



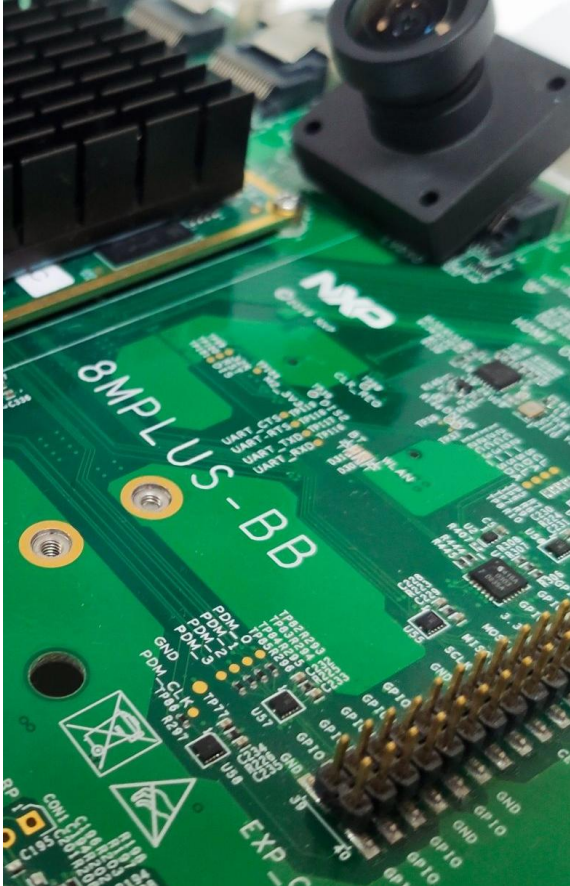
Every Millisecond Counts

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

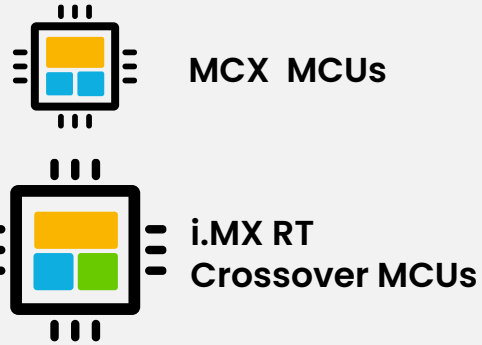
Pavel Macenauer
R&D Manager @ NXP Semiconductors

PyTorch Conference Europe 2026

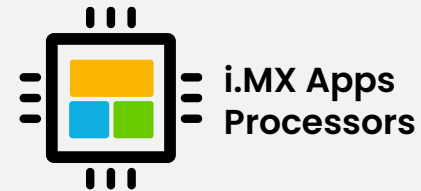
Edge Device Classes



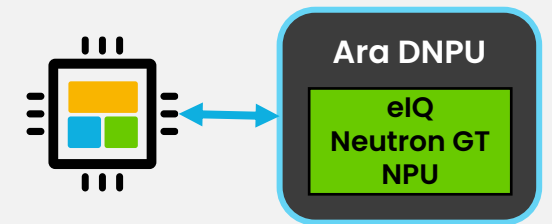
Microcontrollers



Microprocessors



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

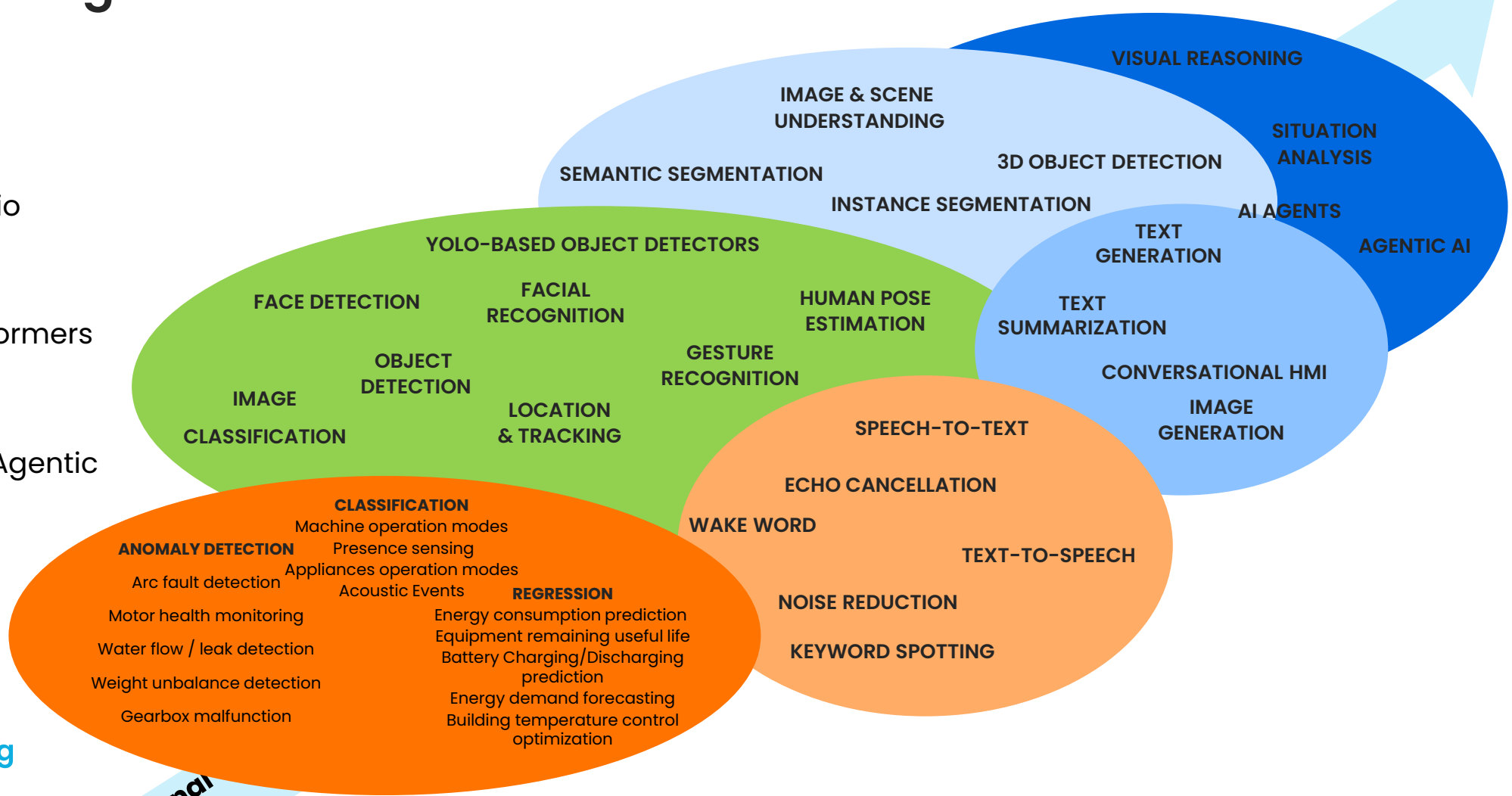
Ara Discrete NPUs
High-performance AI co-processors enabling platform scalability and deployment of Agentic-AI and Physical-AI on the edge

CPU NPU DSP/GPU and other cores

Scalability for Edge AI Use Cases

Legend

- Time Series
- Speech / Audio
- Vision (CNNs)
- Vision Transformers
- LLMs / SLMs
- VLMs / VLAs / Agentic



Our PTC Lightning Talk focus

Computational Demand*



MCX MCUs



i.MX RT Crossover MCUs



i.MX Apps Processors



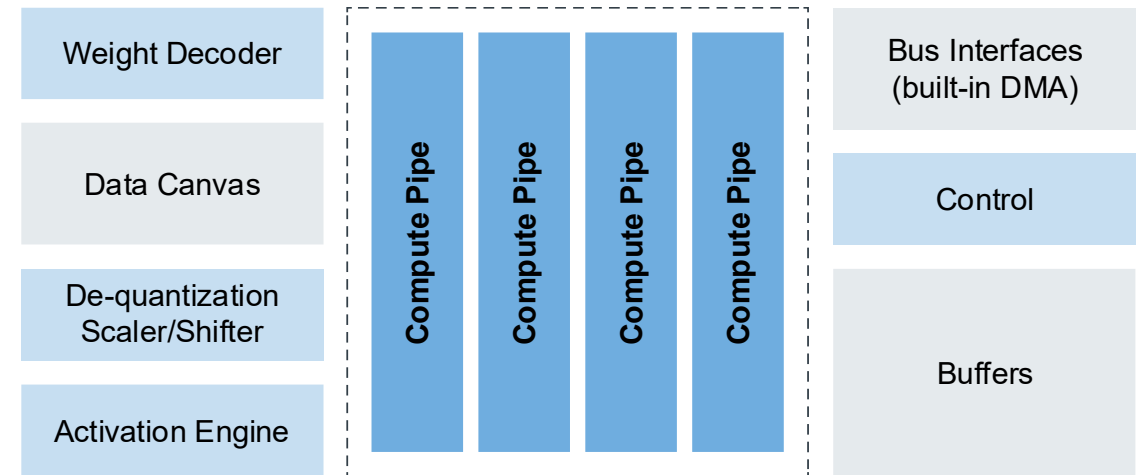
Ara DNPUs

*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.

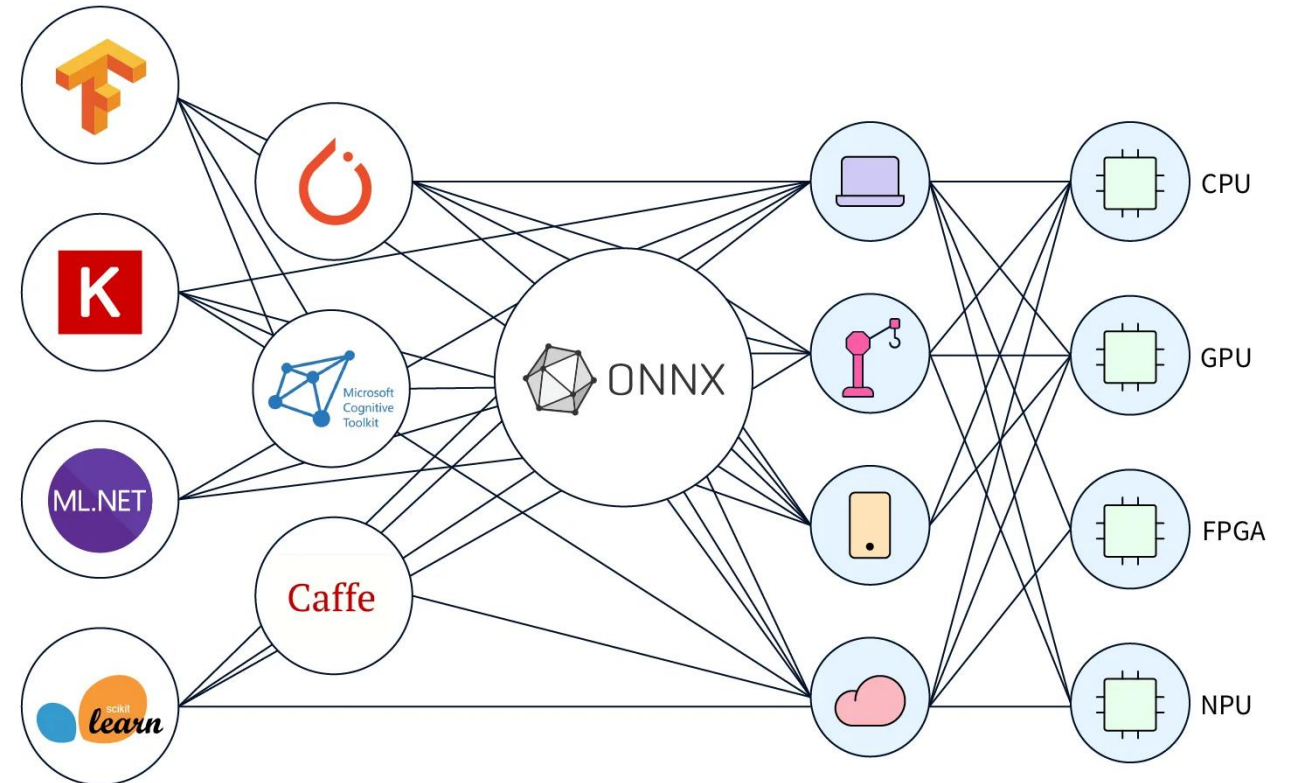
eIQ Neutron NPU Paper: <https://arxiv.org/html/2509.14388v1> "eIQ Neutron: Redefining Edge-AI Inference with Integrated NPU and Compiler Innovations"
eIQ Neutron NPU Product: <https://www.nxp.com/applications/technologies/ai-and-machine-learning/eiq-neutron-npu:EIQ-NEUTRON-NPU>
i.MX95 Application Processors: <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.

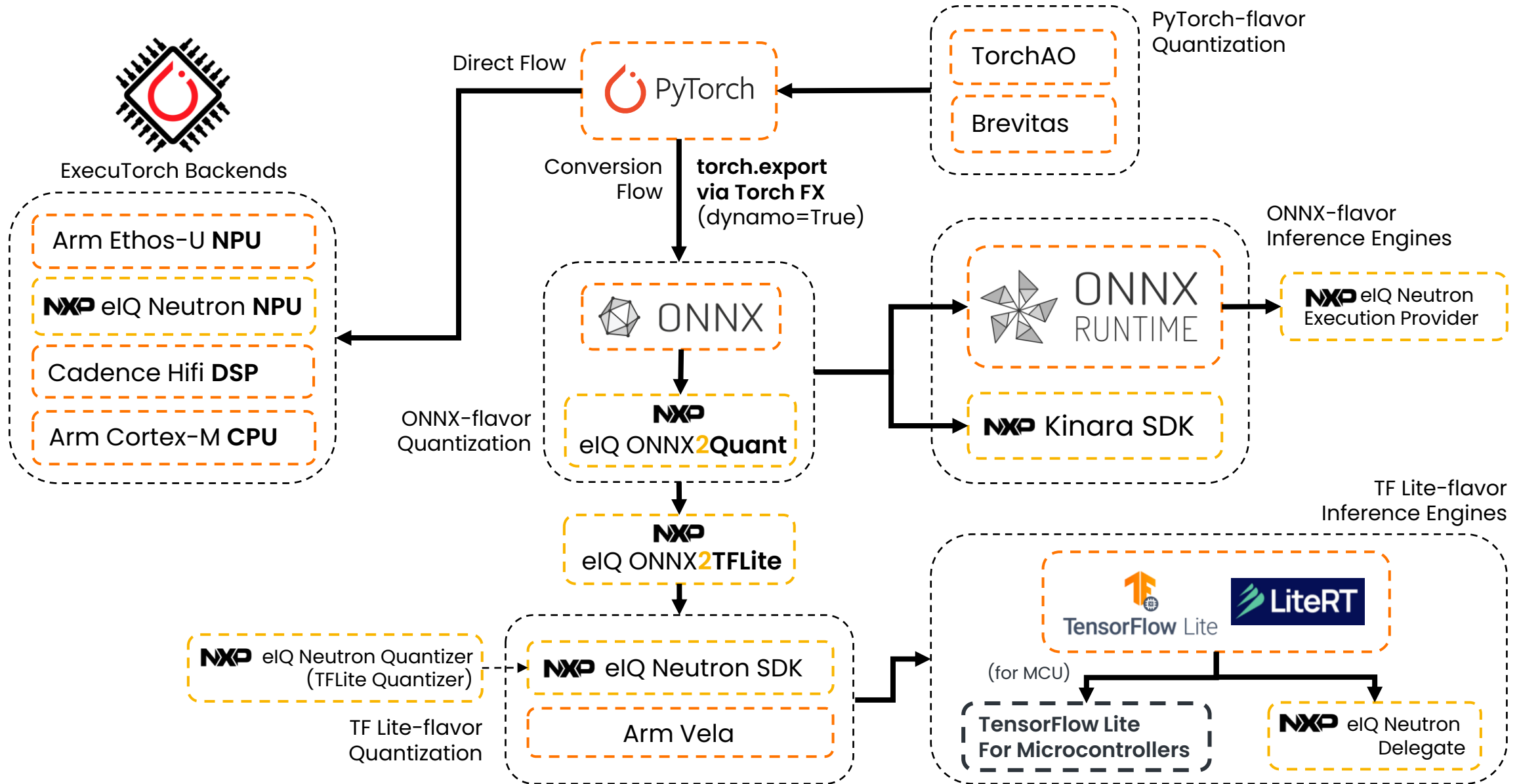
It defines:

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



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

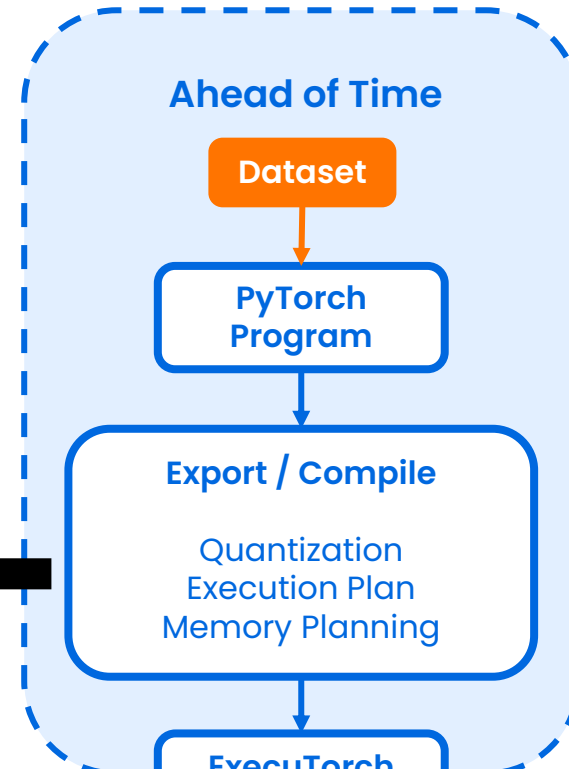
Edge Deployment Strategies




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.

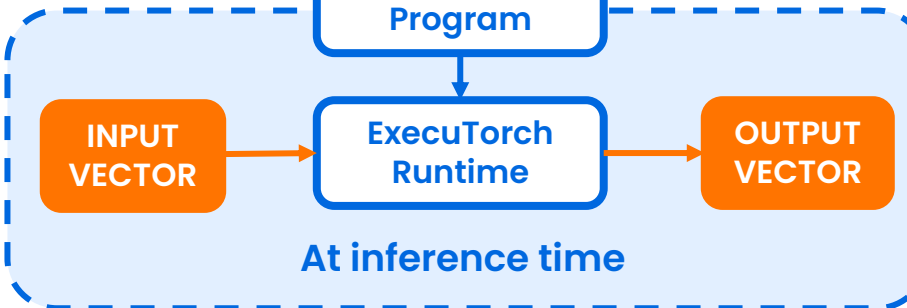


Training is done in  PyTorch typically on server-class machines, GPU farms, in the cloud, etc.

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** ($325 \times 2 \times 64 = 41.6$).

| 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

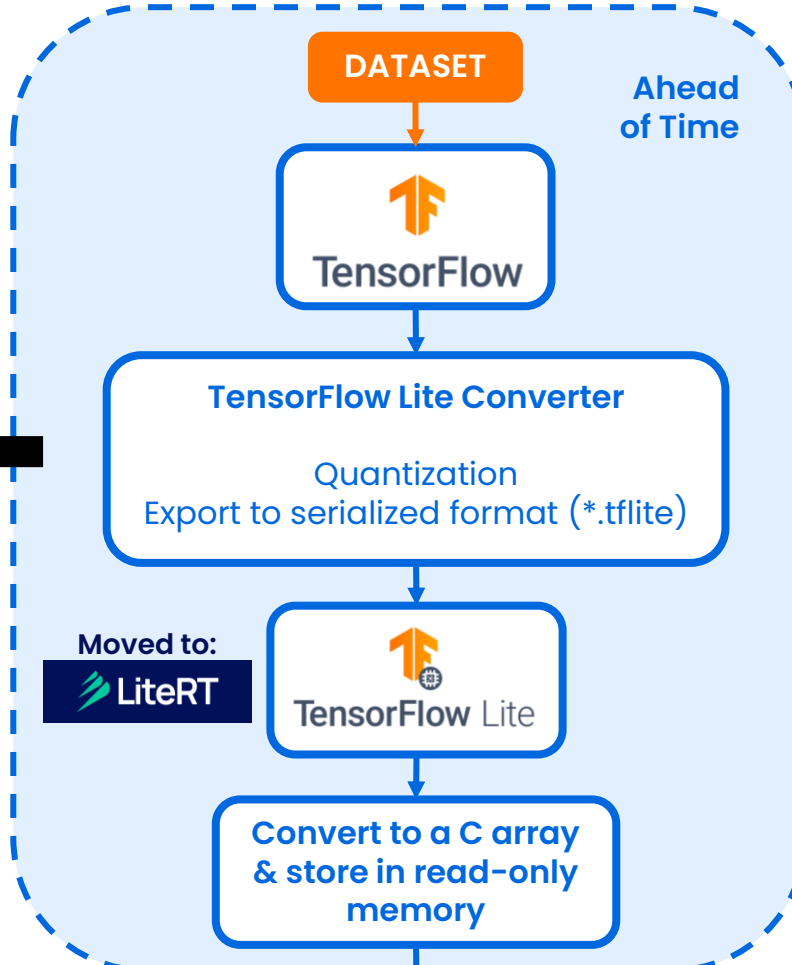
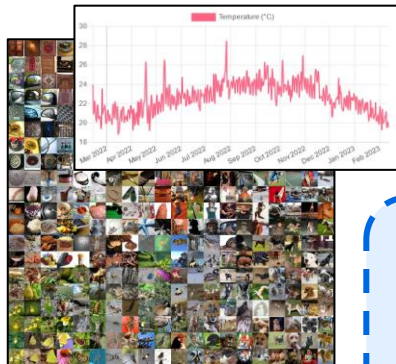
```
import tensorflow as tf
```

```
# 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)
```



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.



<https://github.com/protocolbuffers/protobuf>

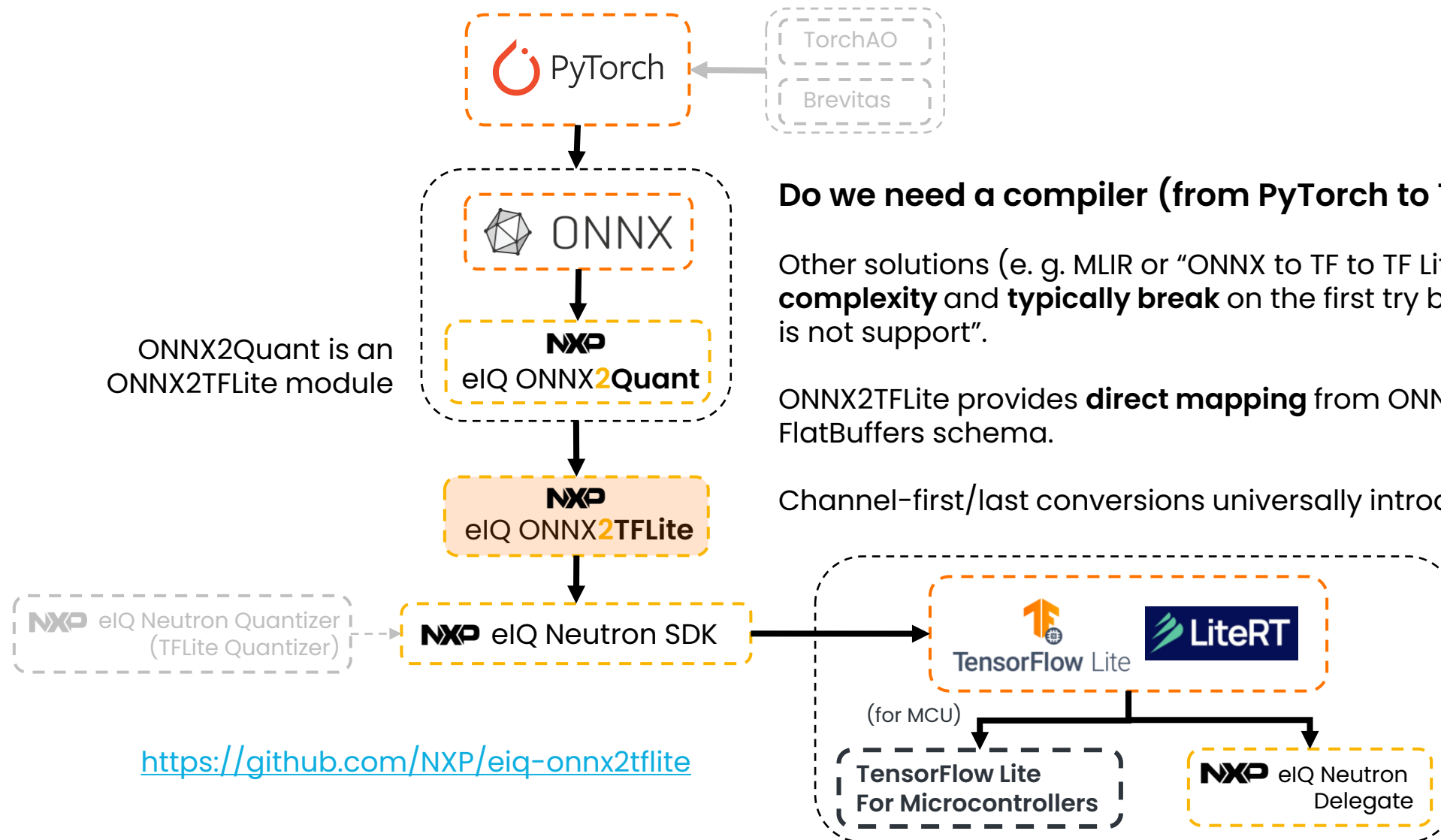
FlatBuffers is an efficient cross platform serialization library for C++, C#, C, Go, Java, Kotlin, JavaScript, Lobster, Lua, TypeScript, PHP, Python, Rust and Swift. It was originally created at Google for game development and other performance-critical applications.



<https://github.com/google/flatbuffers>



eIQ ONNX2Quant & eIQ ONNX2TFLite



ONNX2Quant is an ONNX2TFLite module

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

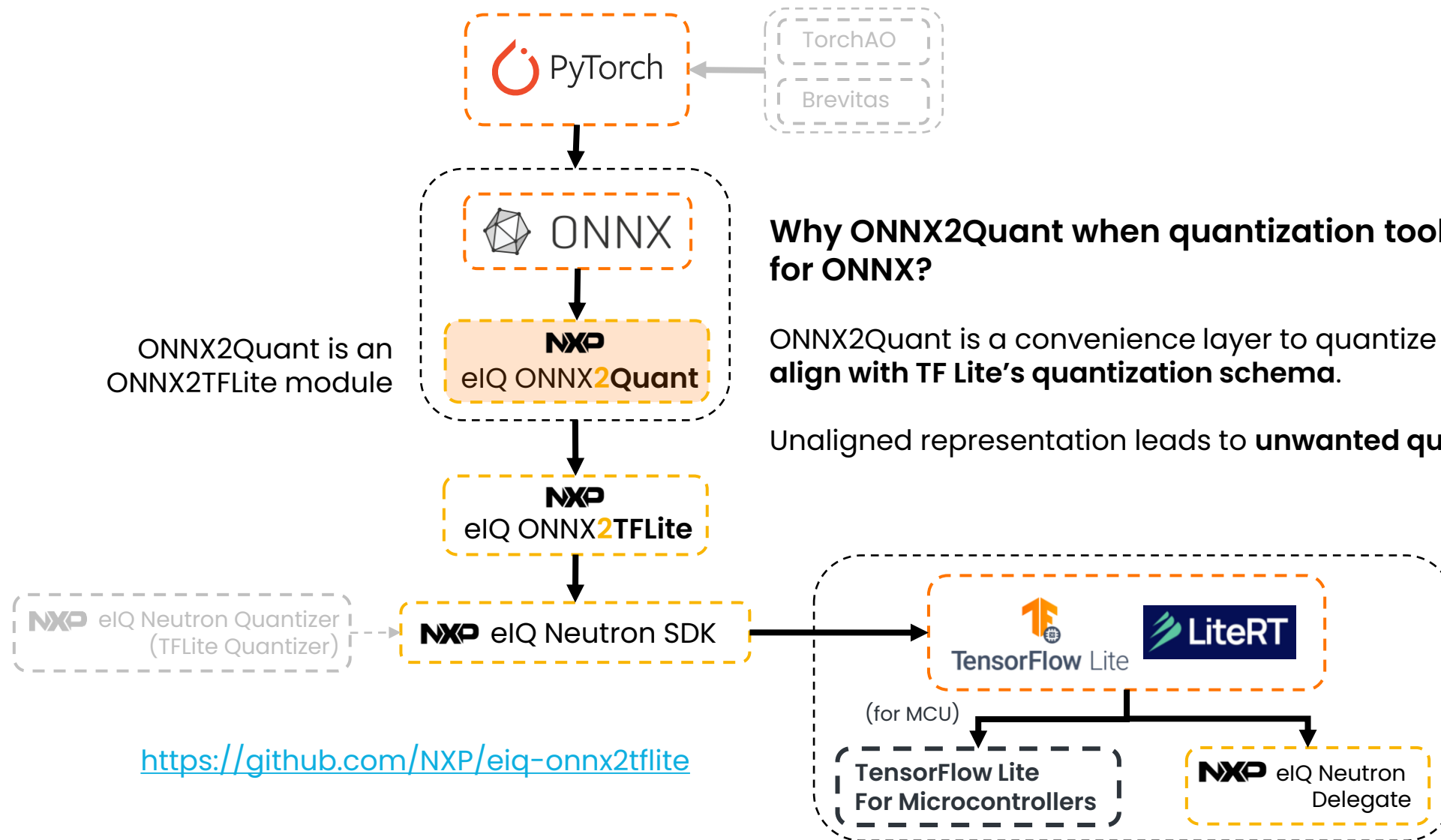
Other solutions (e. g. MLIR or “ONNX to TF to TF Lite”) have **high complexity** and **typically break** on the first try because “something is not support”.

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

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

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

eIQ ONNX2Quant & eIQ ONNX2TFLite



ONNX2Quant is an ONNX2TFLite module

Why ONNX2Quant when quantization tooling already exists for ONNX?

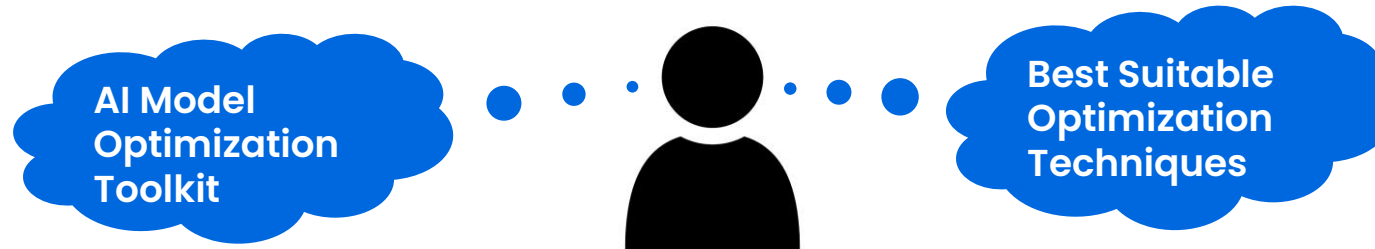
ONNX2Quant is a convenience layer to quantize the ONNX model to align with TF Lite's quantization schema.

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

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

ONNX Optimization Framework – eIQ Olive

“Given a model and targeted hardware, Olive (abbreviation of **Onnx 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>



<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

The screenshot displays the eIQ AI Toolkit interface. On the left is a navigation sidebar with sections: Home, Data management, Models, Datasets, Model manipulation, Model evaluation, and AI Hub Development. The main area is titled 'Model Explorer' and shows a hierarchical view of a model named 'user_loader_module.DS_CNN'. The diagram highlights a specific layer, 'aten.conv2d', which is selected. A legend at the bottom left explains the visual encoding: Op (white), Layer (light blue), Selected op (blue), Inputs (green), and Outputs (orange). A toolbar above the diagram provides search and manipulation functions. On the right, a 'NODE INFO' panel details the selected layer's attributes and inputs.

Home / Models / My models / 0af5e17a-4f27-4b83-96e7-7b90a86342a7

Model Explorer

Search nodes/layers by regex

user_loader_module.DS_CNN

- torch.nn.modules.container.Sequential_model
 - torch.nn.modules.container.Sequential_0
 - aten.conv2d** (Selected op)
 - torch.nn.modules.batchnorm.BatchNorm2d_1 (Output)
 - aten.relu
 - aten.dropout
 - torch.nn.modules.container.Sequential_1

Legend:

- Op
- Layer
- Selected op
- Inputs (if any)
- Outputs (if any)

Zoom: Ctrl+Scroll
Pan: Drag or scroll

▼ NODE INFO

| | |
|-----------|--------------------------------|
| op name | aten.conv2d |
| id | conv2d |
| namespace | user_loader_module.DS_CNN/t... |

▼ ATTRIBUTES

| | |
|----------|--------------------|
| input | x |
| weight | p_model_0_0_weight |
| bias | p_model_0_0_bias |
| stride | [2, 2] |
| padding | [5, 1] |
| dilation | [1, 1] |
| groups | 1 |

▼ INPUTS (3)

- x**
 - shape: torch.float32[1, 1, 49, 10]
 - namespace: inputs
 - values: [-0.29672709107398987, -0.31...
- p_model_0_0_weight**
 - shape: torch.float32[64, 1, 10, 4]
 - namespace: inputs
 - values: [-0.04737124592065811, -0.01...
- p_model_0_0_bias**
 - shape: torch.float32[64]
 - namespace: inputs
 - values: [3.4737701071207994e-7, -5.8...

▼ OUTPUTS (1)

Profiler in eIQ AI Toolkit

Home / Profiling history / ff9638bd-a2f4-4ee2-a7d8-2c7de549b005

Profiling result

Regular inference time ⓘ

∅ duration: **5875.18 ms** Standard deviation: **1.52** 1st run: **5874.60 ms**

Warmup inference time ⓘ

∅ duration: **7468.01 ms** Standard deviation: **0.00** 1st run: **7468.01 ms**

Memory footprint ⓘ

Total: **277.46 MB** Initial: **10.36 MB** Tensor arena size: **14771.20 kB**

Model size ⓘ

12.8 MB

Acceleration ratio ⓘ

100 %

Unnamed profiling result [✎](#) Success

| | | |
|--------------------------------------|---------------------------|---|
| Profiling UUID | Type | Created At |
| ff9638bd-a2f4-4ee2-a7d8-2c7de549b005 | On device | Tuesday, April 7, 2026 2:19 PM |
| Input Model | Target | Engine |
| yolov8n_tfliteconversion | imx8mpevk | NPU |
| Warmup inference number | Regular inference number | Device Address |
| 1 | 10 | http://10.171.65.160/ |

Per node profiling statistics ⌵ ⌶ 🔄 🔍

| Node id | Name | Op name | Processing unit | Execution time (us) | DDR Read Bandwidth (MB) | DDR Write Bandwidth (MB) | GPU Idle cycles | GPU Total cycles |
|---------|------|---------|-----------------|---------------------|-------------------------|--------------------------|-----------------|------------------|
|---------|------|---------|-----------------|---------------------|-------------------------|--------------------------|-----------------|------------------|

Optimization Pipeline in eIQ AI Toolkit

The screenshot displays the eIQ AI Toolkit interface. On the left is a navigation sidebar with the following menu items: Home, Data management (Models, Datasets), Model manipulation (Optimize & convert, Optimization templates, Optimizations history), Model evaluation (Run profiling, Profiling history, My devices), and the NXP eIQ AI Toolkit logo at the top.

The main workspace is divided into three sections:

- Header:** Shows the Optimization UUID (80e1a47d-6b9f-475a-8092-cfcd1e485f0a), Input Model (ResNet1_RealTime_5min_int8_NOW), and Created At (Tuesday, April 7, 2026 2:54 PM).
- Diagram:** A flowchart on a grid background showing the optimization pipeline. It starts with a node labeled "ResNet1_RealTime_5min_int8_NOW" (TFLite), followed by a "NeutronConversion" node, and ends with a node labeled "ResNet1_RealTime_5min_int8_NOW_neutronconversion" (TFLite). A "Reuse configuration" button is located at the top right of the diagram area.
- Logs:** A log viewer on the right side with a "Wrap text" toggle, "Severity" and "Pass" dropdowns, and a download icon. The log entries are as follows:

| Timestamp | Severity | Module | Message |
|---------------------|----------|---|--|
| 2026-04-07 14:54:21 | INFO | run.py:99:run_engine | Running workflow 80e1a47d-6b9f-475a-8092-cf... |
| 2026-04-07 14:54:21 | INFO | cache.py:138: _init_ | Using cache directory: /app_data/optimization |
| 2026-04-07 14:54:21 | INFO | accelerator_creator.py:31:normalize | No accelerators specified. Defau... |
| 2026-04-07 14:54:21 | INFO | accelerator_creator.py:195:create_accelerator | Running workflow or... |
| 2026-04-07 14:54:21 | INFO | engine.py:206:run | Running Olive on accelerator: cpu |
| 2026-04-07 14:54:21 | INFO | engine.py:824: _create_system | Creating target system ... |
| 2026-04-07 14:54:21 | INFO | engine.py:827: _create_system | Target system created in 0.000098 se... |
| 2026-04-07 14:54:21 | INFO | engine.py:830: _create_system | Creating host system ... |
| 2026-04-07 14:54:21 | INFO | engine.py:833: _create_system | Host system created in 0.000010 seco... |
| 2026-04-07 14:54:21 | INFO | engine.py:656: _run_pass | neutronconversion Running pass 0:neu... |
| 2026-04-07 14:54:23 | INFO | neutron_conversion.py:57: _forward_lines | neutronconversion C |
| 2026-04-07 14:54:23 | INFO | neutron_conversion.py:57: _forward_lines | neutronconversion I |
| 2026-04-07 14:54:23 | INFO | neutron_conversion.py:57: _forward_lines | neutronconversion (|
| 2026-04-07 14:54:23 | INFO | neutron_conversion.py:57: _forward_lines | neutronconversion 1 |



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