Every Millisecond Counts

The Fine-tuning Journey of an Ultra-Efficient PyTorch Model for the Edge

Pavel Macenauer
R&D Manager @ NXP Semiconductors

PyTorch Conference Europe 2026

Edge Device Classes

Microcontrollers

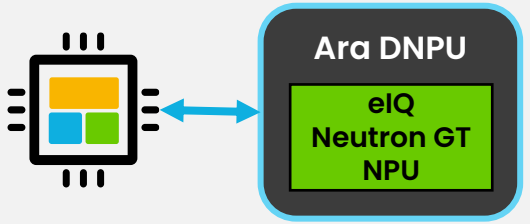
MCX MCUs

i.MX RT Crossover MCUs

Microprocessors

i.MX Apps Processors

Microprocessors with External ML Accelerator



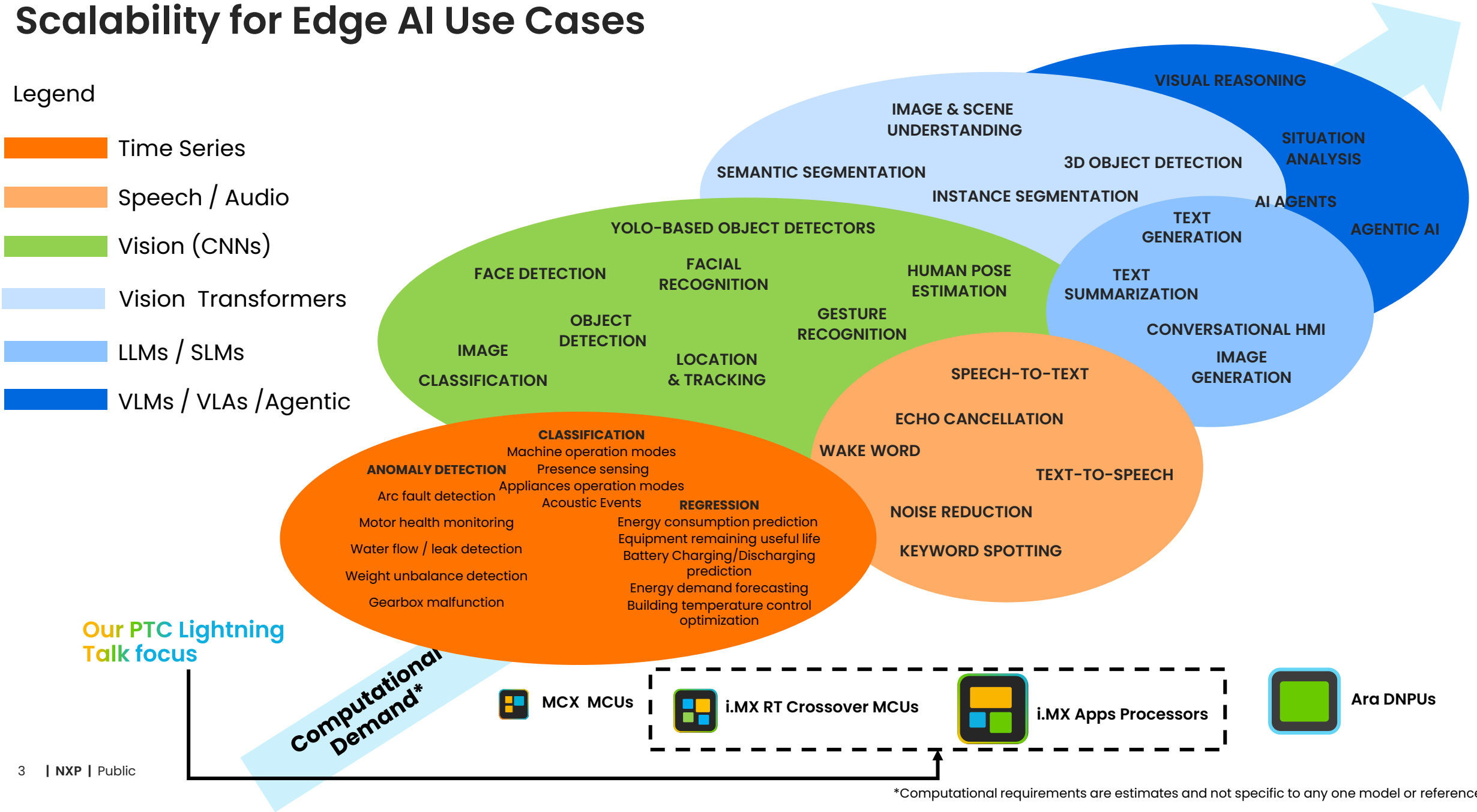
eIQ® Neutron NPU
Highly scalable, flexible and power optimized integrated AI acceleration cores

eIQ® Neutron GT NPU
New class of integrated AI acceleration cores that scales to 100s of TOPs

Ara Discrete NPUs
High-performance AI co-processors enabling platform scalability and deployment of Agentic-AI and Physical-AI on the edge

CPU NPU DSP/GPU and other cores

Scalability for Edge AI Use Cases

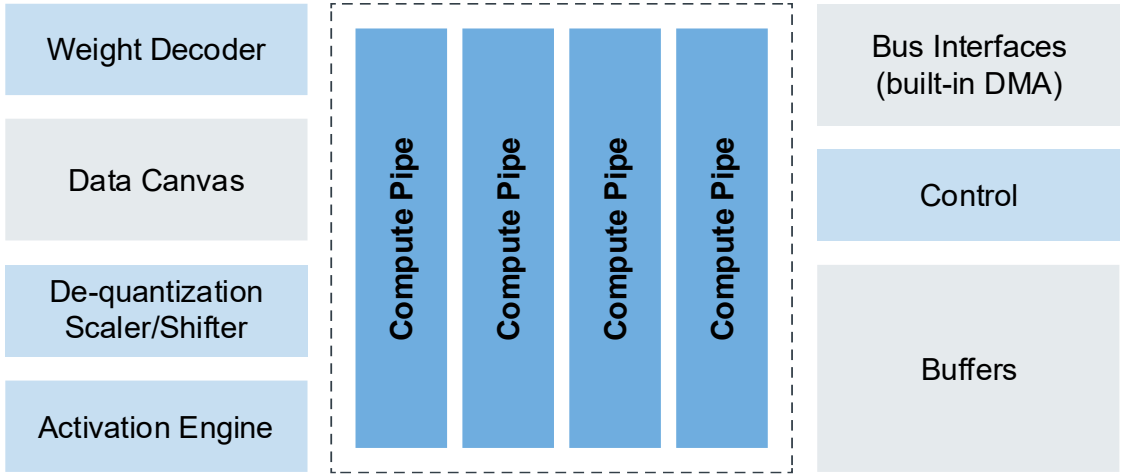


*Computational requirements are estimates and not specific to any one model or reference.

eIQ® Neutron NPU

- **Scalable Neural Processing Unit** (from 32 ops/cycle to 10 000+ Ops/cycle) to run int8 machine learning models.
- **Performance**, **low power** and **low footprint**.
- Support for **CNN**, **RNN**, **TCN**, **Transformers**.
- Programmable through **eIQ AI Toolkit, eIQ Neutron SDK** and inference engines such as ExecuTorch or TF Lite.

eIQ Neutron NPU Architecture



eIQ Neutron NPU Additional Options

- Dedicated Controller Core
- In-line dequantization, Activation and Pooling
- Built in tiny-caching to reduce power consumption and reduce reliance on system memory speed
- Weight Decompression Engine
- Configurable Coupled Memory
- Advanced multi-dimensional DMA for input and output formats, including striding, batching, interleaving, concatenating, etc.

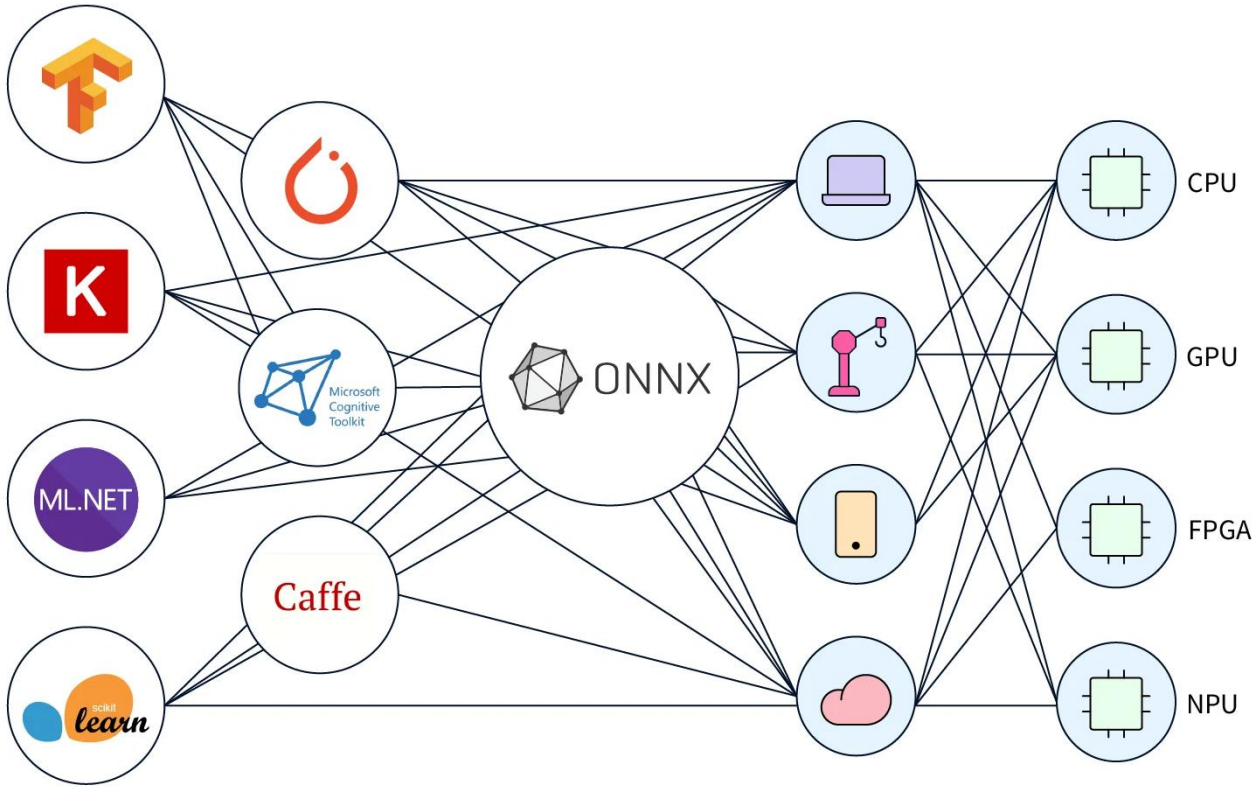
eIQ Neutron NPU Paper: https://arxiv.org/html/2509.14388v1 "eIQ Neutron: Redefining Edge-AI Inference with Integrated NPU and Compiler Innovations"
eIQ Neutron NPU Product: https://www.nxp.com/applications/technologies/ai-and-machine-learning/eiq-neutron-npu:EIQ-NEUTRON-NPU
i.MX95 Application Processors: https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-9-processors/i-mx-95-applications-processor-family-high-performance-safety-enabled-platform-with-eiq-neutron-npu:iMX95

What is ONNX?

Open Neural Network Exchange (ONNX) is an ecosystem providing an open-source format for AI models, both deep learning and traditional ML.

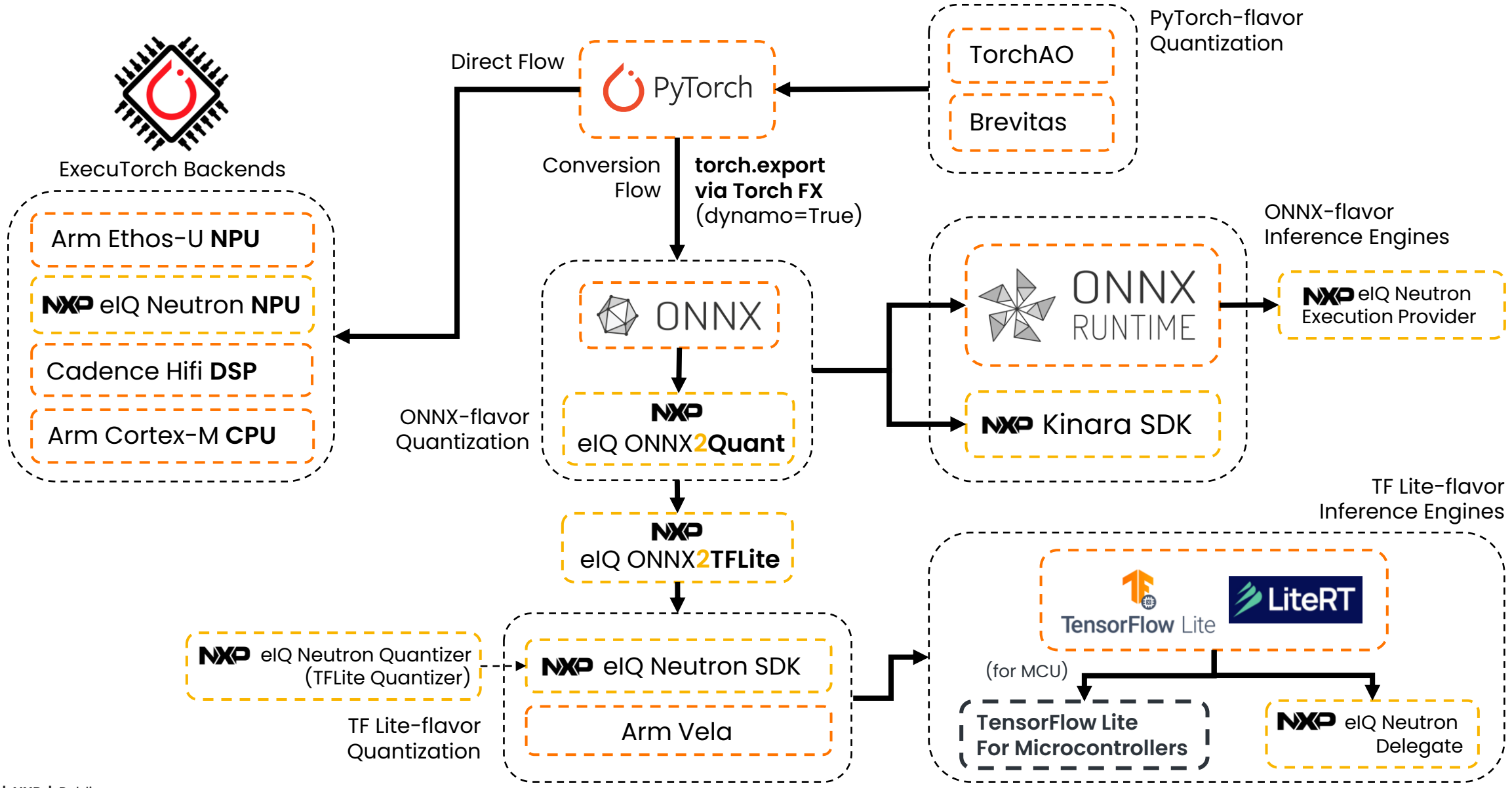
It defines:

- an extensible computation **graph model**
- definitions of built-in **operators**
- standard **data types**



ONNX is widely supported and enables **interoperability** between different frameworks.

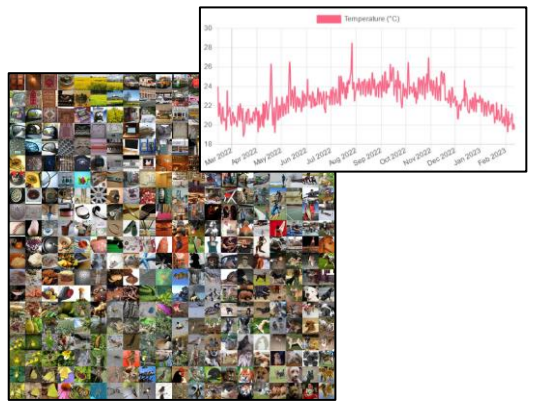
Edge Deployment Strategies

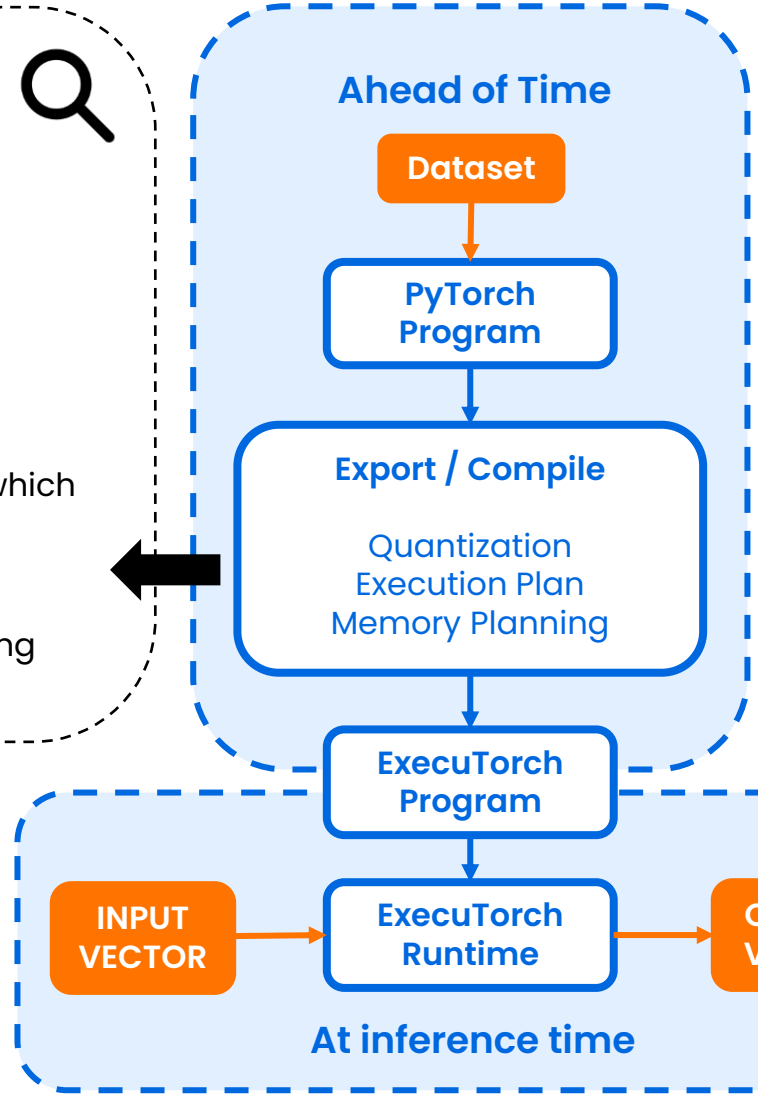


From PyTorch to ExecuTorch

Export / Compile Lowering:

1. **PyTorch Program (nn.Module)**
   PyTorch Python program for training.
2. **ATen Dialect**
   Exported IR graph and
   entry point to ExecuTorch.
3. **Edge Dialect**
   Specializations useful for edge devices which can further run on different hardware.
4. **(optional) Backend Dialect**
   Special variant of edge dialect introducing hardware-awareness.





Training is done in PyTorch typically on server-class machines, GPU farms, in the cloud, etc.

Inference (run) happens on the device. ExecuTorch runs the generated *.pte file.

https://github.com/pytorch/executorch/tree/main/backends/nxp

Challenges of Running a Model on the Edge

Running a model on CPU (x86/aarch64) typically **works right away**. Latency or power are not optimal.

Splitting workload between different platforms (CPU/GPU/NPU/DSP).

Quantization of the model to utilize the accelerator (4/8/16b int) decreases accuracy.

NPU driver does not fully support the **inference framework** (PyTorch, TensorFlow).

- Lacks operator implementation support.
- Different operator definitions.
- Unsupported datatypes (mixed schemas).

The **N3-64 NPU** (eIQ Neutron NPU configuration) consists of 64 MAD blocks for up to **64 MACs/cycle** for 8b x 8b operations. With a maximum frequency of **325 MHz**, the NPU provides up to **41.6 GOPS** (325 x 2 x 64 = 41.6).

Model	Arm Cortex-M33 only	eIQ Neutron NPU fully offloaded	Increase
MLPerf Tiny AD01	2.3 ms	0.134 ms	**17.5x**
MLPerf Tiny KWS	28.5 ms	0.384 ms	**74.1x**
MLPerf Tiny Resnet	121.2 ms	0.717 ms	**169.1x**
MLPerf Tiny VWW	91.4 ms	0.966 ms	**94.6x**

CPU vs. NPU Acceleration using TF Lite Micro on i.MX RT 700 @ 325 MHz (2025)
See https://mlcommons.org/ for benchmarking details.

Platform	Latency	Notes
ARM Cortex-M33 CPU fp32 I/O	77694.018 ms	-
eIQ Neutron NPU fp32 I/O	257.163 ms	-
eIQ Neutron NPU int8 I/O	38.917 ms	fully quantized to int8 including I/O
ARM Cortex-M33 CPU + eIQ Neutron NPU fp32 I/O	124.617 ms	2.1x faster vs. NPU-only variant (justifying mixed backends)

Experimental ExecuTorch Backend latency for fp32/int8 on i.MX RT700 @ 325 MHz
MobileNet v2 224x224x3; ExecuTorch 1.0+ (2026)

From TensorFlow to TF Lite Micro

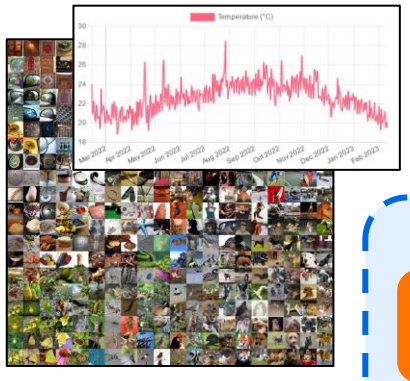
```python
import tensorflow as tf

# Create a model using high-level tf.keras.* APIs
model = tf.keras.models.Sequential([
    # Layers
])

# Compile and train the model.
model.compile(optimizer='sgd', loss='mean_squared_error')
model.fit(x=[-1, 0, 1], y=[-3, -1, 1], epochs=5)
# (to generate a SavedModel)
tf.saved_model.save(model, "saved_model_keras_dir")

# Convert the model.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```



DATASET

Ahead of Time

TensorFlow

TensorFlow Lite Converter

Quantization
Export to serialized format (*.tflite)

Moved to: LiteRT

TensorFlow Lite

Convert to a C array & store in read-only memory

At inference time

INPUT TENSOR

TensorFlow Lite
C++ library (MicroInterpreter)

OUTPUT TENSOR

Protocol Buffers (Protobuf) is an open-source cross-platform data format used to serialize structured data. It is useful in developing programs that communicate with each other over a network or for storing data.

protobuf
Protocol Buffers

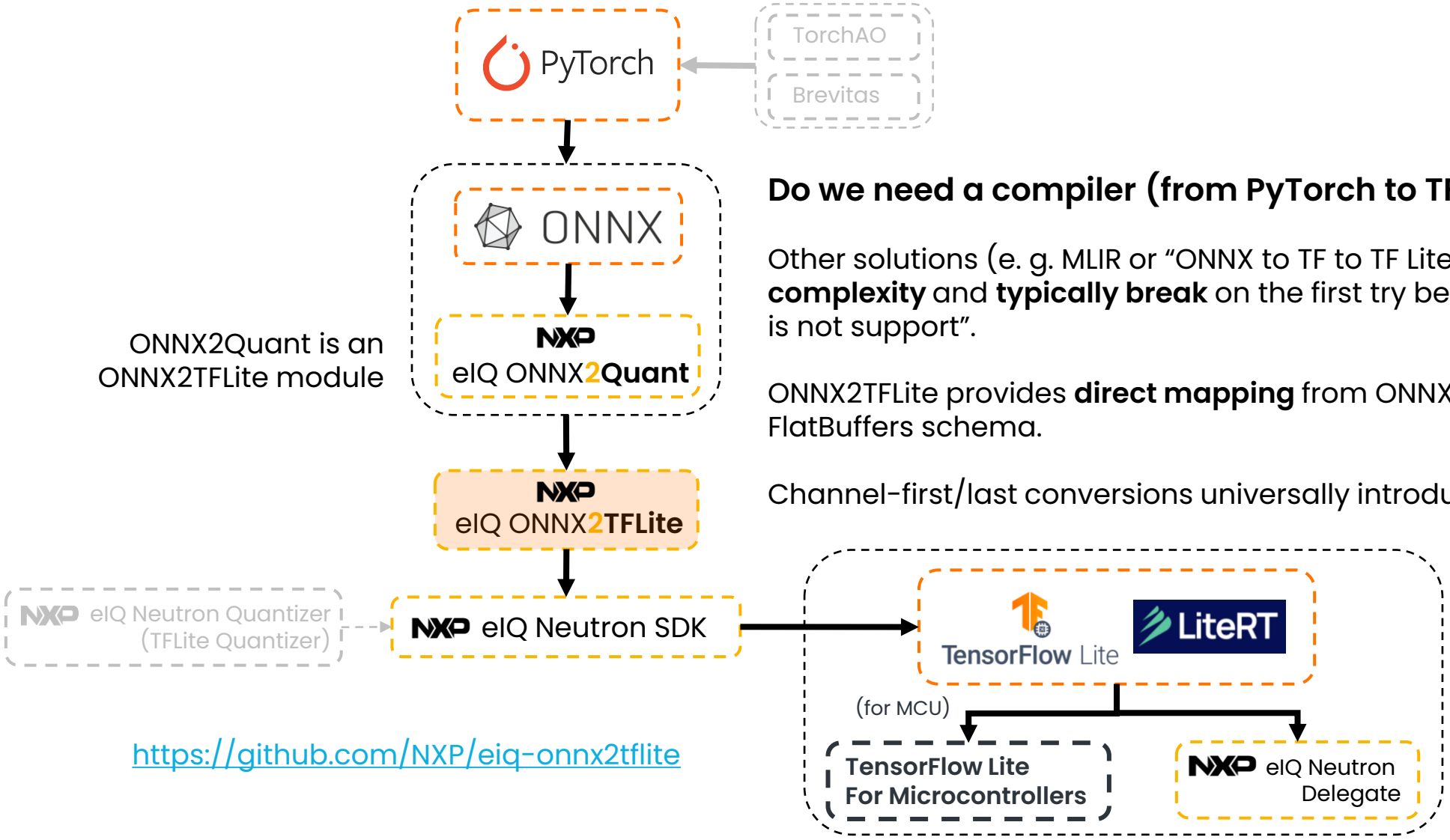
https://github.com/protocolbuffers/protobuf

FlatBuffers is an efficient cross platform serialization library for C++, C#, C, Go, Java, Kotlin, JavaScript, Lobster, Lua, TypeScript, PHP, Python, Rust and Swift. It was originally created at Google for game development and other performance-critical applications.

FlatBuffers

https://github.com/google/flatbuffers

eIQ ONNX2Quant & eIQ ONNX2TFLite



https://github.com/NXP/eiq-onnx2tflite

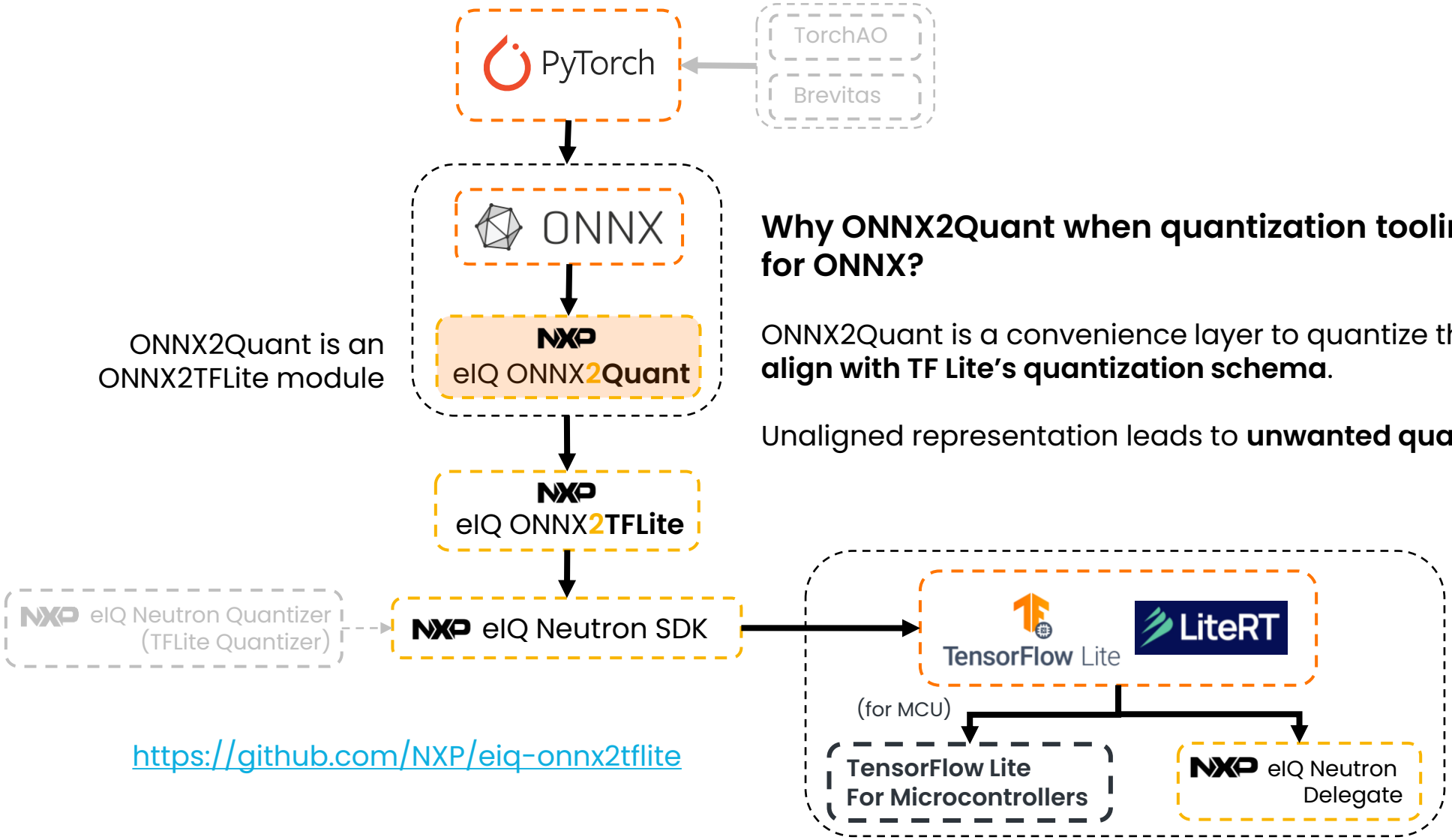
Do we need a compiler (from PyTorch to TF Lite)?

Other solutions (e. g. MLIR or "ONNX to TF to TF Lite") have **high complexity** and **typically break** on the first try because "something is not support".

ONNX2TFLite provides **direct mapping** from ONNX to TF Lite FlatBuffers schema.

Channel-first/last conversions universally introduce **transposes**.

eIQ ONNX2Quant & eIQ ONNX2TFLite



https://github.com/NXP/eiq-onnx2tflite

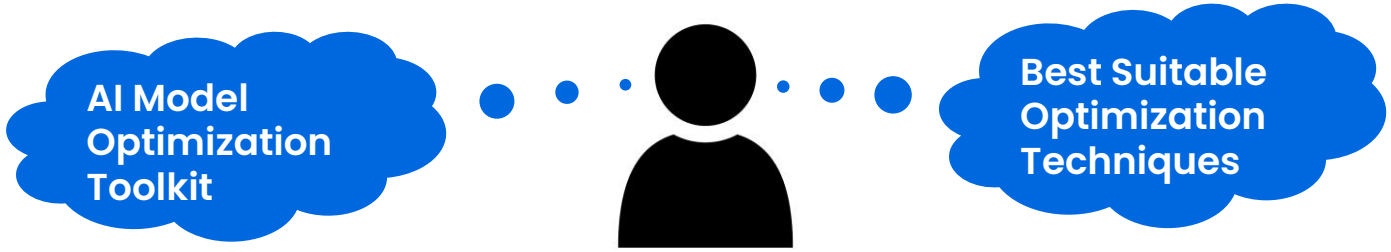
Why ONNX2Quant when quantization tooling already exists for ONNX?

ONNX2Quant is a convenience layer to quantize the ONNX model to **align with TF Lite's quantization schema**.

Unaligned representation leads to **unwanted quant & dequant** ops.

ONNX Optimization Framework - eIQ Olive

"Given a model and targeted hardware, Olive (abbreviation of **O**nnx **LIVE**) composes the **best suitable optimization techniques** to output the most efficient ONNX model(s) for inferencing on the cloud or edge, while taking a set of constraints such as accuracy and latency into consideration."



"Olive: The **AI Model Optimization Toolkit** for the ONNX Runtime"

https://microsoft.github.io/Olive/
https://github.com/microsoft/Olive

Fork → https://github.com/nxp/eiq-olive

Shoutout to eIQ AI Toolkit

eIQ ONNX2TFLite (Quant), **eIQ Olive** are open-sourced:

- https://github.com/NXP/eiq-onnx2tflite
- https://github.com/nxp/eiq-olive

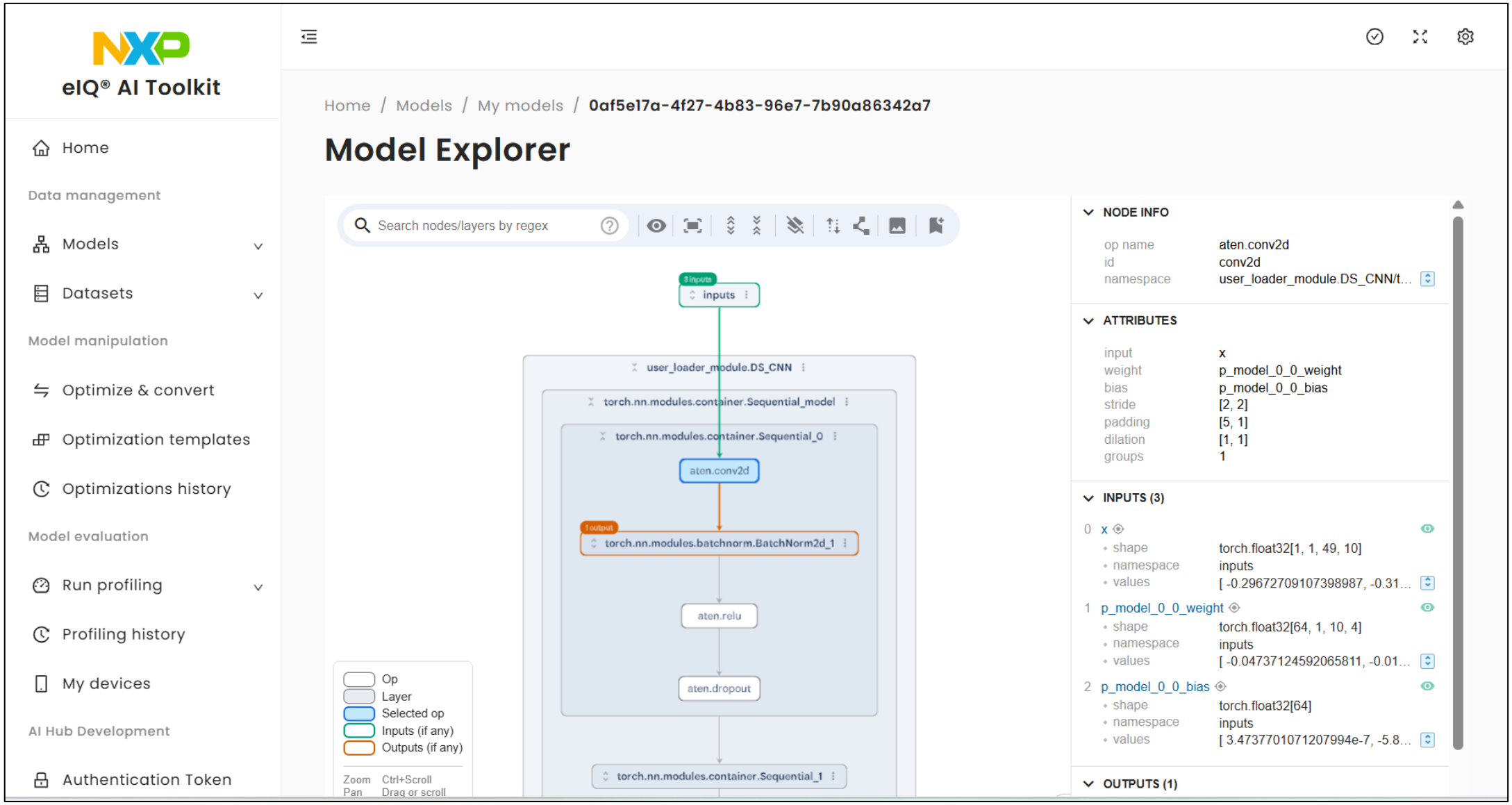
eIQ Neutron SDK (for accelerator enablement) is available on eIQ PyPi & download:

- https://eiq.nxp.com/repository/eiq-neutron-sdk
- http://nxp.com/eiq/toolkit

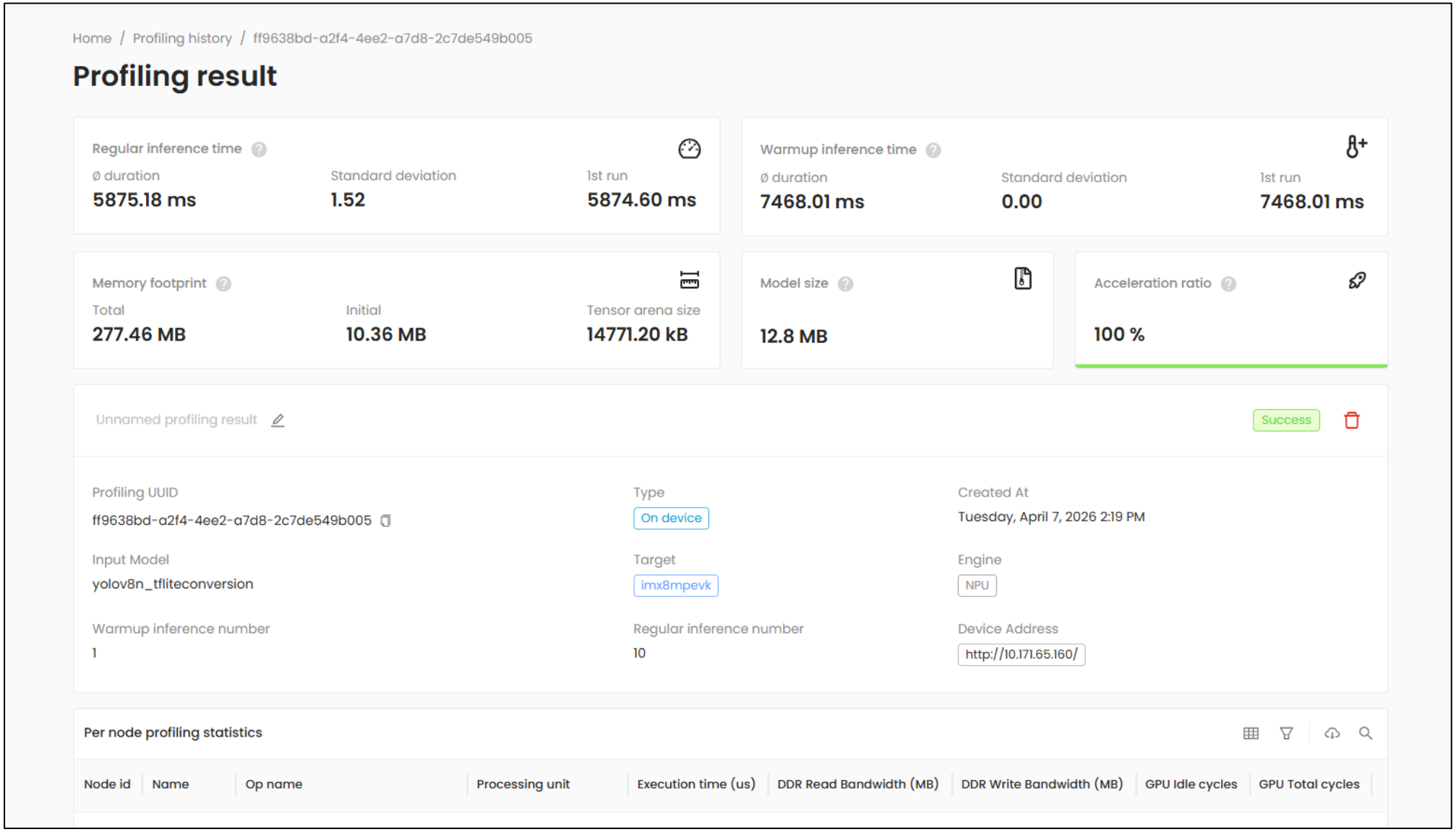
All of these (and more tools) will be available together in the new SDK for model deployment and optimization - **eIQ AI Toolkit!**

- Estimated release **mid-to-end April 2026** on Docker Hub
- Interfaces: GUI, REST API, Python (for individual packages)

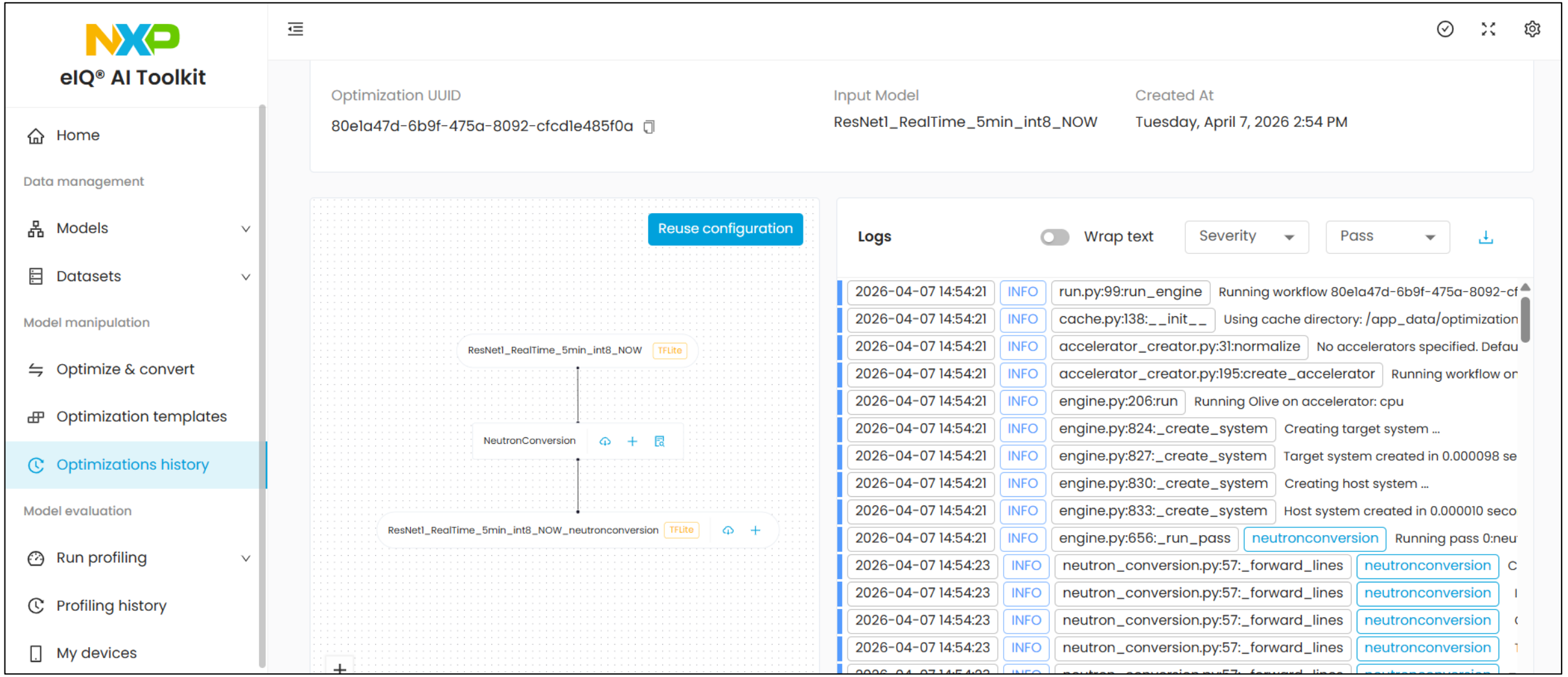
Model Visualization in eIQ AI Toolkit



Profiler in eIQ AI Toolkit



Optimization Pipeline in eIQ AI Toolkit



Pavel Macenauer

R&D Manager @ NXP Semiconductors

pavel.macenauer@nxp.com

Jakub Salamon
Lukas Sztefek
Martin Knutelsky
Barbora Nemcekova
Michaela Sezimova
Jakub Kasem
Irina Korchakova

nxp.com