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
- VR/MR
- Embedded
- Mobile
- Wearables

- Performance
- Privacy
- Personalized
- Enable new experiences
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

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

**Alpha - April 2024**
- Generative AI support

**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
ExecuTorch Overview

- Export
  - PyTorch Program
  - Exported Graph

- Compile
  - ExecuTorch Program

- Load
  - ExecuTorch Runtime

- Inference
  - "dog"

Diagrams:
- torch.nn.Module
- Edge Device
Benefit #1: Export-Based

PyTorch Program → Exported Graph → ExecuTorch Program → ExecuTorch Runtime

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

<table>
<thead>
<tr>
<th>Legacy</th>
<th>ExecuTorch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert to 3rd party formats through HW-specific toolchains ✗</td>
<td>Delegate to specialized HW through consistent entry-points ✓</td>
</tr>
<tr>
<td>Lack of debugging and profiling tools ✗</td>
<td>Native debugging and profiling through SDK ✓</td>
</tr>
<tr>
<td>Conversion failure ✗</td>
<td>Progressive lowering to delegate kernels ✓</td>
</tr>
</tbody>
</table>
Benefit #3: Modularity

Developers can “pick and choose” compilations and transformation steps through well-defined Python APIs.
Benefit #3: Modularity

Example

```
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

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
Visit our poster sessions

ExecuTorch Productivity SDK

ExecuTorch to Arm Delegates
On-Device Generative AI

SOTA Performance (tokens/sec on Mobile CPUs)

<table>
<thead>
<tr>
<th>Model</th>
<th>Android</th>
<th>iOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Llama 2 7B</td>
<td>7-8 (Samsung S22)</td>
<td>6 (iPhone 15 Pro)</td>
</tr>
<tr>
<td>Llama 3 8B</td>
<td>7-8 (One Plus 12)</td>
<td>5+ (iPhone 15 Pro)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>NLP</th>
<th>Vision</th>
<th>Speech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Llama 3 8B</td>
<td>Deeplab_v3</td>
<td>Wav2letter</td>
</tr>
<tr>
<td>Llama 2 7B</td>
<td>Edr</td>
<td>SqueezeSAM</td>
</tr>
<tr>
<td></td>
<td>Emformer_rnnt</td>
<td>Yolo v5</td>
</tr>
<tr>
<td></td>
<td>Conformer</td>
<td>LSTM</td>
</tr>
<tr>
<td></td>
<td>Inception_v3</td>
<td>LearningToPaint_model</td>
</tr>
<tr>
<td></td>
<td>Inception_v4</td>
<td>dcgan_model</td>
</tr>
<tr>
<td></td>
<td>Mobilebert</td>
<td>maml_omniglot_model</td>
</tr>
<tr>
<td></td>
<td>Mobilenet_v2</td>
<td>mnasnet1_0_model</td>
</tr>
<tr>
<td></td>
<td>Mobilenet_v3</td>
<td>shufflenet_v2_x1_0_model</td>
</tr>
<tr>
<td></td>
<td>resnet18_model</td>
<td>squeezenet1_1_model</td>
</tr>
<tr>
<td></td>
<td>resnet50_model</td>
<td>timm_efficientnet_model</td>
</tr>
<tr>
<td></td>
<td>Torchvision_vit</td>
<td></td>
</tr>
</tbody>
</table>
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://pytorch.org/edge/)
- [https://github.com/pytorch/executorch/](https://github.com/pytorch/executorch/)
Copyright Notice

This presentation in this publication was presented at the tinyML® Summit 2024. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyml.org