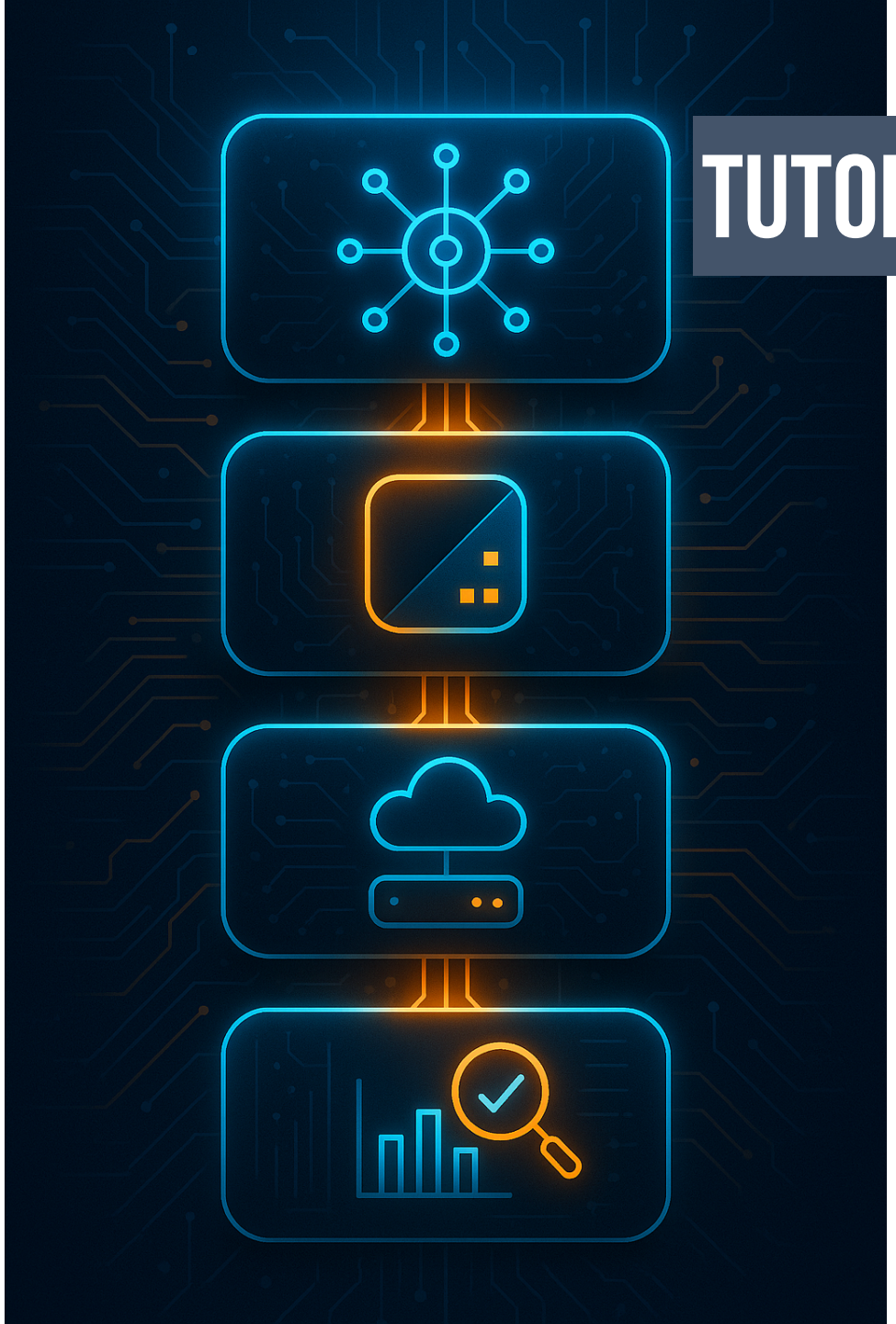
ACCELERATING EDGE AI WITH QUALCOMM AI HUB

CVPR 2025 Tutorial

The IEEE/CVF Conference on Computer Vision and Pattern Recognition 2025

Nashville, TN, USA



TUTORIAL AGENDA

1. Introduction to Qualcomm AI Hub
2. API Token and Python Environment
3. Model Compilation
4. Performance Profiling
5. Model Inference

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

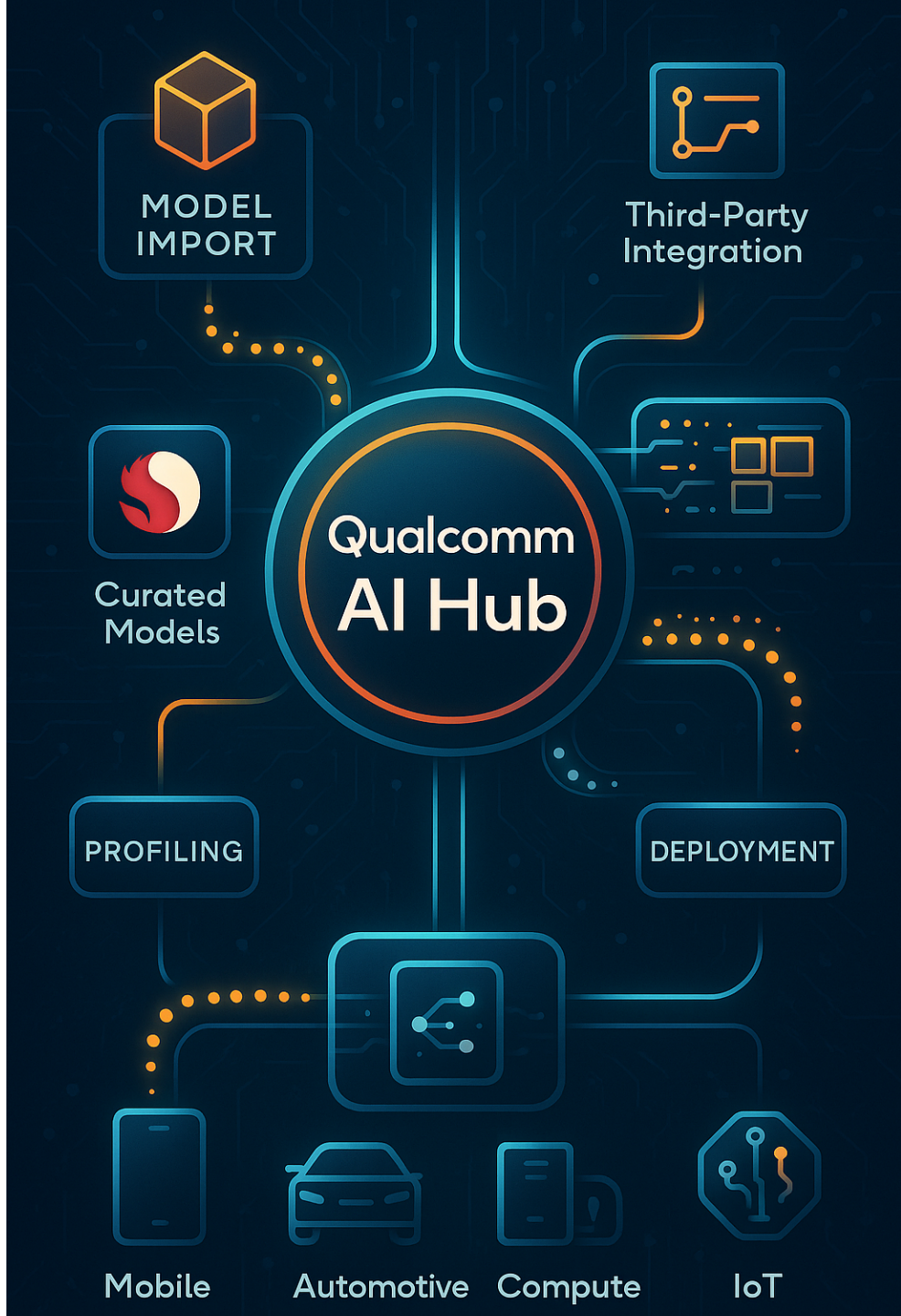
Get comprehensive on-device metrics including runtime, load time, and compute unit utilization.

Validation

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

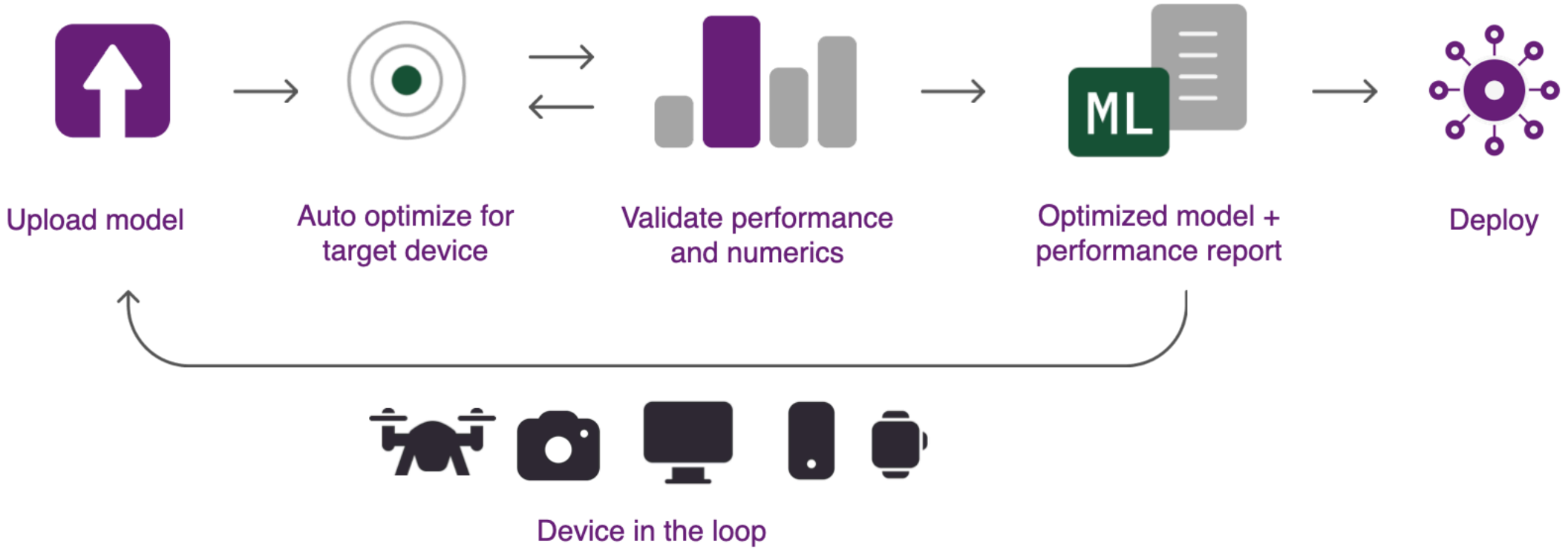
Flexible Deployment

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



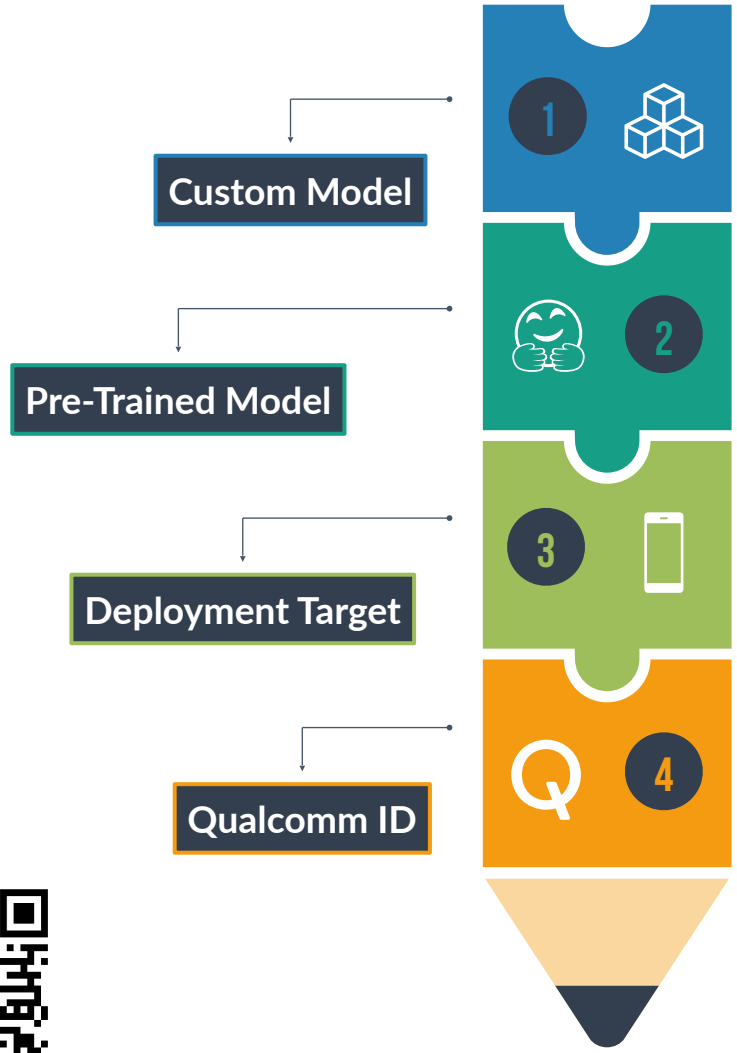
QUALCOMM AI HUB

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

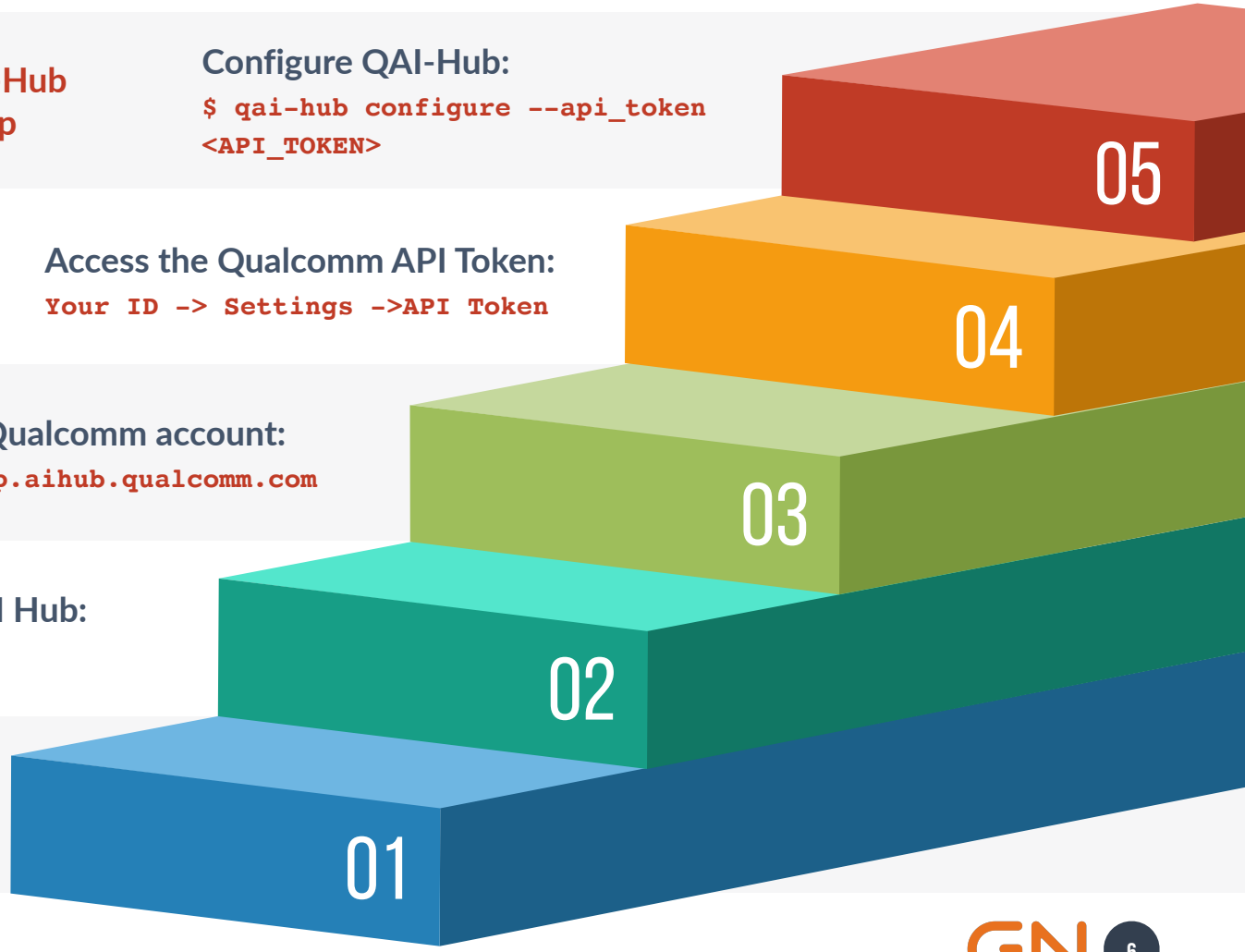
Install Python API for AI Hub:

```
$ 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
```

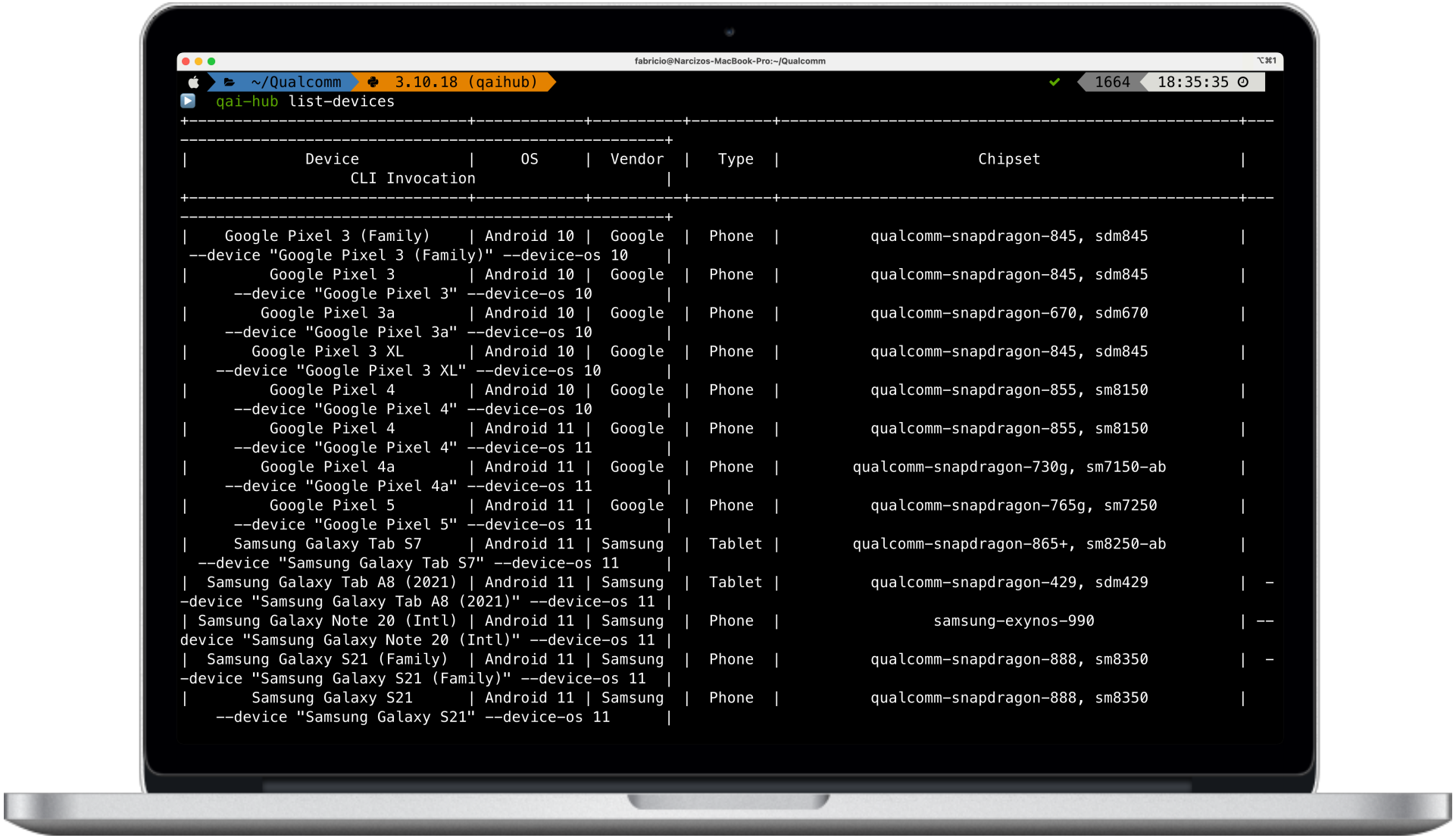
WEB

Access the following website on your browser:

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

CHECK AVAILABLE DEVICES

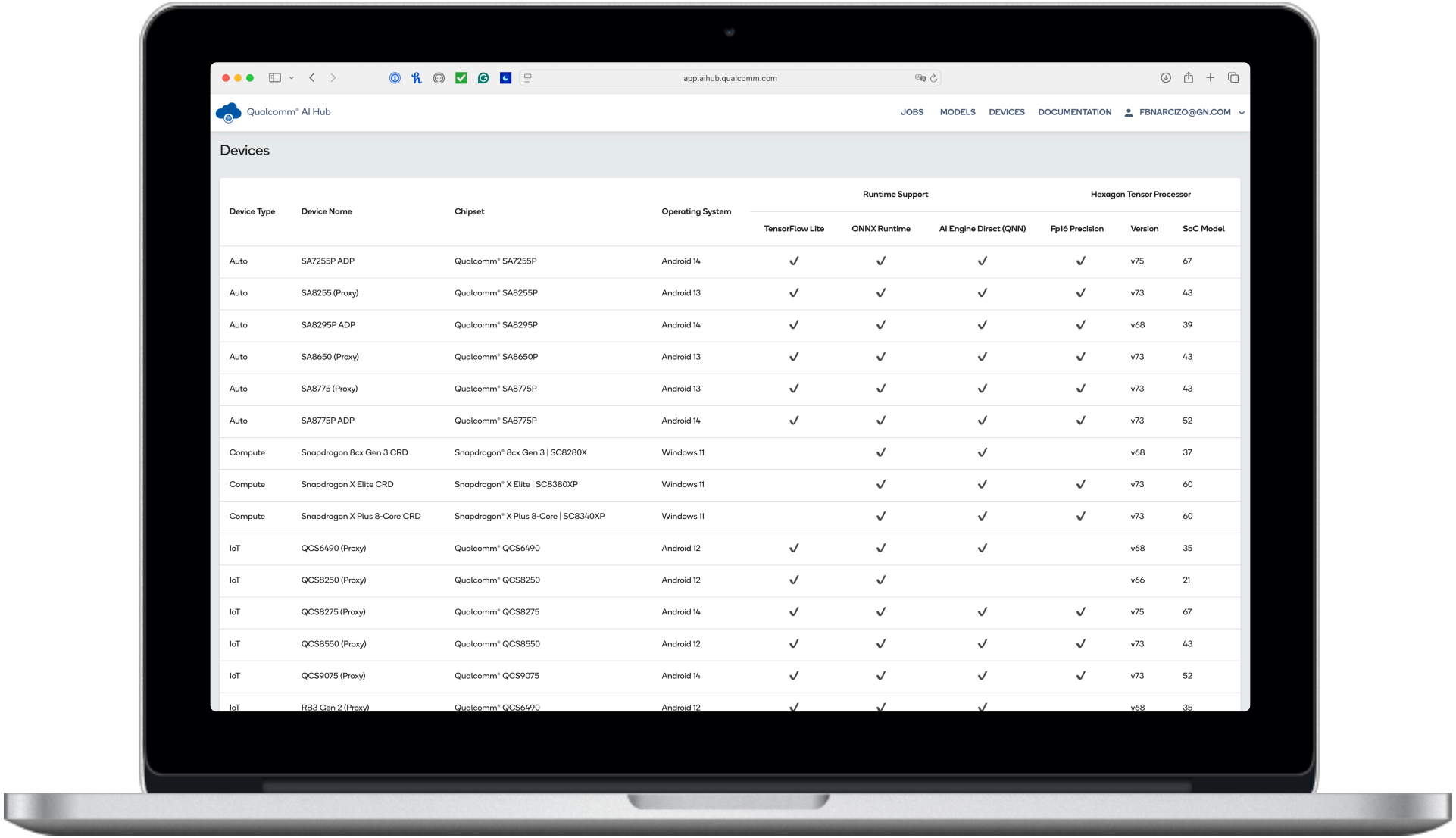
Terminal




```
+-------------------------------+------------+----------+---------+-------------------------------------------------------+---
-----------------------------------------------------+
|            Device             |     OS     |  Vendor  |   Type  |                       Chipset                         |
                CLI Invocation                       |
+-------------------------------+------------+----------+---------+-------------------------------------------------------+---
-----------------------------------------------------+
|    Google Pixel 3 (Family)    | Android 10 |  Google  |  Phone  |          qualcomm-snapdragon-845, sdm845          |
 --device "Google Pixel 3 (Family)" --device-os 10    |
|        Google Pixel 3         | Android 10 |  Google  |  Phone  |          qualcomm-snapdragon-845, sdm845          |
     --device "Google Pixel 3" --device-os 10         |
|       Google Pixel 3a         | Android 10 |  Google  |  Phone  |          qualcomm-snapdragon-670, sdm670          |
     --device "Google Pixel 3a" --device-os 10        |
|      Google Pixel 3 XL        | Android 10 |  Google  |  Phone  |          qualcomm-snapdragon-845, sdm845          |
    --device "Google Pixel 3 XL" --device-os 10        |
|        Google Pixel 4         | Android 10 |  Google  |  Phone  |          qualcomm-snapdragon-855, sm8150          |
     --device "Google Pixel 4" --device-os 10         |
|        Google Pixel 4         | Android 11 |  Google  |  Phone  |          qualcomm-snapdragon-855, sm8150          |
     --device "Google Pixel 4" --device-os 11         |
|       Google Pixel 4a         | Android 11 |  Google  |  Phone  |        qualcomm-snapdragon-730g, sm7150-ab        |
     --device "Google Pixel 4a" --device-os 11        |
|        Google Pixel 5         | Android 11 |  Google  |  Phone  |          qualcomm-snapdragon-765g, sm7250         |
     --device "Google Pixel 5" --device-os 11         |
|     Samsung Galaxy Tab S7     | Android 11 | Samsung  |  Tablet |        qualcomm-snapdragon-865+, sm8250-ab        |
  --device "Samsung Galaxy Tab S7" --device-os 11      |
|  Samsung Galaxy Tab A8 (2021) | Android 11 | Samsung  |  Tablet |          qualcomm-snapdragon-429, sdm429          |  -
-device "Samsung Galaxy Tab A8 (2021)" --device-os 11 |
| Samsung Galaxy Note 20 (Intl) | Android 11 | Samsung  |  Phone  |                 samsung-exynos-990                | --
device "Samsung Galaxy Note 20 (Intl)" --device-os 11 |
|  Samsung Galaxy S21 (Family)  | Android 11 | Samsung  |  Phone  |          qualcomm-snapdragon-888, sm8350          |  -
-device "Samsung Galaxy S21 (Family)" --device-os 11  |
|       Samsung Galaxy S21      | Android 11 | Samsung  |  Phone  |          qualcomm-snapdragon-888, sm8350          |
    --device "Samsung Galaxy S21" --device-os 11       |
```

CHECK AVAILABLE DEVICES

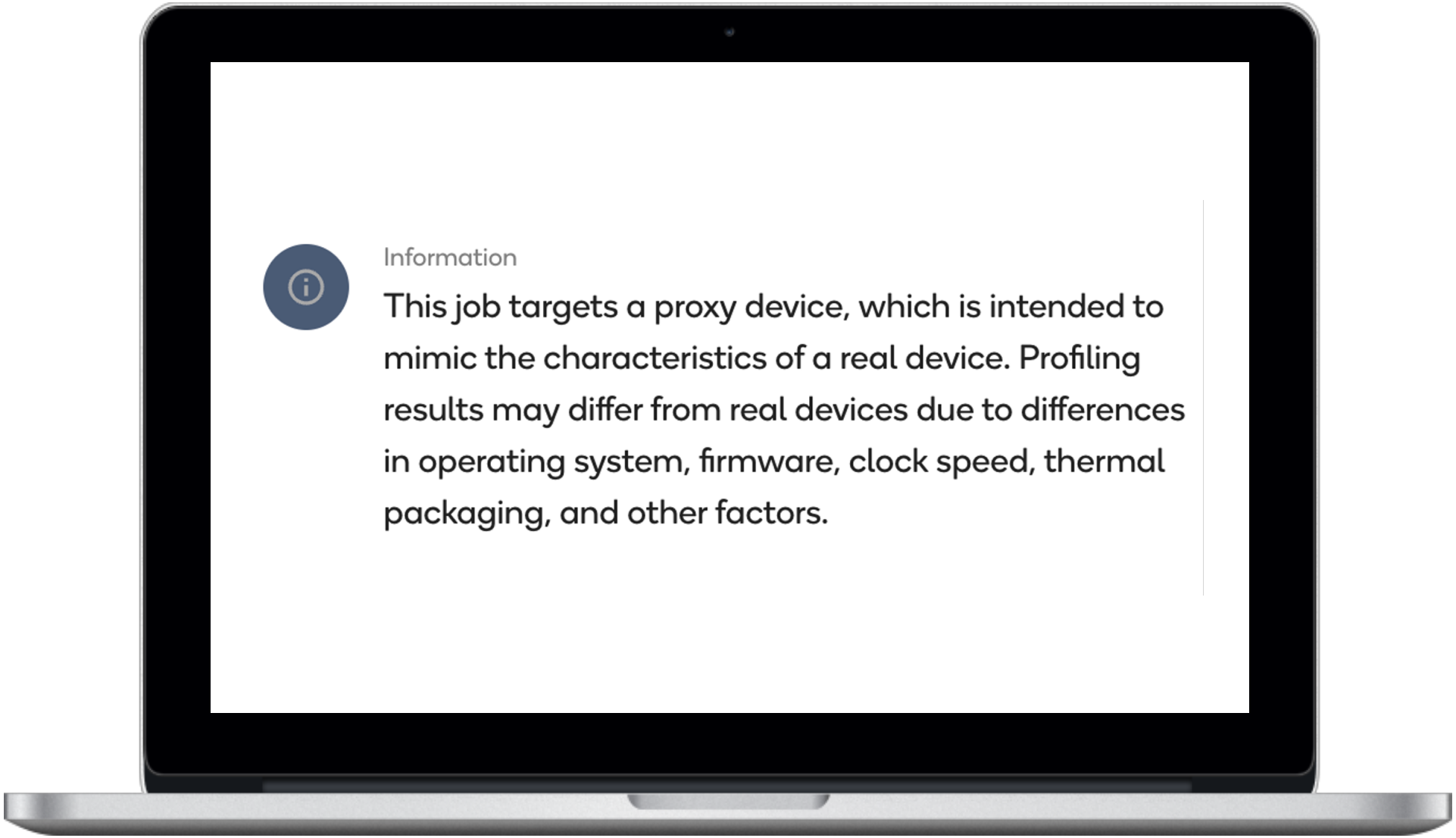
Qualcomm AI Hub



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

WHY IS IT CALLED QCS6490 (PROXY)?

Important Information



CHANGES SPECIFIC FOR OUR APP

InferSNPE Android Application

Data Layer Format

When using SNPE, the model input layers changed from NCHW to NHWC.

Input Format

InferSNPE App requires the input format as NHWC (1, 320, 320, 3).

Output Names

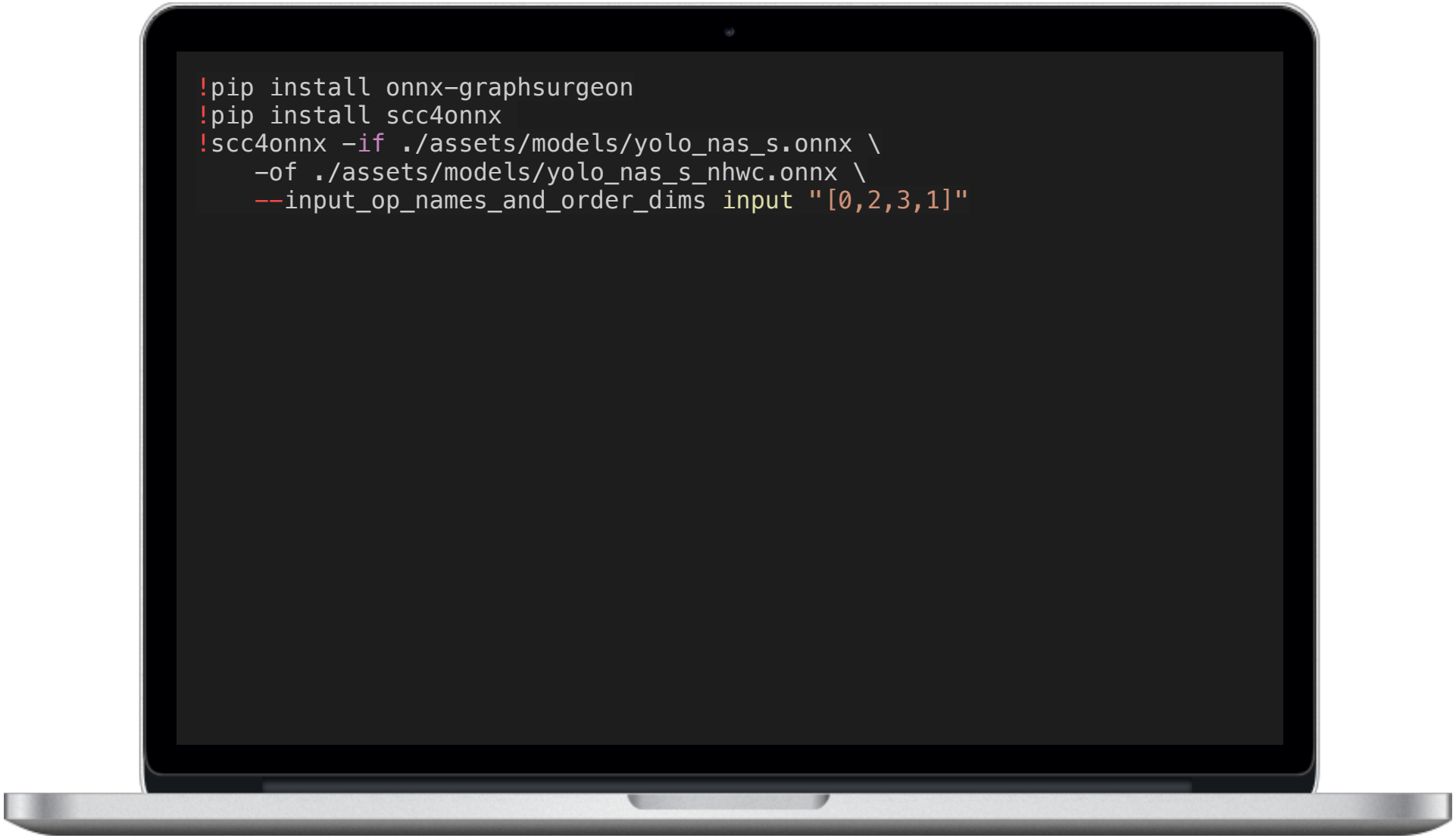
`output_bboxes:` Bounding box coordinates.
`output_classes:` Class prediction scores.

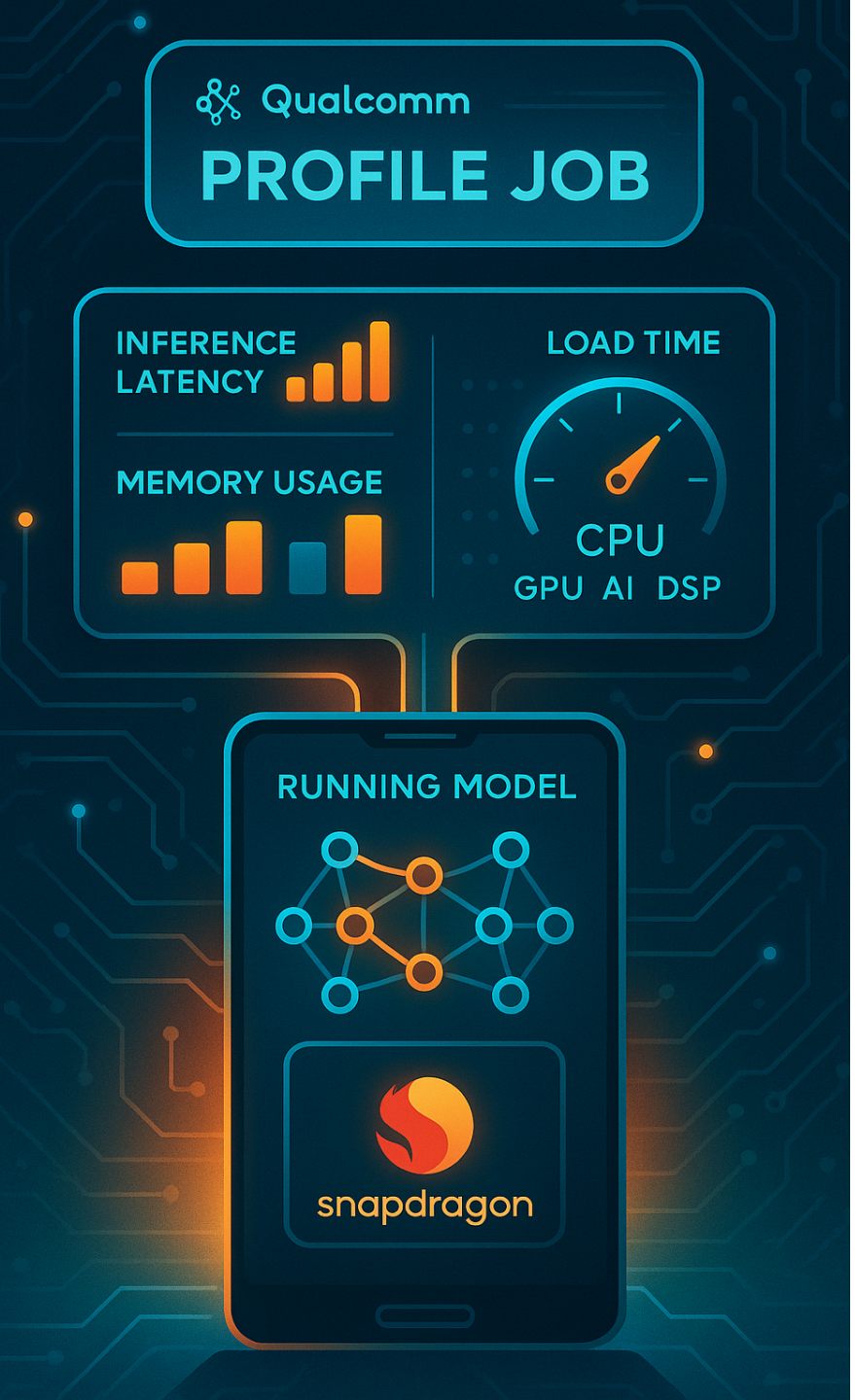
Layer Names

The InferSNPE App looks for these specific names, not generic `output_0` or `output_1`.

HOW TO CHANGE THE INPUT FORMAT

https://github.com/fabricionarcizo/snpe_optimizer/blob/main/notebooks/qai_hub.ipynb





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

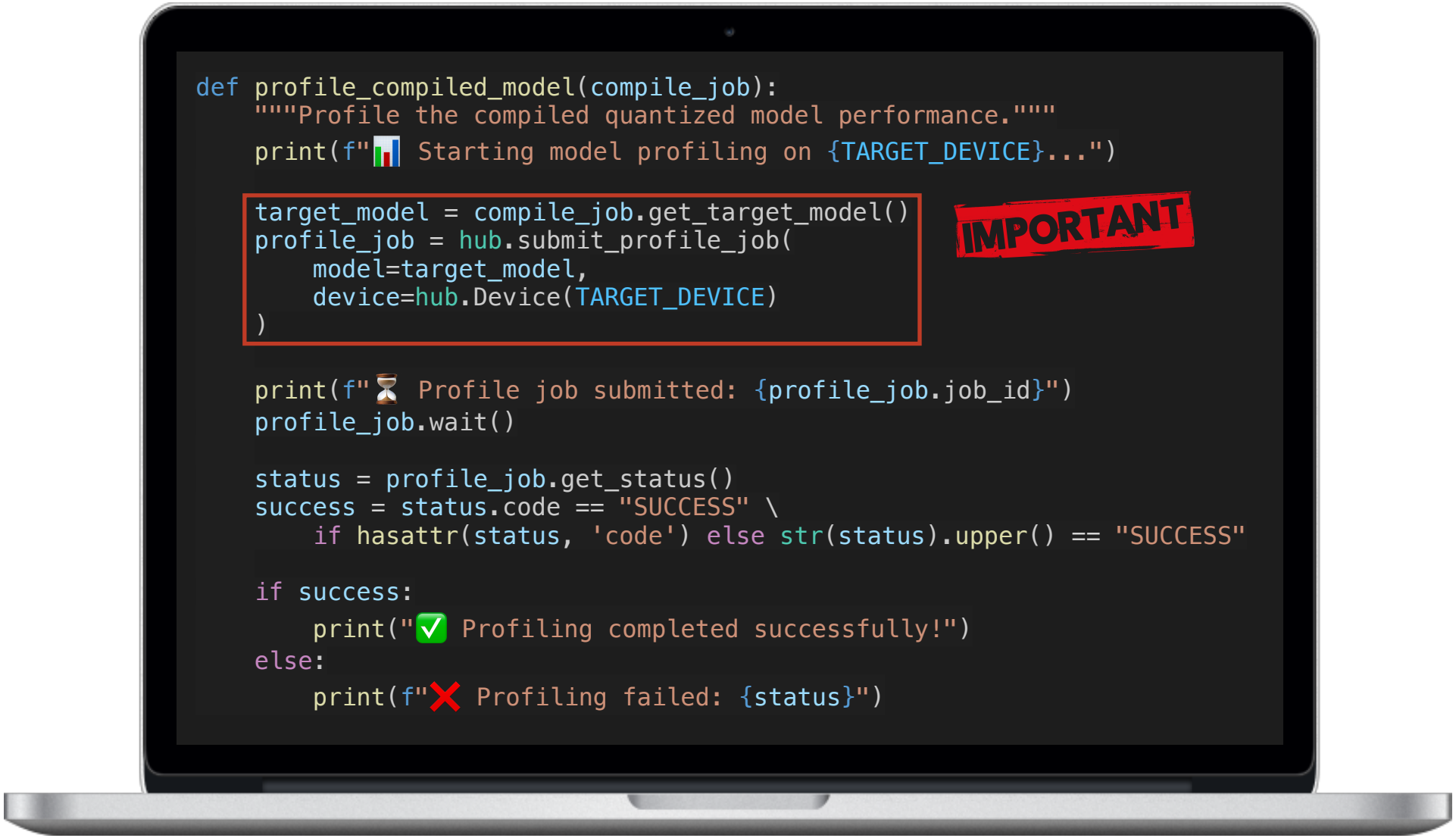
Indicates the total memory consumed during inference, helping identify models too large for constrained devices.

Compute Breakdown

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

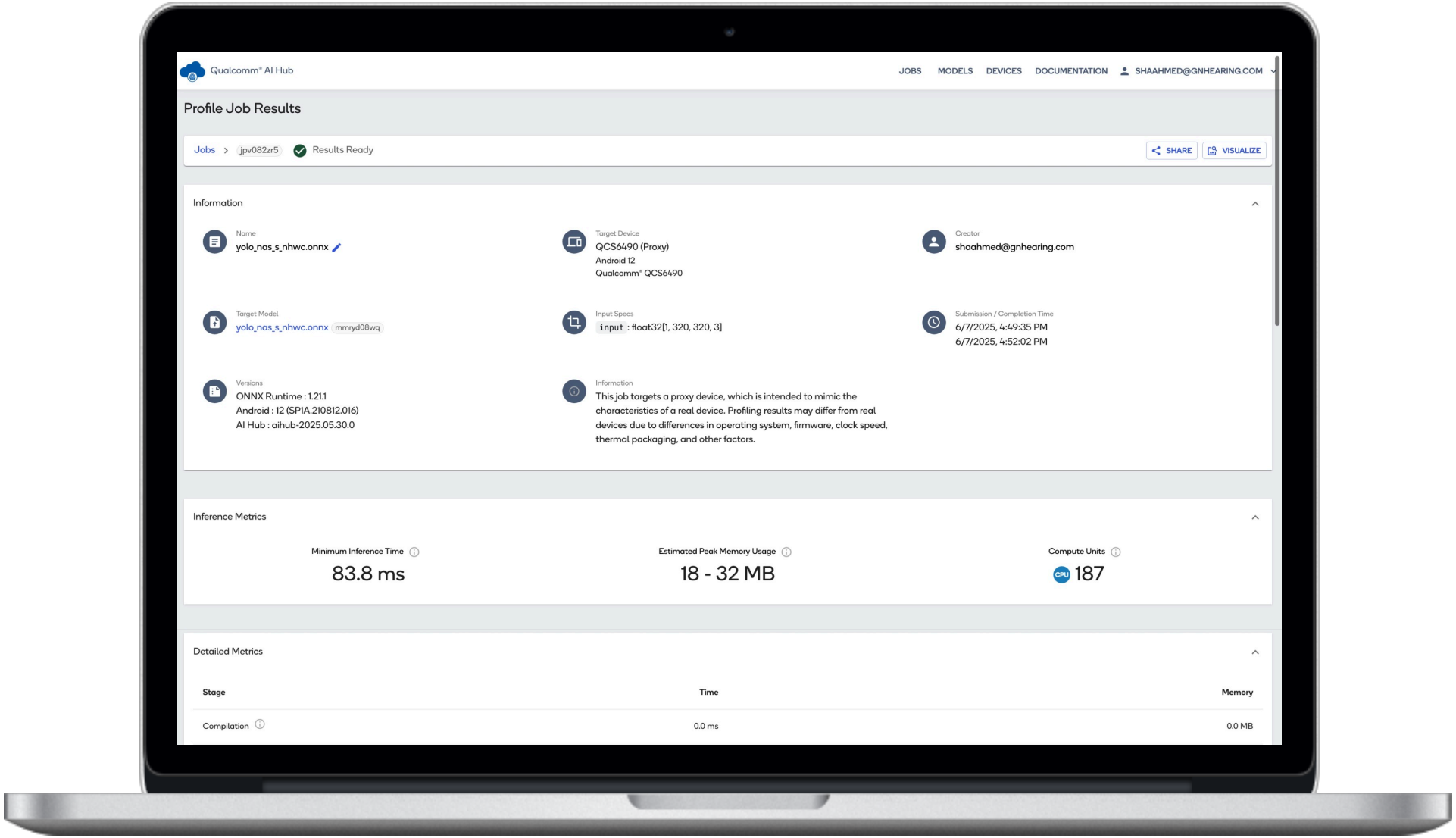
PROFILE ORIGINAL ONNX MODEL

https://github.com/fabricionarcizo/snpe_optimizer/blob/main/notebooks/qai_hub.ipynb



PROFILE ORIGINAL ONNX MODEL

Original YOLO-NAS S Model Inference



CALIBRATION DATA

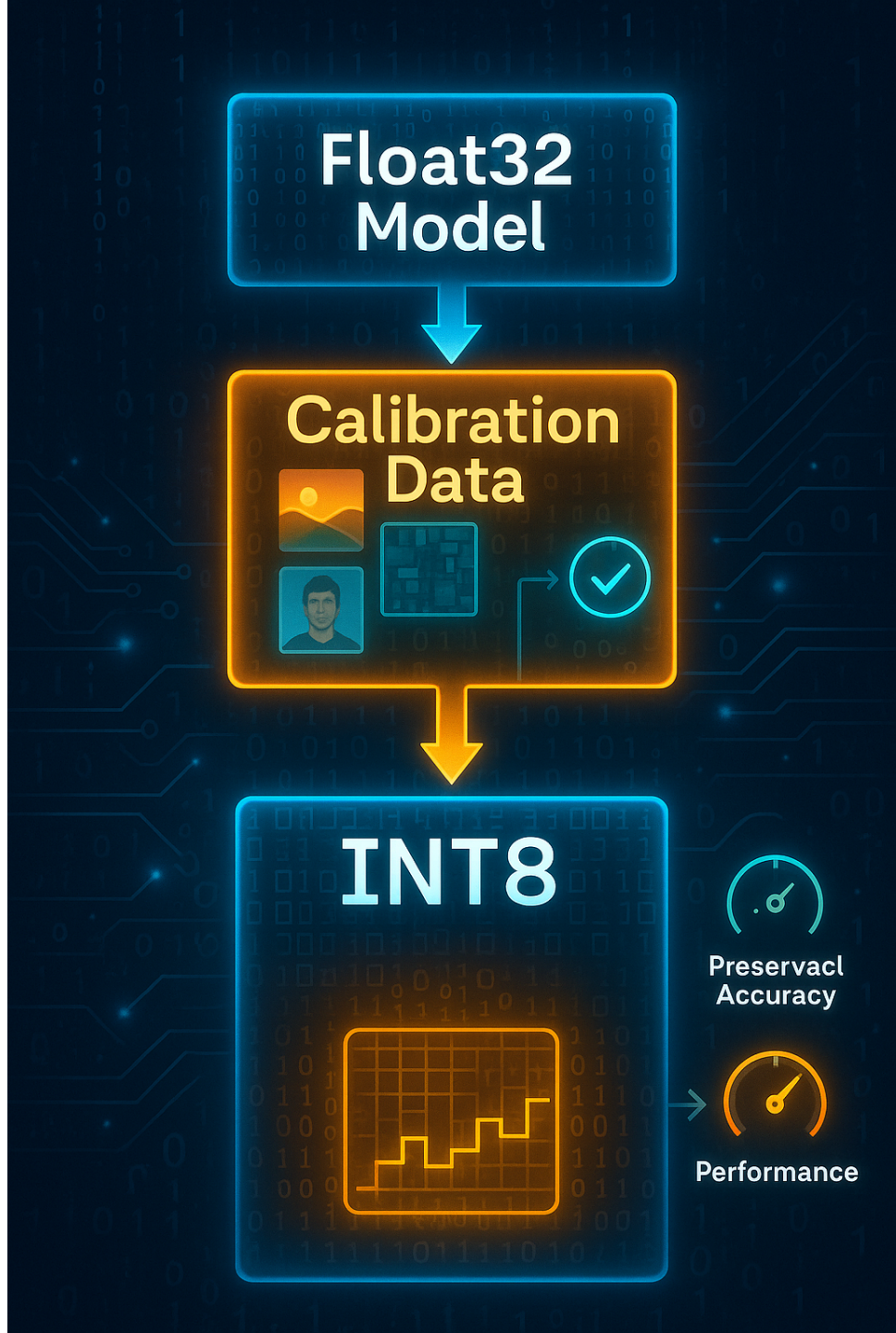
Overview

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.



COMPILE JOB

Transform and Optimize

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

Convert the original model to Qualcomm format.

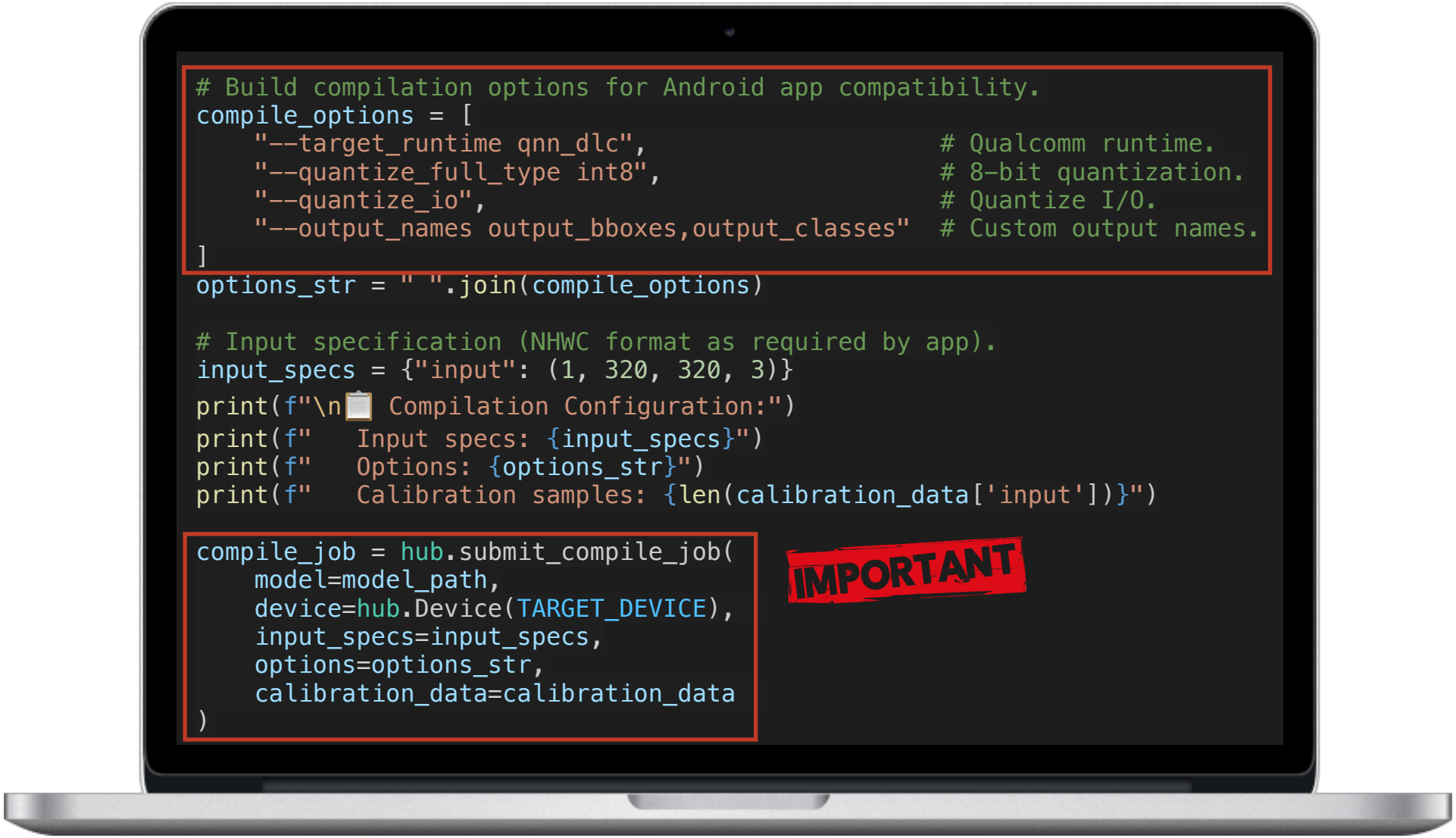
Quantize the Qualcomm model from FP32 to INT8.

Quantize the input and output layers from the INT8 model.

Directly give names for outputs required for InferSNPE App.

COMPILE JOB CODE

https://github.com/fabricionarcizo/snpe_optimizer/blob/main/notebooks/qai_hub.ipynb

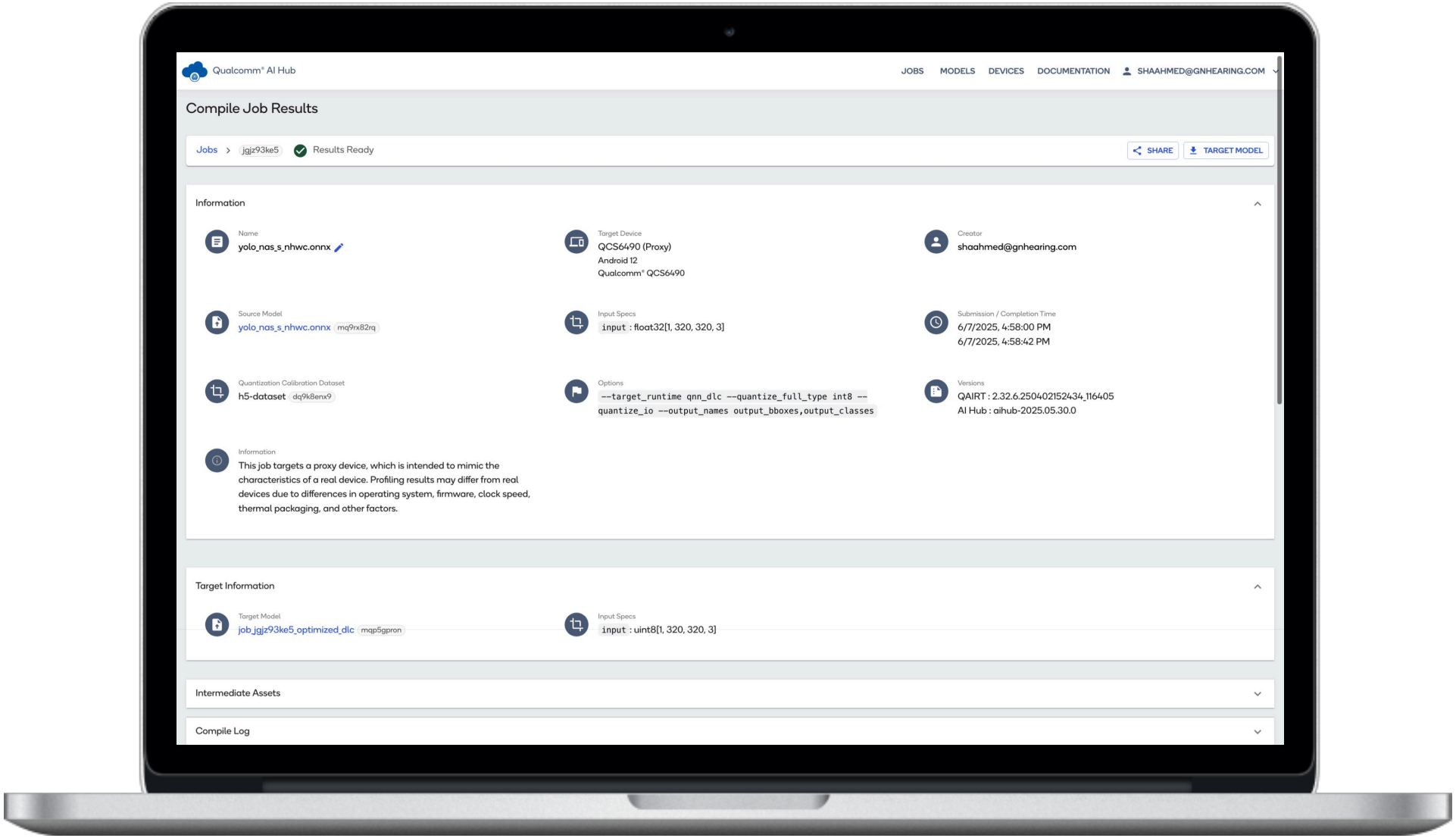


```
# Build compilation options for Android app compatibility.
compile_options = [
    "--target_runtime qnn_dlc",                    # Qualcomm runtime.
    "--quantize_full_type int8",                   # 8-bit quantization.
    "--quantize_io",                               # Quantize I/O.
    "--output_names output_bboxes,output_classes"  # Custom output names.
]
options_str = " ".join(compile_options)

# Input specification (NHWC format as required by app).
input_specs = {"input": (1, 320, 320, 3)}
print(f"\n📋 Compilation Configuration:")
print(f"   Input specs: {input_specs}")
print(f"   Options: {options_str}")
print(f"   Calibration samples: {len(calibration_data['input'])}")

compile_job = hub.submit_compile_job(
    model=model_path,
    device=hub.Device(TARGET_DEVICE),
    input_specs=input_specs,
    options=options_str,
    calibration_data=calibration_data
)
```

IMPORTANT

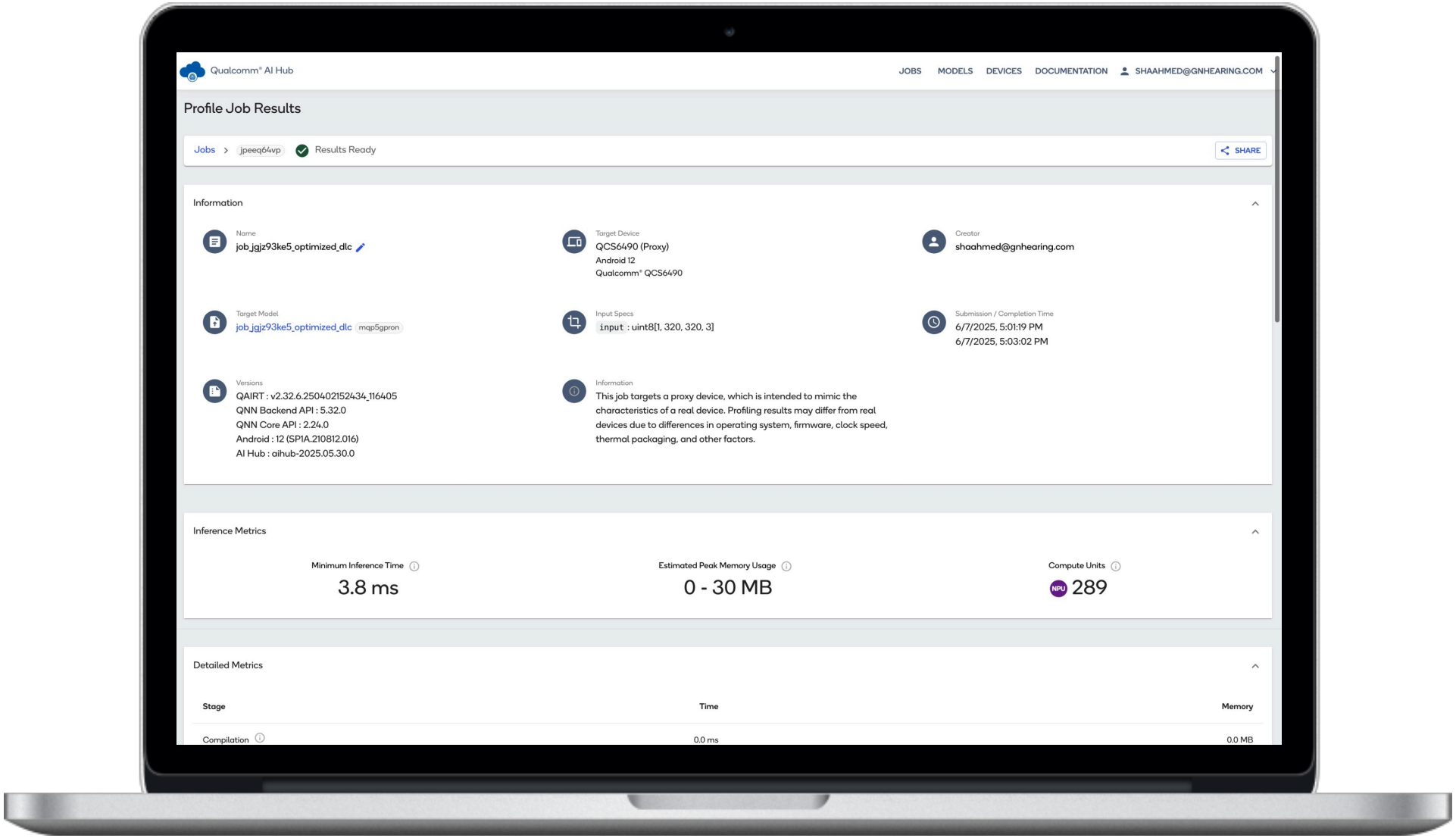
COMPILE JOB

YOLO-NAS S Model



PROFILE AFTER COMPILATION

YOLO-NAS S Model



RUN INFERENCE

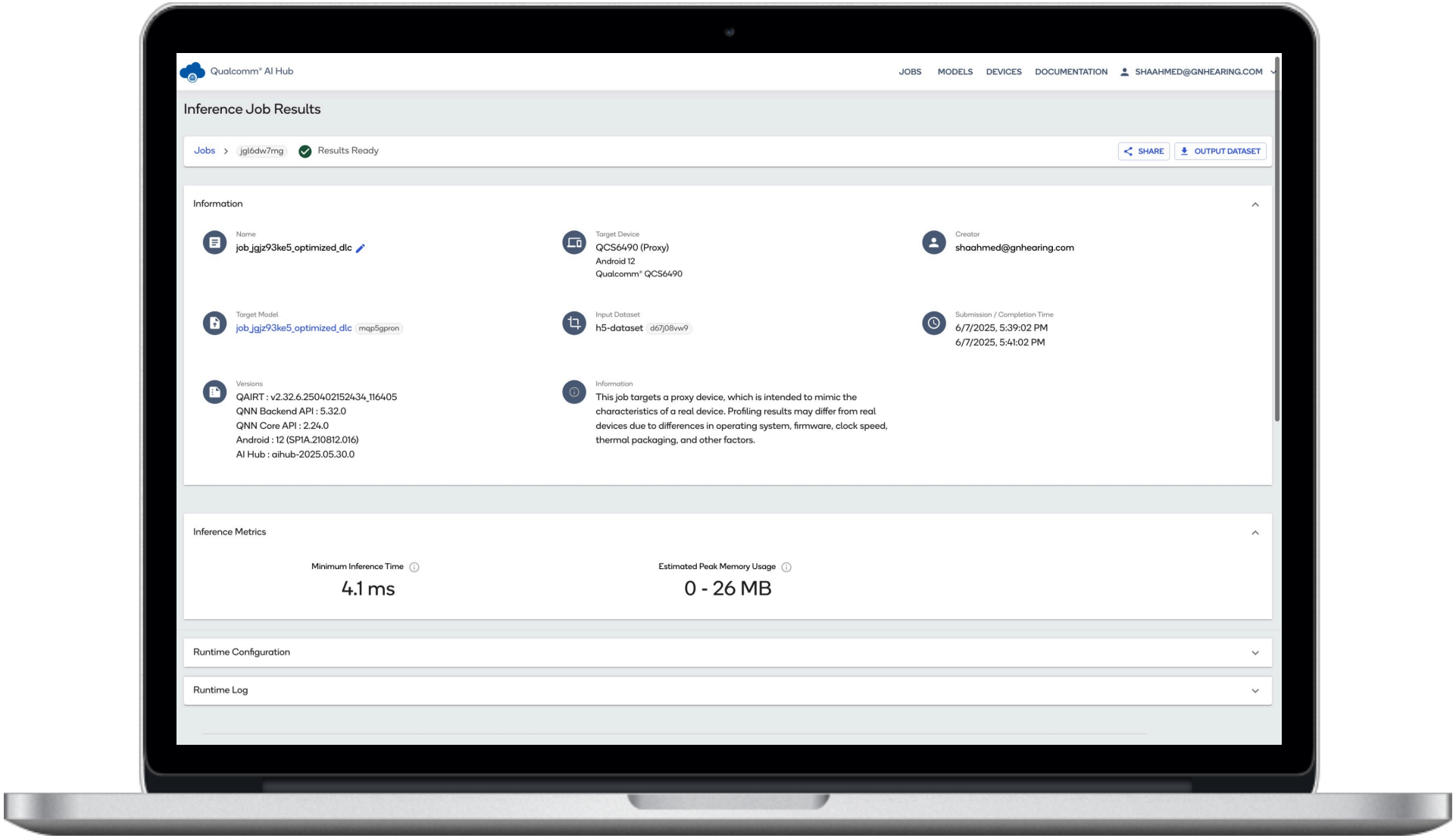
https://github.com/fabricionarcizo/snpe_optimizer/blob/main/notebooks/qai_hub.ipynb

```python
def simple_inference_pipeline(
    image_path: str, compile_job, show_inline: bool = True):
    print(f"🚀 Simple YOLO inference on: {os.path.basename(image_path)}")
    print(f"   Format: int8, NHWC")
    print(f"   Output: Top 10 detections only")

    # 1. Preprocess image.
    input_data, original_image, scale_x, scale_y = \
        simple_preprocess(image_path)

    # 2. Run inference.
    print("🔄 Running inference...")
    target_model = compile_job.get_target_model()

    inference_job = hub.submit_inference_job(
        model=target_model,
        device=hub.Device(TARGET_DEVICE),
        inputs={"input": [input_data]}
    )

    inference_job.wait()
    results = inference_job.download_output_data()
    ...
```

IMPORTANT

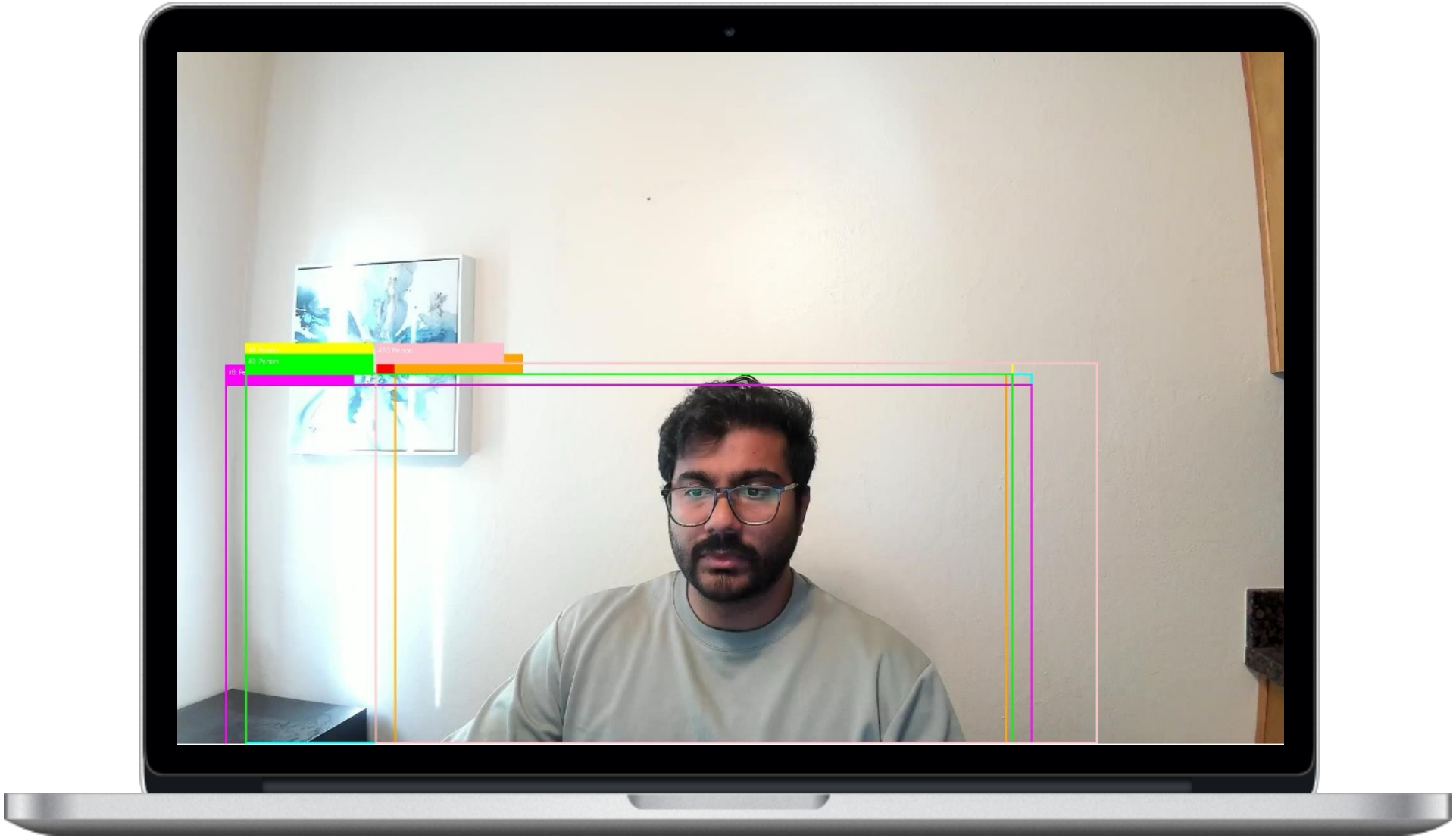
INFERENCE RESULTS

Quantized YOLO-NAS S Model



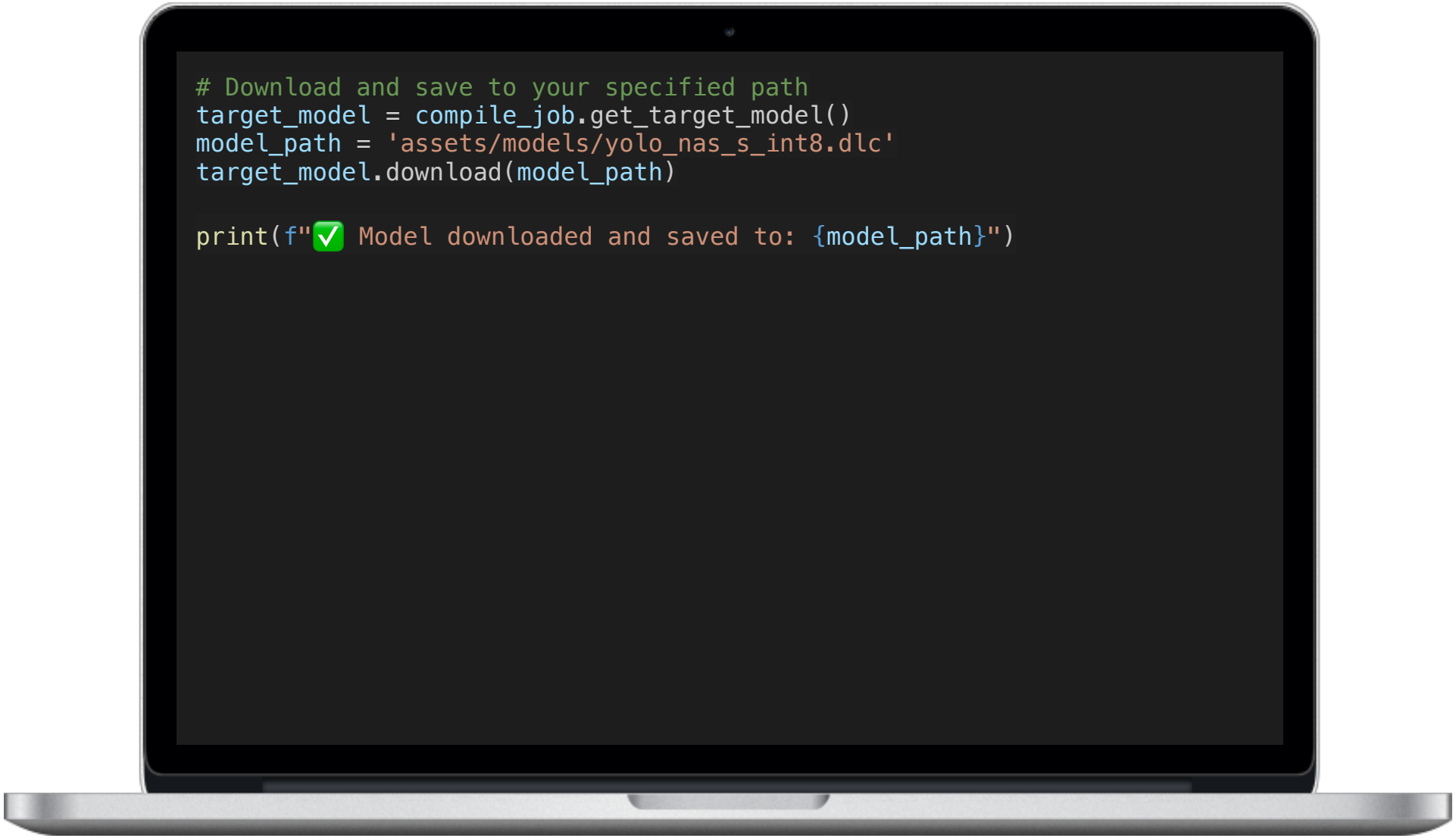
INFERENCE OUTPUT IMAGE

Example



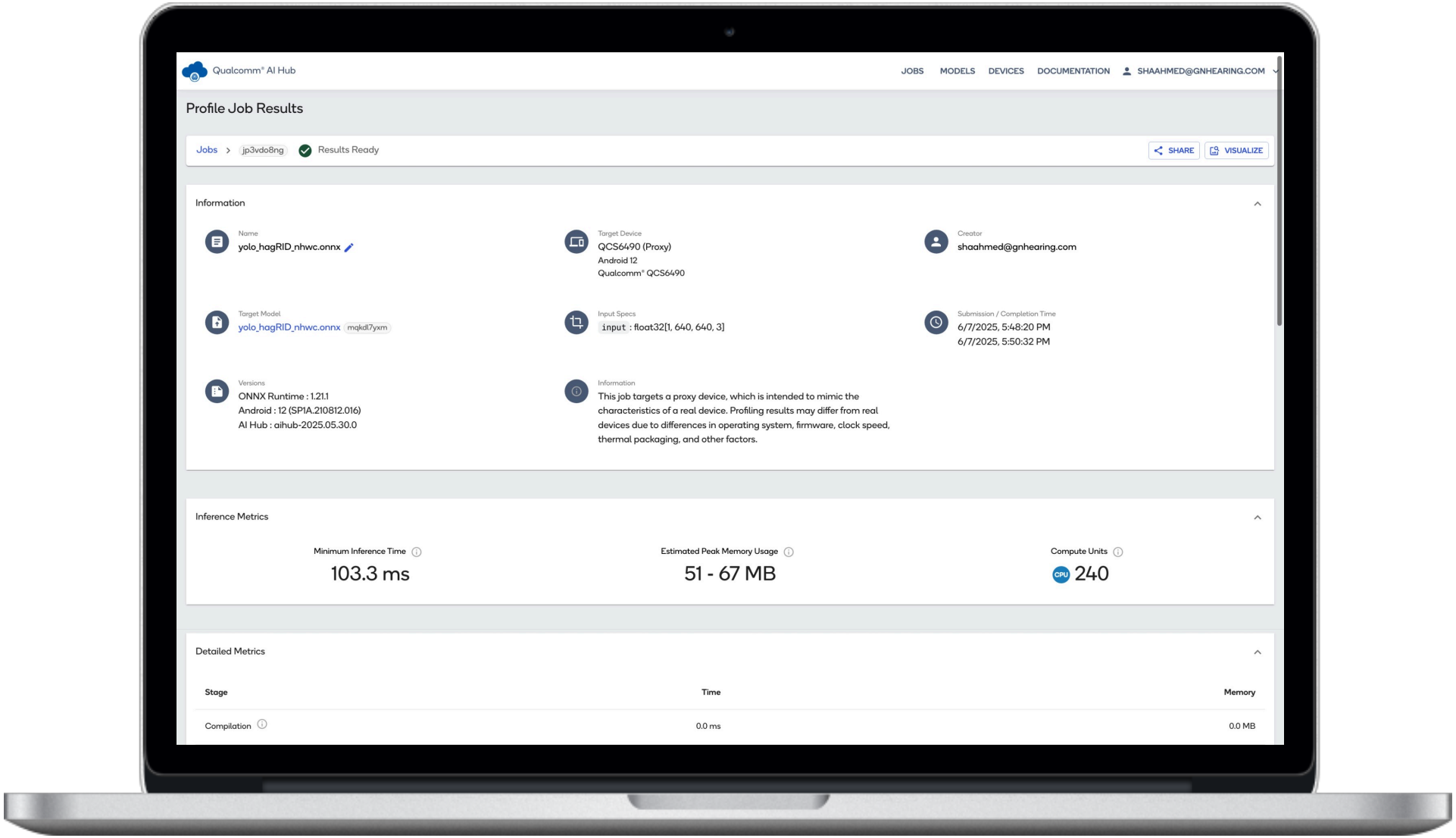
DOWNLOAD QUANTIZED MODEL

https://github.com/fabricionarcizo/snpe_optimizer/blob/main/notebooks/qai_hub.ipynb



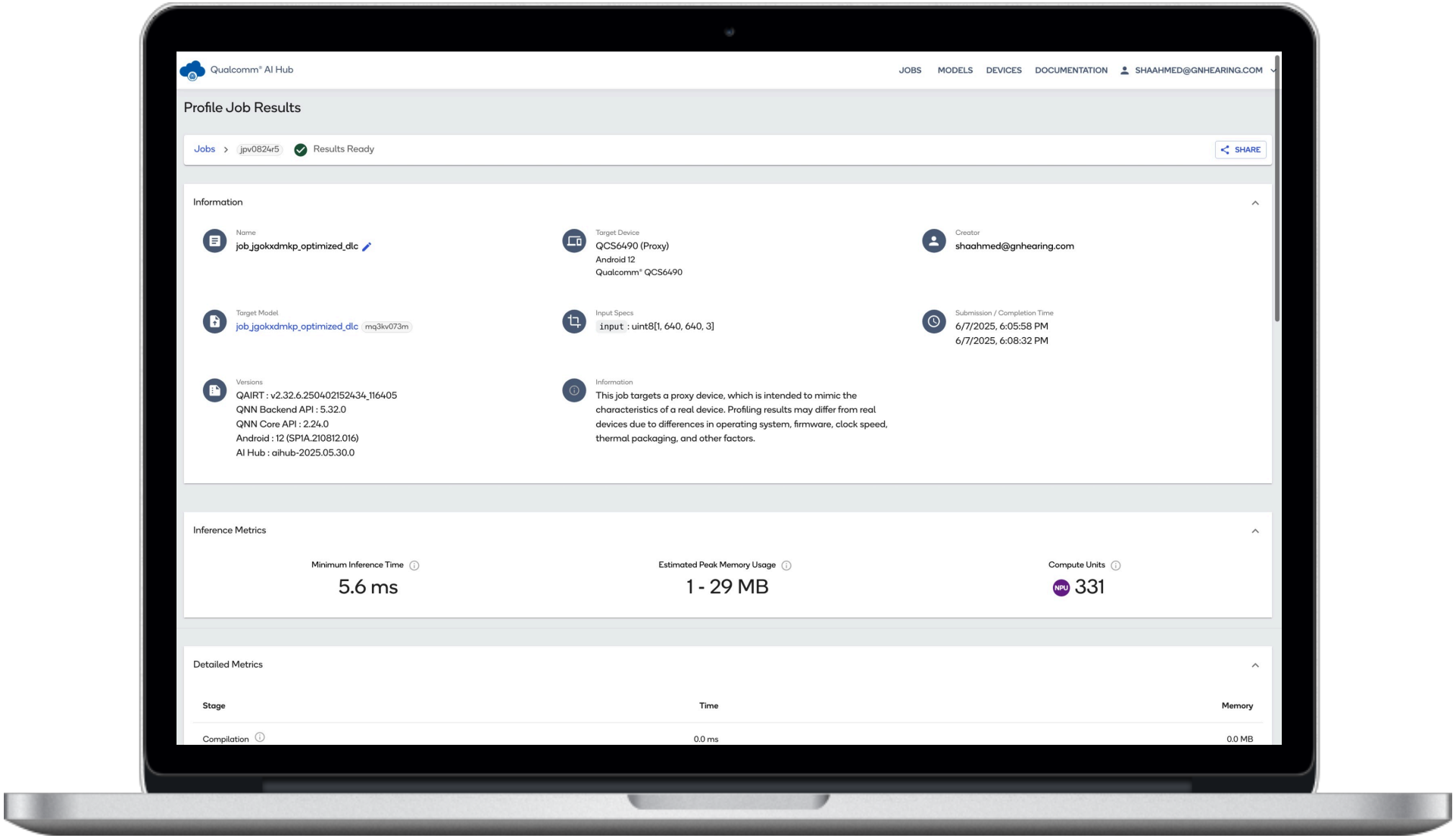
PROFILE ONNX HAGRID

Original YOLO-hagRID Model Inference



PROFILE ONNX HAGRID

Quantized YOLO-hagRID Model Inference

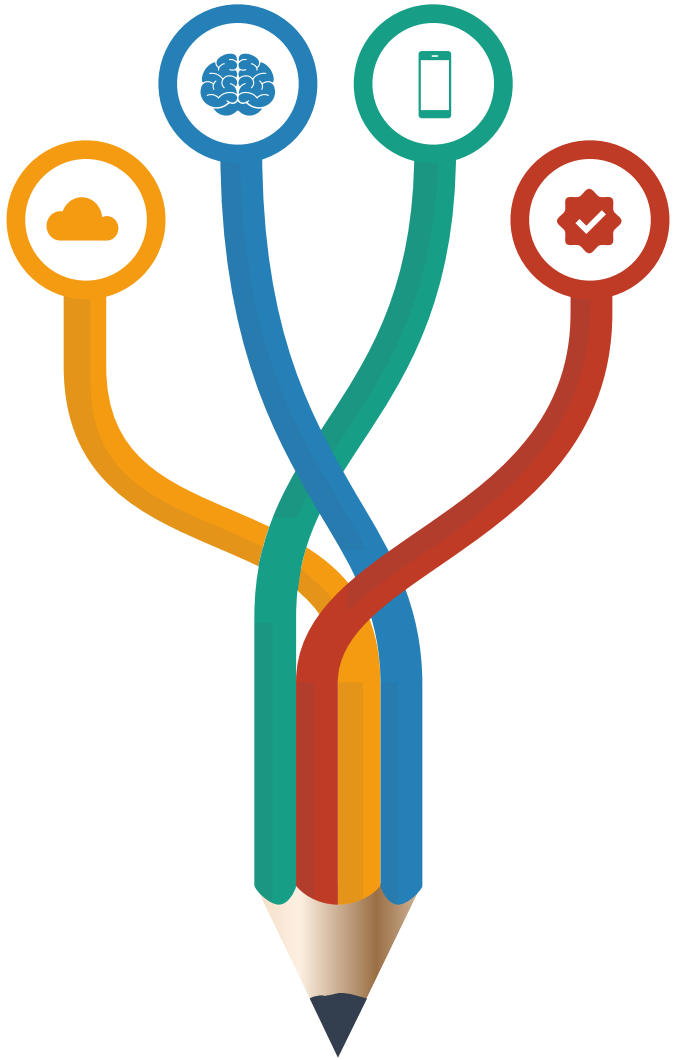


HOW DOES IT WORK?

Overview

Intelligent Translation

Automatically converts models from source frameworks to device-optimized runtime.



Hardware-Aware Optimization

Applies Qualcomm-specific optimizations for maximum performance.

Cloud-Based Validation

Provision real devices in the cloud for accurate performance profiling.

Physical Testing

Validates both performance metrics and numerical correctness on actual hardware.

QUESTIONS & ANSWERS

THANK YOU!