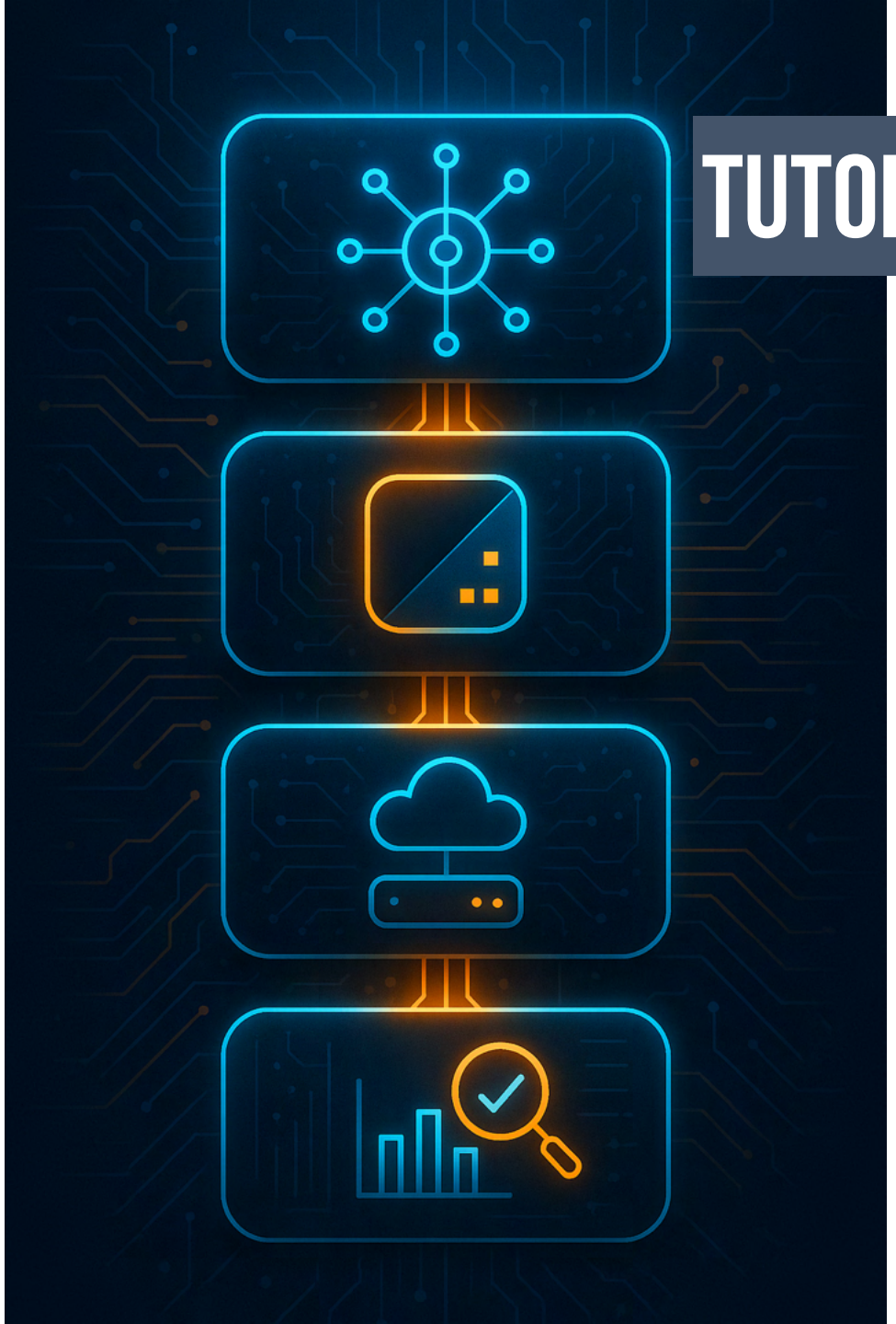
OPTMIZE AND DEPLOY MODELS ON QUALCOMM

CVPR 2026 Tutorial

The IEEE/CVF Conference on Computer Vision and Pattern Recognition 2026

Denver, CO, USA



TUTORIAL AGENDA

1 Post-Training Model Quantization

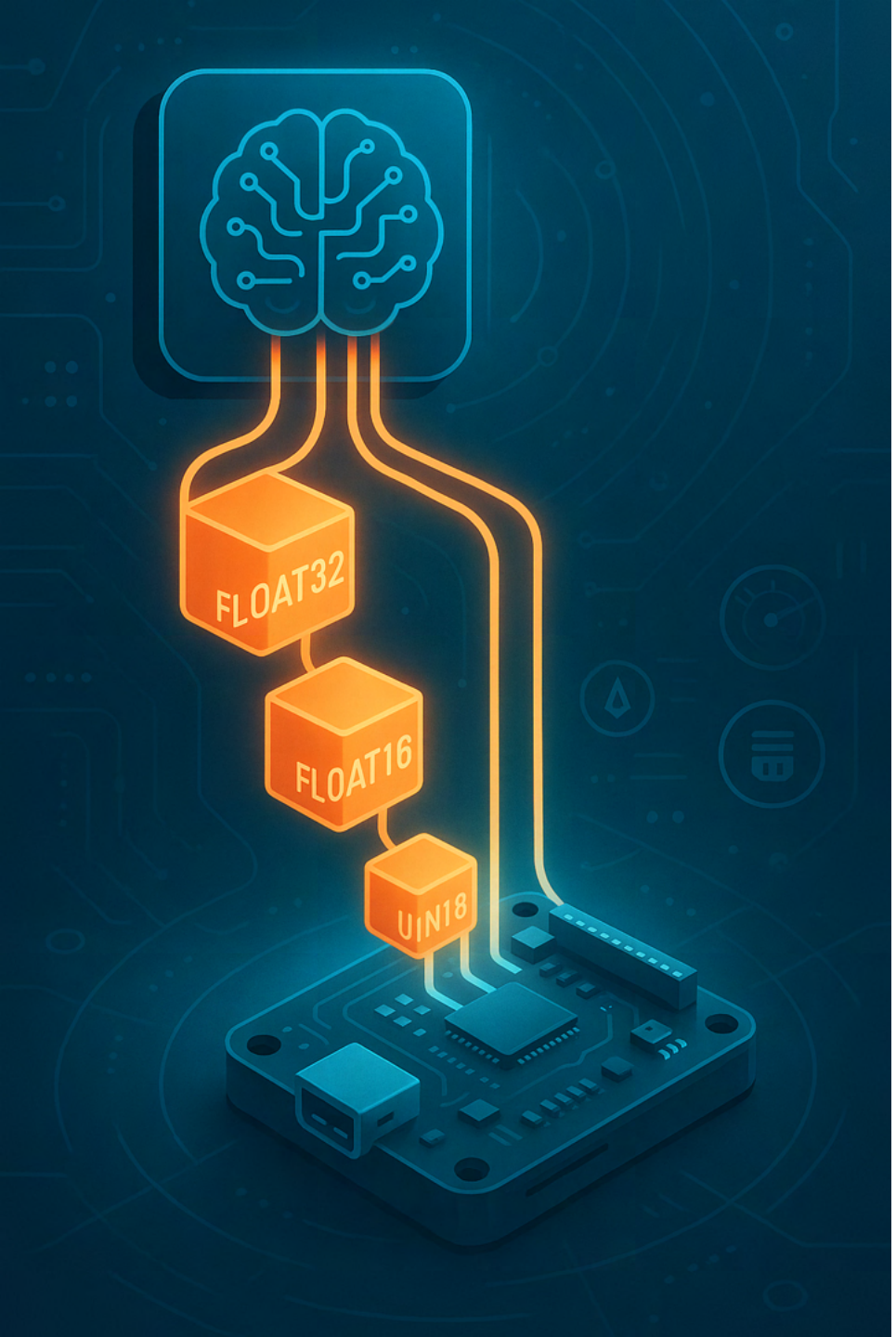
2 Qualcomm Neural Processing SDK

3 Qualcomm AI HUB

4 Model Deployment on Android

5 EdgeVision AI App

POST-TRAINING MODEL QUANTIZATION



INTRODUCTION TO QUANTIZATION

Overview

Quantization is a key technique in **model optimization** for Edge AI, where the precision of model parameters and activations is reduced to accelerate inference and lower memory usage, often with minimal impact on accuracy.

Float32

Offers high precision but requires more memory and computing power.

Float16

Reduces model size and speeds up inference on hardware with native FP16 support, such as GPUs.

Int8

Ideal for edge devices but may require careful calibration to preserve accuracy.

AI PROCESSING UNITS

Hardware

Edge AI devices incorporate diverse processing units, each optimized for different types of computations. Quantization enables these units to run AI models more efficiently by aligning data precision with hardware capabilities.

CPU

A general-purpose processor ideal for model control logic and less demanding inference.

GPU

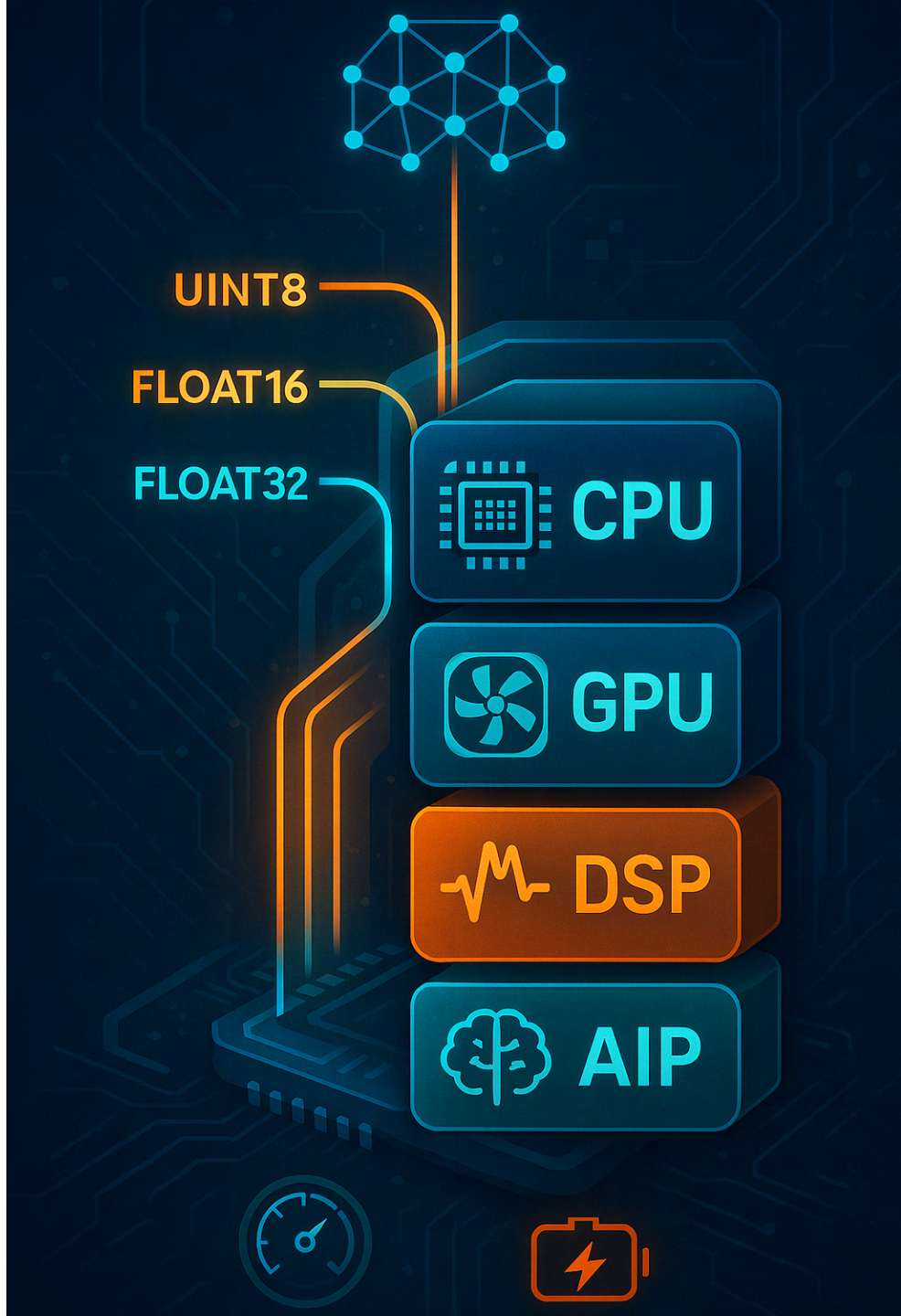
Optimized for parallel floating-point operations, well-suited for CV models with large matrix computations.

DSP

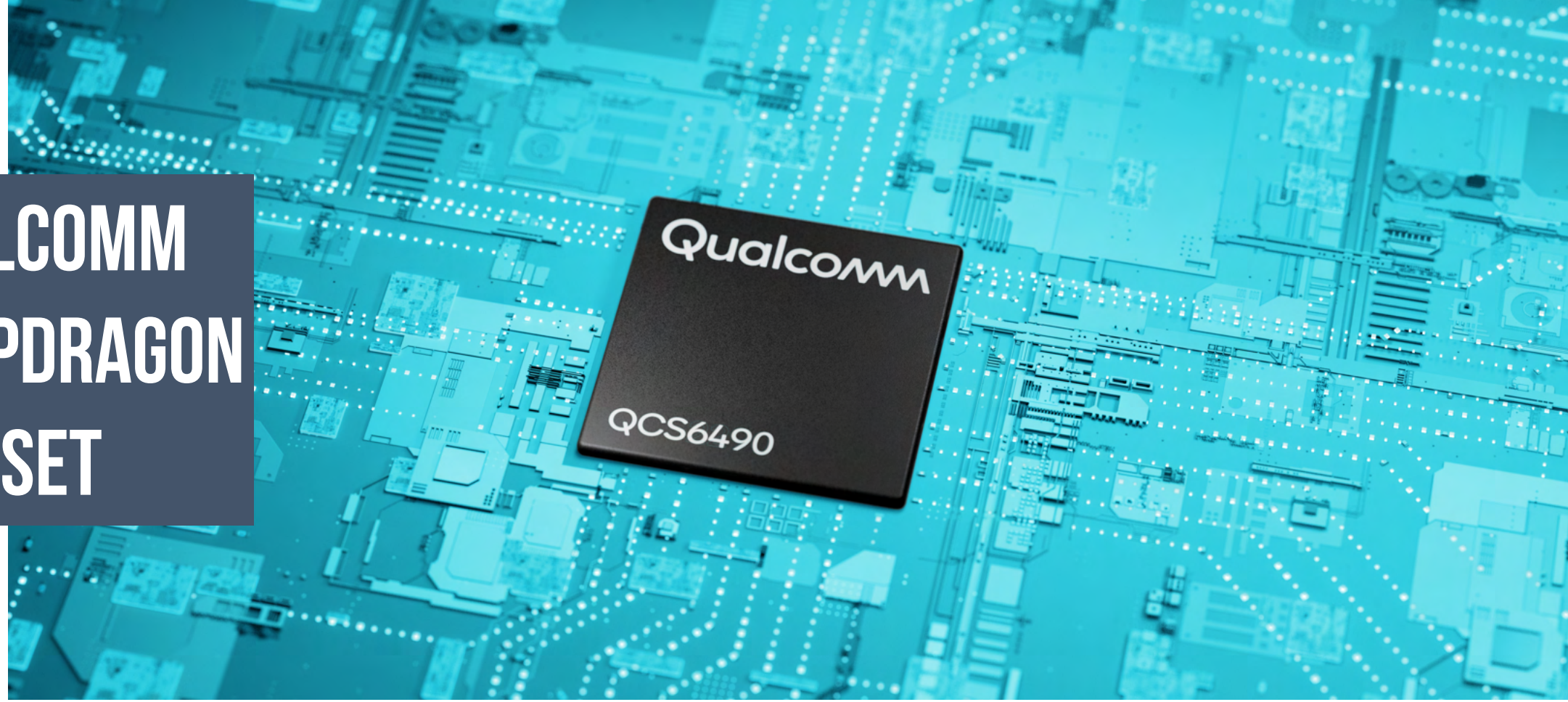
Designed for low-power execution of quantized models (especially UINT8); provides efficient fixed-point computation for always-on tasks.

AIP

Dedicated accelerator explicitly built for deep learning. It delivers high-throughput inference with minimal latency across quantized and mixed-precision models.



QUALCOMM SNAPDRAGON CHIPSET



The Qualcomm QCS6490 is a high-performance chipset designed for intelligent edge computing. It integrates multiple processing engines—CPU, GPU, DSP, and a dedicated AI Processor (AIP)—allowing for efficient on-device AI inference. This makes it ideal for applications in robotics, industrial IoT, and smart cameras. Its compatibility with quantized models and optimization through the SNPE SDK makes it a powerful platform for deploying real-time AI models at the edge.

DEVICES WITH QCS6490

Model Deployment

I/O and Interfaces

Offers GPIO, UART, SPI, I2C, and USB interfaces for hardware debugging, prototyping, and integration with sensors and peripherals.

Edge AI and DSP

Supports AI model deployment using the SNPE SDK, providing direct access to CPUs, GPUs, DSPs, and NPUs.

Headless Configuration

Typically does not include a built-in front camera or display, focusing instead on edge deployment scenarios.

Qualcomm RB3 Gen 2 Vision Development Kit

Android Fairphone 5 5G

Display and Cameras

Comes with high-resolution screens, front/rear cameras, and touch input, making it ideal for real-world testing of vision models and user interfaces.

Power Management

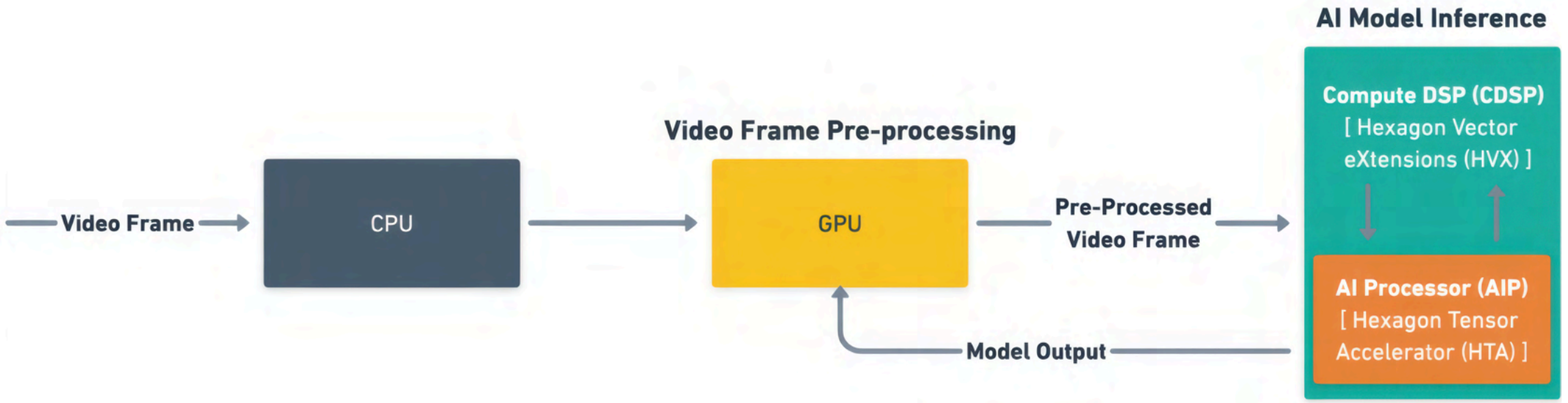
Includes mobile-optimized thermal and power regulation, enabling sustained AI workloads within user-friendly temperature thresholds.

Real-World Application

Allows testing in actual user contexts, making it ideal for validating usability and responsiveness of AI models.

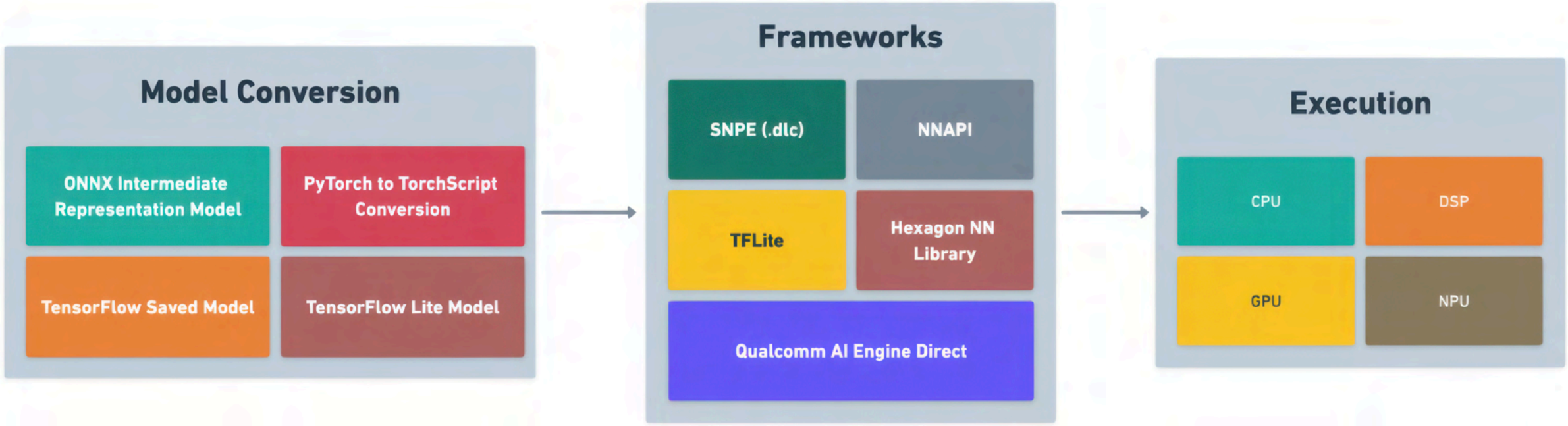
QUALCOMM INFERENCE END-TO-END

Workflow



WORKFLOW FOR MODEL DEPLOYMENT

Deploying Machine Learning Models on Qualcomm Hardware



ONNX VS DLC

Model Formats

ONNX (*Open Neural Network Exchange*) and DLC (*Deep Learning Container*) are two key model formats in the Edge AI deployment pipeline. Converting from ONNX to DLC is a crucial step in model optimization.

ONNX

Open format that enables models trained in different DL frameworks to be converted and reused across various tools and platforms.

DLC

Qualcomm's optimized model format for deployment with SNPE, tailored for efficient execution on DSP, GPU, and AIP.

Conversion Pipeline

Models are exported to ONNX and then converted to DLC using SNPE tools for quantization and hardware-specific tuning.

Execution Compatibility

ONNX is used in general-purpose runtimes, while DLC is mandatory for leveraging full acceleration on Qualcomm devices.

QUALCOMM NEURAL PROCESSING SDK

NEURAL PROCESSING SDK

Overview

The Qualcomm Neural Processing SDK for AI is a suite of tools and libraries for running Artificial Intelligence models directly on Snapdragon devices.

Model Conversion

A set of tools to convert AI models to optimized Qualcomm formats, such as Deep Learning Containers (DLCs).

Quantization

Reduces numerical precision to accelerate inference and save memory, and optimize models for DSP/HTP chipsets.

Multi-Runtime

Automatically runs models on CPU, GPU, or DSP/HTP.

Benchmark Tools

Measures model performance, latency, and efficiency.

QUALCOMM AI ECOSYSTEM

Overview

QAIRT

Qualcomm AI Runtime SDK is a modern runtime based on QNN for optimized, cross-platform AI inference.

QAI HUB

Qualcomm AI Hub is a cloud platform for compiling, profiling, optimizing, and benchmarking models for Snapdragon devices.

QUALCOMM AI ECOSYSTEM

Qualcomm AI Runtime SDK

SNPE

Snapdragon Neural Processing Engine

High-level SDK for converting and executing AI models using CPU, GPU, and DSP/HTP on Snapdragon devices.

QNN

Qualcomm AI Engine Direct

Low-level framework that provides direct access to Qualcomm accelerators for optimized neural network execution.

GENIE

Gen AI Inference Extensions

Qualcomm's framework for local execution of Large Language Models (LLMs) and on-device generative AI applications.

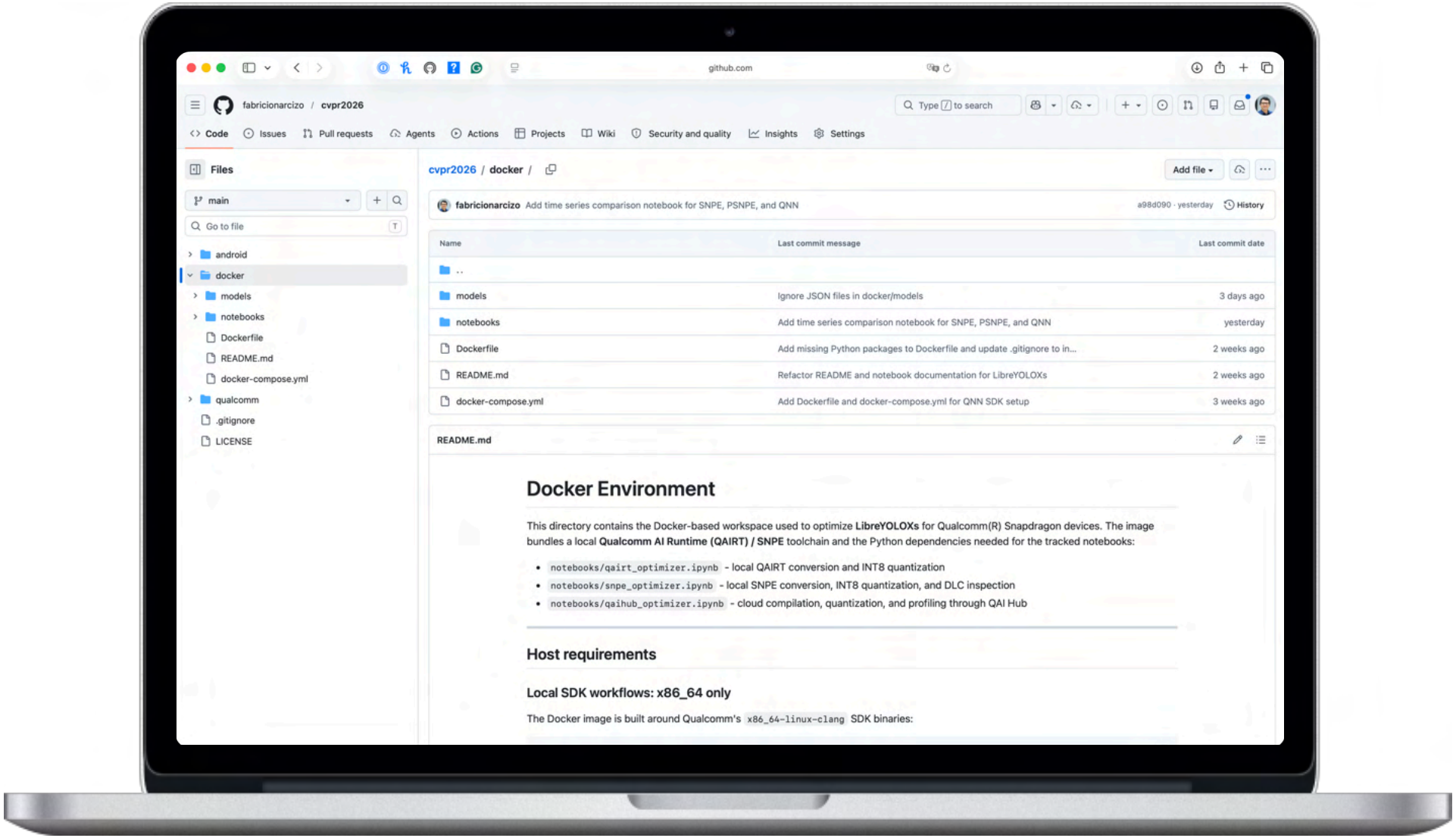
QTLD

Qualcomm TFLite Delegate

It works by delegating TFLite (TensorFlow Lite) operations out to the Qualcomm AI Engine Direct framework.

MODEL OPTIMIZER PLATFORM

https://github.com/fabricionarcizo/cvpr2026/tree/main/docker



MODEL OPTIMIZER PLATFORM

Local Optimization

It contains tools and scripts for optimizing AI models for deployment on edge devices. It includes model conversion, quantization, and benchmarking utilities, as well as Jupyter Notebooks for model optimization and evaluation.

Docker

SNPE Optimizer provides pre-configured Docker images for development and conversion.

Linux (x86_64)

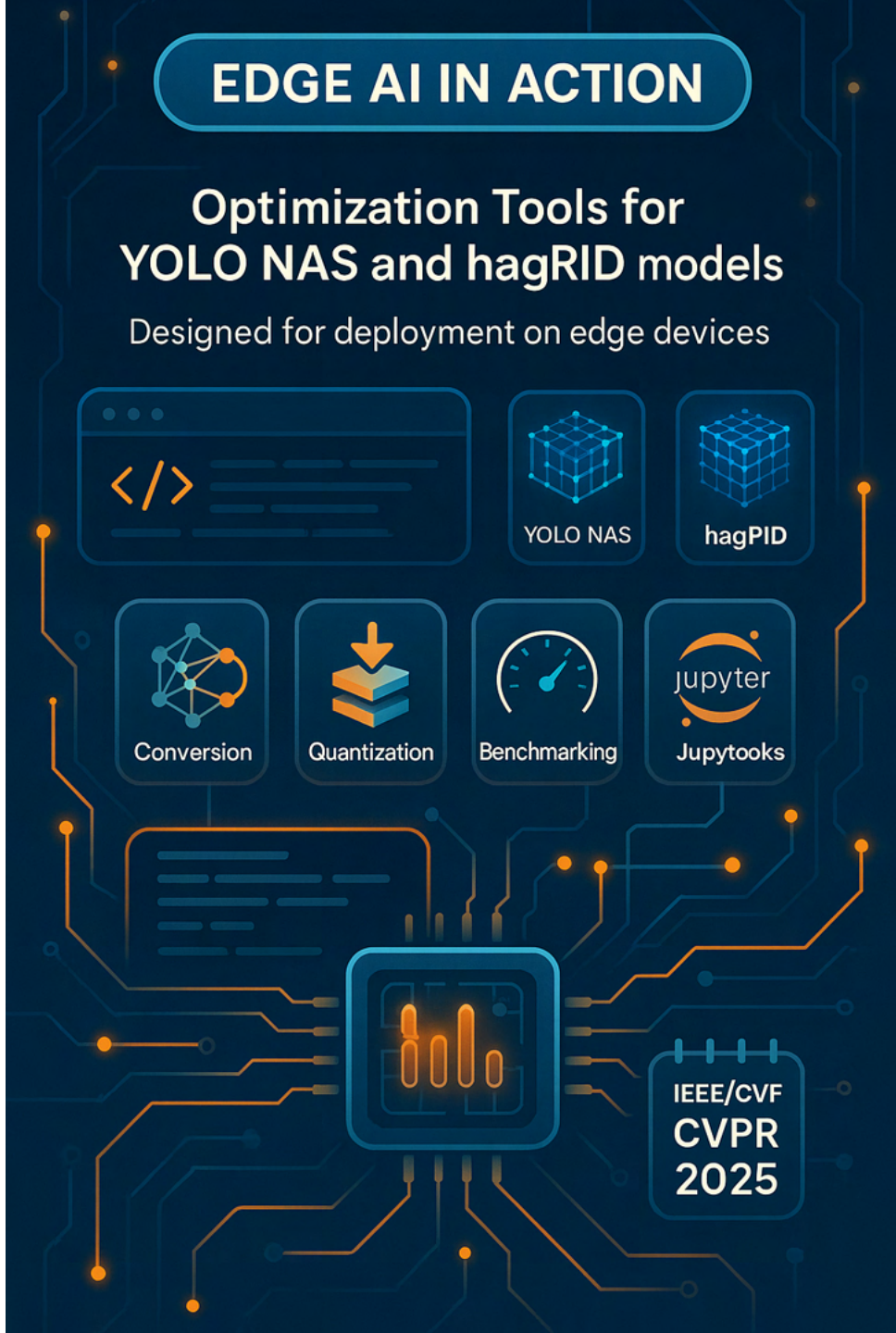
Official support for Ubuntu-based systems ensures stability and compatibility with the SNPE toolchain.

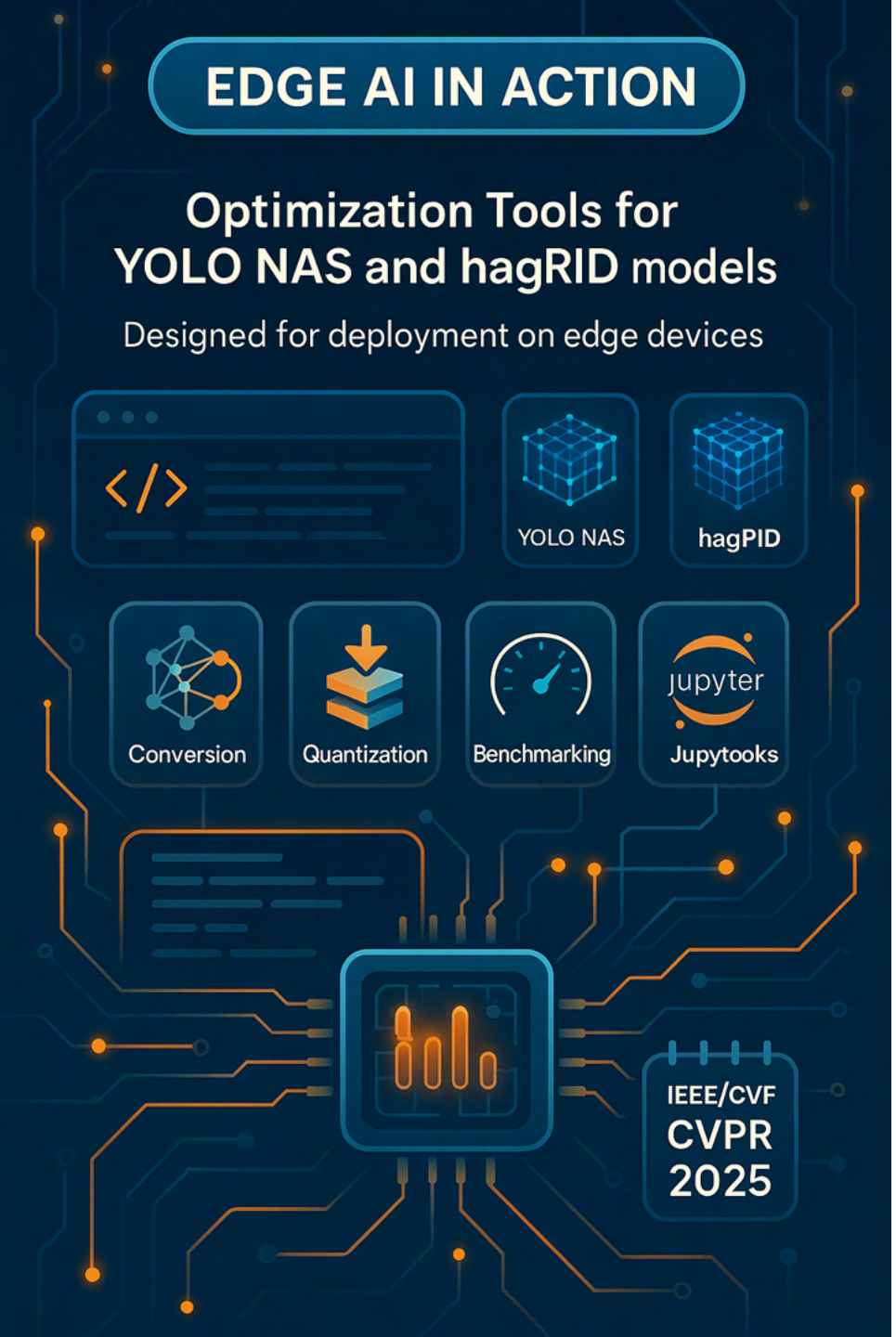
Intel CPU Architecture

Required for compatibility with SNPE tools; ARM-based developer machines are not supported for model conversion.

Android NDK

Necessary to build and deploy Android-native binaries for executing models on actual devices using the SNPE runtime.





MODEL OPTIMIZER PLATFORM

Folder Structure

The Model Optimizer Platform uses volumes to persist data stores implemented by the container engine. These are the primary volumes used in this platform:

models

Pretrained and optimized model files (ONNX, DLC, TFLite, and TensorFlow SavedModel formats).

notebooks

Jupyter notebooks for model optimization, export, and evaluation. Contains validation and raw data folders.

JUPYTER NOTEBOOKS

Model Quantization Workflow

SNPE OPTIMIZER

End-to-end optimization workflow for a LibreYOLO object detection model with the Snapdragon Neural Processing Engine SDK.

QAIRT OPTIMIZER

Deploying and accelerating a LibreYOLO object detector using the Qualcomm AI Runtime SDK.

QAI HUB OPTIMIZER

Streamlined compilation, quantization, and profiling of LibreYOLO with Qualcomm AI Hub.

MODEL OPTIMIZER PLATFORM

Setup and Execution

Model Optimization

Access the model optimization:

```
http://localhost:8888/lab/tree/
*_optimizer.ipynb
```

Docker Container

Build and start the container:

```
$ docker compose up --build -d
```

QAI Hub

Set the QAI Hub API token:

```
$ nano notebooks/.env
QAI_HUB_API_TOKEN=your_api_token
```

Docker Folder

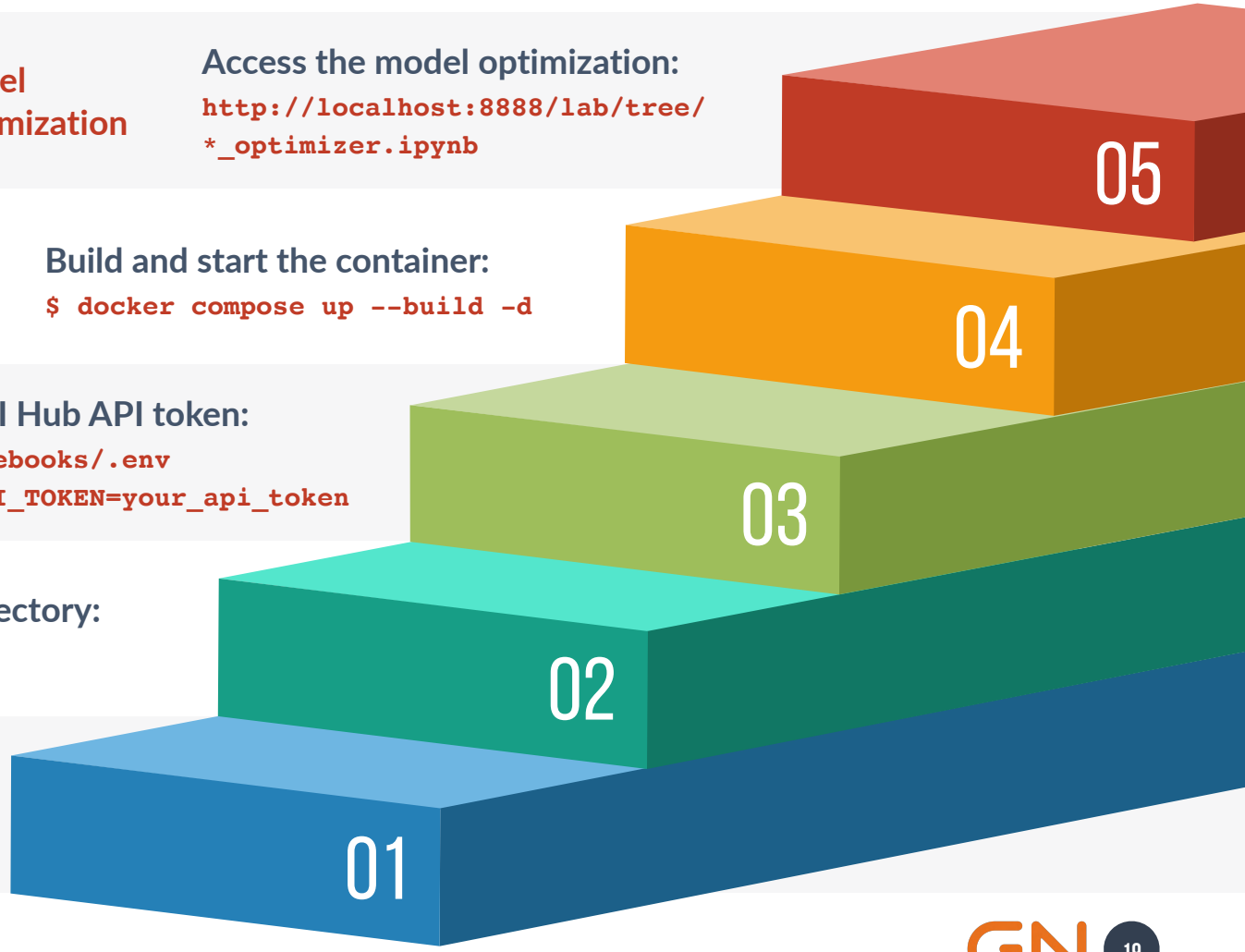
Change the working directory:

```
$ cd cvpr2026/docker
```

GitHub Clone

Clone the GitHub repository:

```
$ git clone https://github.com/
fabricionarcizo/cvpr2026
```



MODEL OPTIMIZER PLATFORM

Optimization Steps

EXPORT ONNX

Export a pre-trained model to the ONNX format, typically by using a tool like PyTorch ONNX exporter or a similar tool specific to your model's framework.

DOWNLOAD DATASET

We will use the dataset to calculate the ranges and calibrate the quantization parameters.

MODEL CONVERSION

Use any conversion tool to convert a non-quantized model ONNX to a non-quantized DLC file (F32).

MODEL QUANTIZATION

Use any quantization tool to quantize the Float32 model to one of the supported fixed-point formats (UINT8).

BINARY GENERATOR

Once conversion/quantization finishes, generate the model's binary to run on the target device's OS.

LIBREYOLO MODELS

Object Detection

LibreYOLO is an MIT-licensed computer vision library that supports both training and inference across multiple models. It offers an intuitive high-level Python and CLI interface while maintaining compatibility with standard YOLO-format datasets, enabling existing workflows to transition with minimal modifications.

Official Website

The website contains links for the models, datasets, and docs.

GitHub Repository

You can install the Python library directly from the GitHub repository.

CALIBRATION DATA

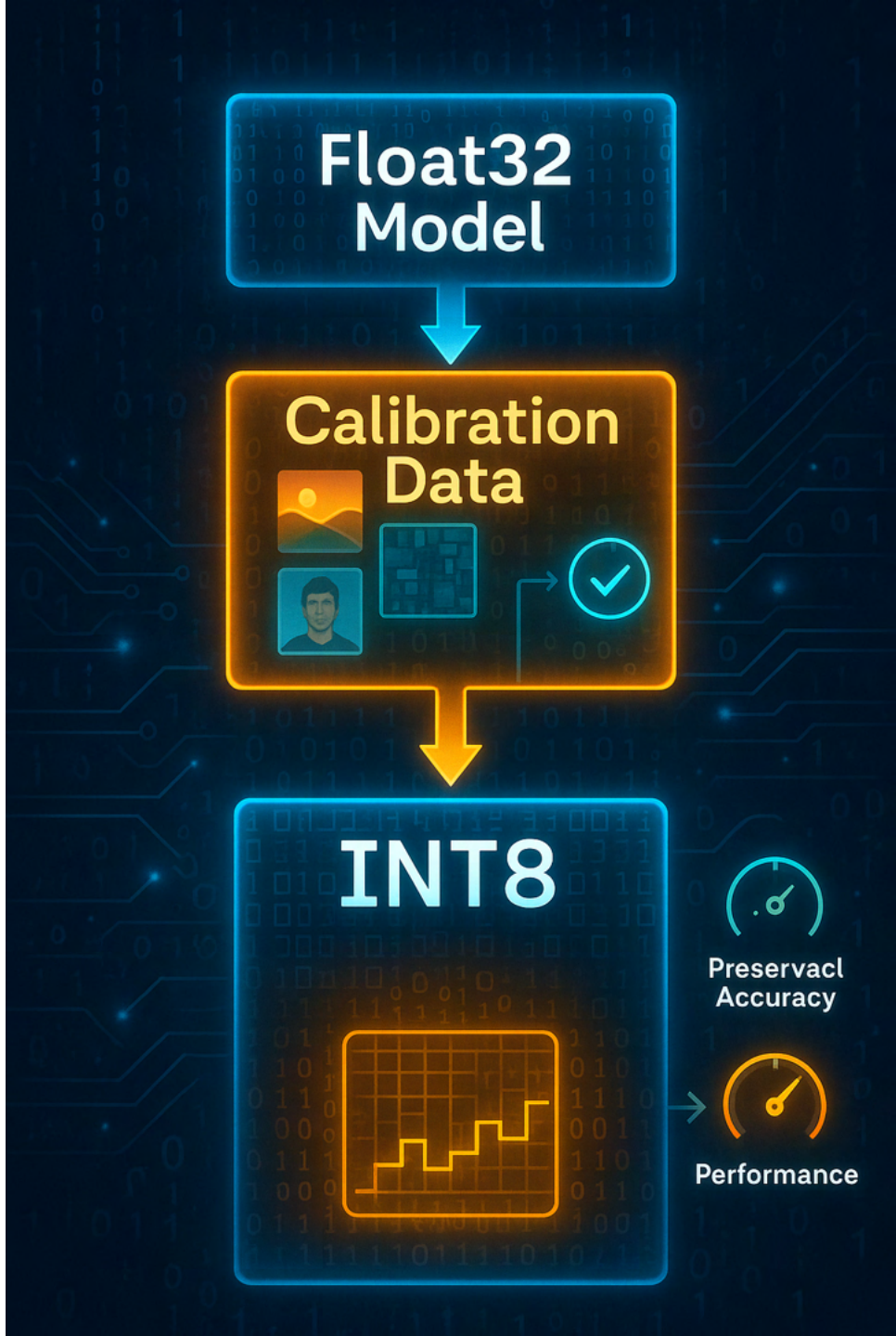
Overview

WHAT IS IT?

Representative input samples used during model quantization to preserve accuracy when converting from `float32 → int8`

REASONS

Edge devices need 8-bit models for speed and efficiency. Naive quantization can destroy model accuracy. We need to understand the typical value ranges in each model layer.



COCO DATASET

Common Object in Context

80 Object Categories

Includes a broad range of everyday objects like people, vehicles, animals, and household items to support general-purpose detection.

Rich Annotations

Each image is annotated with bounding boxes, segmentation masks, keypoints, and contextual metadata.

Real-World Scenarios

Features complex, cluttered scenes that closely resemble real-life environments—ideal for training edge-focused models.

GETTING COCO DATASET

Validation Dataset

Validation Dataset

To evaluate object detection models like LibreYOLOXs, downloading and preparing the COCO validation dataset is essential.

Data Annotation

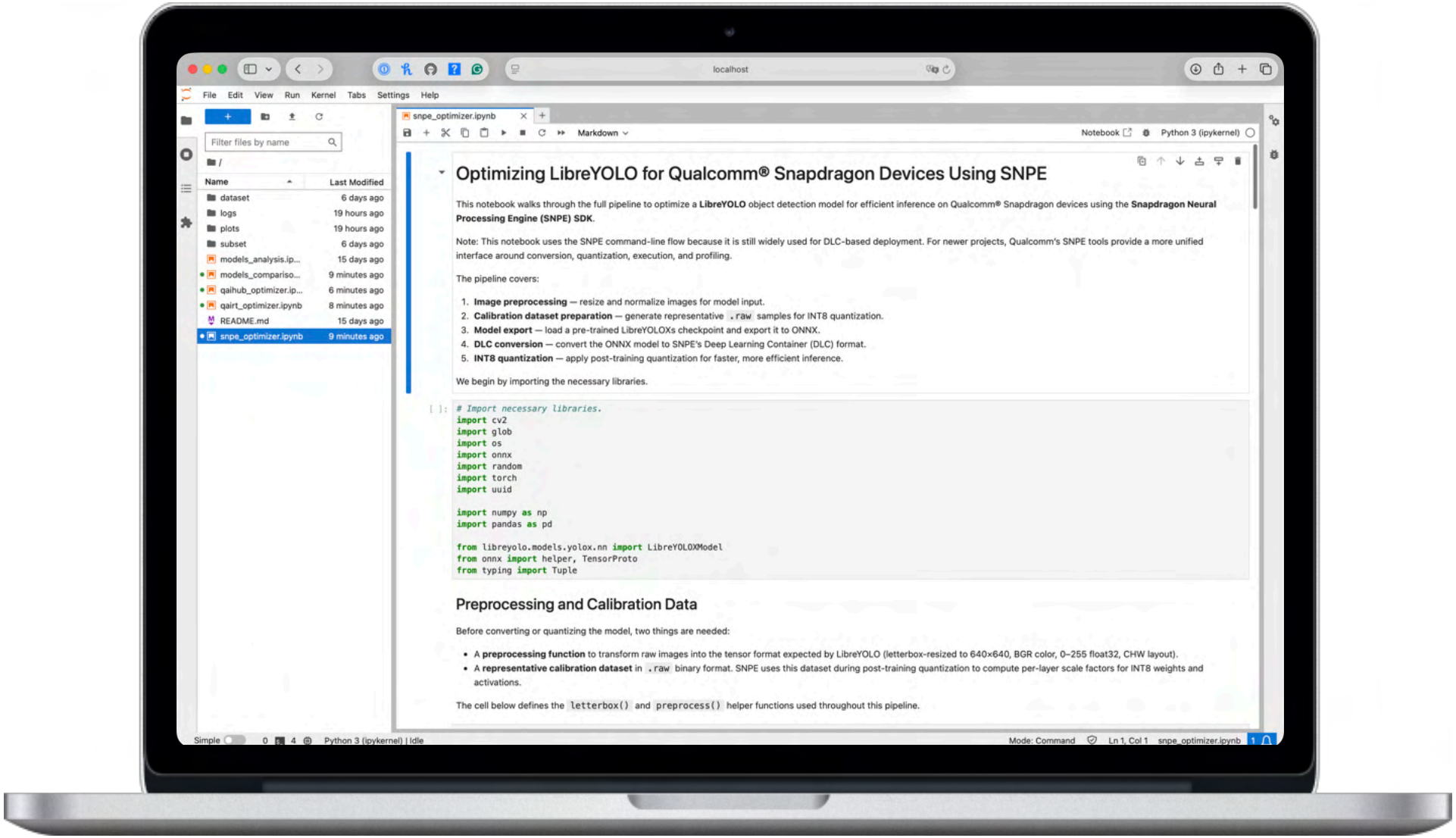
The validation set provides a standardized benchmark to assess model accuracy, bounding box quality, and object recall.

Optimization Phase

The validation split (commonly `val2017`) enables faster iteration and tuning during the optimization phase.

SNPE OPTIMIZER NOTEBOOK

http://localhost:8889/lab/tree/snpe_optimizer.ipynb



SNPE SDK

Qualcomm Neural Processing SDK for AI

The SNPE SDK is Qualcomm's official toolkit for deploying AI models on Snapdragon-powered devices.

HOW TO OPTIMIZE AN AI MODEL USING V2.35.0.250530?

CONVERSION TOOLS

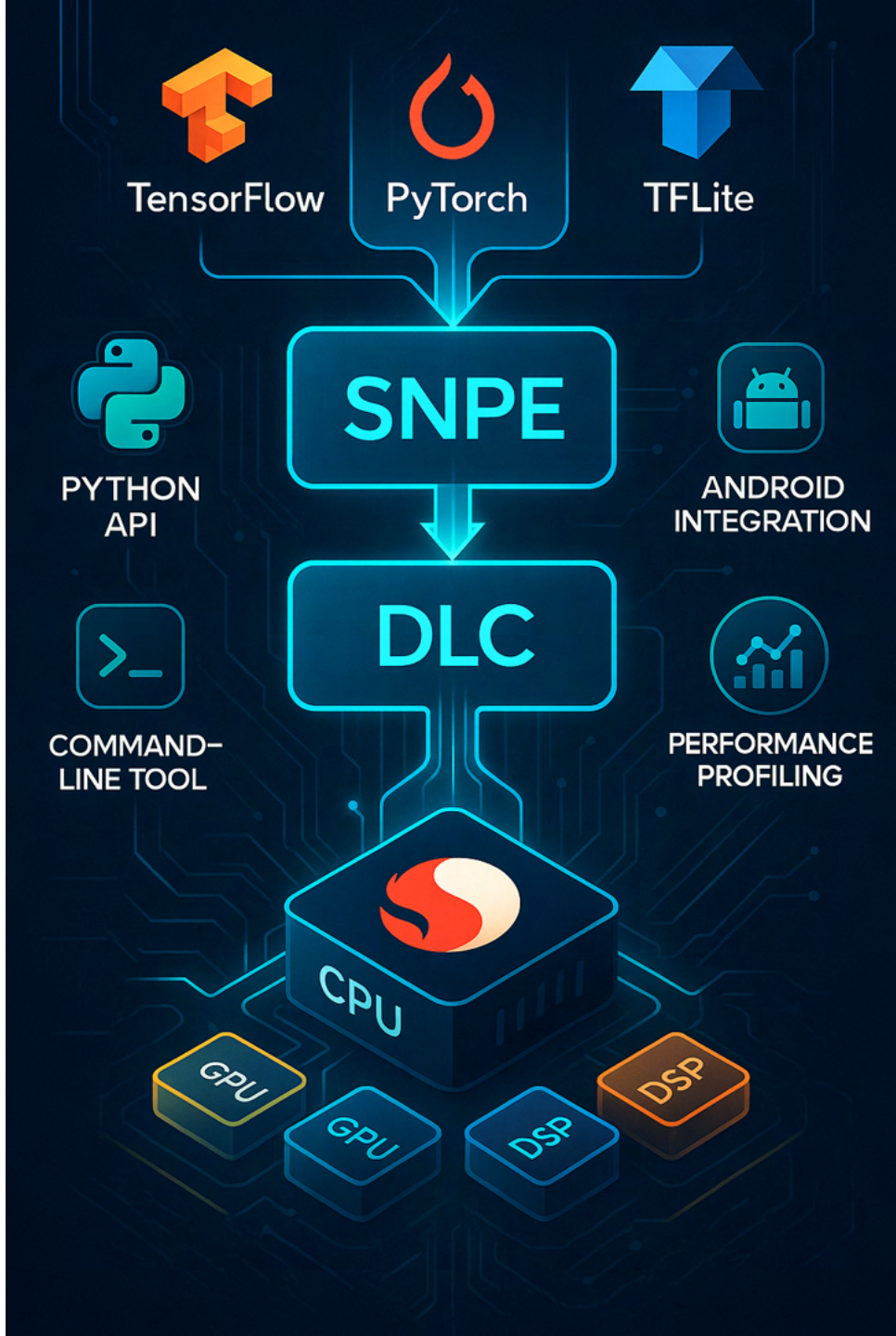
Converts models from ONNX/TF to DLC, with optional quantization for UInt8 inference and graph optimizations.

COMMAND-LINE

Offers interfaces for automation, scripting, and integration into custom workflows and CI/CD pipelines.

PERFORMANCE PROFILING

Tools for benchmarking inference speed, memory usage, and layer-by-layer diagnostics.



SNPE SDK COMMANDS

https://www.qualcomm.com/developer/software/neural-processing-sdk-for-ai

1 **snpe-onnx-to-dlc**
Converts ONNX models to DLC (a widely used format-agnostic format).

2 **snpe-pytorch-to-dlc**
Converts PyTorch models into DLC format for Snapdragon inference.

3 **snpe-tensorflow-to-dlc**
Converts TensorFlow models to DLC.

4 **snpe-tflite-to-dlc**
Converts TFLite models to DLC.

5 **snpe-dlc-info**
Displays information about a DLC model.

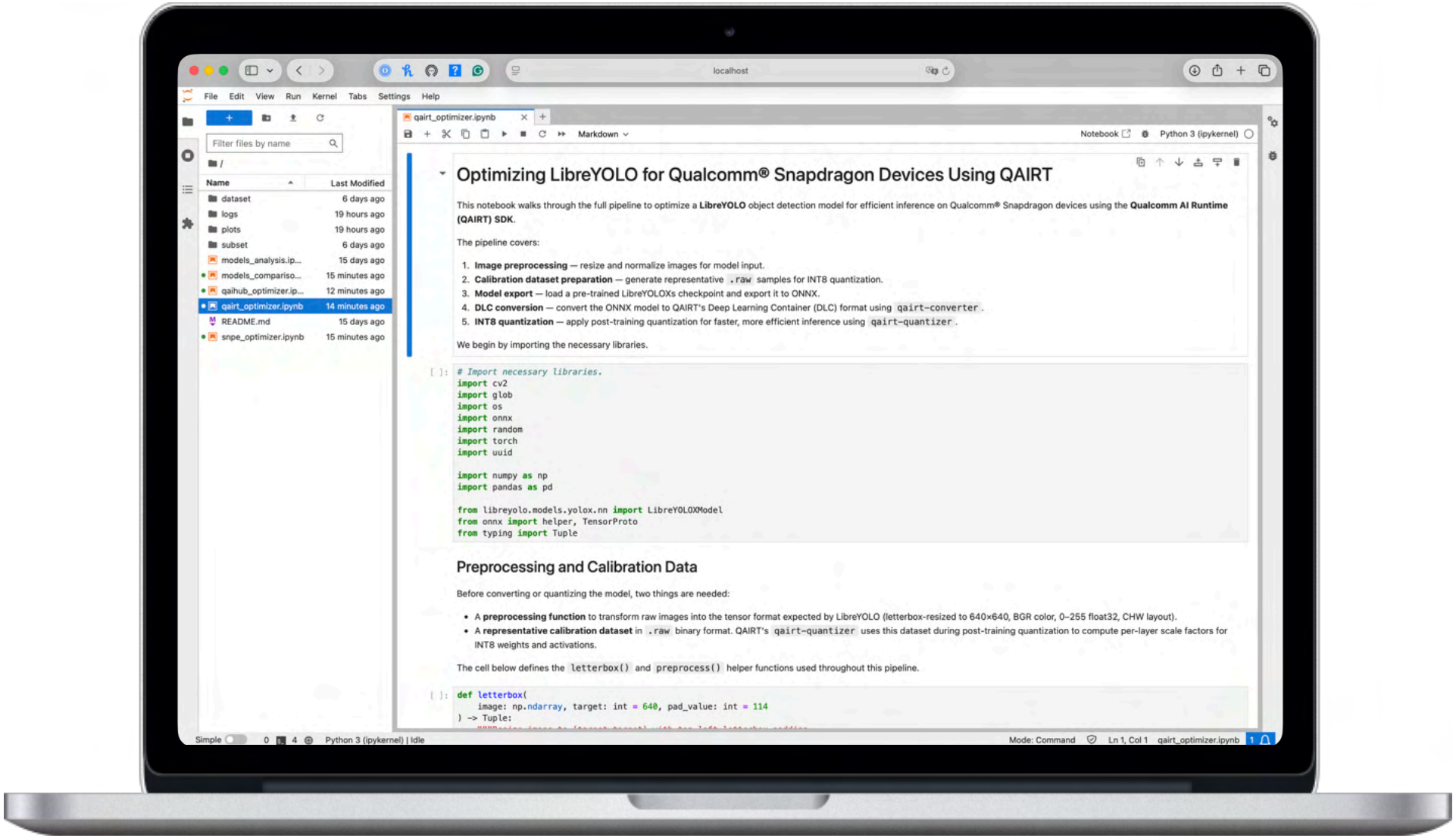
6 **snpe-throughput-net-run**
Measures inference speed and throughput.

7 **snpe-dlc-diff**
Compares two DLC models to detect changes.

8 **snpe-dlc-viewer**
GUI-based tool to inspect DLC structure.

QAIRT OPTIMIZER NOTEBOOK

http://localhost:8889/lab/tree/qairt_optimizer.ipynb



QAIRT SDK

Qualcomm AI Runtime SDK

Unified runtime framework for deploying and accelerating AI inference across Qualcomm heterogeneous compute platforms.

HOW TO OPTIMIZE AN AI MODEL USING V2.35.0.250530?

MULTI-RUNTIME

Run AI models efficiently on CPU, GPU, and Hexagon NPU/DSP accelerators.

OPTIMIZED DEPLOYMENT

Model conversion, quantization, compilation, and runtime optimization based on SNPE and QNN.

CROSS-PLATFORM

Enables AI deployment across Android, Linux, IoT, automotive, and embedded systems.

QAIRT SDK COMMANDS

https://www.qualcomm.com/developer/software/neural-processing-sdk-for-ai

1 **qairt-accuracy-evaluator**

Evaluates model inference accuracy by comparing outputs against reference results or datasets.

2 **qairt-converter**

Converts AI models from frameworks such as ONNX or TensorFlow into QAIRT-compatible DLC format.

3 **qairt-dlc-diff**

Compares two DLC models to identify structural or parameter differences.

4 **qairt-dlc-info**

Displays detailed information about a DLC model, including layers, tensors, and input/output specifications.

5 **qairt-dlc-to-json**

Exports DLC model metadata and structure into a JSON representation for inspection and analysis.

6 **qairt-quantizer**

Quantizes AI models to lower-precision formats, such as INT8, for faster, more efficient edge inference.

QUALCOMM AI HUB

QUALCOMM AI HUB

Overview

Qualcomm AI Hub is a developer-centric platform that streamlines the deployment of on-device AI for Snapdragon-powered hardware. It enables seamless workflows—from model import and optimization to profiling and deployment.

Model Conversion

Transform trained models (e.g., PyTorch, ONNX) for optimal on-device performance.

Performance Profiling

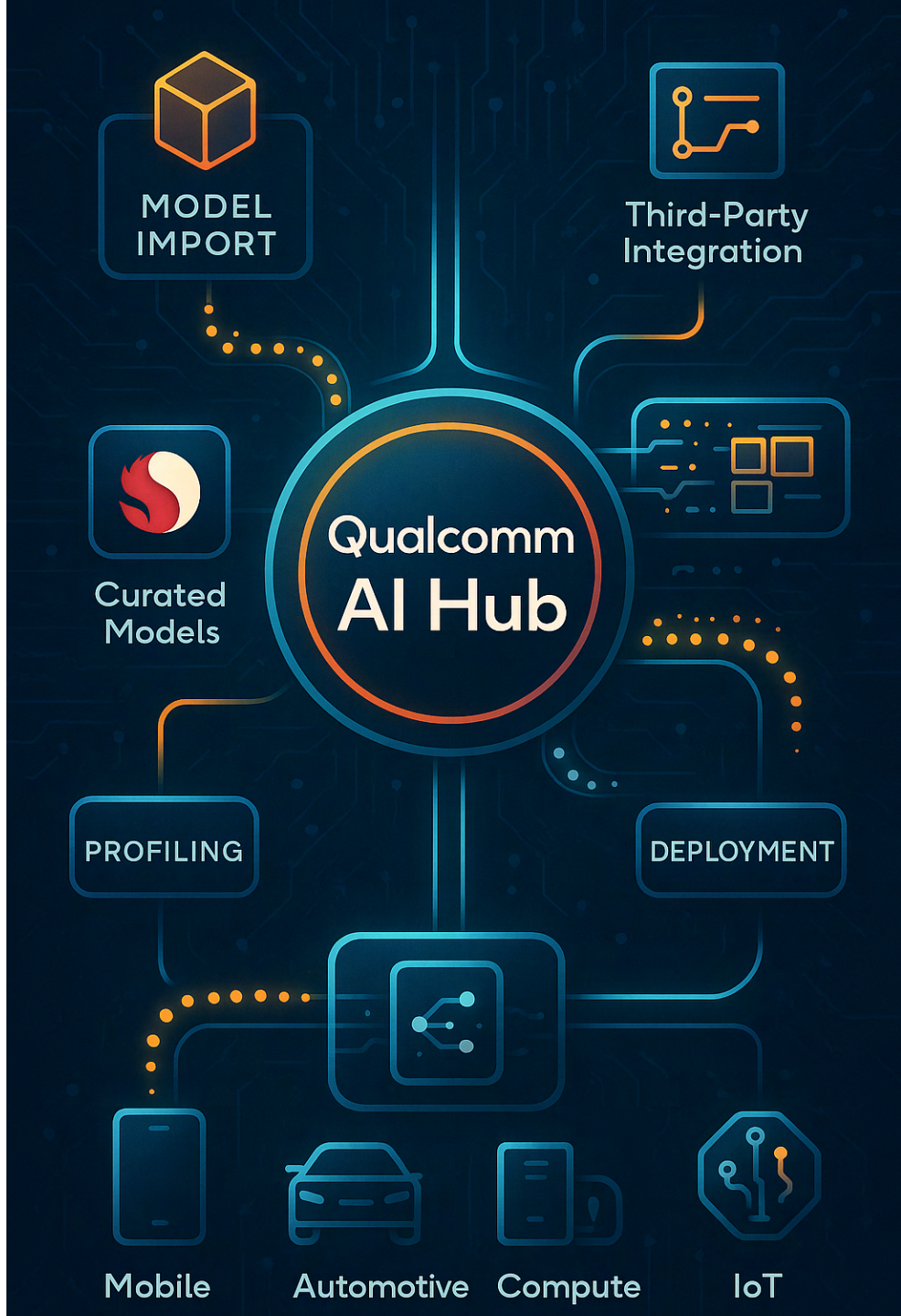
Get comprehensive on-device metrics including runtime, load time, and compute unit utilization.

Validation

Verify numerical correctness by comparing on-device inference outputs against reference model outputs to ensure fidelity.

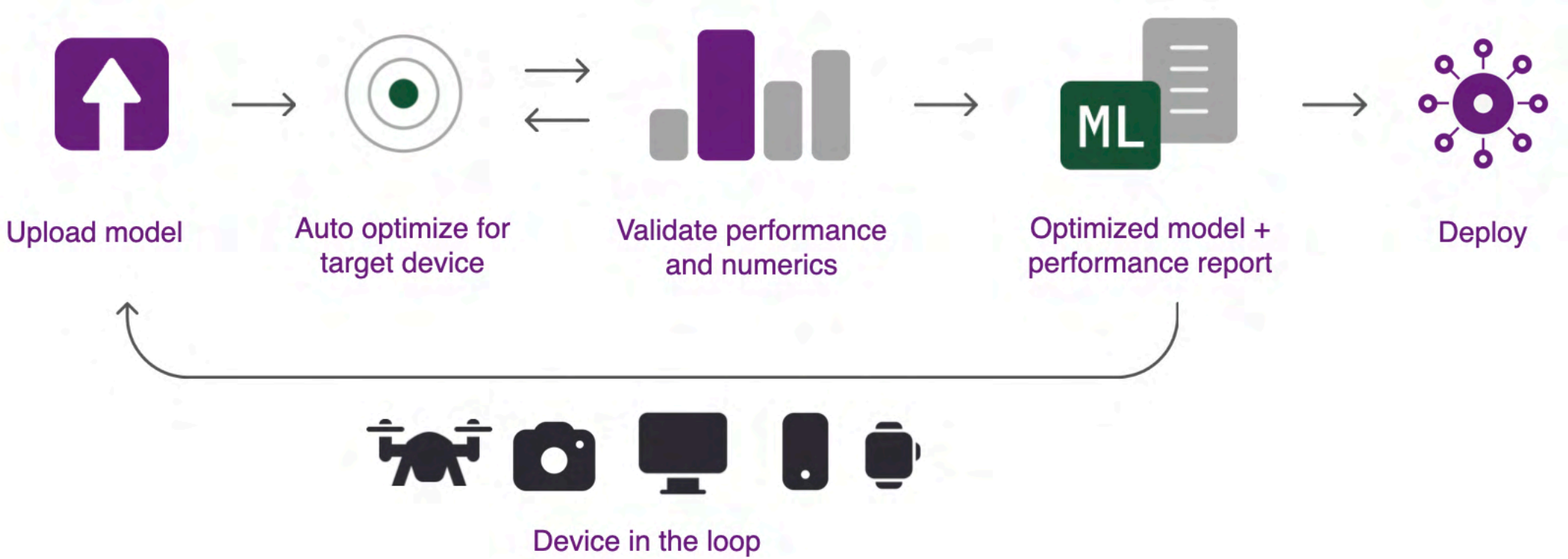
Flexible Deployment

Receive device-ready artifacts (e.g., DLC files and runtime config) and sample apps for easy integration into your Edge AI projects.



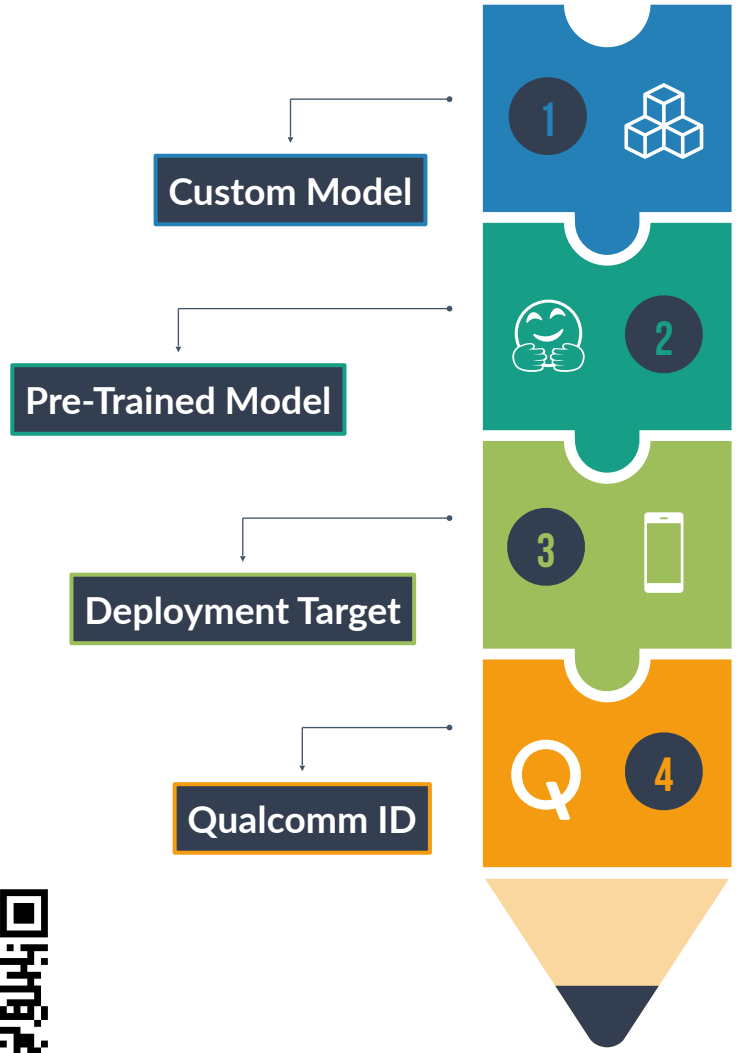
QUALCOMM AI HUB

How does it work?



QUALCOMM AI HUB

What do you need?



01
Custom Model
A trained model that can be in Pytorch, TFLite, or ONNX format.

02
Pre-Trained Model
Qualcomm also has several models available on GitHub and Hugging Face.

03
Deployment Target
This can be a specific device (FairPhone 5, Pixel 6) or a range of devices.

04
Qualcomm ID
An account on Qualcomm AI HUB.

QUALCOMM AI HUB

Installation

QAI-Hub Setup

Configure QAI-Hub:

```
$ qai-hub configure --api_token <API_TOKEN>
```

Qualcomm Account ID

Access the Qualcomm API Token:

```
Your ID -> Settings ->API Token
```

Qualcomm Sign In

Sign in to Qualcomm account:

```
https://app.aihub.qualcomm.com
```

Install QAI-Hub

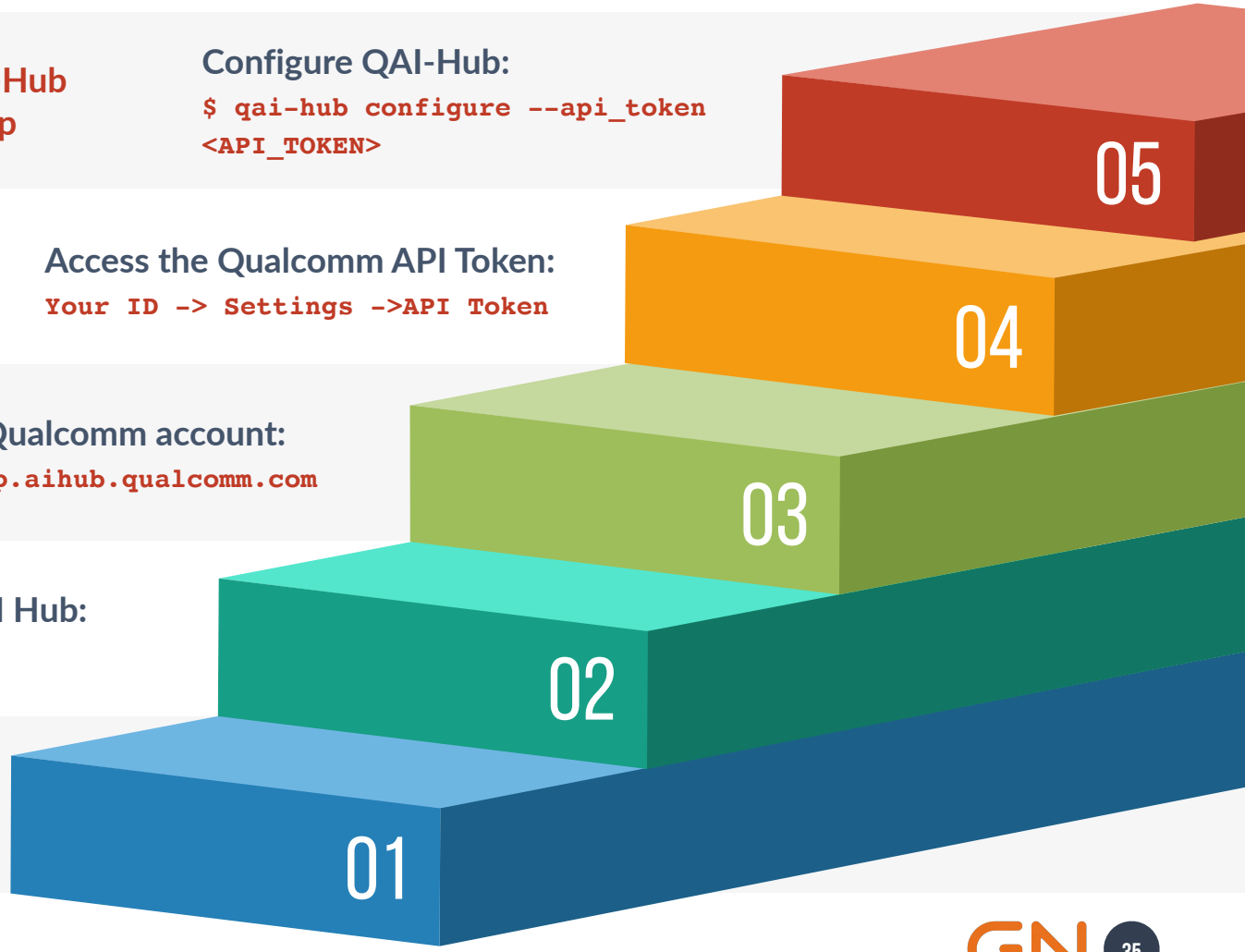
Install Python API for AI Hub:

```
$ pip install qai-hub
```

Python Environment

Create a Python Environment:

```
$ conda create -n qaihub python=3.10 -y
```



CHECK AVAILABLE DEVICES

List of Devices

> Choose based on device type: Automotive, IOT, XR, Windows, or Mobile.

CLI

Type this command on your Terminal:

```
$ qai-hub list-devices
```

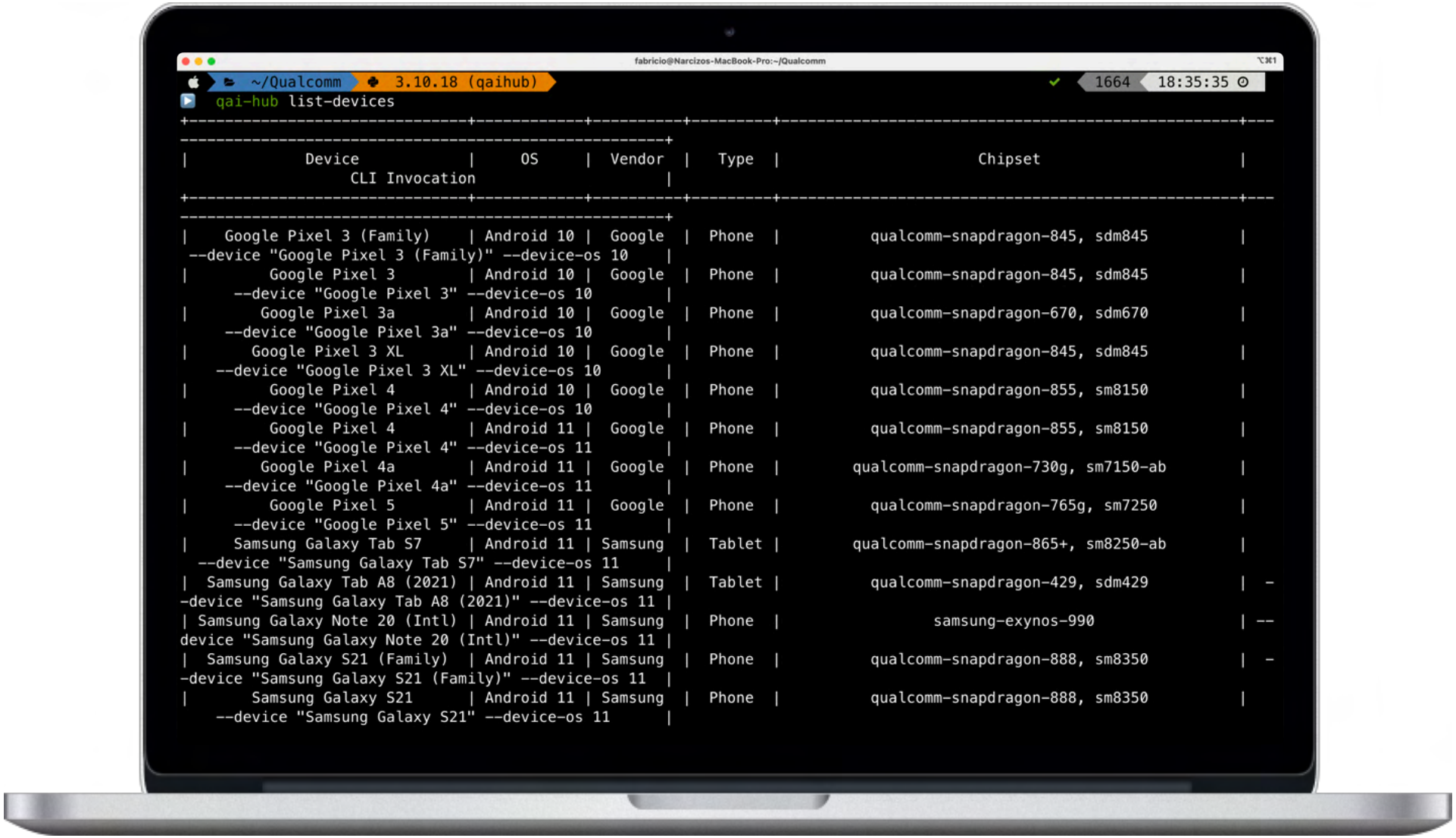
WEB

Access the following website on your browser:

`https://app.aihub.qualcomm.com/devices`

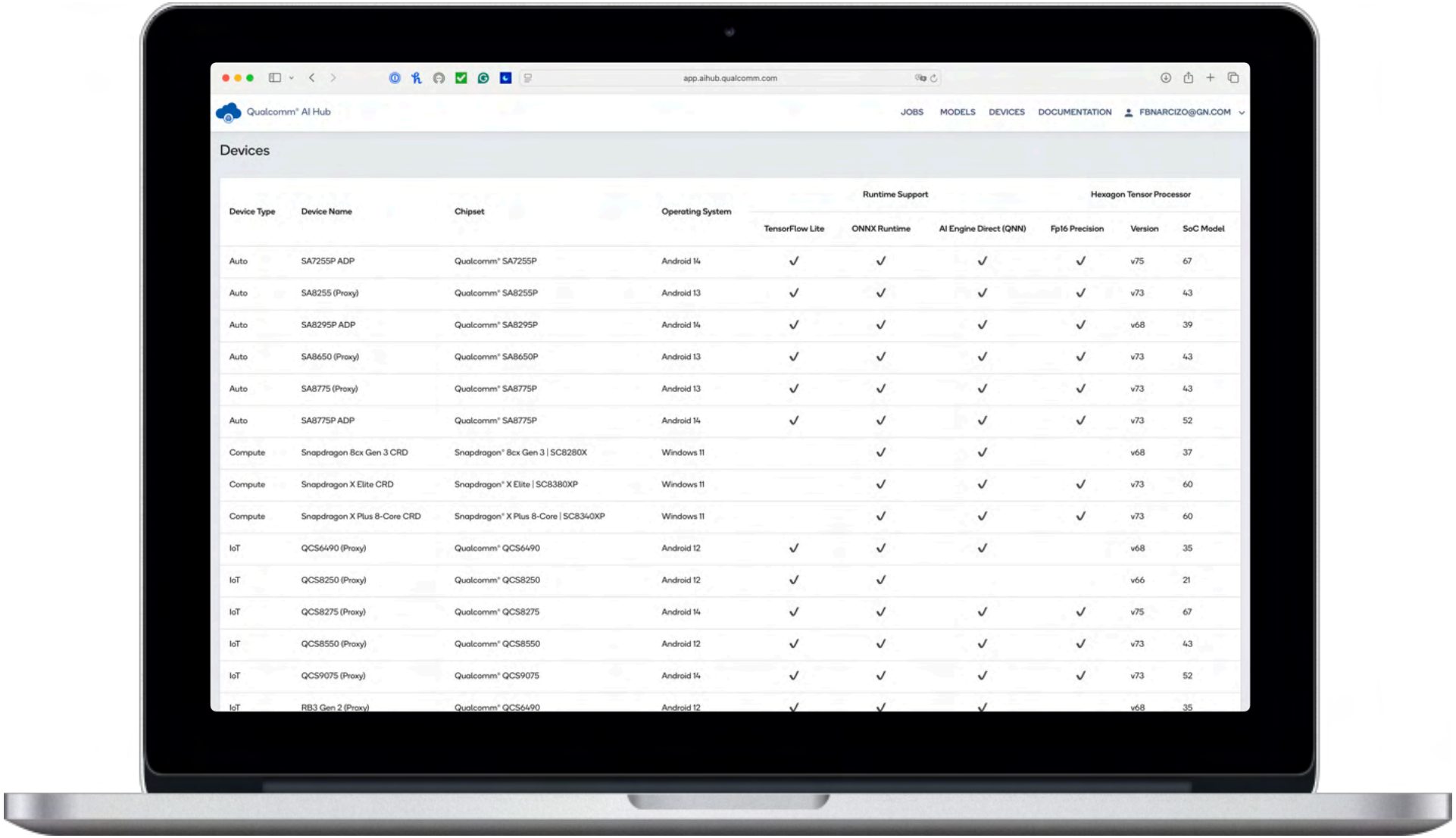
CHECK AVAILABLE DEVICES

Terminal



CHECK AVAILABLE DEVICES

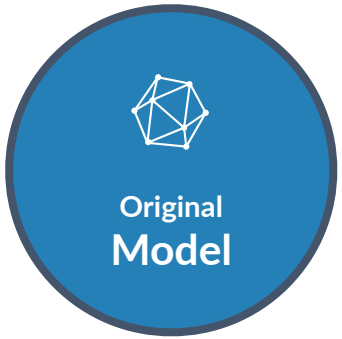
Qualcomm AI Hub



Device Type	Device Name	Chipset	Operating System	Runtime Support			Hexagon Tensor Processor		
				TensorFlow Lite	ONNX Runtime	AI Engine Direct (QNN)	Fp16 Precision	Version	SoC Model
Auto	SA7255P ADP	Qualcomm® SA7255P	Android 14	✓	✓	✓	✓	v75	67
Auto	SA8255 (Proxy)	Qualcomm® SA8255P	Android 13	✓	✓	✓	✓	v73	43
Auto	SA8295P ADP	Qualcomm® SA8295P	Android 14	✓	✓	✓	✓	v68	39
Auto	SA8650 (Proxy)	Qualcomm® SA8650P	Android 13	✓	✓	✓	✓	v73	43
Auto	SA8775 (Proxy)	Qualcomm® SA8775P	Android 13	✓	✓	✓	✓	v73	43
Auto	SA8775P ADP	Qualcomm® SA8775P	Android 14	✓	✓	✓	✓	v73	52
Compute	Snapdragon 8cx Gen 3 CRD	Snapdragon® 8cx Gen 3 | SC8280X	Windows 11		✓	✓		v68	37
Compute	Snapdragon X Elite CRD	Snapdragon® X Elite | SC8380XP	Windows 11		✓	✓	✓	v73	60
Compute	Snapdragon X Plus 8-Core CRD	Snapdragon® X Plus 8-Core | SC8340XP	Windows 11		✓	✓	✓	v73	60
IoT	QCS6490 (Proxy)	Qualcomm® QCS6490	Android 12	✓	✓	✓		v68	35
IoT	QCS8250 (Proxy)	Qualcomm® QCS8250	Android 12	✓	✓			v66	21
IoT	QCS8275 (Proxy)	Qualcomm® QCS8275	Android 14	✓	✓	✓	✓	v75	67
IoT	QCS8550 (Proxy)	Qualcomm® QCS8550	Android 12	✓	✓	✓	✓	v73	43
IoT	QCS9075 (Proxy)	Qualcomm® QCS9075	Android 14	✓	✓	✓	✓	v73	52
IoT	RB3 Gen 2 (Proxy)	Qualcomm® QCS6490	Android 12	✓	✓	✓		v68	35

COMPILE JOB

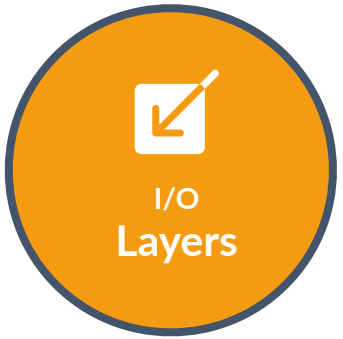
Transform and Optimize



Input model from PyTorch, ONNX, and TensorFlow in FP32.

Convert the original model to Qualcomm format.

Quantize the Qualcomm model from FP32 to INT8.

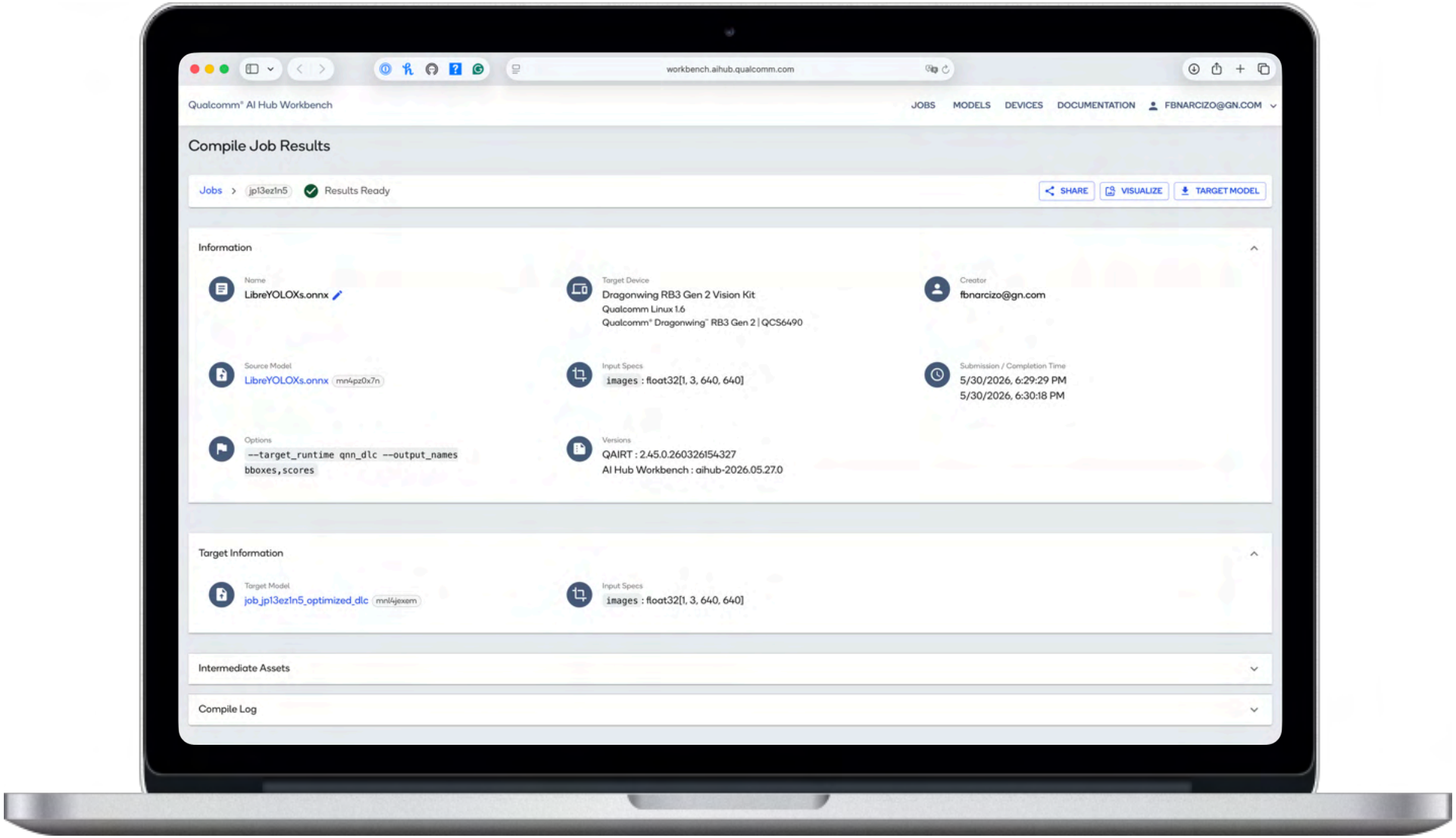


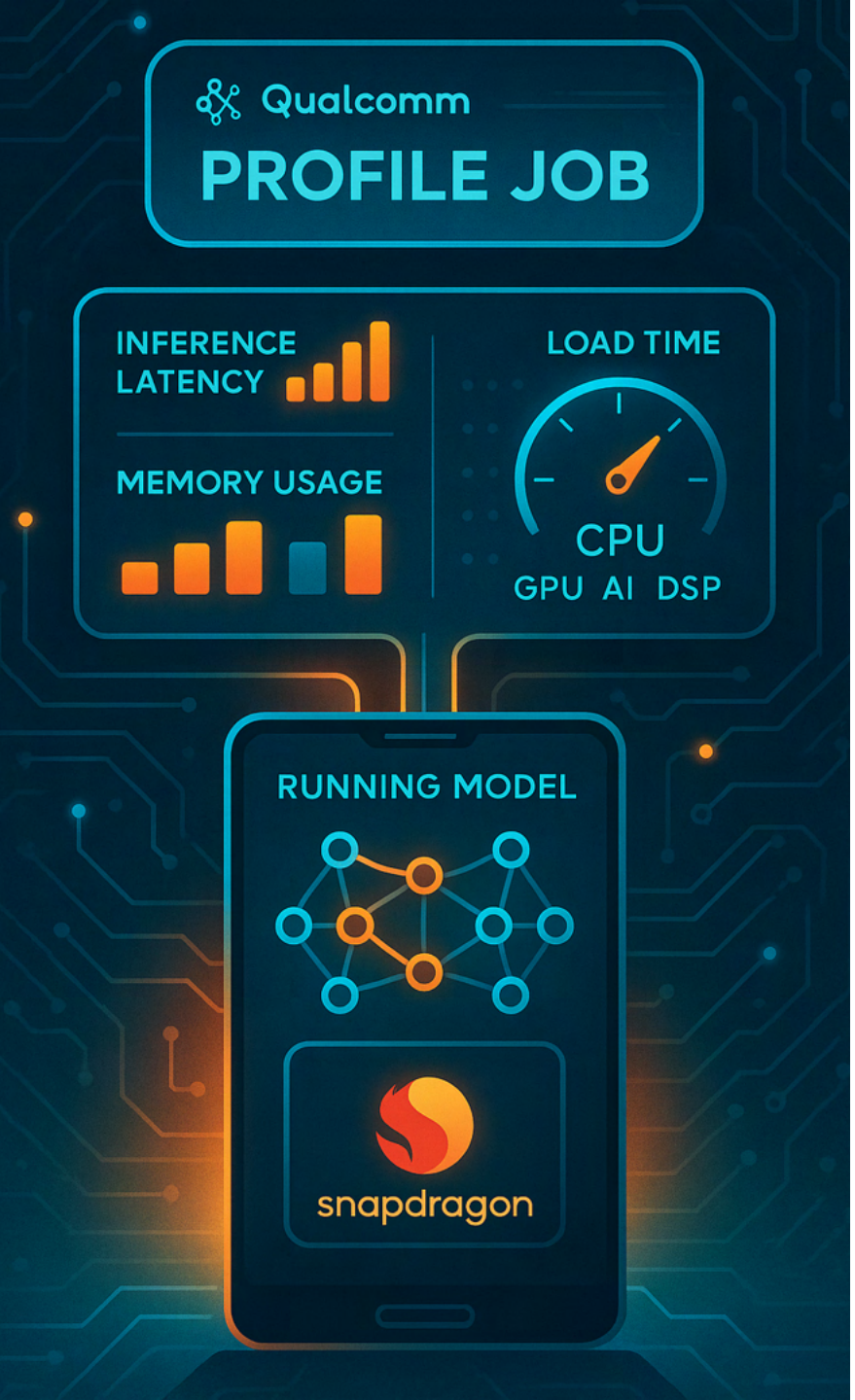
Quantize the input and output layers from the INT8 model.

Directly give names for outputs required for InferSNPE App.

COMPILE JOB

LibreYOLOXs Model





PROFILE JOB

Measure Performance

The Profile Job feature in Qualcomm AI Hub enables detailed performance benchmarking of AI models on actual Snapdragon hardware. It provides insights into how efficiently a model runs, revealing critical deployment metrics.

Inference Latency

Measures how long it takes for the model to produce outputs once inputs are received—critical for real-time applications.

Load Time

Reports how long it takes to initialize and load the model into memory, including compilation overhead if applicable.

Memory Footprint

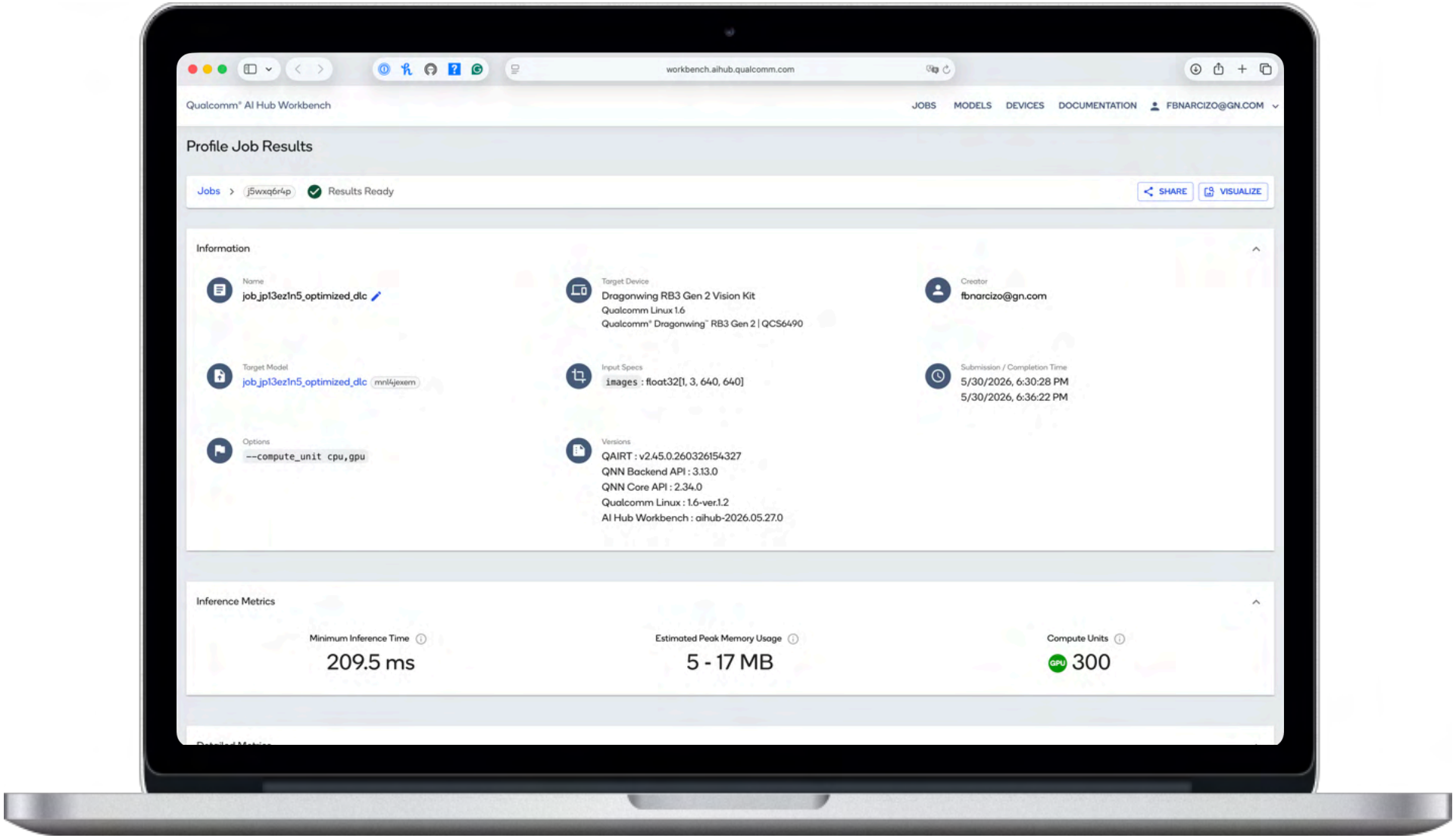
Indicates the total memory consumed during inference, helping identify models too large for constrained devices.

Compute Breakdown

Displays the distribution of processing workload across CPU, GPU, and DSP, enabling better runtime allocation and performance tuning.

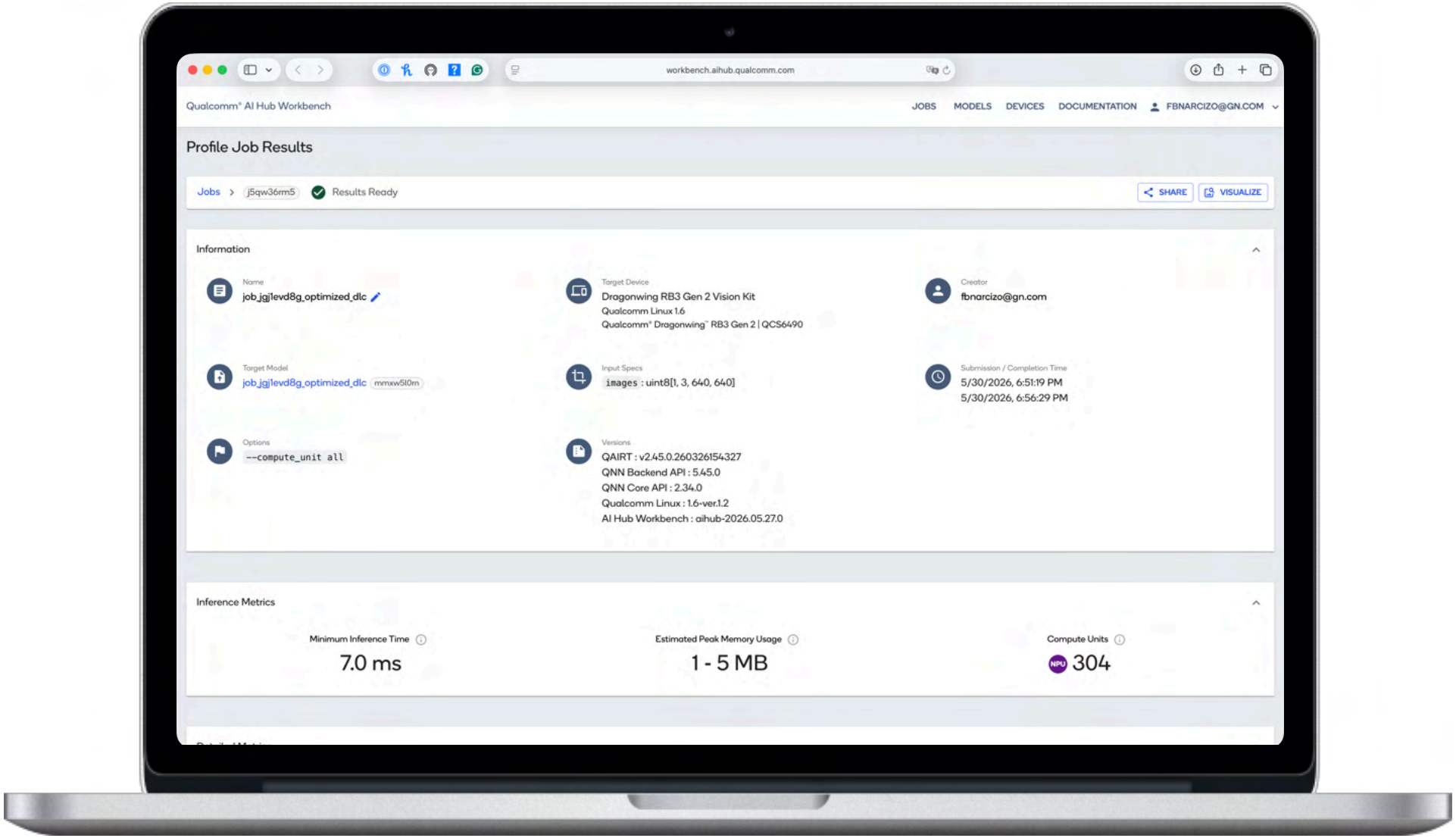
PROFILE ORIGINAL MODEL

Original LibreYOLOXs Model Inference



PROFILE QUANTIZED MODEL

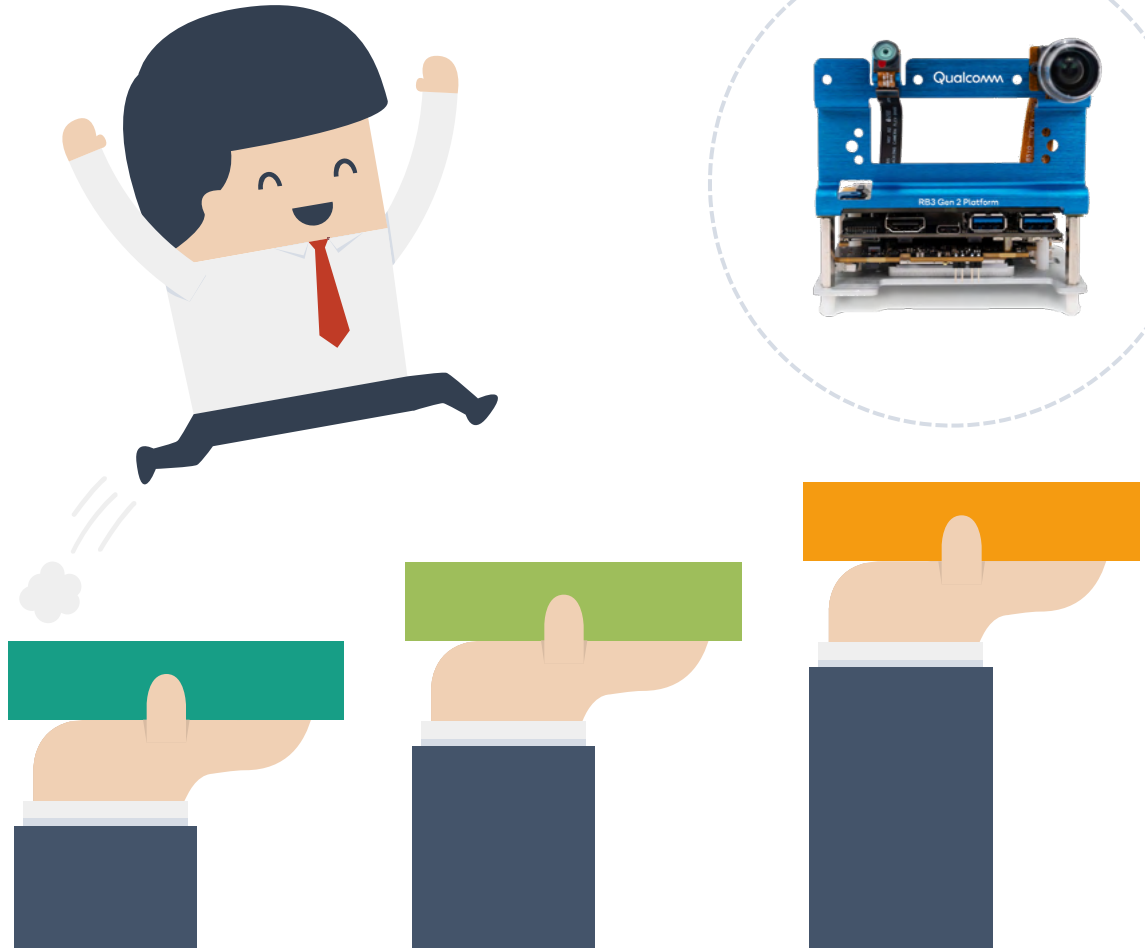
Quantized LibreYOLOXs Model Inference



MODEL DEPLOYMENT ON ANDROID

QUALCOMM RB3 DEV KIT

Overview



01 **Microsoft Windows**

Windows 11 IoT Enterprise LTSC 24H2 Evaluation.

02 **Linux / Ubuntu**

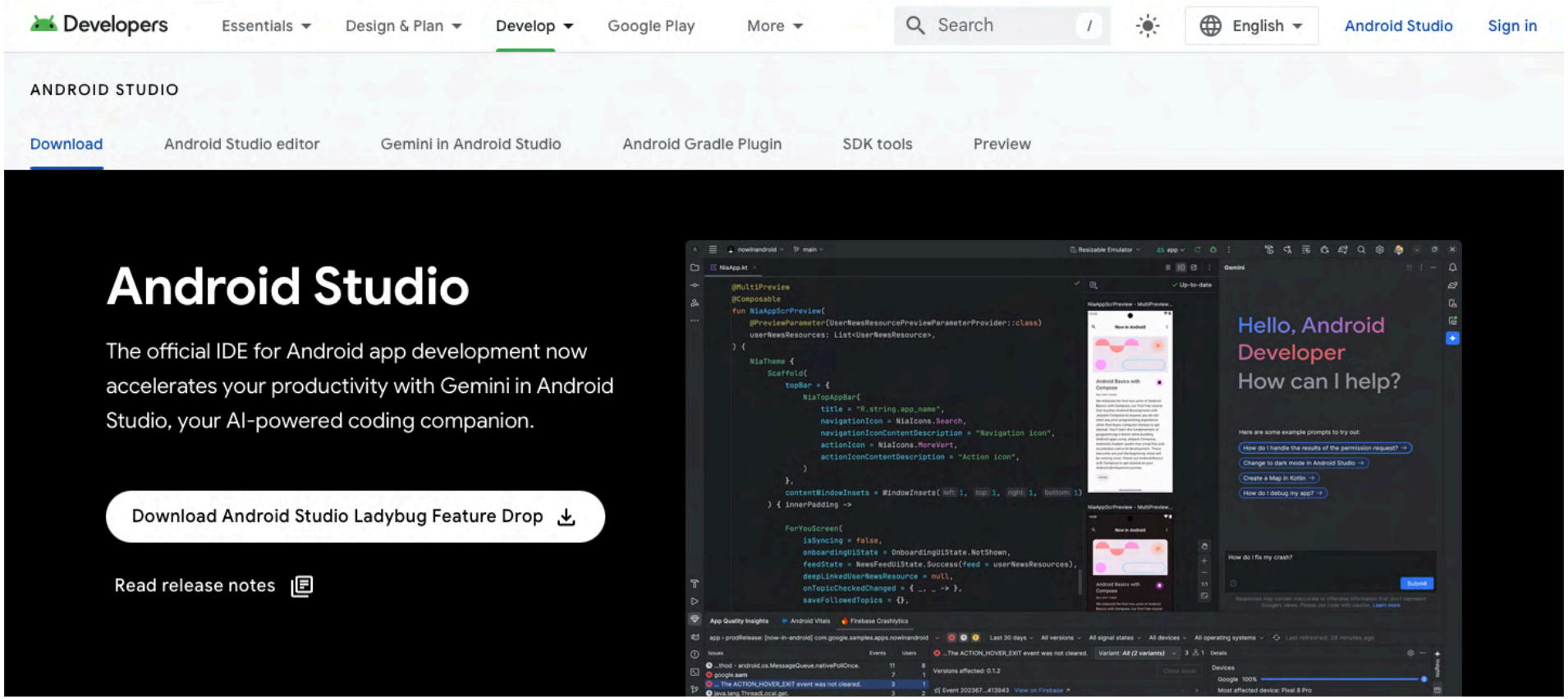
Two versions of Qualcomm Linux 1.x/2.x, and Ubuntu on Qualcomm IoT Platforms.

03 **Android**

We wrote a tutorial showing how to compile and deploy Android 15 on Qualcomm RB3 (*available on our repo*).

DEVELOPING ANDROID APPS

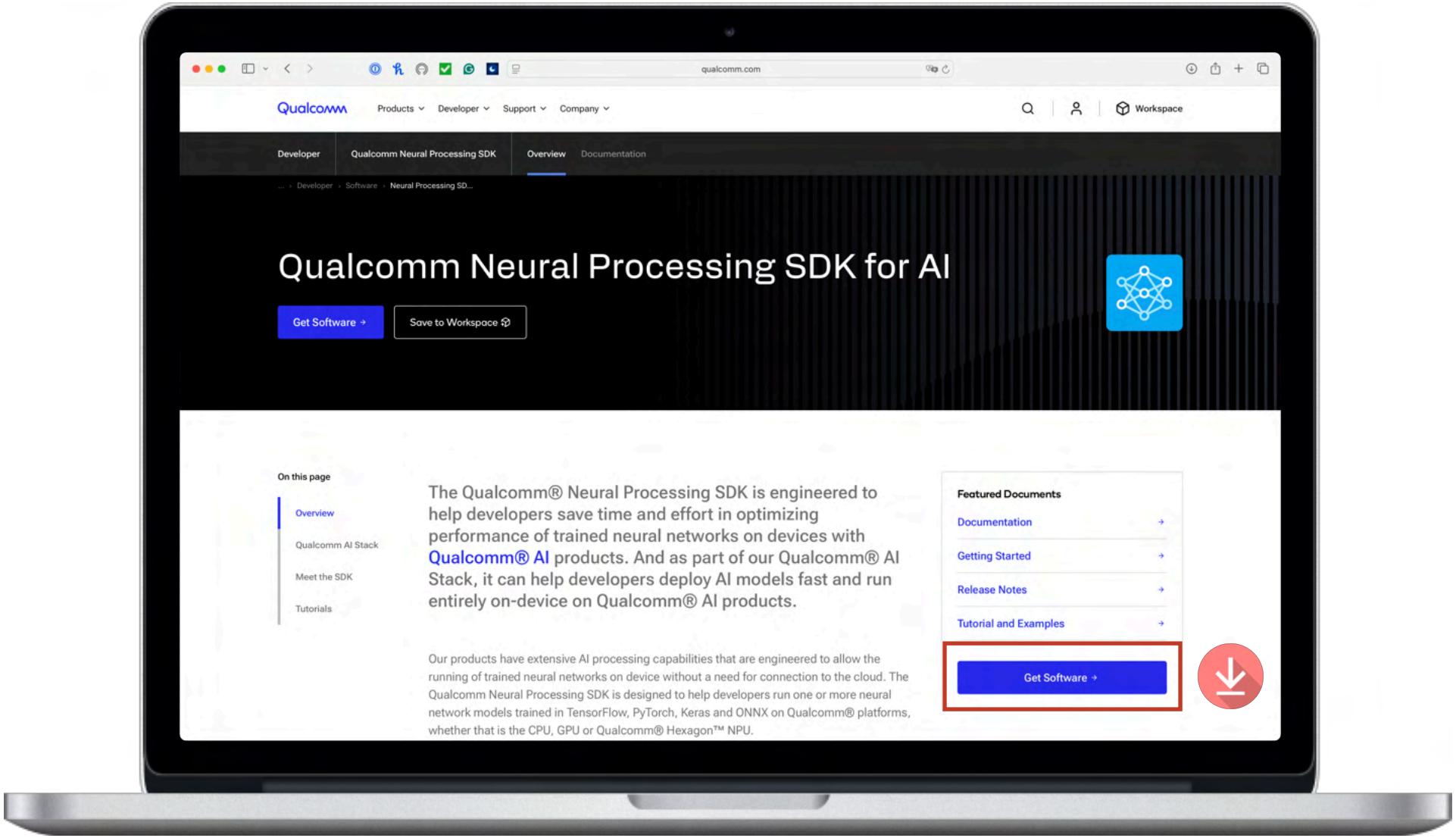
Android Studio



HTTPS://DEVELOPER.ANDROID.COM/STUDIO

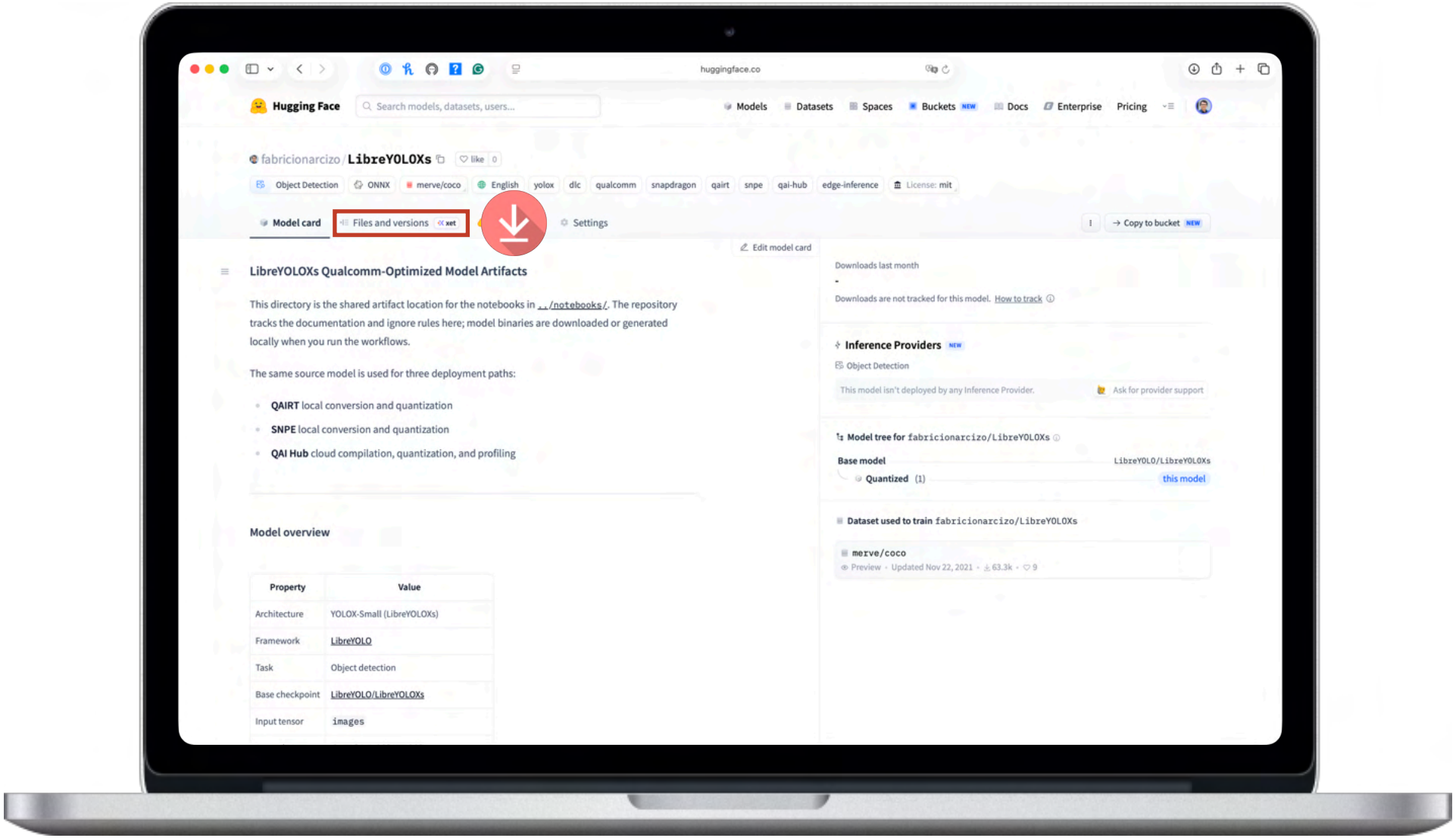
QUALCOMM NEURAL PROCESSING SDK FOR AI

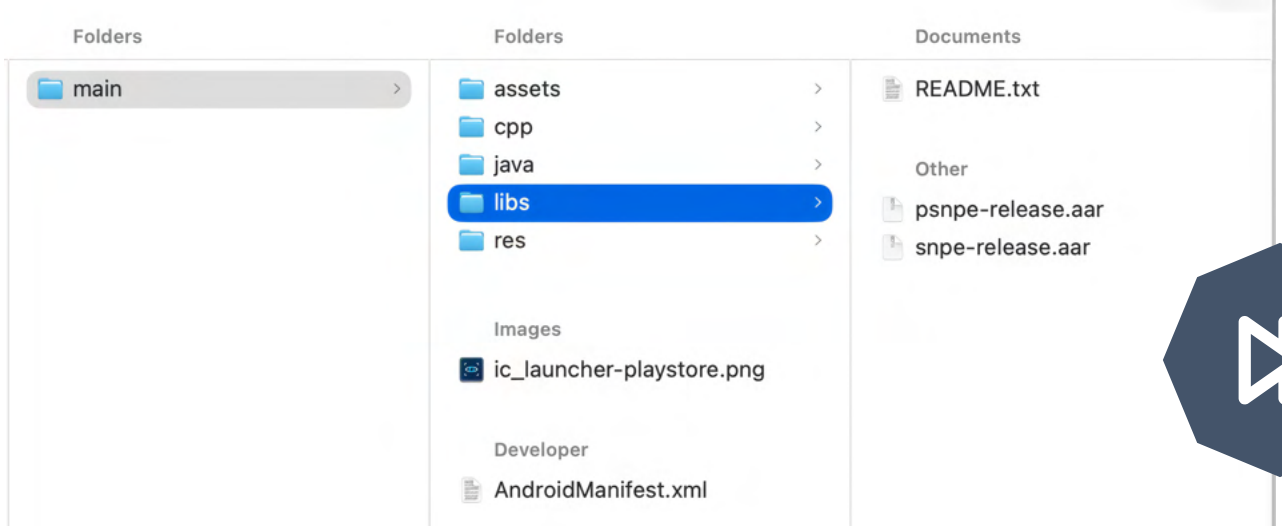
https://www.qualcomm.com/developer/software/neural-processing-sdk-for-ai



QUANTIZED LIBREYOLOXS MODELS

https://huggingface.co/fabricionarcizo/LibreYOLOXs



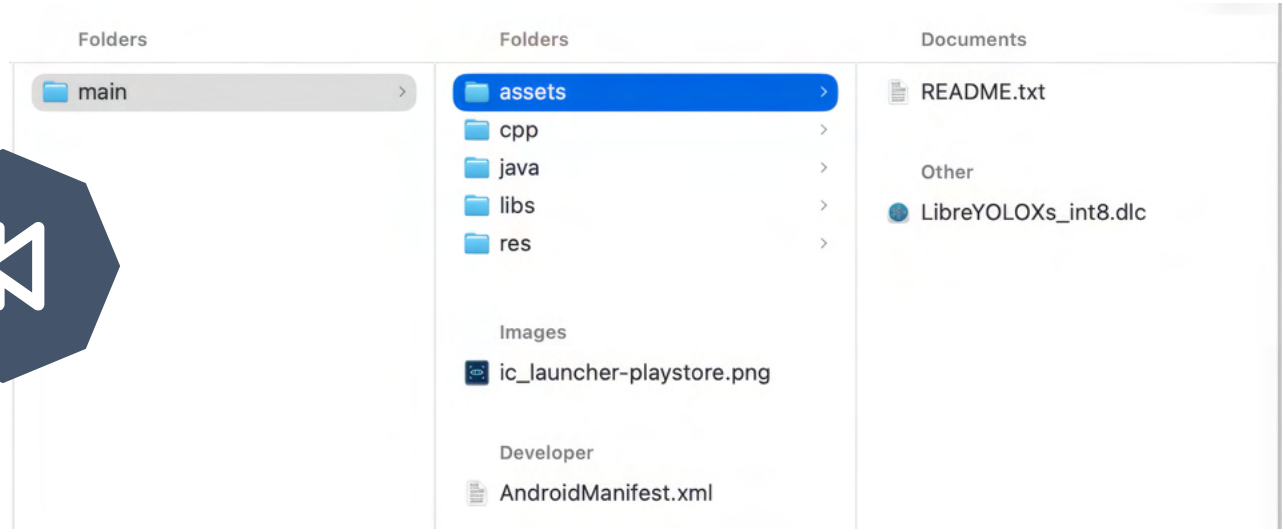


[P]SNPE-RELEASE.AAR (LIBS)

Android library package that includes all necessary SNPE binaries, Java interfaces, and native shared libraries for runtime execution.

DLC FILES (ASSETS)

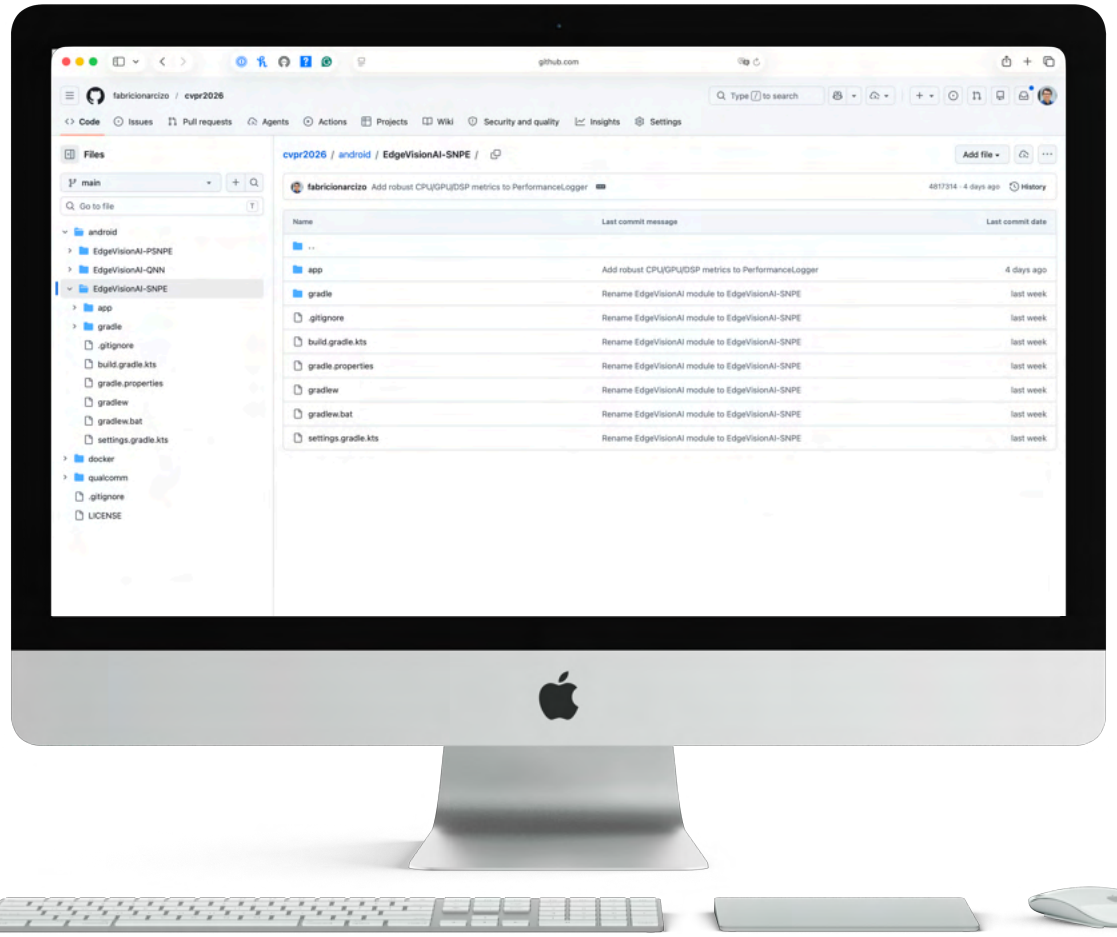
Hardware-optimized model files generated by SNPE tools from ONNX or TensorFlow sources; required for deployment on Qualcomm chipsets.



EDGEVISION AI APP

https://github.com/fabricionarcizo/cvpr2026/tree/main/android

SAMSUNG
Galaxy S25



EDGEVISION AI

App Versions

SNPE

A high-level Qualcomm SDK that simplifies deploying and running AI models across CPU, GPU, and DSP/HTP accelerators on Snapdragon devices.

PSNPE

Extend the standard SNPE workflow by enabling parallel and asynchronous inference execution, improving throughput and performance for multi-model or multi-stream AI workloads.

QNN

Interact with a lower-level runtime and APIs that provide finer control over graph execution, backend selection, and hardware optimization.

CUSTOM OVERLAY VIEW

Drawing Bounding Boxes

Dynamic Scaling

Maps bounding boxes from image to view coordinates while maintaining aspect ratio.

Front Camera Support

Flips boxes horizontally if using a front-facing camera.

Styled Drawing

Uses customizable Paint objects for boxes, text, and background.

Custom View

Draws detection results, such as bounding boxes and labels, over the camera preview or images.

Live Updates

The method updateDetections() triggers redraws with new detection results.

SNPEMODEL CLASS

Overview

SNPEModel is a custom utility class designed to simplify interaction with the SNPE runtime on Android. It also ensures consistency in how DLC models are handled across different app components.

Model Initialization

Loads the DLC file, configures runtime (CPU, GPU, or DSP), and sets up input/output layers.

Input Preprocessing

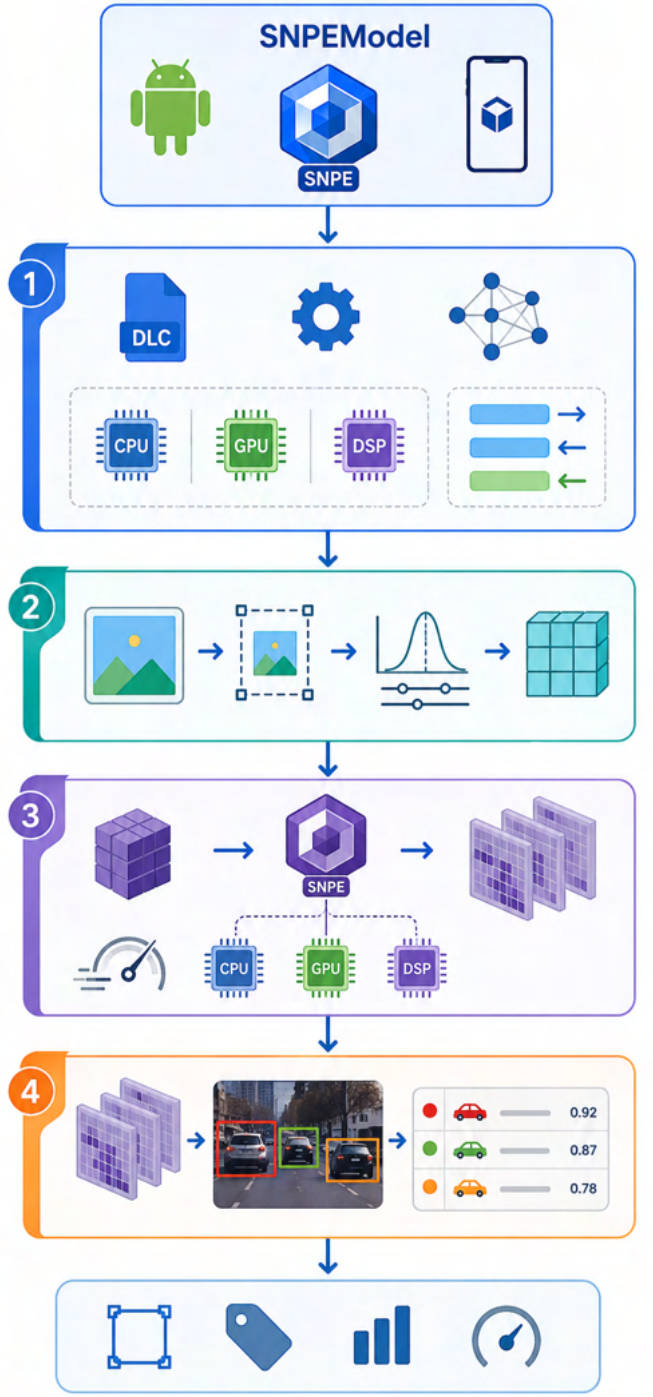
Handles resizing, normalization, and data formatting of images before feeding them to the model.

Inference Execution

Executes the model and retrieves raw output tensors in a hardware-optimized and asynchronous manner.

Output Parsing

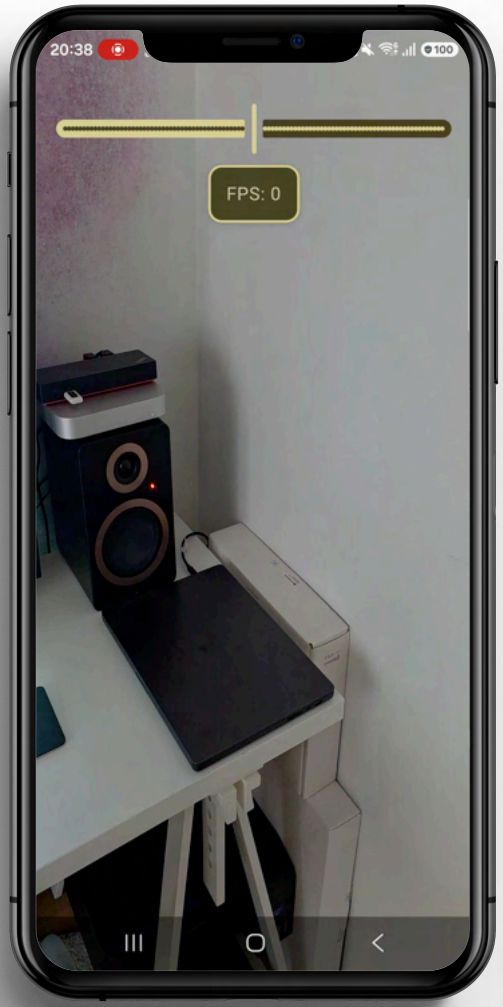
Interprets model outputs into usable objects like bounding boxes, labels, and confidence scores.



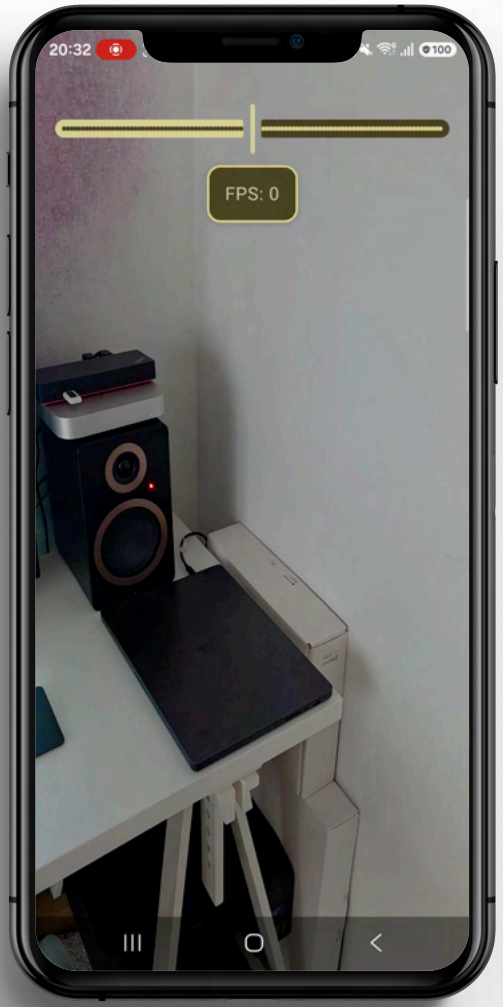
EDGEVISION AI

Benchmarking

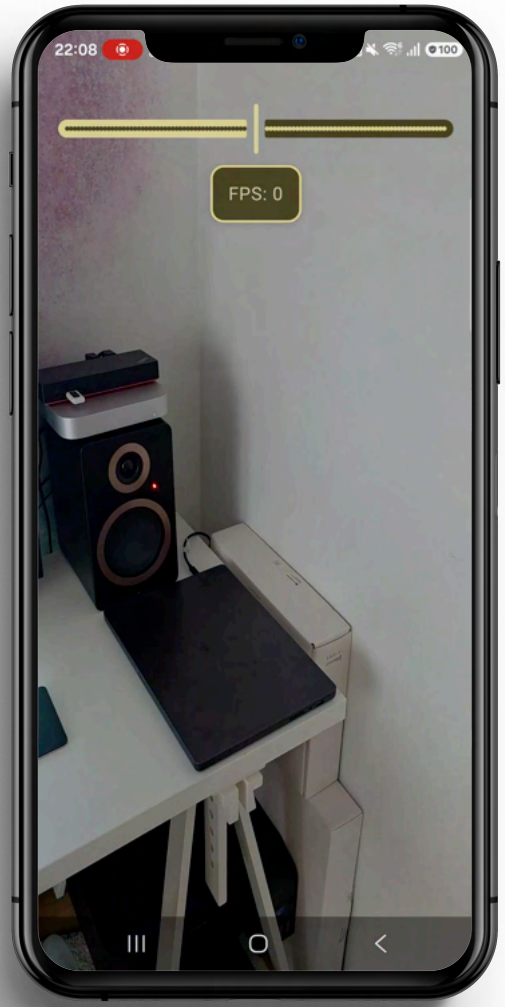
SNPE



PSNPE



QNN



QUANTIZED MODEL PROFILING

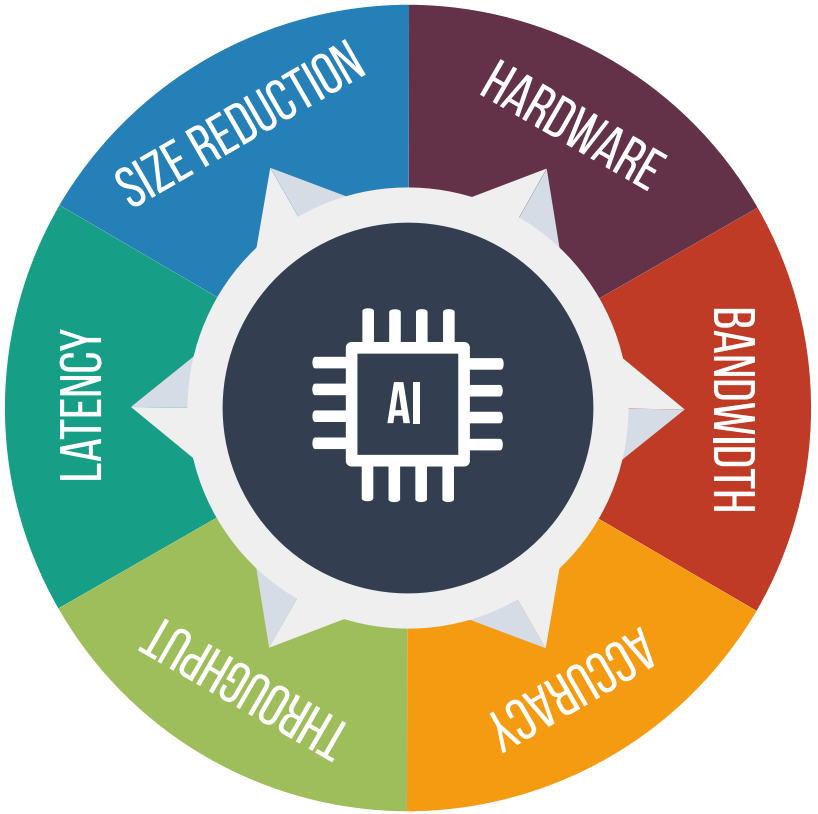
UNDERSTANDING KEY METRICS

Model Deployment

Size Reduction
Shrinking model dimensions to fit edge device constraints.

Latency
Timing the delay from input to decision output by the model.

Throughput
Assessing the number of inferences a model can process per unit time.



Hardware Compatibility
Evaluating the hardware is compatible to execute the quantized models

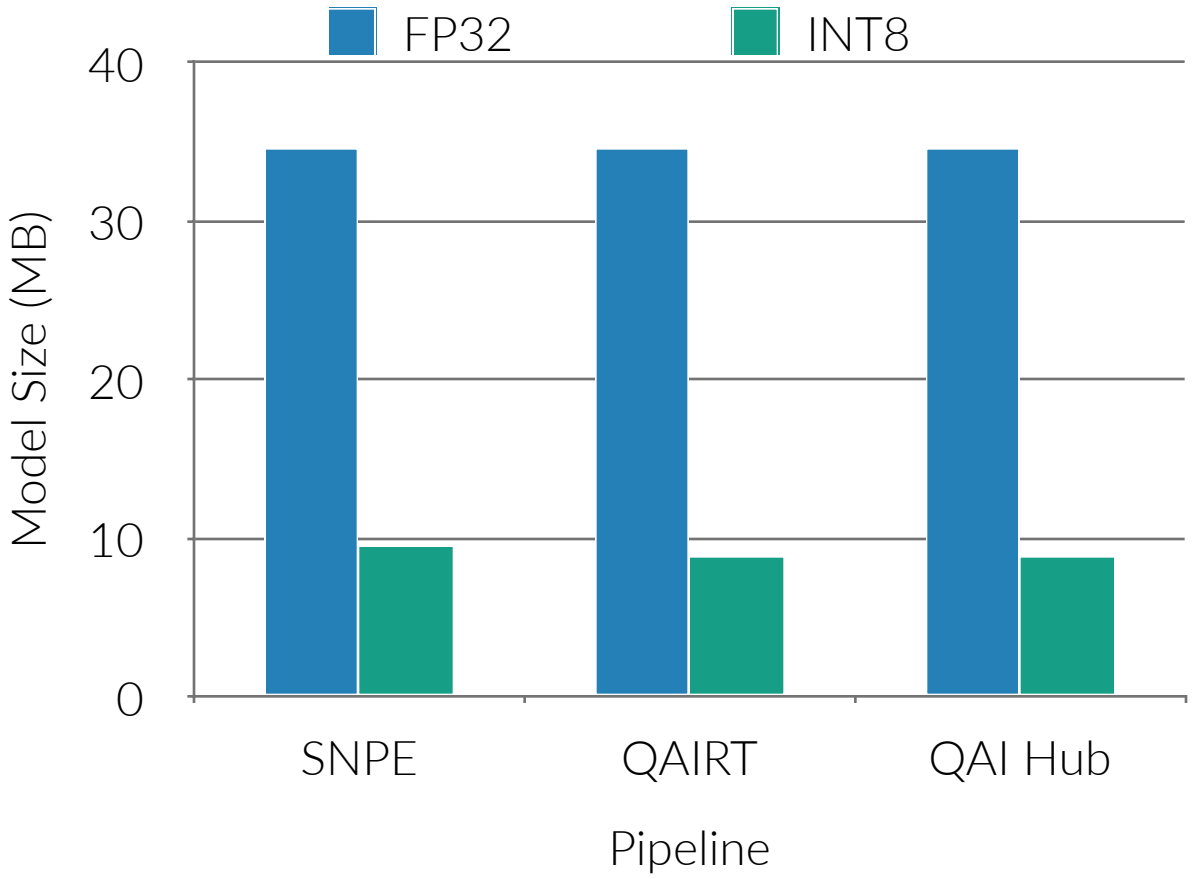
Memory Bandwidth
Gauging the rate at which data is transferred to and from the device's memory.

Accuracy
Measuring the model's correctness in predictions against real-world data.

QUANTIZATION IMPACT ON MODEL SIZE

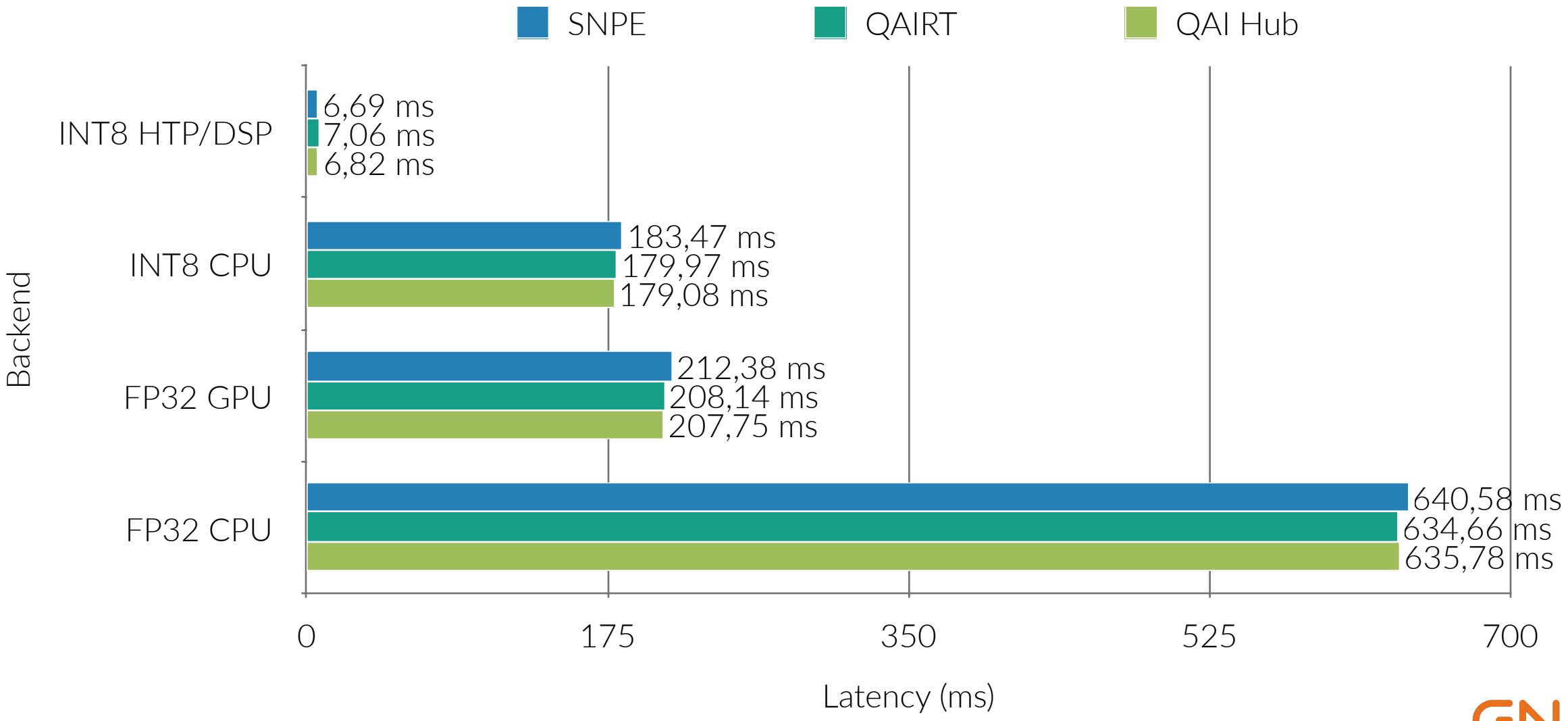
INT8 Quantization Reduced Model Size by ~4×

Pipeline	FP32 Size	INT8 Size	Compression
SNPE	34.51 MB	8.84 MB	3.63×
QAIRT	34.51 MB	8.84 MB	3.90×
QAI Hub	34.51 MB	9.52 MB	3.89×



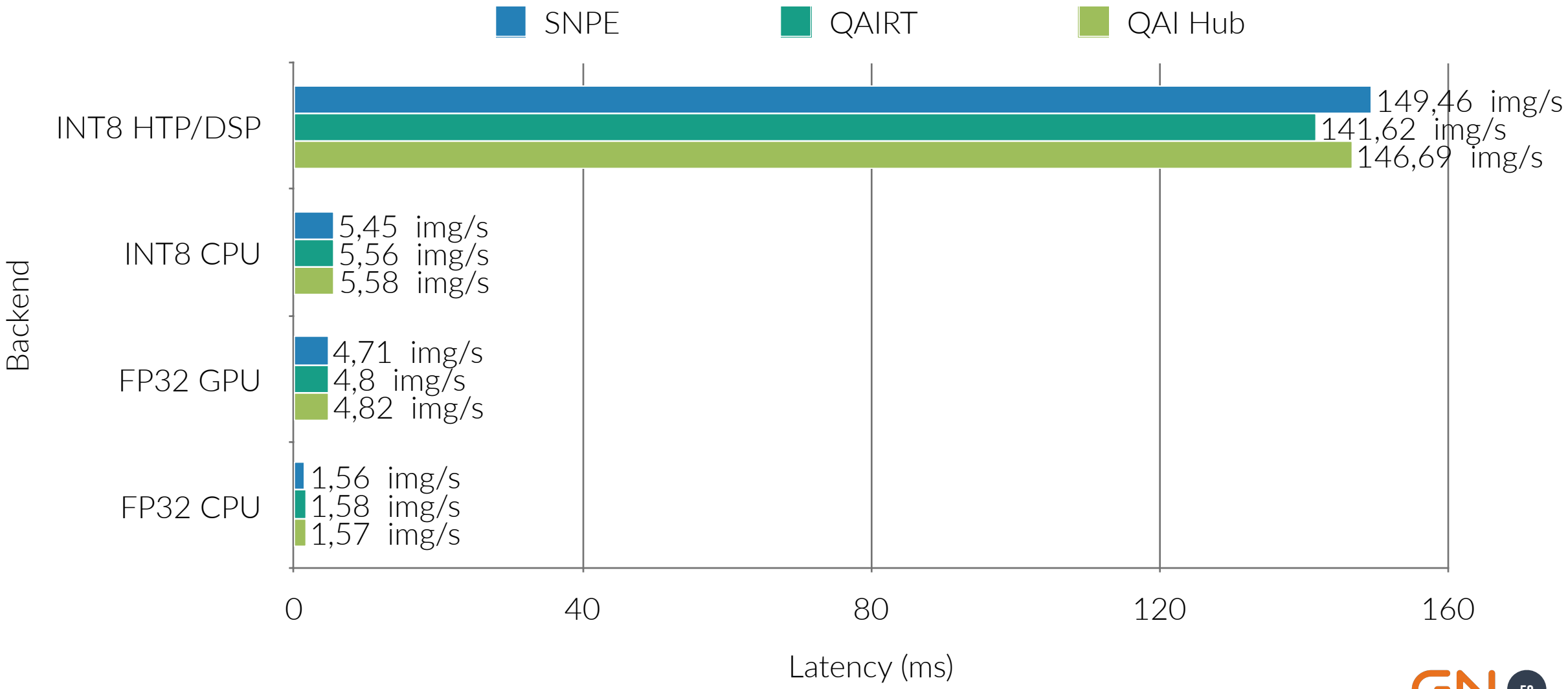
LATENCY ANALYSIS

HTP/DSP Acceleration Achieved Massive Latency Reduction



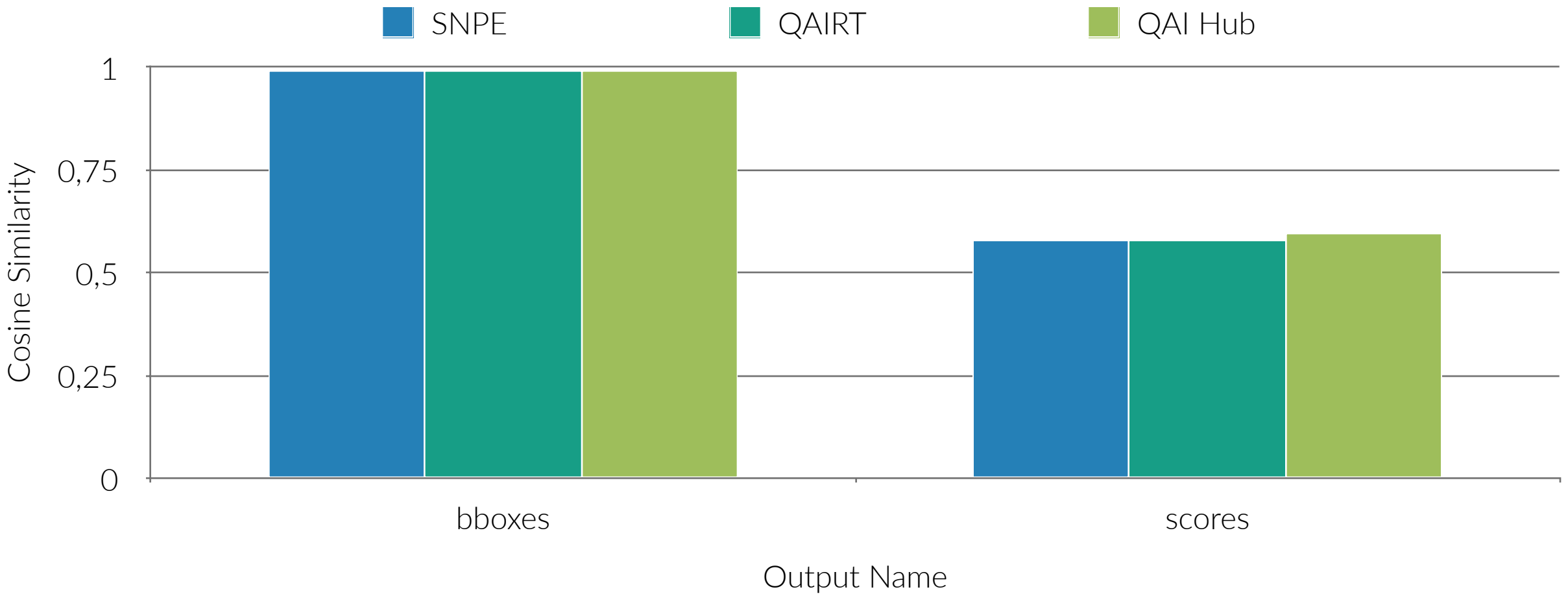
THROUGHPUT ANALYSIS

Real-Time Capable Deployment



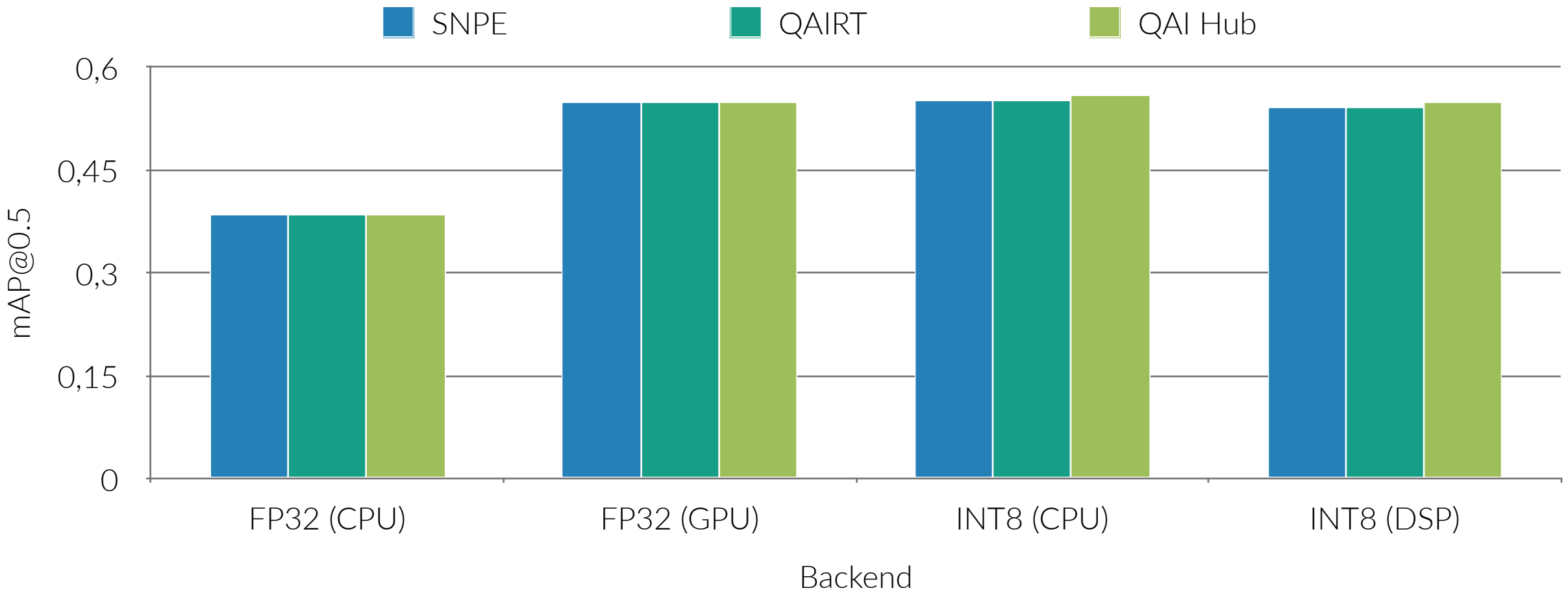
TENSOR SIMILARITY ANALYSIS

Numerical Similarity After Quantization



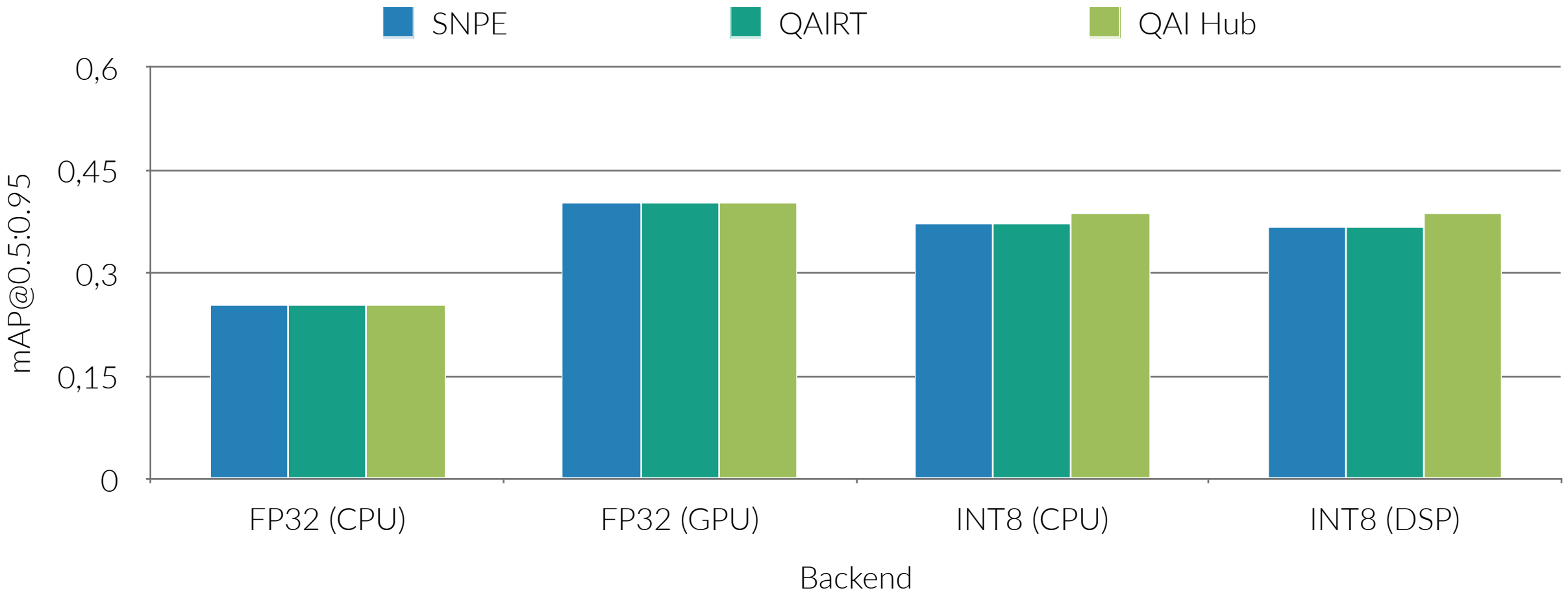
DETECTION ACCURACY RESULTS

Detection Accuracy Remained Highly Stable



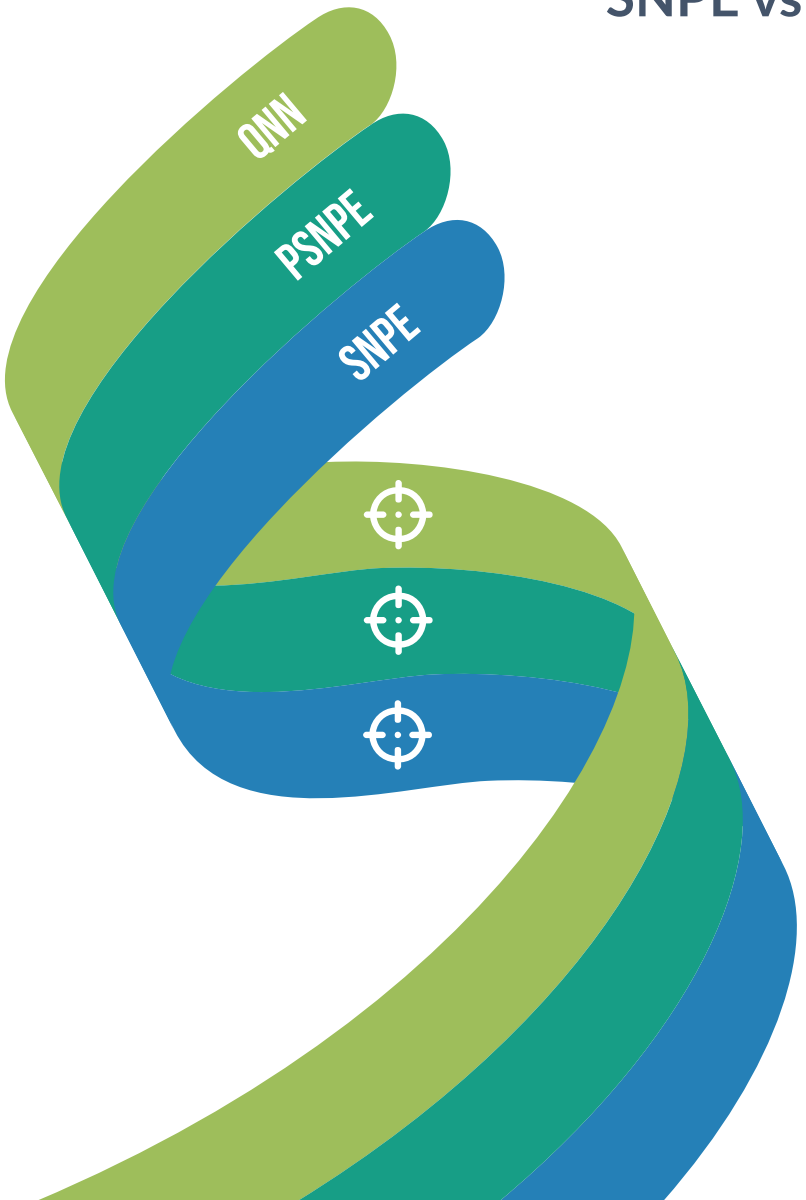
DETECTION ACCURACY RESULTS

Detection Accuracy Remained Highly Stable



RUNTIME FRAMEWORK COMPARISON

SNPE vs PSNPE vs QNN Runtime Analysis



Evaluate runtime behavior across Qualcomm inference frameworks using synchronized time-series profiling.

SNPE

Snapdragon Neural Processing Engine

PSNPE

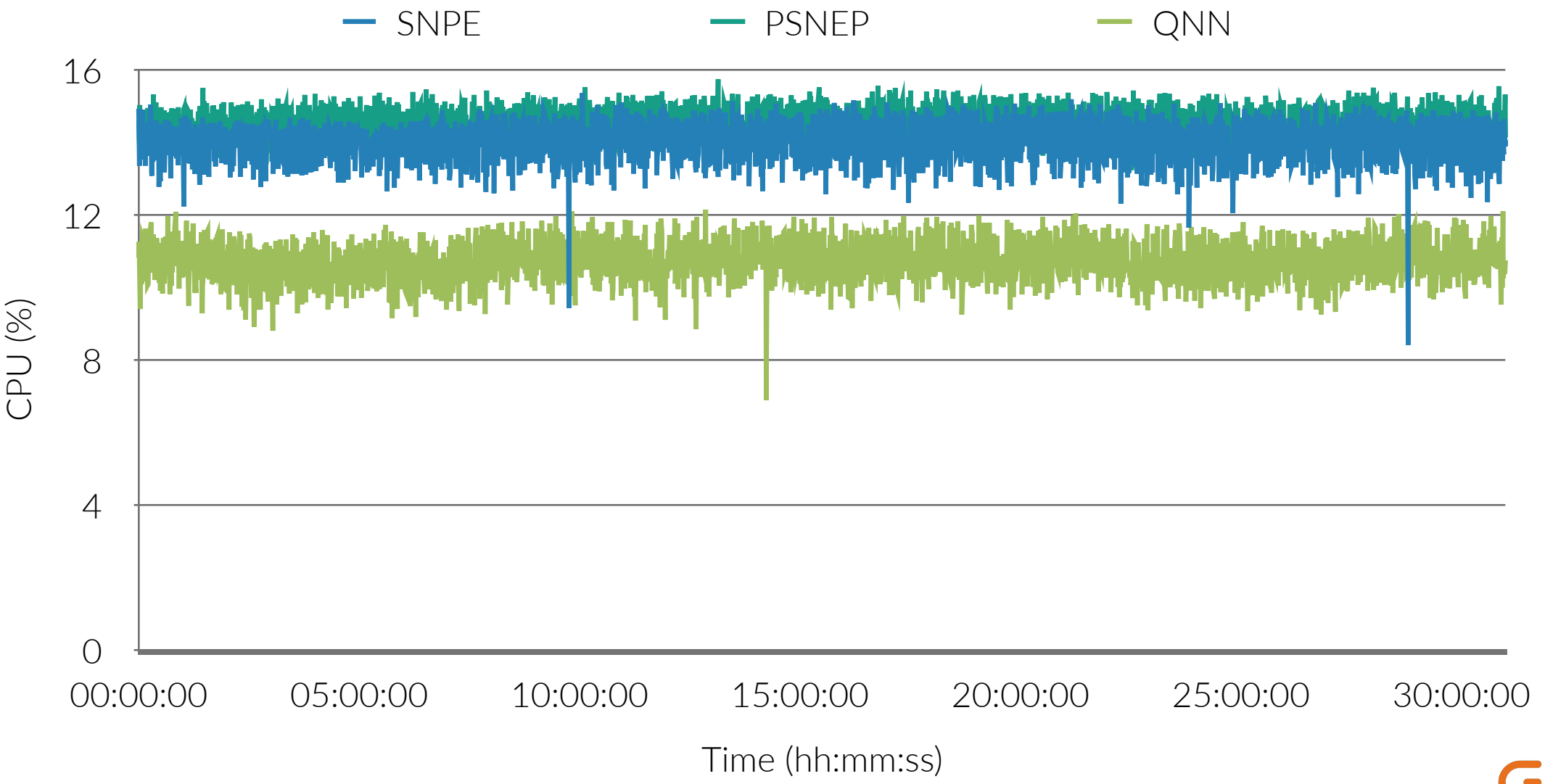
Parallel SNPE

QNN

Qualcomm Neural Network SDK

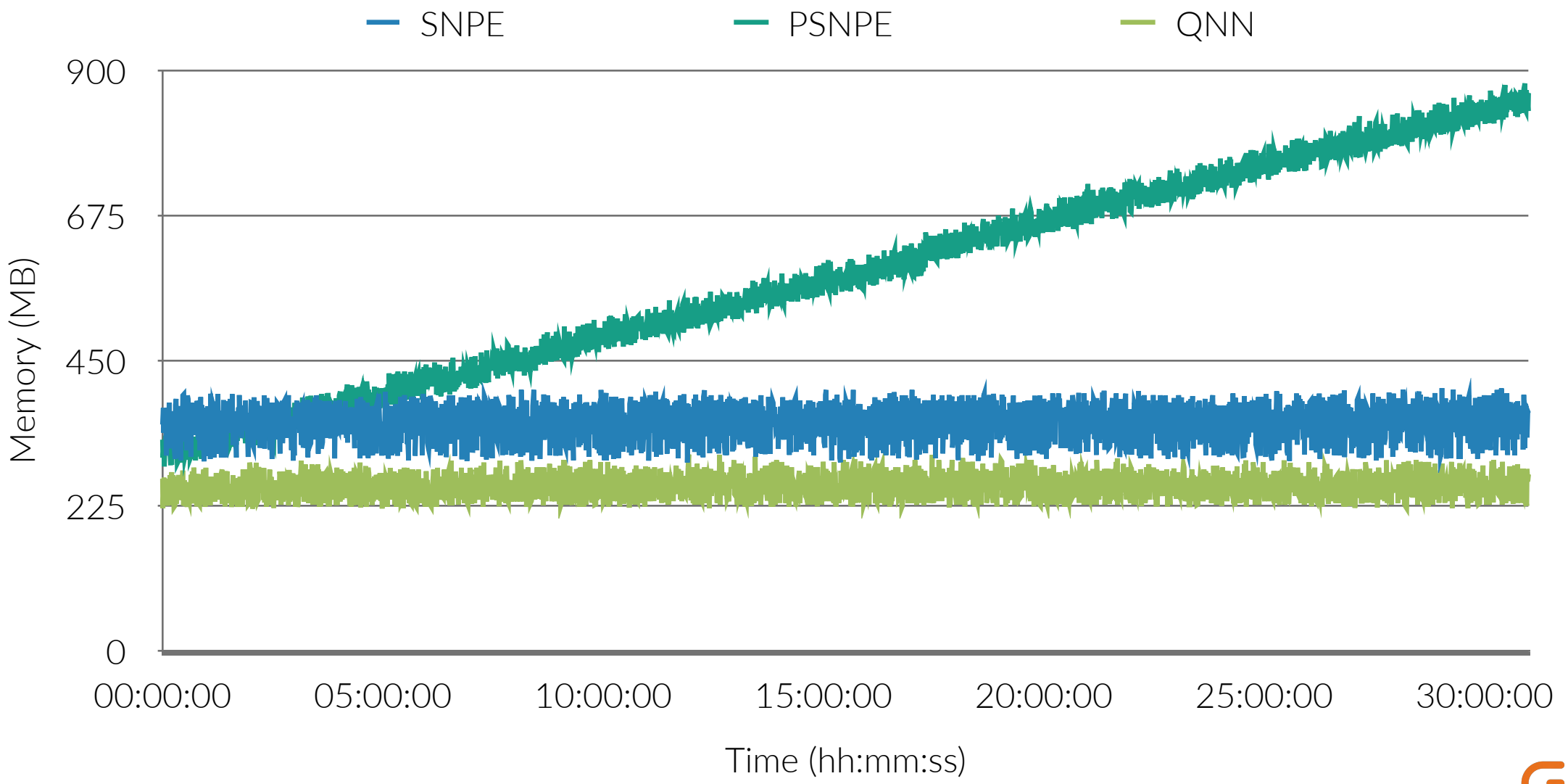
CPU UTILIZATION COMPARISON

CPU Usage Across Frameworks



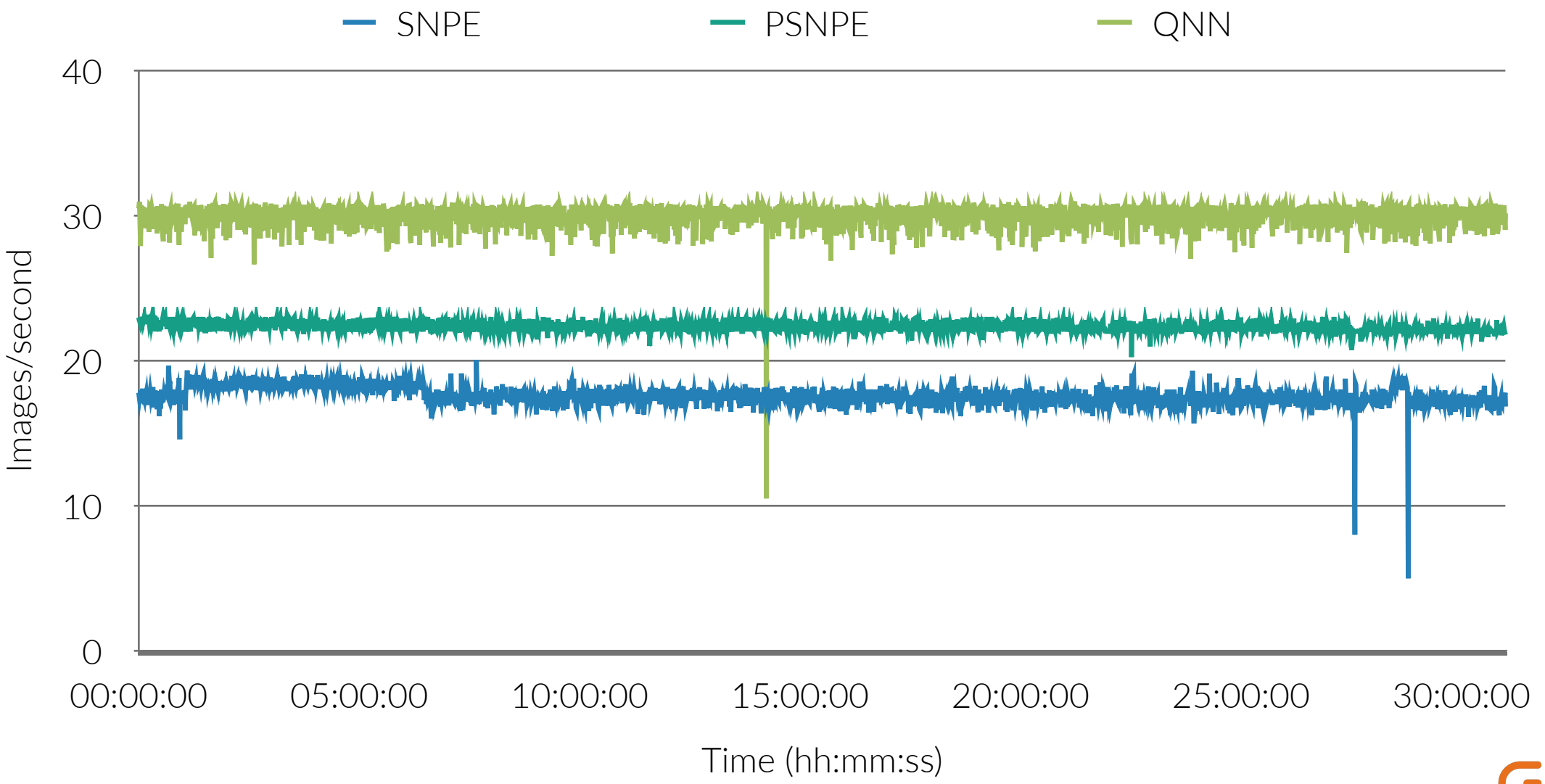
MEMORY CONSUMPTION COMPARISON

Memory Footprint Analysis



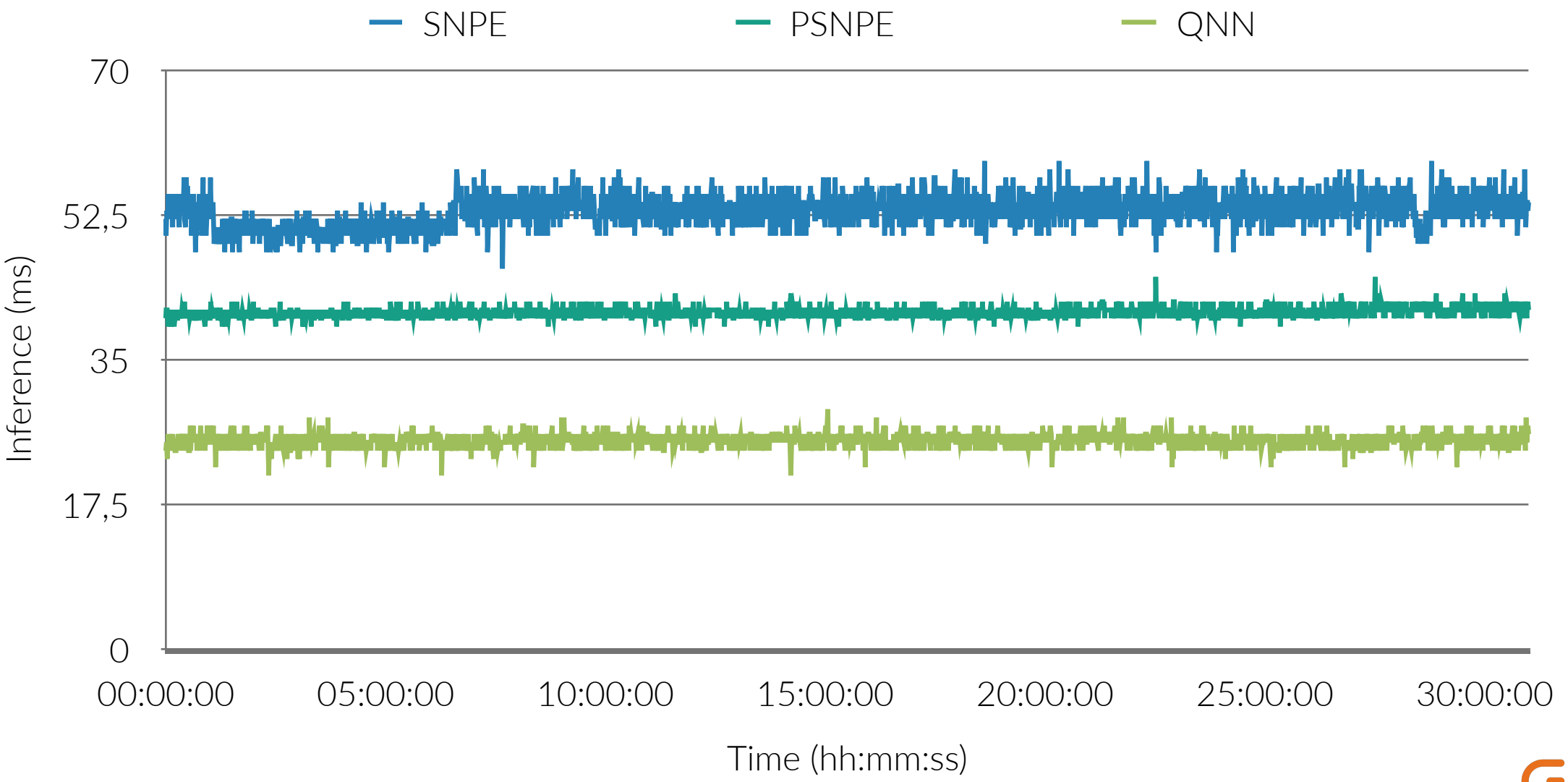
FPS COMPARISON

Throughput Stability Over Time



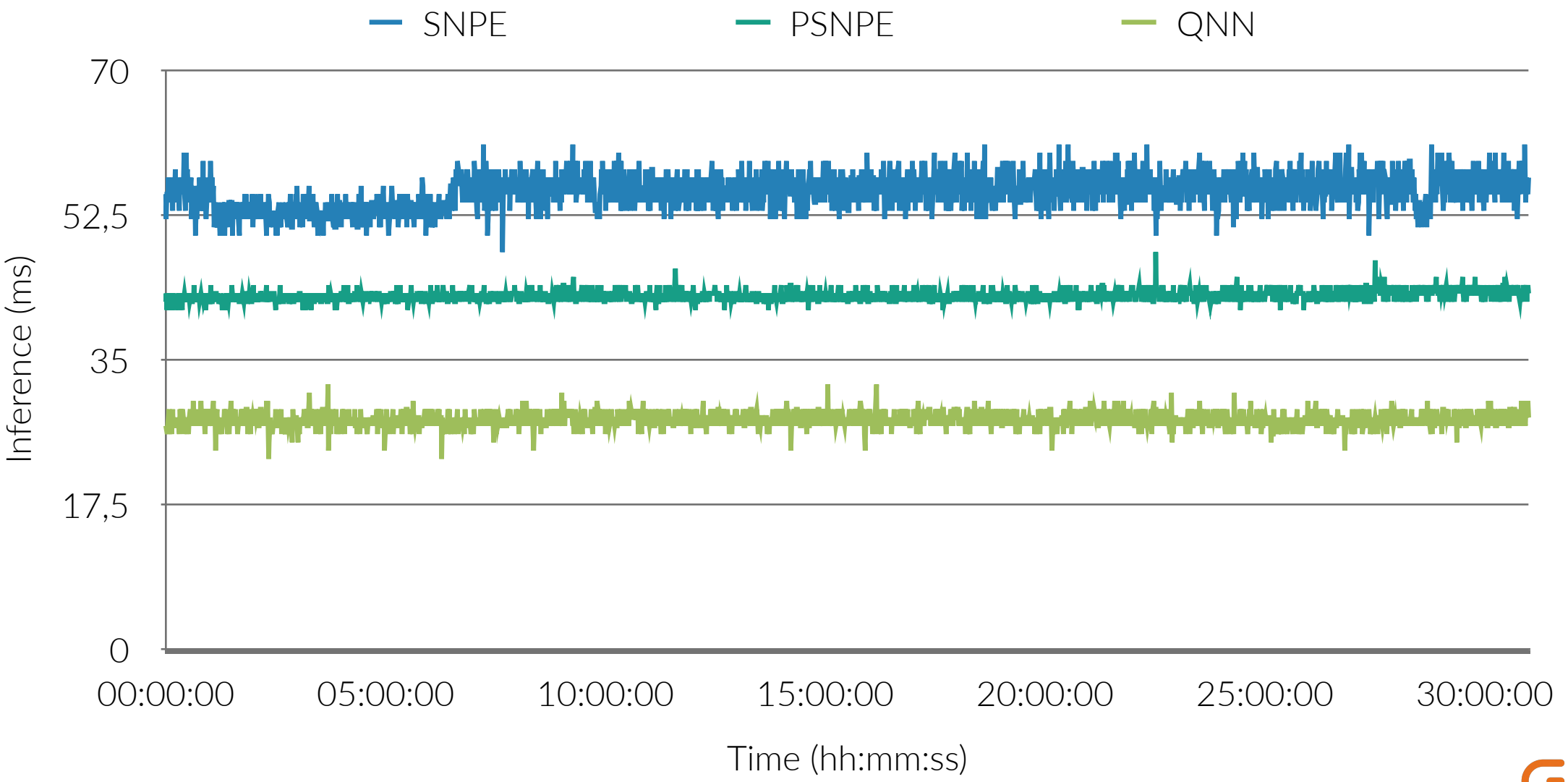
INFERENCE LATENCY COMPARISON

Inference Latency Analysis



TOTAL PIPELINE INFERENCE

End-to-End Pipeline Performance



FINAL INTERPRETATION

Overview

THAT'S SO GREAT IDEA

1. **Model Size**
   Quantization reduced model size by 3.9031 x in the best observed pipeline (QAIRT).

2. **Best Latency**
   INT8 improved latency to a best observed value of 6.6910 ms on SNPE (HTP/DSP).

3. **mAP Analysis**
   mAP changed by 52.31% after quantization in the strongest measured pipeline.

4. **Bboxes Drift**
   Output drift (measured here as MAE) was 7.3494 pixels.

5. **Qualcomm Neural Network SDK**
   QNN is a modern runtime architecture and performs better HTP integration.

QUESTIONS & ANSWERS

BREAK (15 MINUTES)

THANK YOU!