tinyML® Foundation

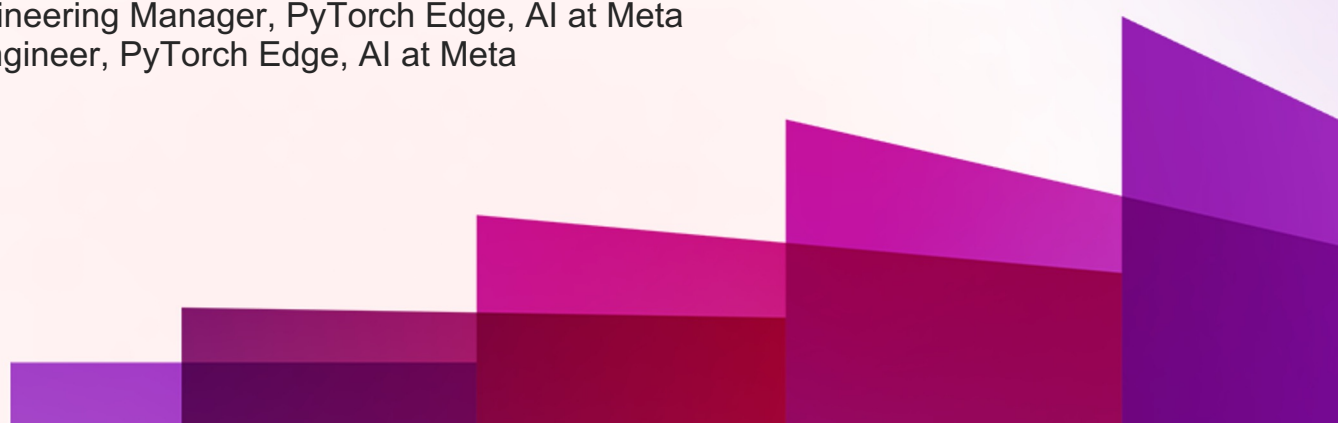
Enabling Ultra-low Power Machine Learning at the Edge

tinyML Summit April 22 - 24, 2024

www.tinyML.org

ExecuTorch: A PyTorch Software Stack for On-Device Machine Learning Execution

Mengtao (Martin) Yuan - Engineering Manager, PyTorch Edge, AI at Meta
Mergen Nachin - Software Engineer, PyTorch Edge, AI at Meta



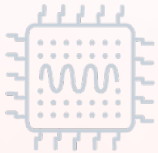
Opportunities

AR

Laptop

- Performance
- Privacy
- Personalized

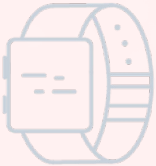
VR/MR



Embedded

- Enable new experiences

Mobile



Wearables

Executing PyTorch on the Edge =

PyTorch Edge

- ExecuTorch (alpha release)
  - Export and PyTorch 2.x-based
  - iOS, Android, Embedded
  - On-device generative AI support

⭐ *Shipped on Ray-Ban|Meta Smart Glasses and Quest 3 VR Headsets.*

⭐ *Supporting Meta Apps (we have begun the rollout of ExecuTorch with Instagram and are integrating with Meta's Family of Apps)*

- PyTorch Mobile (Legacy)
  - TorchScript-based
  - iOS, Android

Status and Timeline

Alpha - April 2024
Generative AI support

Preview / MVP - Oct 2023
Hardware partnerships
Early user feedback
Code access

Beta - Sept 2024
Hardening
Strong performance
Community involvement
Compatibility policy

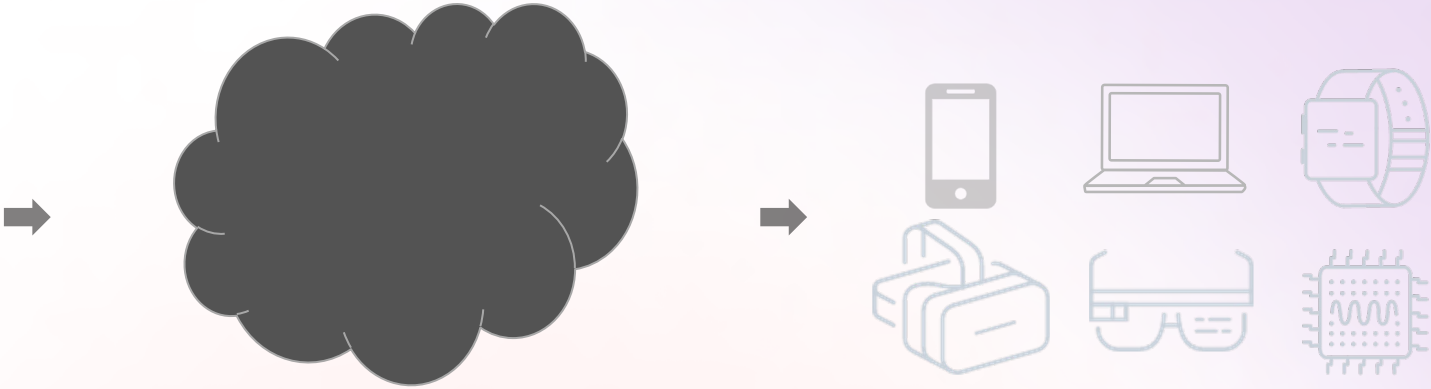
Focus

- **Portability**
  - Developers can run on wide range of devices. Runtime is 40KB
- **Productivity**
  - Developers can easily customize and deploy to production from original PyTorch models
- **Performance**
  - Provide good performance through compilation

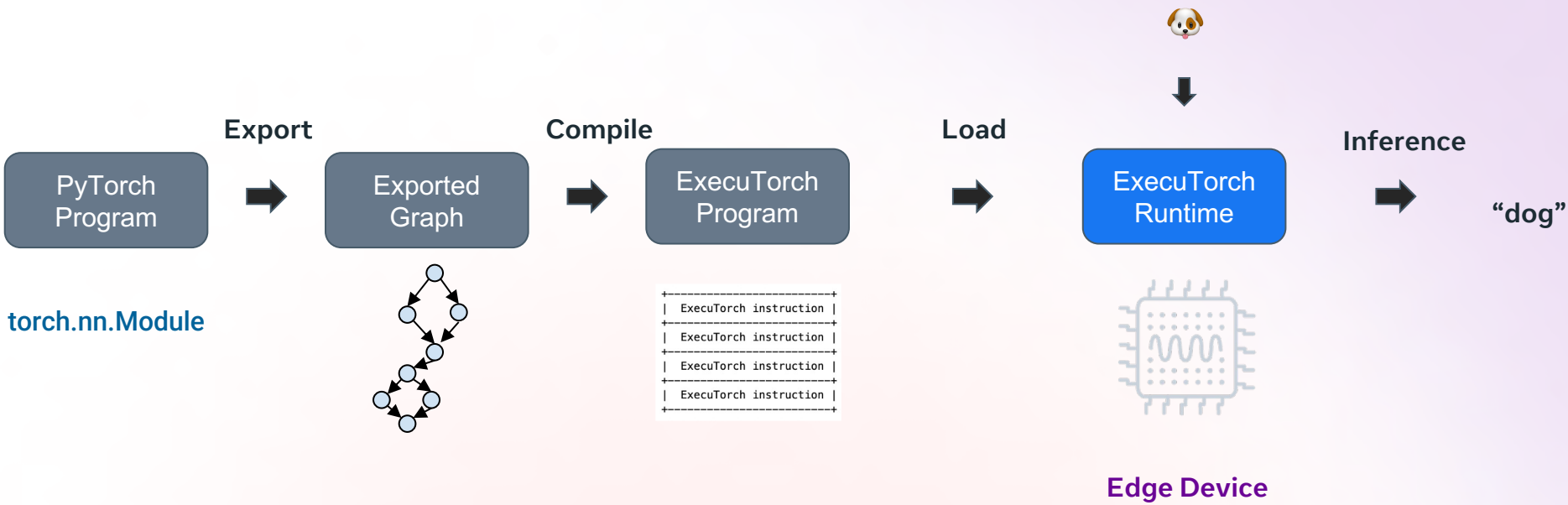
Problem Statement

PyTorch Program

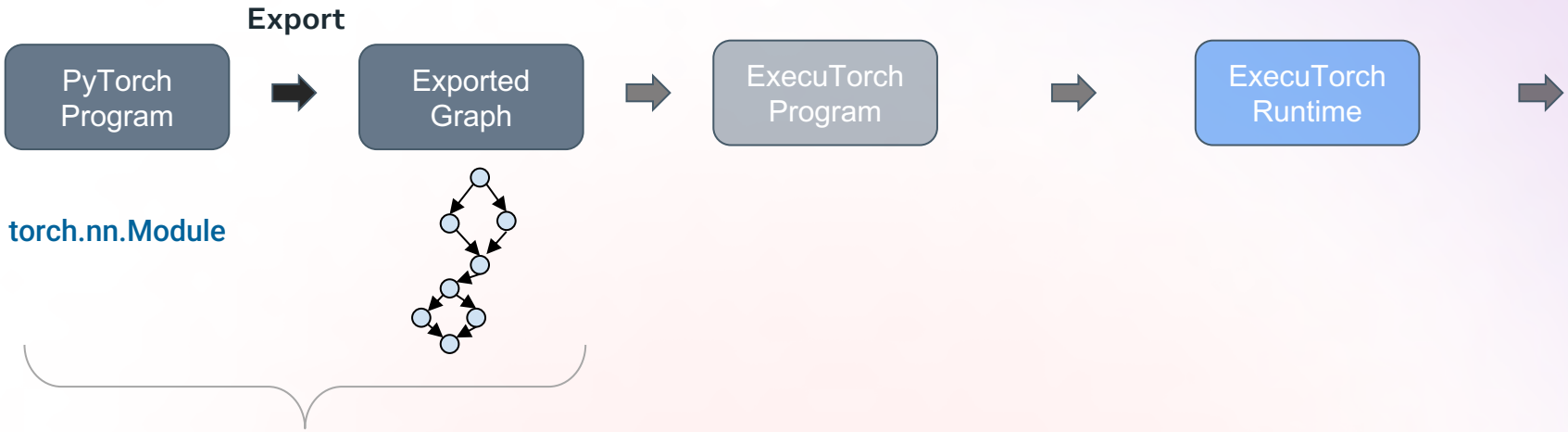
torch.nn.Module



ExecuTorch Overview



Benefit #1: Export-Based



PyTorch 2.x export mechanism

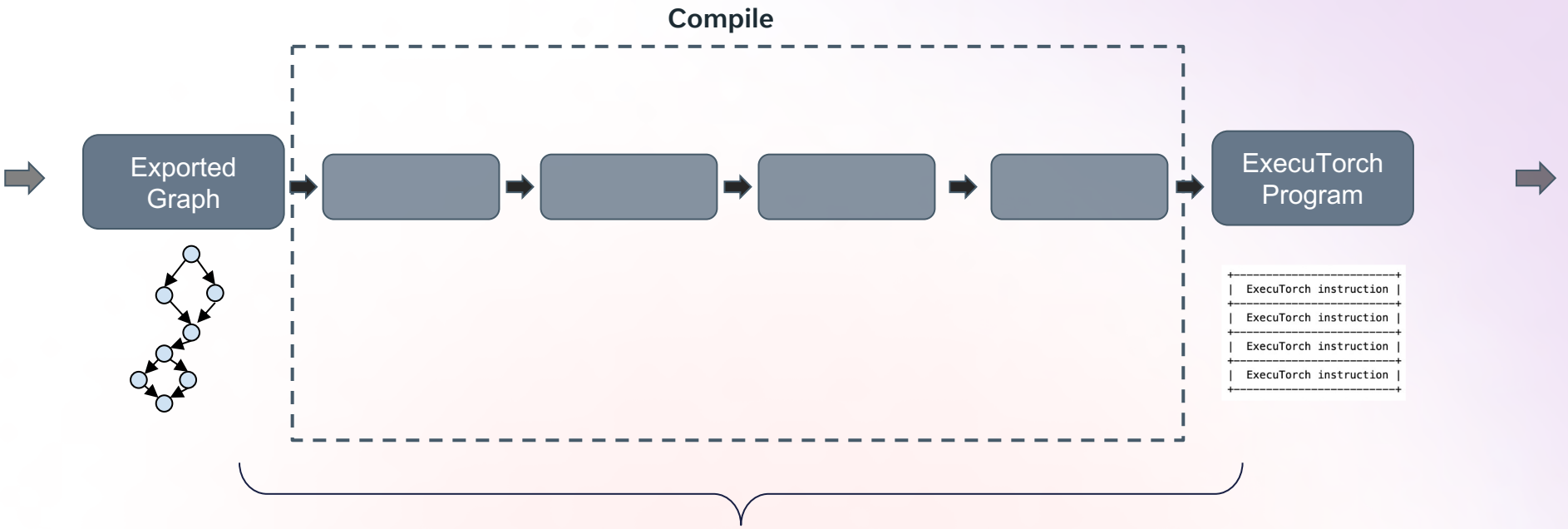
- **Export that is concise yet can capture wide range of dynamism**
- **Standardized Core ATen Operators (~300)**
- **Consistency between authoring and deployment**

Benefit #2: PyTorch Ecosystem

No intermediate conversion

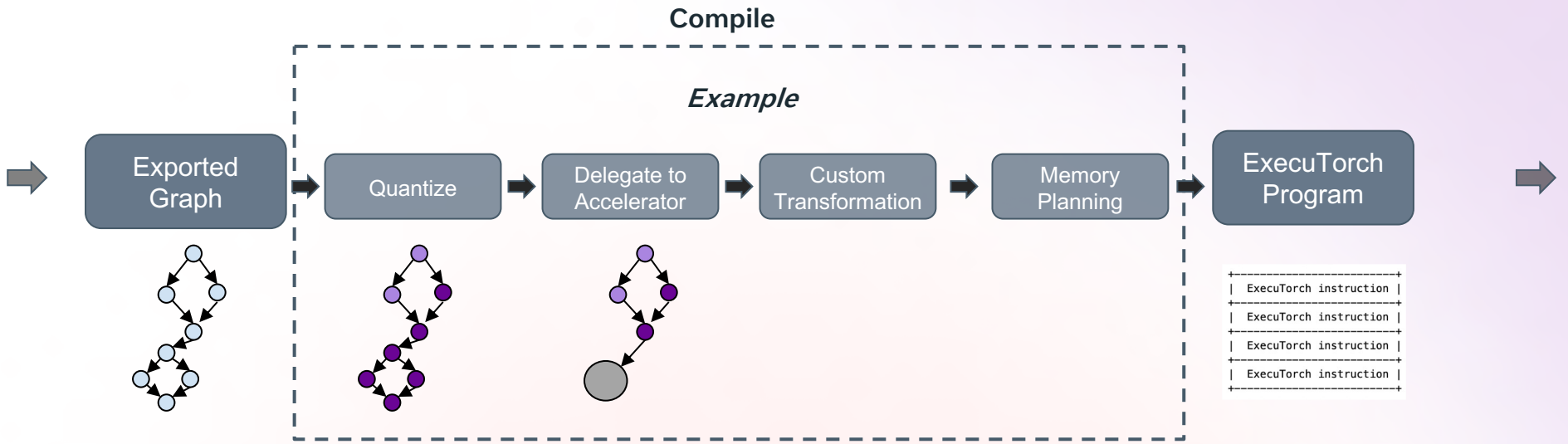
Legacy	ExecuTorch
Convert to 3rd party formats through HW-specific toolchains ❌	Delegate to specialized HW through consistent entry-points ✅
Lack of debugging and profiling tools ❌	Native debugging and profiling through SDK ✅
Conversion failure ❌	Progressive lowering to delegate kernels ✅

Benefit #3: Modularity



Developers can “pick and choose” compilations and transformation steps **through well-defined Python APIs.**

Benefit #3: Modularity

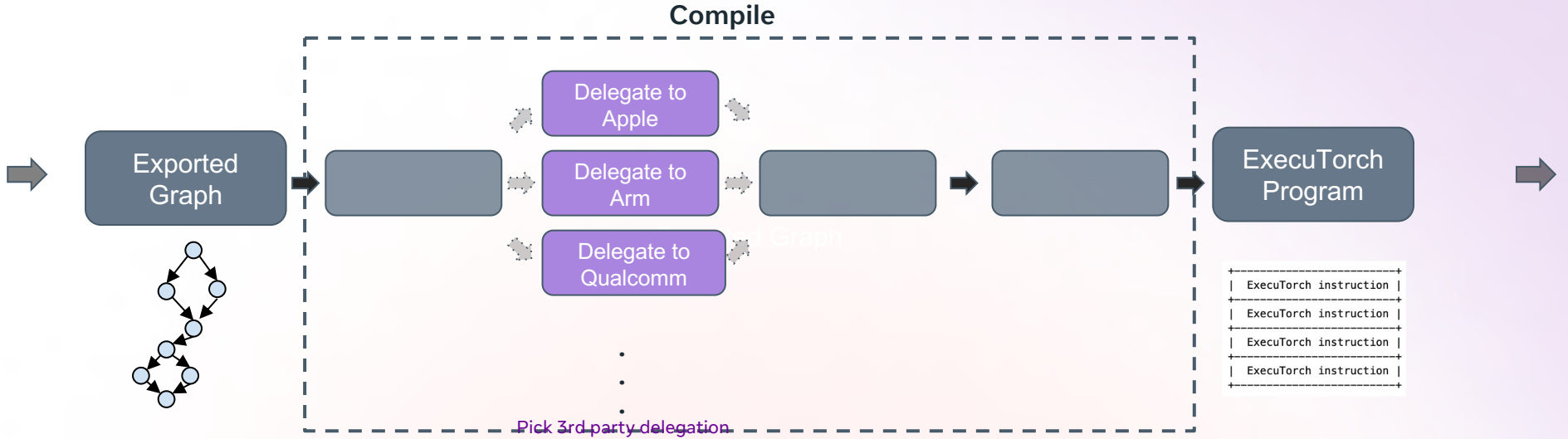



```
torch.export()
```

```
prepared_graph = prepare_pt2e(
   pre_autograd_graph, quantizer
)
converted_graph = convert_pt2e(
   prepared_graph
)
```

```
lowered_module = to_backend(
  converted_graph,
  BackendPartitioner,
)
```

Benefit #4: 3rd Party Compilers and OSS Ecosystem



- Partners:
  - Follows PyTorch 2.x export IR and Core ATen Operator.
  - Implements well-defined compiler and quantization APIs.
  - Contributes to OSS

- Developers are still in PyTorch ecosystem but can improve their performance on a target hardware

Benefit #5: Portable and Lightweight Runtime

ExecuTorch Program

Load

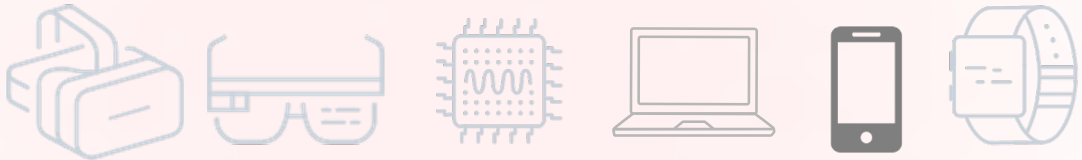
```
+-------------------------+
|  ExecuTorch instruction |
+-------------------------+
|  ExecuTorch instruction |
+-------------------------+
|  ExecuTorch instruction |
+-------------------------+
|  ExecuTorch instruction |
+-------------------------+
```

ExecuTorch Runtime

Runtime is **portable and lightweight**. Can run on mobile, embedded and microcontrollers.

"Embedded friendly" C++. Examples:

- No need dynamic memory allocation in heap
- Minimal dependency on C++ Standard Libraries
- No assumption of OS and filesystems
- Small runtime size. Link only selected kernels
- Core ATen compliant reference kernels available
- Can link against 3rd party kernels and delegates



Inference

Visit our poster sessions

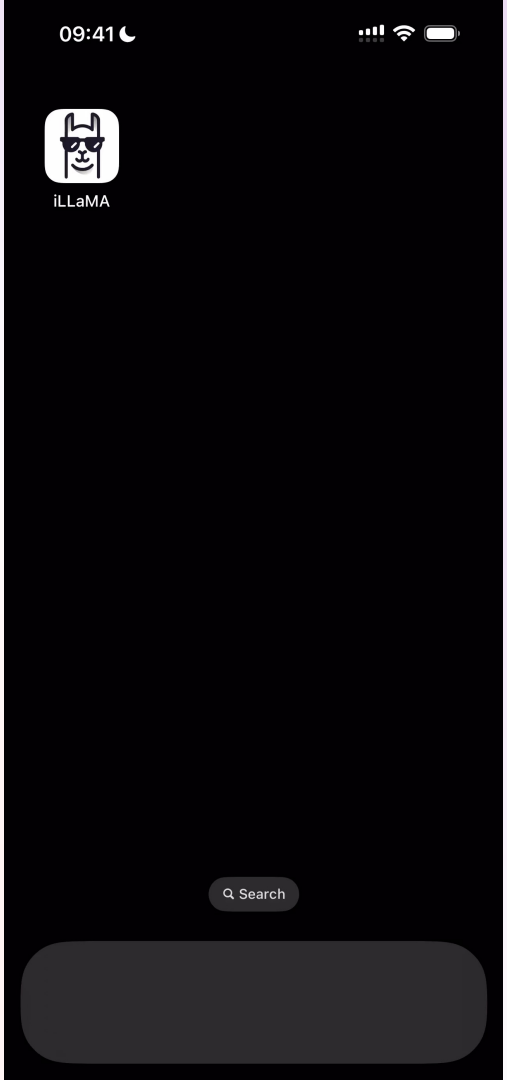
ExecuTorch Productivity SDK

ExecuTorch to Arm Delegates

On-Device Generative AI

SOTA Performance (tokens/sec on Mobile CPUs)

	Android	iOS
Llama 2 7B	7-8 (Samsung S22)	6 (iPhone 15 Pro)
Llama 3 8B	7-8 (One Plus 12)	5+ (iPhone 15 Pro)



Techniques Used

- AOT with PyTorch 2.x Export and ExecuTorch Compilation
- 4-bit group-wise weight quantization
- XNNPACK Delegate for best performance on CPU (WIP on other backends)
- Multi-Threading
- KV Cache support through PyTorch mutable buffer
- Custom ops for SDPA, with kv cache and multi-query attention
- ExecuTorch Runtime + tokenizer and sampler
- Improved and built on top of ExecuTorch Core stack

We welcome the community to try:

- Bring your own model to ExecuTorch!
- Colab notebook
- Android and iOS demos
- Evaluation and Benchmarking
- Documentations and Instructions

Foundational improvements since last Oct

- PyTorch mutable buffers
- Constant data segment for more efficient serialization
- Better Kernel coverage
- SDK - better profiling and debugging within delegates
- API improvements/simplification
- Vulkan delegate for mobile GPU
- Data Type based selective build for optimizing binary size
- Compatible with TorchTune
- More models supported across different backends

A growing list in NLP, vision, and speech
Enabled with one or more backends

Llama 3 8B
Llama 2 7B
Deeplab_v3
Edsr
Emformer_rnnt
Conformer
Inception_v3
Inception_v4
Mobilebert
Mobilenet_v2
Mobilenet_v3
resnet18_model
resnet50_model
Torchvision_vit

Wav2letter
SqueezeSAM
Yolo v5
LSTM
LearningToPaint_model
dcgan_model
maml_omniglot_model
mnasnet1_0_model
shufflenet_v2_x1_0_model
squeezenet1_1_model
timm_efficientnet_model
functorch_dp_cifar10_model
lennard_jones_model

Going forward

- More generative AI support through ExecuTorch
  - Other backends: CoreML, Qualcomm, MPS, MediaTek
  - Other LLMs: Mistral/Mixtral, Gemma, Mamba, Phi, Qwen, Baichuan, etc.
  - Multi modality: Llava, etc.
- Hardening
- Backward/Forward compatibility policy
- Performance improvements
- Community involvement
- On-device training

Get Access

Please try it, get involved, give feedback.

- https://pytorch.org/edge/
- https://github.com/pytorch/executorch/

Copyright Notice

www.tinyml.org