PyTorch CONFERENCE EUROPE 2026

ExecuTorch on Microcontrollers

RJ Ascani
Meta Superintelligence Labs

Matthias Cremon
Meta Reality Labs

ExecuTorch: AI at the Edge

Run PyTorch models on devices

Physical AI

For phones, glasses, wearables,
personal computers...

Lightweight & Portable

Minimal runtime footprint
design for resource-constrained environments

General Availability and Production ready

Powers all Meta Technologies on-device inference.
External adoption and ecosystem integration.

12+

Hardware backends

Incl. XNNPACK, CUDA, Vulkan, Metal, Qualcomm NPU

TinyML

Power

Ambient always-on sensing
Extremely long battery-life

Cost

High performance MCUs under 10€
Cost effective MCUs under 1€

Low-connectivity

Bring intelligence to remote markets

Silicon Landscape

MCUs & DSPs

Arm Cortex-M
Cadence HiFi & Vision DSPs

Micro NPUs

Arm Ethos-U
NXP eIQ® Neutron

Resource constrained

Typically less than 10 MiB of RAM
Compute ranges from <10 GOPS to <5 TOPS
Power measured in milliWatts

ExecuTorch for TinyML

Device Aware Ahead of Time (AOT) Model Compilation

torch.export() uses the same technology used in PyTorch 2.x to capture PyTorch programs for fast execution

Easy to use quantization through TorchAO

Graph compilers of Export IR (exir) enables common optimization patterns

Customizable AOT workflow enables optimizing deployment for even better performance

ExecuTorch for TinyML

Optimized Runtime Architecture for Embedded Systems

OS-agnostic cross platform core runtime under 50 KiB

Hierarchical memory planning enables leveraging different memory types

Operator & dtype selective builds

Constant tensors zero copy from Flash

Quick Deployment Tips

- Quantize to INT8 or INT4 (limited support)
  - Pro Tip: Use `QuantizeInputs()` & `QuantizeOutputs()`
- Use selective build
  - `EXECUTORCH_SELECT_OPS_MODEL=your_model.pte`
  - `MAX_KERNEL_NUM=N`
- Enable size optimization flags
  - `EXECUTORCH_OPTIMIZE_SIZE=ON` (-Os, no exceptions or RTTI)
  - `EXECUTORCH_ENABLE_LOGGING=OFF` (~50 KiB)
  - `EXECUTORCH_ENABLE_PROGRAM_VERIFICATION=OFF` (~20 KiB)
- Use `BufferDataLoader` for firmware-embedded .pte files
- Enable dtype selective build when using portable kernels
  - `EXECUTORCH_ENABLE_DTYPE_SELECTIVE_BUILD=ON`
- Match your memory hierarchy to your hardware

Case Studies

Model	Task	Hardware	Flash Usage (Model)	Flash Usage (Code)	RAM Usage	Latency
Tiny MLP	Digit Classifier	RPi Pico 2 Cortex-M33	29.1 KiB	58 KiB	11 KiB	3.5 ms @ 150 MHz
DS-CNN	Keyword Spotting	Cortex-M55	49 KiB	133 KiB	26 KiB	6 ms @ 400 MHz
MV1-0.25	Visual Wake Words	Cortex-M55	291 KiB	90 KiB	98 KiB	21 ms @ 400 MHz
RNNT Encoder	Transcription	Cadence HiFi4	312 KiB	179 KiB	114 KiB	12 ms @ 500 MHz*
ResNet 18	Image Classifier	Cadence HiFi4	11,444 KiB	169 KiB	2,145 KiB	129 ms @ 500 MHz*
Conformer (7.5s window)	ASR	Arm Ethos-U85 & Cortex-M55	10,154 KiB	66 KiB	3,242 KiB	57 ms @ 1 GHz

* Simulated w/ 0 wait state memory

Partners, Products & Integrations

arm

Cortex-M CPU
Ethos-U NPU

cadence®

HiFi, Vision DSPs

NXP

eIQ Neutron NPUs

ALIF SEMICONDUCTOR

Ensemble
Balletto

Meta

Zephyr® Project

Zephyr RTOS

Expanding the reach of ExecuTorch for Microcontrollers

Expand HW Support

Products choose the best hardware for their use cases

SDK Integrations

Meet the embedded developers where they are

Now

Develop Arm Cortex-M and Cadence Vision DSP backends

Zephyr RTOS External Module, Arm ML Embedded Kit and CMSIS-Pack

Next

Improve CPU + NPU backend composability

Arduino

pytorch.org

Get started!

github.com/pytorch/executorch