
# Optimizing Edge AI with ExecuTorch and Arm® Ethos™ NPU

Erik Lundell 16-05-2025
Arm

# PyTorch and ExecuTorch

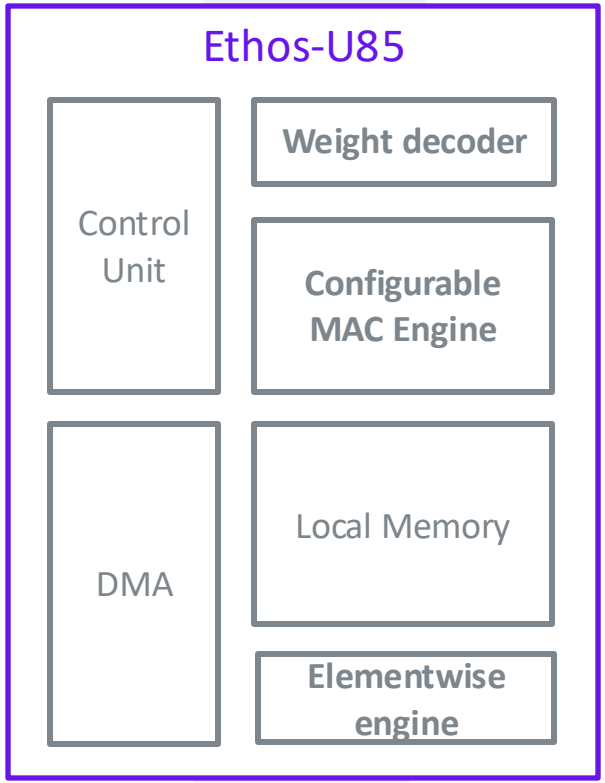
- Open-source Machine Learning framework by Meta.
- *“The debuggable, hackable and flexible framework”*
- 81 sub-repositories, 3500 contributors in 2024.
- ExecuTorch is one of them:
    - Alpha October 2023 – Proof of concept Ethos-U support included.
    - Currently Beta, v0.6 – Much improved Ethos-U support.
    - Full release scheduled October 2025 – Aiming for full Ethos-U support.

# Why ExecuTorch?

- ExecuTorch is PyTorch's complete solution for deploying AI on edge.
    - No python required at runtime, even bare-metal is supported.
- A solution to a fragmented ecosystem.
- Use the same framework for a multitude of platforms from high-end phones to micro controllers.
- While still getting optimized performance for each platform.

# Ethos-U Overview

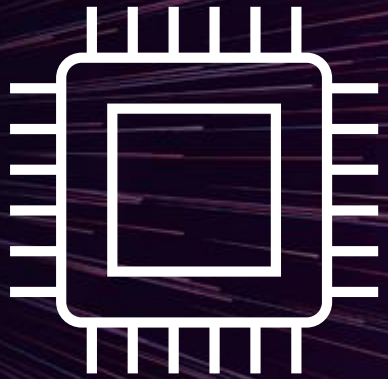
- Low-power NPU for IoT use cases.
- Three generations: Ethos-U55, Ethos-U65, and Ethos-U85.
- Flexible CPU-integration.
- Latest gen. offers up to 2048 MACs, 4 TOPs@1GHz.
- Capable of running full networks.

# So, let's deploy a PyTorch model on edge

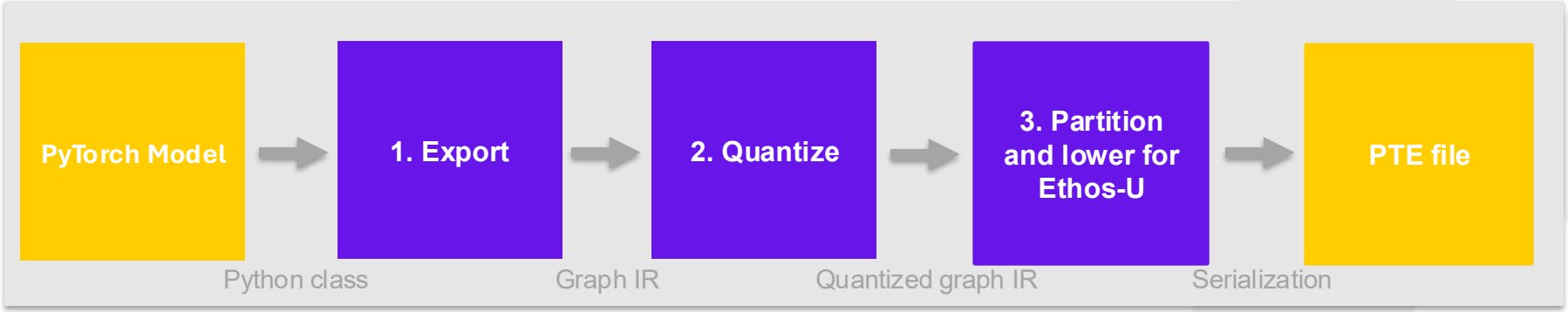
Goal:

- Execute inferences on an embedded platform.
- Accelerate the workload using Ethos-U.
- Simulate execution to ease development.



# Ahead of Time flow

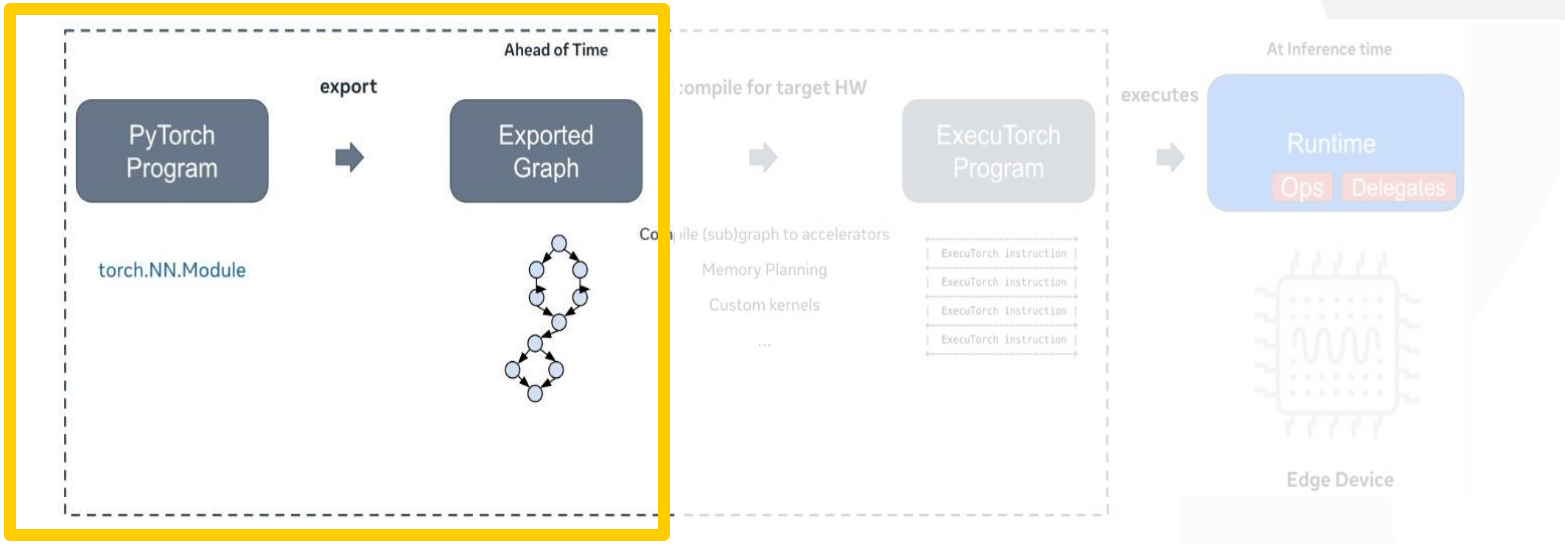
Produces a PTE file with one or more “delegate” sections, targeting specific hardware.



# 1. Exporting the model

- ExecuTorch utilizes PyTorch's export technology.
- Traces a complete graph Ahead of Time.
- Produces an Intermediate Representation (IR) graph in the ATen dialect.

```
graph_module = export(model, example_inputs)
```



# 2. Quantizing the model

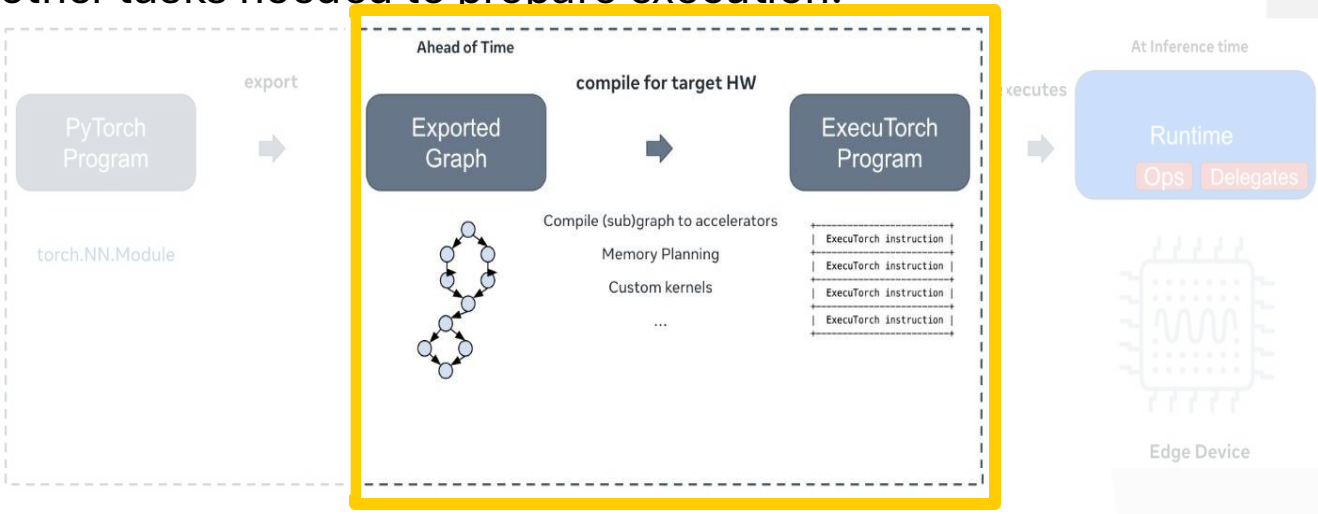
- The traced graph can be quantized automatically by PyTorch's export quantization.
- Necessary to execute on the integer-based Ethos-U.
- The EthosUQuantizer ensures proper quantization for the target hardware, while maintaining flexibility.

```
quantizer = EthosUQuantizer(compile_spec)
graph_module = prepare_pt2e(graph_module, quantizer)
graph_module(*example_inputs)
graph_module = convert_pt2e(graph_module)
```

# 3. Lowering to ExecuTorch program

- Massage our IR graph to an artifact to be consumed by the runtime.
- Partition the IR graph for different backends.
  - CPU/GPU/NPU.
- Compile partitions, creating delegates.
- ... other tasks needed to prepare execution.

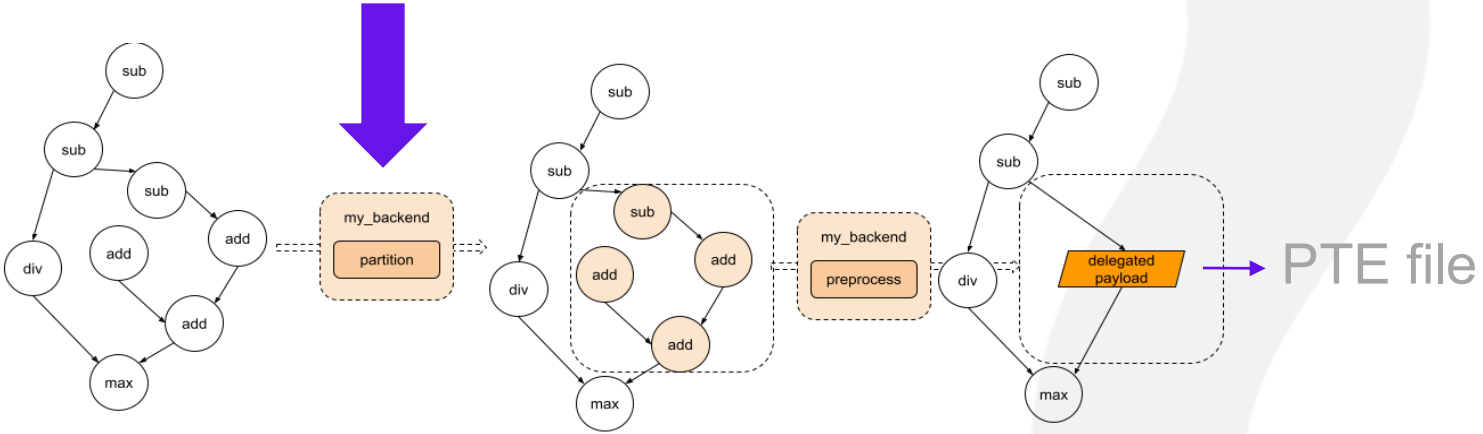
```
to_edge_transform_and_lower(
      model,
      [ethos_u_partitioner],
      compile_config
).to_executorch(executorch_config)
```



# 3a. Partitioning

- The partitioner selects parts of the IR graph to be consumed by a backend.
- EthosUPartitioner partitions everything that is supported unless explicitly specified.
- Possible to have many partitions of each backend, and many backends.

```
to_edge_transform_and_lower(
      model,
      [ethos_u_partitioner],
      compile_config
).to_executorch(executorch_config)
```

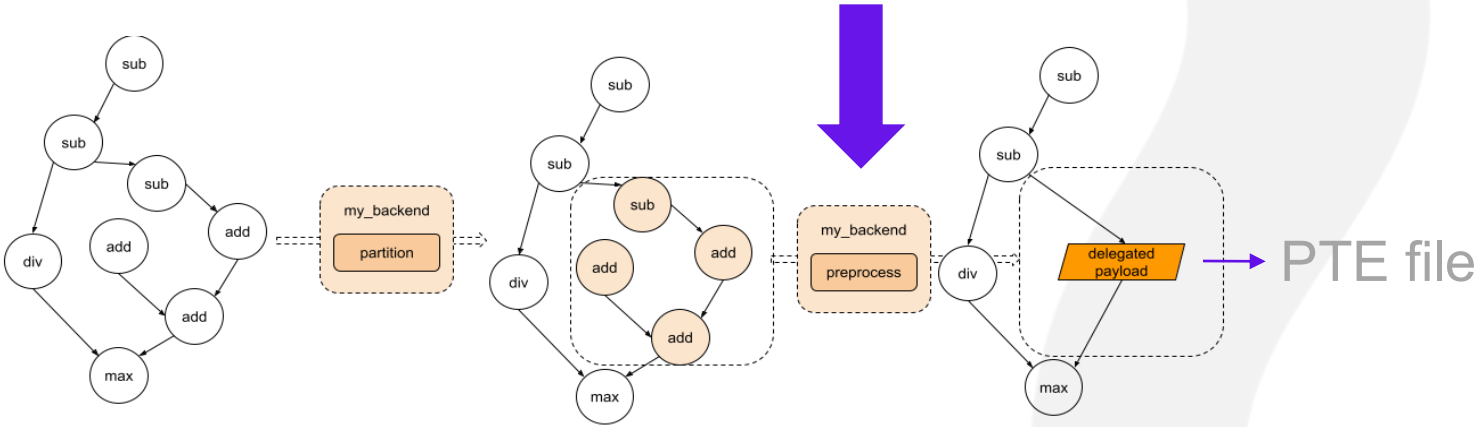



https://PyTorch.org/ExecuTorch/stable/compiler-delegate-and-partitioner.html

# 3b. Compiling

- Done per partition.
- The backend first lowers the IR to TOSA IR.
- It then compiles the TOSA IR to produce a command stream for Ethos-U.
- The command stream is baked into the PTE file.

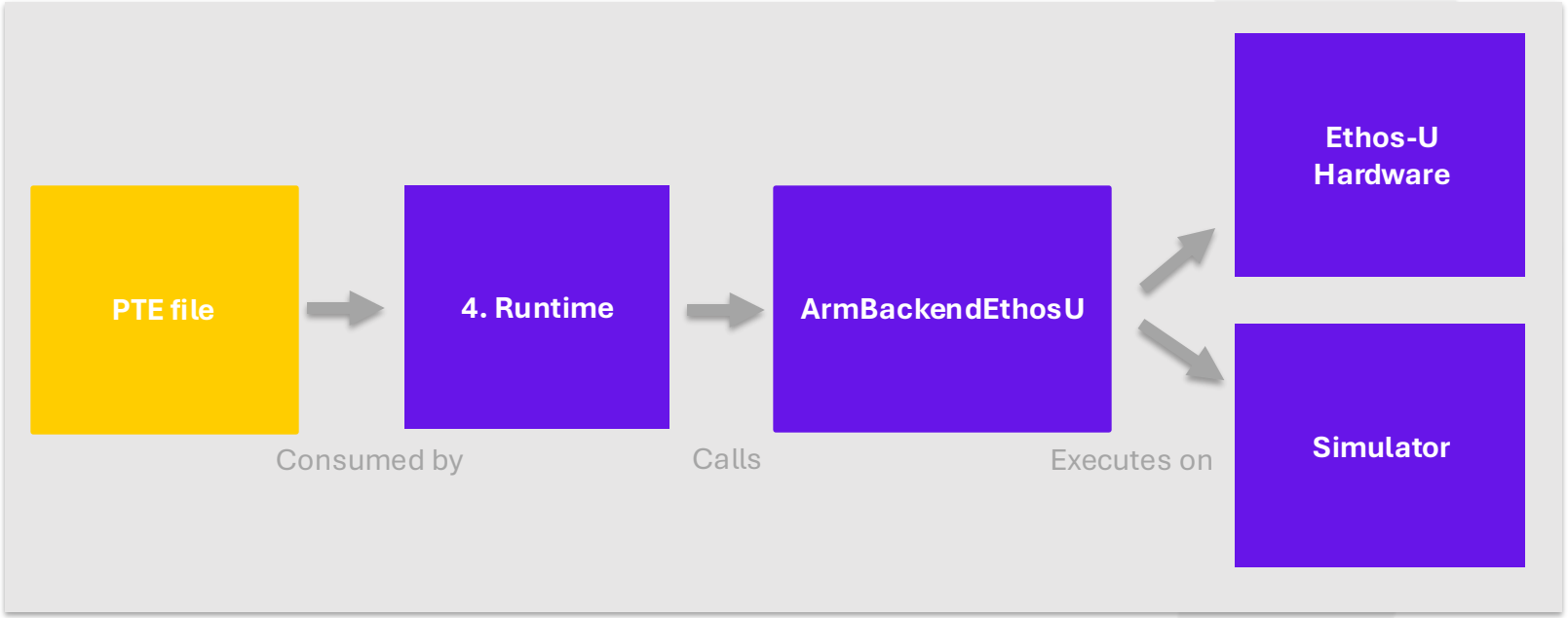
```
to_edge_transform_and_lower(
      model,
      [ethos_u_partitioner],
      compile_config
).to_executorch(executorch_config)
```




https://PyTorch.org/ExecuTorch/stable/compiler-delegate-and-partitioner.html

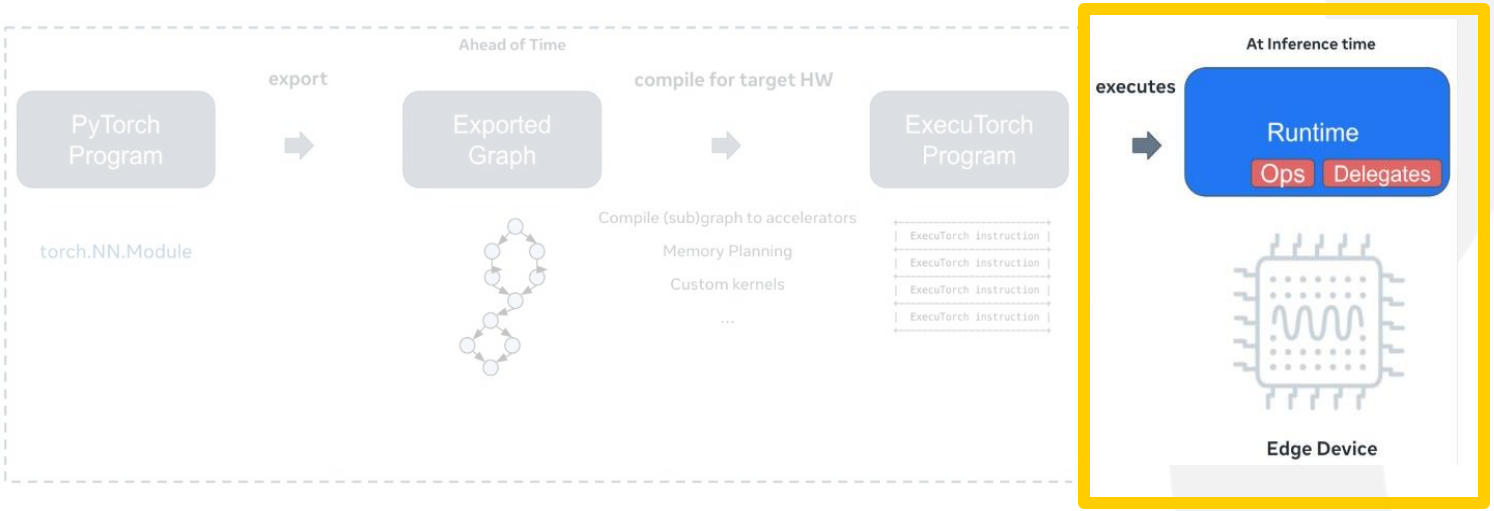
# Runtime flow

Consumes the PTE file, executing operators or delegating sections.



# 4. Running on bare metal

- A custom application uses the Executorch runtime, ~50kB.
- The runtime handles calling operator kernels and delegate backends.
- Example application in `examples/arm/arm_executor_runner.cpp`.
    - Reads a PTE file and input, runs a single inference, writes output.



# 5. Simulating an embedded system

- Arm provides simulators (Fixed Virtual Platforms, FVPs) of the Arm® Corstone™ reference systems.
- We have integrated the freely available:
  - Corstone-300 FVP: includes Arm® Cortex®-M55 and Ethos-U55.
  - Corstone-320 FVP: includes Arm® Cortex®-M85 and Ethos-U85.
- Check functionality and get a rudimentary performance estimate.
- Easy to use with the utility script backends/arm/scripts/run_fvp.sh

# Try it out!

- It is ready today, with examples.
- We are happy to receive feedback and issues being raised
  - Tag Github issues with **partner:arm**
  - There is an ExecuTorch discord with a channel **#arm-ethos-u**

- **Install ExecuTorch**
  - https://pytorch.org/executorch/stable/getting-started-setup.html
  - https://pytorch.org/executorch/stable/executorch-arm-delegate-tutorial.html
- **Learn more:**
  - https://learn.arm.com/learning-paths/embedded-and-microcontrollers/introduction-to-tinyml-on-arm/
  - https://github.com/PyTorch/ExecuTorch/tree/main/backends/arm/


# Thank You!

# Tensor Operator Set Architecture (TOSA)

- Arm developed standard set of approx. 70 operators commonly used in ML models.
- Profiles and extensions provide flexibility.
- Software and hardware agnostic.
- Representation agnostic.
- But with a toolchain.