RASPBERRY PI КАМЕРА ЗІ ШТУЧНИМ ІНТЕЛЕКТОМ

RASPBERRY PI AI CAMERA

У статті наведена коротка інформація про 12-мегапіксельну камеру Raspberry Pi зі штучним інтелектом, побудовану на основі датчика зображення Sony IMX 500. Розглянуті основні характеристики камери та рекомендації по роботі з нею.

Abstract – The article provides brief information about the Raspberry Pi's 12-megapixel camera with artificial intelligence, built on the basis of the Sony IMX 500 image sensor. The main characteristics of the camera and recommendations for working with it are considered.

В. Макаренко

V. Makarenko

Компанія Raspberry Pi Foundation окрім модулів Raspberry Pi випускає модулі відеокамер. Однією з останніх розробок є 12-мегапіксельна камера зі штучним інтелектом що використовує датчик зображення Sony IMX 500 для забезпечення високої продуктивності і можливостями штучного інтелекту для будь-якої програми камери [1].

Камера Raspberry Pi AI (рис. 1), сумісна з усіма комп'ютерами Raspberry Pi, використовує переваги інтелектуального датчика зору Sony IMX500, щоб допомогти створювати додатки vision AI і моделі нейронних мереж з використанням вбудованого процесора штучного інтелекту. Тісна інтеграція IMX500 зі стеком програмного забезпечення для камер Raspberry Pi дозволяє користувачам розгортати власні моделі нейронних мереж з мінімальними зусиллями.

Рис. 1. Raspberry Pi AI Camera

Для початку розглянемо посібник, який допоможе запустити на камері готову або виготовлену на замовлення модель нейронної мережі.

Ці інструкції передбачають, що ви використовуєте камеру штучного інтелекту, підключену до плати Raspberry Pi 4 Model B або Raspberry Pi 5. З невеликими змінами ви можете слідувати цим інструкціям на інших моделях Raspberry Pi з роз'ємом камери, включаючи Raspberry Pi Zero 2 W та Raspberry Pi 3 Model B+.

По-перше, переконайтеся, що на вашому Raspberry Pi встановлено найновіше програмне забезпечення. Виконайте наступну команду для оновлення:

```
sudo apt update && sudo apt full-upgrade
```

Встановлення вбудованого ПЗ IMX500

При запуску AI-камера повинна завантажити вбудоване ПЗ. Щоб встановити це вбудоване ПЗ на Raspberry Pi, треба виконати таку команду:

```
sudo apt install imx500-all
```

Ця команда:

- встановлює файли / lib/firmware / imx500_loader.apk та / lib/firmware / imx500_firmware.для файлів вбудованого ПЗ, необхідних для роботи датчика IMX500
- розміщує кілька файлів вбудованого програмного забезпечення для моделей нейронних мереж у/usr/share / imx500-models/
- встановлює програмне забезпечення для подальшої обробки IMX500 у rpicam-apps
- встановлює інструменти для упаковки мережевих моделей Sony.

Драйвер пристрою IMX500 kernel завантажує всі файли вбудованого ПЗ при запуску камери. Це може зайняти кілька хвилин, якщо вбудована модель нейронної мережі не була попередньо кешована. У наведених нижче демонстраціях на консолі відображається індикатор виконання, що показує хід

завантаження вбудованого ПЗ.

Після встановлення ПЗ необхідно зробити перезавантаження Raspberry Pi, виконаши команду:

```
sudo reboot
```

Як тільки всі системні пакети будуть оновлені, а вбудовані файли встановлені, можна запускати прикладні програми. Raspberry Pi AI Camera повністю інтегрується з програмами libcamera, програмами rpicam та Pi camera 2.

Додатки rpicam-apps

Програми для камер rpicam-apps включають етапи виявлення об'єктів IMX500 та оцінки пози, які можна запускати в процесі постобробки. Додаткові відомості про процес постобробки наведені в документації з постобробки. Приклади використовують файли JSON для подальшої обробки, розташовані в /usr/share/rpicam-assets/.

Виявлення об'єктів

Нейронна мережа MobileNet SSD виконує базове виявлення об'єктів, надаючи обмежувальні рамки і значення достовірності для кожного знайденого об'єкта. Файл imx500_mobilenet_ssd.json містить параметри конфігурації для етапу подальшої обробки результатів виявлення об'єктів IMX500 з використанням нейронної мережі MobileNet SSD.

Файл imx500_mobilenet_ssd.json запускає конвеєр після обробки, який містить два етапи:

- imx500_object_detection, який вибирає обмежувальні рамки та значення довіри, що генеруються нейронною мережею, у вихідному тензорі
- object_detect_draw_cv, який малює обмежувальні рамки та написи на зображенні.

Тензор MobileNet SSD не вимагає додаткової постобробки на Raspberry Pi для отримання остаточного результату у вигляді обмежувальних рамок. Всі операції по виявленню об'єктів виконуються безпосередньо камерою штучного інтелекту.

Наступна команда запускає rpicam-hello з подальшою обробкою виявлення об'єктів:

```
rpicam-hello -t 0s --post-process-file /usr/share/rpi-camera-assets/imx500_mobilenet_ssd.json --viewfinder-width 1920 --viewfinder-height 1080 --framerate 30
```

Після виконання команди ви побачите видошукач, який накладає обмежувальні рамки на об'єкти, розпізнані нейронною мережею (рис. 2).

Щоб записати відео з накладанням об'єктів виявлення, використовуйте команду rpicam-vid:

Рис. 2. Обмежувальні рамки на об'єктах, розпізнаних нейронною мережею

```
rpicam-vid -t 10s -o output.264 --post-process-file /usr/share/rpi-camera-assets/imx500_mobilenet_ssd.json --width 1920 --height 1080 --framerate 30
```

Розпізнанвання imx500_object_detection можна налаштувати різними способами.

Наприклад, параметр max_detections визначає максимальну кількість об'єктів, які конвеєр виявить у будь-який момент часу. поріг визначає мінімальне значення довіри, необхідне для того, щоб конвеєр міг розглядати будь-які вхідні дані як об'єкт.

Необроблені вихідні дані цієї мережі можуть бути досить зашумленими, тому на цьому етапі також виконується деяка часова фільтрація та застосовується гістерезис. Щоб вимкнути цю фільтрацію необхідно видалити конфігураційний блок temporal_filter.

Оцінка пози

Нейронна мережа PoseNet виконує оцінку пози, позначаючи ключові точки на тілі, пов'язані з суглобами та кінцівками. Файл imx500_posenet.json містить параметри конфігурації для етапу постобробки оцінки пози IMX500 за допомогою нейронної мережі PoseNet. Він активує конвеєр постобробки, який містить два етапи:

- imx500_posenet, який отримує вихідний тензор із нейронної мережі PoseNet
- plot_pose_cv, який малює лінії, що накладаються на зображення.

Камера штучного інтелекту виконує базове виявлення, але вихідний тензор потребує додаткової пост-обробки на хості Raspberry Pi, щоб отримати

остаточний результат.

Наступна команда запускає rpicam-hello з постобробкою оцінки пози (рис. 3):

```
rpicam-hello -t 0s --post-process-file /usr/share/rpi-camera-assets/imx500_posenet.json --viewfinder-width 1920 --viewfinder-height 1080 --framerate 30
```

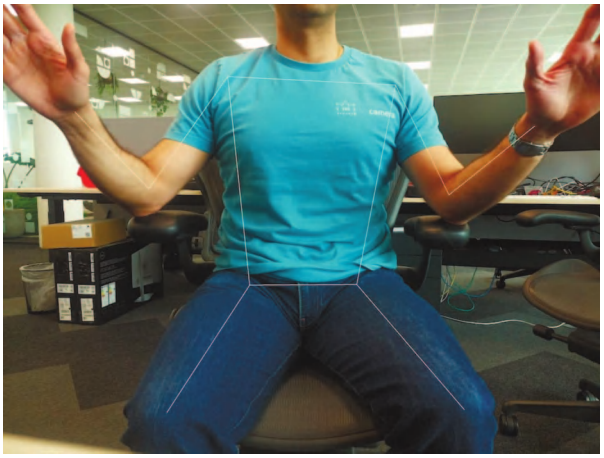


Рис. 3. Результат постообробки оцінки пози

Налаштувати етап imx500_posenet можна різними способами.

Наприклад, max_detections визначає максимальну кількість тіл, які конвеєр виявить у будь-який момент часу. threshold визначає мінімальне значення достовірності, необхідне для того, щоб конвеєр розглядав вхідні дані як тіло.

Picamera2

Приклади класифікації зображень, виявлення об'єктів, сегментації об'єктів і оцінки пози за допомогою Picamera2 (див. у репозиторії picamera2 Url: GitHub https://github.com/raspberrypi/picamera2/tree/main/examples/imx500).

Більшість прикладів використовують OpenCV для додаткової обробки. Щоб установити залежності, необхідні для запуску OpenCV, треба виконати таку команду:

```
sudo apt install python3-opencv python3-munkres
```

Необхідно завантажити репозиторій picamera2 на свій Raspberry Pi, щоб запустити приклади. Приклади файлів знаходяться у кореневому каталозі з додатковою інформацією у файлі readme.md.

Запустіть наступний скрипт зі сховища, щоб запустити виявлення об'єктів YOLOv8:

```
python imx500_object_detection_demo.py --model /usr/share/imx500-models/imx500_network_ssd_mobilenetv2_fpnlite_320x320_pp.rpk
```

Щоб спробувати оцінити позу в Pi camera 2, запустіть наступний скрипт зі сховища:

```
python imx500_pose_estimation_higherhrnet_demo.py
```

Структура AI-камери Raspberry Pi відрізняється від традиційних камер на базі штучного інтелекту, як показано на рис. 4.

На рис. 4,а наведена структура традиційної системи камер зі штучним інтелектом. У такій системі камера передає зображення на Raspberry Pi. який обробляє зображення, а потім виконує логічний висновок за допомогою штучного інтелекту. Традиційні системи можуть використовувати зовнішні прискорювачі штучного інтелекту (як показано на рис.4,а) або покладатися виключно на центральний процесор.

На рис.4,б наведена структура системи, що використовує IMX500. Модуль камери містить процесор обробки сигналів зображення (ISP), який перетворює необроблені дані зображення з камери у вхідний тензор. Модуль камери відправляє цей тензор безпосередньо в прискорювач штучного інтелекту всередині камери, який генерує вихідні тензори, що містять результати логічного висновку. Прискорювач штучного інтелекту надсилає ці тензори до Raspberry Pi. Немає необхідності ні в зовнішньому прискорювачі, ні в Raspberry Pi для запуску програмного забезпечення нейронної мережі на центральному процесорі.

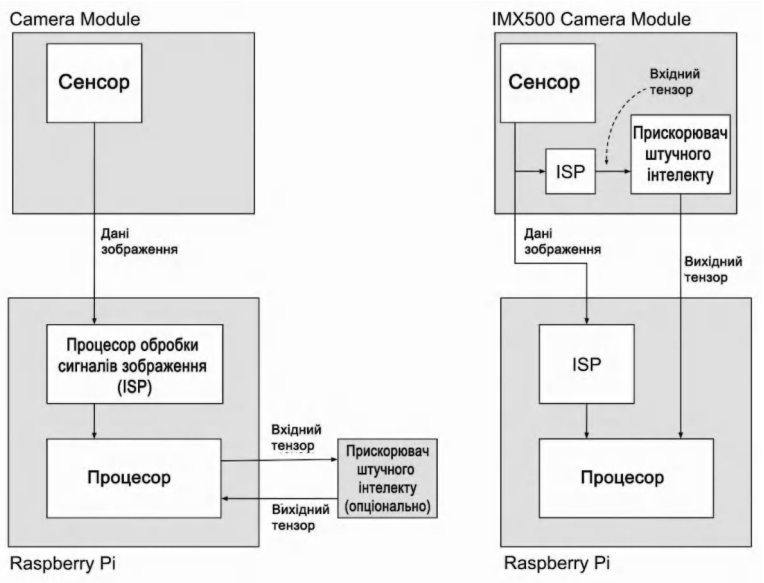
Щоб повністю зрозуміти цю систему, треба ознайомитись з наступними поняттями:

Вхідний тензор

Частина зображення з сенсора, передана в механізм штучного інтелекту для виведення. Створюється невеликим вбудованим провайдером, який також обробляє і масштабує зображення з камери до розмірів, очікуваних завантаженою нейронною мережею. Вхідний тензор, як правило, недоступний для додатків, хоча до нього можна отримати доступ для налагодження.

Область інтересу (ROI)

Визначає, яка саме частина зображення датчика обрізається перед зміною масштабу до розміру, необхідного для нейронної мережі. Може бути запитана і встановлена додатком. В якості одиниць вимірювання завжди використовуються пікселі на виході датчика з повним дозволом. Налаштування ROI за



а) *б)*

Рис. 4. Структура звичайної камери зі штучним інтелектом (а) та Raspberry Pi AI-камери (б)

замовчуванням використовує повне зображення, отримане з датчика, без обрізання даних.

Вихідні тензори

Результати логічного висновку, виконаного нейронною мережею. Точна кількість і форма виходу залежать від нейронної мережі. Код програми повинен розуміти, як поводитися з тензорами.

Архітектура системи обробки зображень

На рис. 5 наведені програмні компоненти камери (виділені зеленим кольором), що використовуються

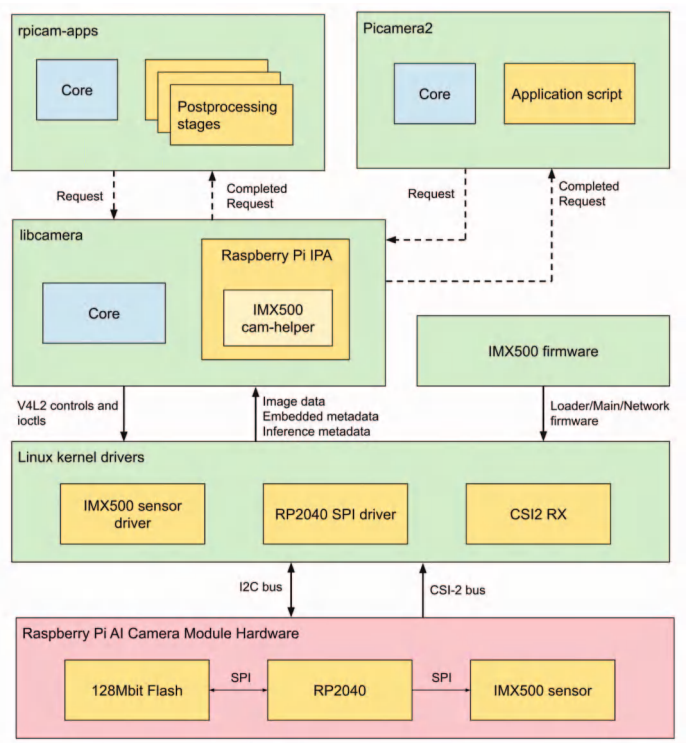


Рис. 5. Програмні компоненти Raspberry Pi AI-камери

в наведеному прикладі використання зображень/виводу з апаратним забезпеченням модуля камери Raspberry Pi AI (виділено червоним кольором).

При запуску сенсорний модуль IMX500 завантажує вбудоване ПЗ для запуску певної моделі нейронної мережі. Під час потокової передачі IMX500 генерує як потік зображень, так і потік логічних висновків. Цей потік логічних висновків містить вхідні та вихідні дані моделі нейронної мережі, також відомі як тензори вводу/виводу.

Драйвери пристроїв

На найнижчому рівні драйвер сенсорного ядра IMX500 налаштовує модуль камери по шині I2C. драйвер CSI2 (CFE на Pi 5, Unicam на всіх інших платформах Pi) налаштовує приймач на запис потоку даних зображення в буфер кадрів разом з потоками вбудованих даних і даних логічного виводу в інший буфер в пам'яті.

Файли вбудованого ПЗ також передаються по шині I2C. На більшості пристроїв для цього використовується стандартний протокол I2C, але в Raspberry Pi 5 використовується спеціальний високошвидкісний протокол. Драйвер RP2040 SPI в ядрі управляє передачею файлів вбудованого ПЗ, оскільки для передачі використовується мікроконтролер RP2040. Мікроконтролер забезпечує передачу даних I2C з ядра в IMX500 по шині SPI. Крім того, RP2040 кешує файли вбудованого ПЗ у вбудованій пам'яті. Це дозволяє уникнути необхідності передачі цілих двійкових файлів вбудованого ПЗ по шині I2C, що значно прискорює завантаження прошивки, яка використовувалась.

Libcamera

Після того як libcamera знімає з черги буфери даних зображення та логічного виводу з ядра, спеціальна бібліотека cam-helper IMX500 (частина Raspberry Pi IPA в libcamera) аналізує буфер логічного виводу (висновків) для доступу до тензорів введення/виведення. Ці тензори упаковані як спеціальні елементи керування libcamera для Raspberry Pi від постачальника. Libcamera повертає елементи керування, наведені в табл. 1.

Програма rpicam

Програма rpicam-apps надає базовий клас етапу постобробки IMX500, який реалізує помічники для етапів постобробки IMX500 – IMX500PostProcessingStage. Потрібно використовувати цей базовий клас для отримання нового етапу постобробки для будь-якої моделі нейронної мережі, що працює на IMX500. Для прикладу наведено imx500_object_detection.cpp:

```
class ObjectDetection : public IMX500PostProcessingStage
{
public:
    ObjectDetection(RPiCamApp *app) : IMX500PostProcessingStage(app) {}
    char const *Name() const override;
    void Read(boost::property_tree::ptree const
```

Таблиця 1. Елементи керування що повертаються libcamera

Елемент	Опис елемента управління
CnnOutputTensor	Масив з плаваючою комою, що зберігає вихідні тензори.
CnnInputTensor	Масив з плаваючою комою, що зберігає вхідний тензор.

CnnOutputTensorInfo	Специфічні для мережі параметри, що описують структуру вихідних тензорів:
	struct OutputTensorInfo {
	uint32_t tensorDataNum;
	uint32_t numDimensions;
	uint16_t size[MaxNumDimensions];
	};
	struct CnnOutputTensorInfo {
	char networkName[NetworkNameLen];
	uint32_t numTensors;
	OutputTensorInfo info[MaxNumTensors];
	};

CnnInputTensorInfo	Специфічні для мережі параметри, що описують структуру вхідного тензора:
	struct CnnInputTensorInfo {
	char networkName[NetworkNameLen];
	uint32_t width;
	uint32_t height;
	uint32_t numChannels;
	};

```
&params) override;
        void Configure() override;
        bool Process(CompletedRequestPtr &completed_request) override;
    };
```

Для кожного кадру, отриманого програмою, викликається функція Process() (ObjectDetection::Process() у наведеному вище прикладу). У цій функції можна отримати вихідний тензор для подальшої обробки або аналізу:

```
    auto output = completed_request->metadata.get(controls::rpi::CnnOutputTensor);
    if (!output)
    {
     LOG_ERROR("No output tensor found in metadata!");
      return false;
    }
    std::vector<float> output_tensor(output->data(), output->data() + output->size());
```

Після завершення кінцеві результати можуть бути візуалізовані або збережені в метаданих і використані на іншому наступному етапі або самому додатку верхнього рівня. У випадку об'єктного висновку:

```
    if (objects.size())
    completed_request->post_process_metadata.Set("object_detect.results", objects);
```

На етапі постобробки object_detect_draw_cv, що наведений нижче, отримує ці результати з метаданих і малює обмежувальні прямокутники на зображенні у функції ObjectDetectDrawCvStage::Process():

```
    std::vector<Detection> detections;
    completed_request->post_process_metadata.Get("object_detect.results", detections);
```

У табл. 2 міститься повний список допоміжних функцій, наданих IMX500PostProcessingStage.

Picamera2

Інтеграція IMX500 в Pi camera 2 дуже схожа на те, що доступно в програмах rpicam. Picamera2 має допоміжний клас IMX500, який забезпечує ту саму функціональність, що і базовий клас rpicam-apps IMX500PostProcessingStage. Його можна імпортувати в будь-який сценарій Python за допомогою скрипта:

```
    from picamera2.devices.imx500 import IMX500
    # This must be called before instantiation of Picamera2
    imx500 = IMX500(model_file)
```

Щоб отримати вихідні тензори, необхідно їх витягнути з елементів керування. Потім можна засто-

Таблиця 2. список допоміжних функцій IMX500PostProcessingStage

Функція	Опис
Read()	Зазвичай викликається з <похідного класу>:: Read (), Ця функція зчитує параметри конфігурації для вхідного тензорного аналізу та збереження. Ця функція також зчитує рядок файлу моделі нейронної мережі ("network_file") і налаштовує вбудоване програмне забезпечення для завантаження під час відкриття камери (camera open).
Process()	Зазвичай викликається з < похідного класу>:: Process (), ця функція обробляє і зберігає вхідний тензор у файл, якщо того вимагає конфігураційний файл JSON.
SetInferenceRoiAbs()	Встановлює прямокутник обрізання абсолютної області інтересу (ROI) на сенсорному зображенні, який буде використовуватися для виведення результатів на IMX500.
SetInferenceRoiAuto()	Автоматично обчислює прямокутник обрізання області, що цікавить (ROI) на сенсорному зображенні, щоб зберегти вхідне тензорне співвідношення сторін для даної нейронної мережі.
ShowFwProgressBar()	Відображає індикатор виконання на консолі, що показує хід завантаження вбудованого ПЗ нейронній мережі в IMX500.
ConvertInferenceCoordinates()	Перетворює з вхідного тензорного координатного простору в кінцевий простір вихідного зображення ISP. Існує ряд операцій масштабування/обрізання / перекладу оригінального сенсорного зображення на повністю оброблене вихідне зображення ISP. Ця функція перетворює координати, надані вихідним тензором, в еквівалентні координати після виконання цих операцій.

сувати додаткову обробку у своєму сценарії Python.

Наприклад, в разі використання логічного висновку об'єкта, такому як imx500_object_detection_demo.py, обмежувальні рамки об'єкта та значення довіри витягуються з parse_detections () і малюються на зображенні в draw_detections():

```
class Detection:
    def __init__(self, coords, category, conf, metadata):
        """Create a Detection object, recording the bounding box, category and confidence."""
        self.category = category
        self.conf = conf
        obj_scaled = imx500.convert_inference_coords(coords, metadata, picam2)
        self.box = (obj_scaled.x, obj_scaled.y, obj_scaled.width, obj_scaled.height)

def draw_detections(request, detections, stream="main"):
    """Draw the detections for this request onto the ISP output."""
    labels = get_labels()
    with MappedArray(request, stream) as m:
        for detection in detections:
            x, y, w, h = detection.box
            label = f"{labels[int(detection.category)]} ({detection.conf:.2f})"
            cv2.putText(m.array, label, (x + 5, y + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
            cv2.rectangle(m.array, (x, y), (x + w, y + h), (0, 0, 255, 0))
        if args.preserve_aspect_ratio:
            b = imx500.get_roi_scaled(request)
            cv2.putText(m.array, "ROI", (b.x + 5, b.y + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 1)
            cv2.rectangle(m.array, (b.x, b.y), (b.x + b.width, b.y + b.height), (255, 0, 0, 0))

def parse_detections(request, stream='main'):
    """Parse the output tensor into a number of detected objects, scaled to the ISP output."""
    outputs = imx500.get_outputs(request.get_metadata())
    boxes, scores, classes = outputs[0][0], outputs[1][0], outputs[2][0]
    detections = [ Detection(box, category, score, metadata)
                  for box, score, category in zip(boxes, scores, classes) if score > threshold]
    draw_detections(request, detections, stream)
```

На відміну від прикладу з rpicam-apps, у цьому прикладі не застосовується додаткова гістерезисна або часова фільтрація.

Клас IMX500 у Pi camera 2 надає допоміжні функції наведені в табл. 3.

Розгортання моделі

Щоб розгорнути нову модель нейронної мережі для камери Raspberry Pi зі штучним інтелектом, необхідно виконати такі дії:

1. Надати модель нейронної мережі.
2. Кількісно обробити і стиснути модель, щоб вона могла працювати з використанням ресурсів, доступних в модулі камери IMX500.
3. Перетворити стиснуту модель у формат IMX500.
4. Упакувати модель у файл вбудованого програмного забезпечення, який можна завантажити в камеру під час виконання.

Перші три кроки зазвичай виконуються на більш потужному комп'ютері, наприклад на настільному комп'ютері або сервері. Завершальний етап упаковки необхідно виконати на Raspberry Pi.

Створення моделей

Існуючі моделі можна повторно використовувати або створювати нові, використовуючи популярні фреймворки, такі як TensorFlow або PyTorch. Для отримання додаткової інформації треба відвідати офіційний веб-сайт розробника ANDROID.

Квантування та стиснення

Для квантування і стиснення моделей використовується інструментарій Sony Model Compression Toolkit. Щоб встановити інструментарій, необхідно виконати наступну команду:

```
pip install model_compression_toolkit
```

Інструментарій стиснення моделей генерує квантовану модель у наступних форматах:

- Keras (TensorFlow)
- HA NX (PyTorch)

Перетворення моделі

Щоб перетворити модель, спочатку необхідно встановити інструменти конвертера:

TensorFlow

```
pip install imx500-converter[tf]
```

Завжди необхідно використовувати ту саму вер-

сію TensorFlow, яка використовувалась для стиснення моделі.

PyTorch

```
pip install imx500-converter[pt]
```

Якщо вам потрібно встановити обидва пакети, використовуйте два окремі віртуальні середовища Python. Це запобігає конфліктам між TensorFlow та PyTorch.

Потім потрібно перетворити модель:

TensorFlow

```
imxconv-tf -i <compressed Keras model> -o <output folder>
```

PyTorch

```
imxconv-pt -i <compressed ONNX model> -o <output folder>
```

Обидві команди створюють вихідну папку, що містить звіт про використання пам'яті та файл packerOut.zip.

Для оптимального використання пам'яті, доступної для прискорювача на сенсорі IMX500, треба додати до вищевказаних команд --no-input-persistency. Однак це призведе до відключення генерації вхідного тензора та повернення до програми налагодження.

Для отримання додаткової інформації про про-

Таблиця 3. Допоміжні функції Pi camera 2

Функція	Опис
IMX500.get_full_sensor_resolution()	Повертає повну роздільну здатність датчика IMX500.
IMX500.config	Повертає словник конфігурації нейронної мережі.
IMX500.convert_inference_coords(coords, metadata, picamera2)	Перетворює координати з вхідного тензорного координатного простору в кінцевий вихідний простір зображень ISP. Необхідно передати метадані зображення Picamera2 для зображення та об'єкта Picamera2. Існує ряд операцій масштабування/обрізання / перекладу оригінального сенсорного зображення на повністю оброблене вихідне зображення ISP. Після виконання цих операцій ця функція перетворює координати, надані вихідним тензором, в еквівалентні координати.
IMX500.show_network_fw_progress_bar()	Відображає індикатор виконання на консолі, що показує хід завантаження вбудованого ПО нейронній мережі в IMX500.
IMX500.get_roi_scaled(request)	Повертає область інтересу (ROI) у координатному просторі вихідного зображення провайдера.
IMX500.get_isp_output_size(picamera2)	Повертає розмір вихідного зображення провайдера.
IMX5000.get_input_size()	Повертає розмір вхідного тензора на основі використовуваної моделі нейронної мережі.
IMX500.get_outputs(metadata)	Повертає вихідні тензори з метаданих зображення Pi camera 2.
IMX500.get_output_shapes(metadata)	Повертає форму вихідних тензорів з метаданих зображення Pi camera 2 для використовуваної моделі нейронної мережі.
IMX500.set_inference_roi_abs(rectangle)	Встановлює прямокутник обрізання області інтересу (ROI), який визначає, яка частина зображення з датчика перетворюється на вхідний тензор, який використовується для виведення на IMX500. Область, що цікавить, повинна бути вказана в одиницях пікселів при повній роздільній здатності датчика у вигляді кортежу (x_offset, y_offset, ширина, висота).
IMX500.set_inference_aspect_ratio(aspect_ratio)	Автоматично обчислює прямокутник обрізки області інтересу (ROI) на сенсорному зображенні для збереження заданого співвідношення сторін. Щоб співвідношення сторін ROI точно відповідало вхідному тензору для цієї мережі, використовуйте imx500.set_inference_aspect_ratio(imx500.get_input_size()).
IMX500.get_kpi_info(metadata)	Повертає показники продуктивності на рівні кадру, що реєструються IMX500 для заданих метаданих зображення.

цес перетворення моделі потрібно звернутись до офіційної документації конвертера Sony IMX 500.

Упаковка моделі

На завершальному етапі модель буде упакована у файл RPK. Під час запуску моделі нейронної мережі необхідно завантажити цей файл у камеру штучного інтелекту. Перш ніж продовжити, необхідно виконати наступну команду, щоб встановити необхідні інструменти **sudo apt install imx500-tools**.

Щоб упакувати модель у файл RPK, необхідно виконати таку команду:

imx500-package -i <path to packerOut.zip> -o <output folder>

Ця команда повинна створити файл під назвою network.rpk у вихідній папці. Необхідно передати назву цього файлу програмам камери IMX500.

Більш повний набір інструкцій і додаткові відомості про використовувані інструменти наведені в документації до Sony IMX 500 Packager [2].

ЛІТЕРАТУРА

1. Raspberry Pi AI Camera. Url: https://www.raspberrypi.com/products/ai-camera/

2. IMX500 Packager User Manual. Url: https://developer.aitrios.sony-semicon.com/en/raspberrypi-ai-camera/documentation/imx500-packager?version=2024-11-21&progLang=