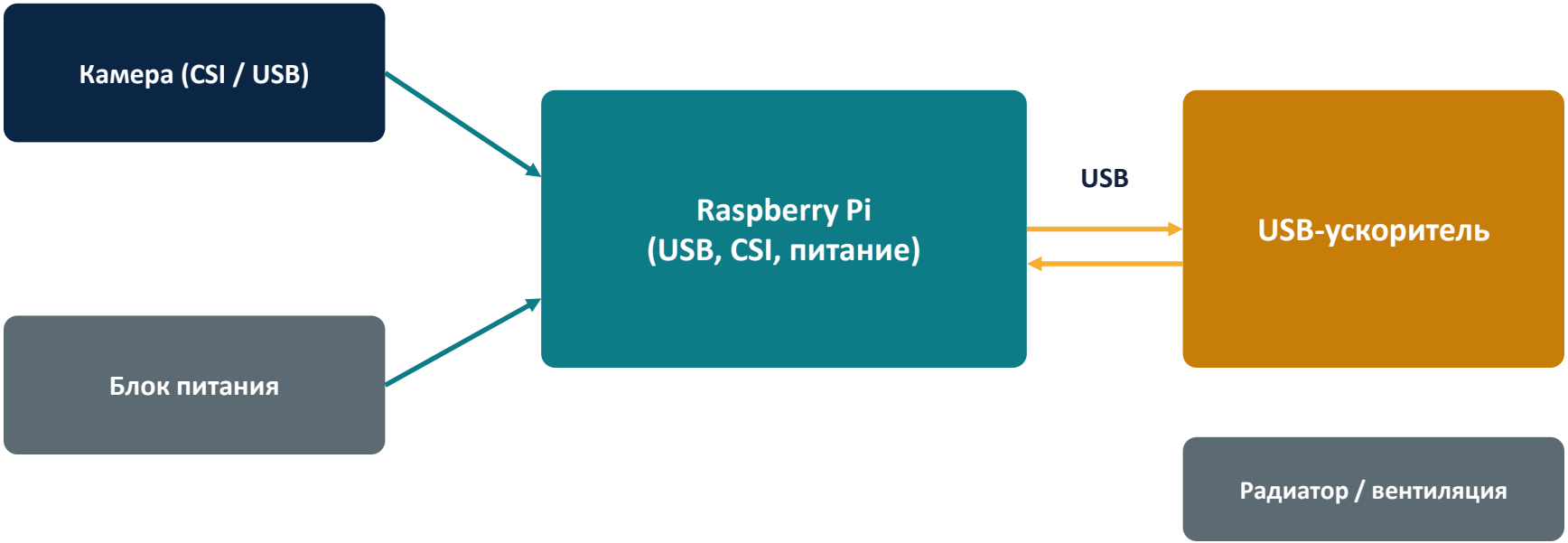


Рис. 15. Контроль питания и температуры важен при длительном инференсе.

4 Подготовка нейронной модели

4.1. Общий конвейер

Обучение модели обычно проводится на рабочей станции или сервере. На Raspberry Pi выполняется инференс: изображение приводится к нужному размеру, нормализуется и подается в модель. Результат преобразуется в метки классов, координаты объектов или вероятности событий. Для Edge TPU необходим этап компиляции совместимой квантованной TFLite-модели; для OpenVINO — экспорт и проверка модели в поддерживаемом формате.

4.2. Квантование

Квантование переводит вещественные веса и активации в дискретное представление, например INT8. Упрощенная симметричная схема: $q = \mathrm{clip}(\mathrm{round}(x/s), -128, 127)$, где s — масштаб квантования. При восстановлении $\hat{x} = s \cdot q$. Ошибка квантования $e_q = x - \hat{x}$ может влиять на точность классификации. Для асимметричной схемы используется нулевая точка z: $q = \mathrm{round}(x/s)+z$.

Посттренировочное квантование требует репрезентативного набора калибровочных данных. Для сложных моделей может понадобиться обучение с учетом квантования (QAT). После преобразования необходимо повторно измерить качество на независимой тестовой выборке.

6 Подготовка модели для двух ускорителей

Intel NCS2



Google Coral

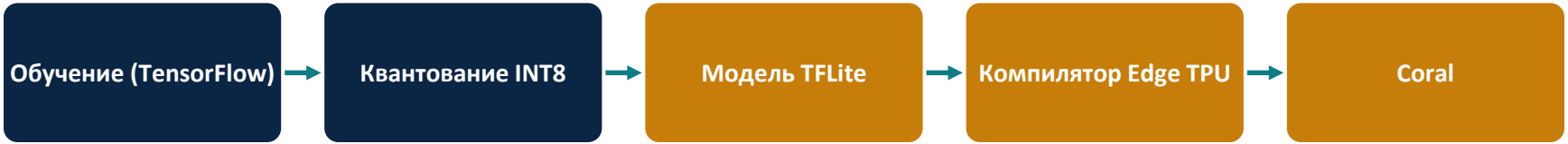


Рис. 14. Два маршрута подготовки: формат и инструменты зависят от ускорителя.

5 Программный конвейер компьютерного зрения

Ниже показан платформенно-нейтральный псевдокод. Конкретный вызов интерпретатора зависит от выбранного ускорителя и версии SDK. Такая форма позволяет отделить алгоритм управления от драйверов оборудования.

```
camera = open_camera()
model = load_compatible_model()
while system_is_running():

  frame = camera.read()
```

Для IoT-мониторинга результаты могут публиковаться через MQTT или сохраняться в локальную базу. В управляющем контуре следует использовать контроль времени жизни сообщения, подтверждение получения команды и безопасное состояние при потере связи. Решение нейросети не должно напрямую обходить аварийные блокировки.

6 Математическая оценка производительности

6.1. Время обработки одного кадра

Полная задержка определяется суммой стадий: $T_{total} = T_{capture} + T_{pre} + T_{transfer} + T_{infer} + T_{post} + T_{control}$. Если средняя длительность обработки кадра равна T_{total}, то теоретическая пропускная способность при последовательной обработке $FPS \approx 1/T_{total}$, где время выражено в секундах. Для конвейерной параллельной обработки пропускная способность ограничивается наиболее медленной стадией, а не обязательно суммой.

Пример (условные экспериментальные значения): захват — 12 мс, предобработка — 8 мс, передача — 3 мс, инференс — 25 мс, постобработка — 7 мс, управляющее решение — 5 мс. Тогда $T_{total} = 60$ мс и $FPS \approx 16{,}7$ кадра/с. Это расчетный пример, а не паспортная характеристика Intel или Coral.

6.2. Энергетическая эффективность

Энергия на один обработанный кадр: $E_{frame} = P_{avg}/FPS$, где P_{avg} — средняя мощность узла в ваттах. Например, при $P_{avg} = 9$ Вт и $FPS = 15$ получаем $E_{frame} = 0{,}6$ Дж/кадр. Для корректного сравнения необходимо измерять мощность всей системы, а не только USB-ускорителя.

6 Математическая оценка производительности (продолжение)

6.3. Точность распознавания

Для задачи бинарной классификации используются Precision = TP/(TP+FP), Recall = TP/(TP+FN), F1 = 2·Precision·Recall/(Precision+Recall). Для детекции объектов также применяются IoU и mAP. Следует дополнительно фиксировать частоту ложных тревог, так как ошибки модели могут приводить к нежелательным управляющим воздействиям.

6.4. Критерий выбора платформы

При многокритериальном выборе можно использовать нормированную функцию $J = w_1 \cdot L_norm + w_2 \cdot E_norm + w_3 \cdot C_norm + w_4 \cdot (1 - Q_norm)$, где L — задержка, E — энергия, C — стоимость, Q — качество распознавания; сумма весов $w_i = 1$. Меньшее значение J соответствует более предпочтительному решению при заданных весах. Нормировка и веса должны быть явно обоснованы.

7 Практический пример: интеллектуальный контроль конвейера

Рассмотрим учебный прототип обнаружения дефектных изделий на конвейере. Камера Raspberry Pi формирует видеопоток; модель классифицирует изображение как «норма» или «дефект»; нейроускоритель сокращает время инференса. Управляющий сигнал направляется на исполнительный механизм только при соблюдении временного окна и разрешающих сигналов датчиков положения.

Для объекта с постоянной скоростью v и расстоянием d от камеры до сортировочного механизма доступное время реакции $T_{available} = d/v$. При $d = 0{,}8$ м и $v = 0{,}4$ м/с получаем $T_{available} = 2$ с. Необходимо обеспечить $T_{total} + T_{actuator} + T_{margin} < T_{available}$. Например, при $0{,}06 + 0{,}25 + 0{,}20 = 0{,}51$ с условие выполняется. В реальном проекте учитываются распределения задержек, джиттер и отказоустойчивость.

Контур испытаний: записать не менее 1000 размеченных кадров из разных условий освещения; разделить выборку на обучающую, валидационную и тестовую без утечки близких кадров; проверить качество на независимых данных; выполнить серии измерений задержки и мощности; испытать отключение камеры, USB и сети; документировать результаты.

8 Ограничения, безопасность и научная воспроизводимость

При выборе аппаратной платформы важно учитывать жизненный цикл SDK и совместимость ОС. Старые устройства могут не поддерживать актуальные версии библиотек. Драйверы, исполняемые файлы и модели необходимо получать из доверенных источников; сетевой доступ ограничивать; результаты инференса логировать с временными метками. Для управления физическим объектом следует реализовать сторожевой таймер, аппаратный аварийный останов и предсказуемое безопасное состояние.

В научном отчете фиксируются: модель Raspberry Pi, ревизия ускорителя, версия ОС и ядра, версии runtime и библиотек, модель нейросети, размер входа, точность/квантование, размер тестовой выборки, FPS, медиана и 95-й перцентиль задержки, энергопотребление, температура, условия эксперимента и доверительные интервалы.

9 Выводы

Нейронные ускорители расширяют возможности Raspberry Pi по локальной обработке данных и позволяют строить интеллектуальные IoT-прототипы с уменьшенной зависимостью от облачных вычислений. Однако практическая эффективность определяется совместимостью ПО, структурой нейросети, задержками пред- и постобработки, энергобюджетом и требованиями безопасности. Intel NCS2 и Google Coral полезны для сравнительного изучения подходов к Edge AI, но их применение должно сопровождаться проверкой актуальной поддержки.

10 Контрольные вопросы

1. Что понимается под Edge AI и чем он отличается от облачного инференса?
2. Какие задачи выполняет Raspberry Pi при использовании USB-нейроускорителя?
3. Почему нейронный ускоритель нельзя считать самостоятельным универсальным микроконтроллером?
4. Каковы архитектурные особенности Intel NCS2?
5. Каковы особенности Google Coral Edge TPU?
6. Почему совместимость SDK и ОС является критическим фактором?
7. Какие операции входят в конвейер обработки видеокадра?
8. Как работает квантование INT8 и что такое масштаб квантования?
9. Для чего используется репрезентативный калибровочный набор?
10. Как вычислить полную задержку интеллектуального IoT-узла?
11. Чем задержка одного кадра отличается от пропускной способности?

10 Контрольные вопросы (продолжение)

12. Как оценить энергию на один обработанный кадр?
13. Какие метрики характеризуют качество классификации и детекции?
14. Какие аппаратные меры безопасности необходимы для управления конвейером?
15. Какие параметры следует указывать для воспроизводимости эксперимента?

11 Рекомендуемая литература и ресурсы

- **Перри Ли. Архитектура интернета вещей. М.:** ДМК Пресс, 2018.
- Матюшин А. О. Программирование микроконтроллеров: стратегия и тактика. М.: ДМК, 2016.
- **Raspberry Pi Documentation:** https://www.raspberrypi.com/documentation/
- **OpenVINO Documentation:** https://docs.openvino.ai/ (проверять совместимость исторических версий с NCS2).
- **Coral Documentation:** https://coral.ai/docs/
- **TensorFlow Lite Documentation:** https://ai.google.dev/edge/litert
- **OpenCV Documentation:** https://docs.opencv.org/