arm

Accelerating On-Device ML Inference with ExecuTorch and Arm SME2

Jason Zhu, PhD
Sr. Principal Engineer

Tyler Mullenbach, PhD
Sr. Director AI Engineering

Apr 7th 2026

Arm: Computing for All

>325B

ARM-BASED CHIPS SHIPPED TO-DATE

Arm is delivering the most power-efficient and most high-performance compute platform across multiple markets.

Scalable Matrix Extension 2 (SME2)

SME2 is the latest CPU extension on Arm Lumex CSS, designed to accelerate matrix-oriented compute workloads directly on device.

MediaTek launched **Dimensity 9500** in Sep 2025

Samsung launched **Exynos 2600** in Feb 2026

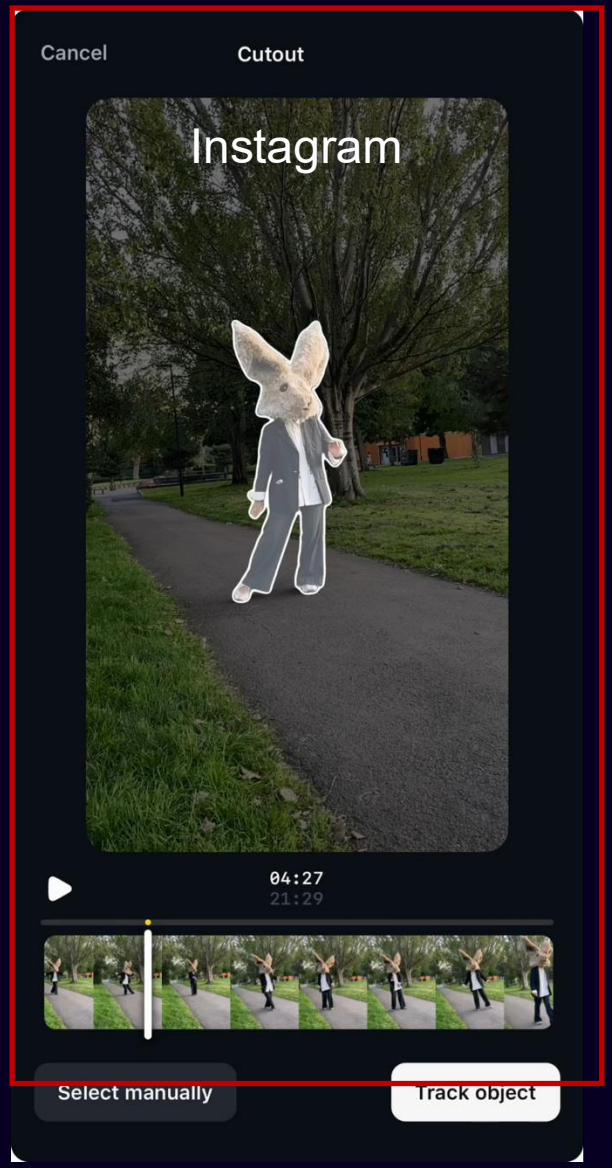
Vision workloads are evolving rapidly

2012: AlexNet

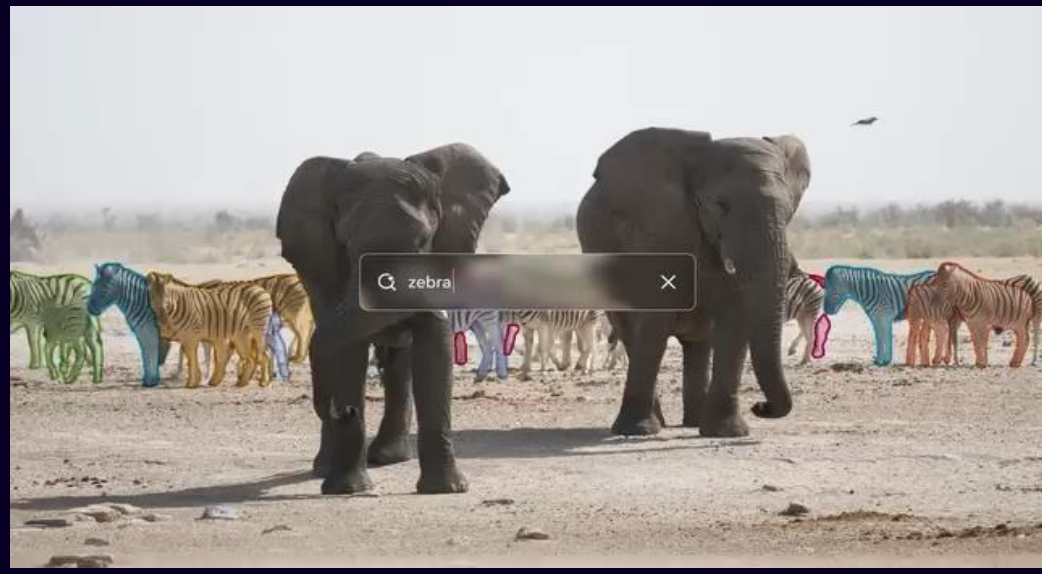
A Krizhevsky at al. ImageNet 2012

Static, simple task

2024: SAM2



2025: SAM3

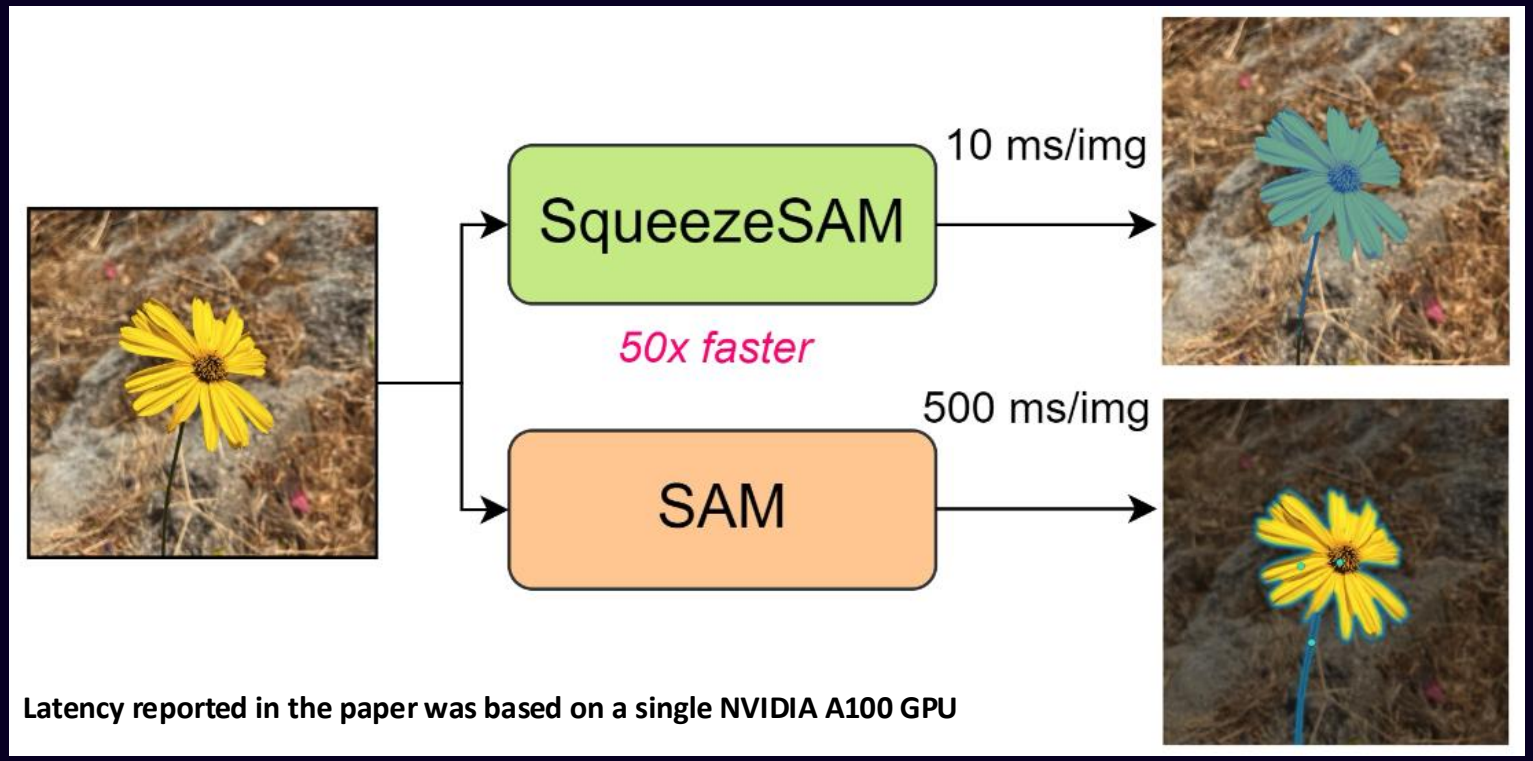


Interactive, richer vision
Low latency becomes critical

Meta's reference blogs
https://ai.meta.com/blog/instagram-edits-cutouts-segment-anything
https://ai.meta.com/blog/segment-anything-model-3/

SqueezeSAM: a mobile-first segmentation model

Used in interactive cutout experiences like Instagram Cutouts



Why SqueezeSAM is a strong case study

Mobile-first modern architecture

- Designed for promptable, real-time mask prediction
- Hybrid conv + transformer design

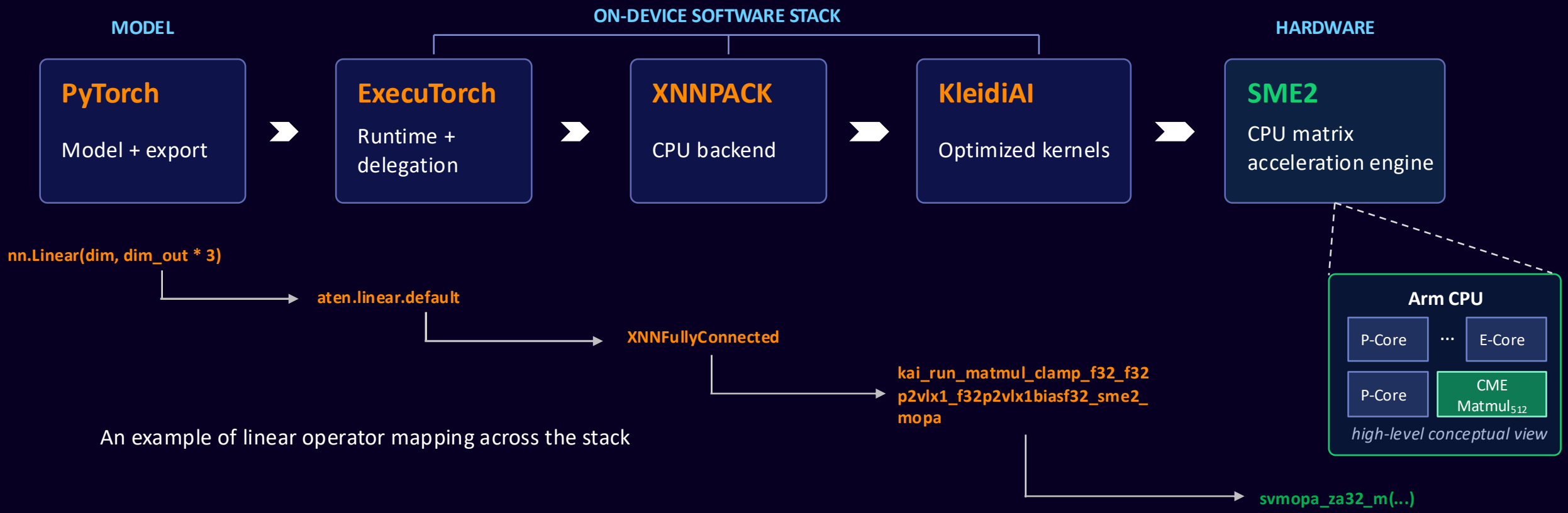
Compute heavy vision workload

- 10.2M params, ~26B MACs @ 512 × 512
- Most of compute is Conv2d, so efficient kernel execution is critical

V. Balakrishnan et al. arXiv 2312.06736, 2024

How PyTorch reaches SME2

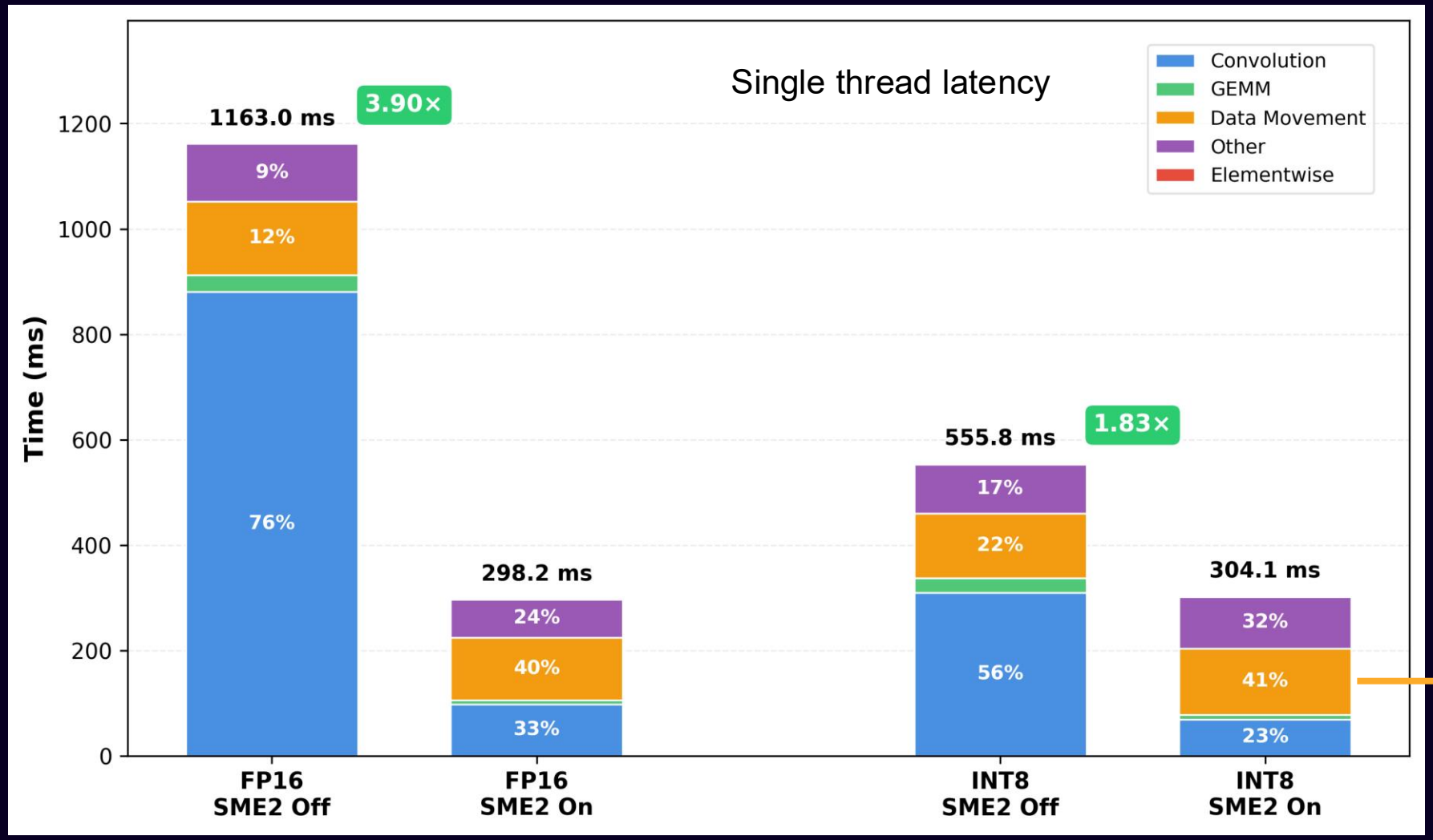
Goal: make interactive models practical on CPU



An example of linear operator mapping across the stack

SqueezeSAM: major end-to-end gains from SME2

As math accelerates, different bottlenecks appear



What this means for developers

1) Conv/GEMM are the biggest SME2 wins
2) FP16 can approach INT8 latency
3) Data movement becomes the next bottleneck
4) Ops outside the fast path matter more

Data movement total
119 ms · 40% of runtime

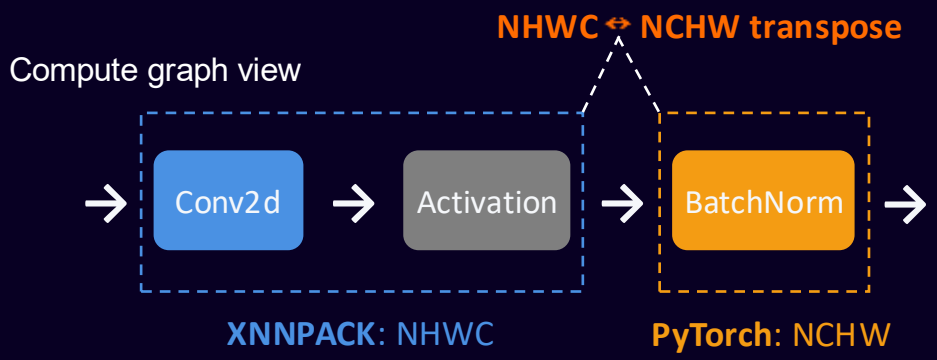
Transpose overhead
123 ops · ~98 ms

https://pytorch.org/blog/accelerating-on-device-ml-inference-with-executorch-and-arm-sme2/

Faster matrix compute shifts the bottleneck

A layout and operator-coverage issue, not just a hardware limit

Data movement overhead root cause



1. Activation blocks Conv-BN fusion
2. Conv prefers NHWC, BN stays NCHW

Fix / design guidance

Existing models: Delegate BN

Result: Transpose time ↓ 70%: 98 ms → 30 ms

New models: prefer fusible pattern

Conv → BN → Act preserves layout continuity

SME2 delivers real gains across model families

Operator profiling reveals what to optimize next

SqueezeSAM

Biggest wins		
Convolution	881 → 98 ms	**9.0x**
GEMM	32 → 8 ms	**4.0×**
What remains		
Data Movement	139 → 119 ms	**1.2x**
Portable	110 → 72 ms	**1.5×**
E2E latency	1163 → 298 ms	**3.9x**

LLM 2B (prefill_256)

Only top operators are shown here

Biggest wins		
Batch MatMul	3403 → 477 ms	**7.1x**
FC (QC4W)	1870 → 203 ms	**9.2×**
What remains		
Softmax	190 → 190 ms	**1.0x**
Conditional Select	71 → 80 ms	**0.9×**
E2E throughput	44 → 223 t/s	**5.1x**

SME2 speeds up matrix-heavy operations most;
the remaining costly operations tell you what to optimize next.

arm

Acknowledgments

We would like to thank the Meta ExecuTorch team Bilgin Cagatay, Mergen Nachin, Digant Desai, Gregory Comer, and Andrew Caples for their guidance on real-world use cases and their contributions to inference optimization implementations. We also thank Ray Hensberger, Ed Miller, Mary Bennion, and Shantu Roy from Arm for their support and guidance throughout this work.

More details can be found in our PyTorch blogpost:
https://pytorch.org/blog/accelerating-on-device-ml-inference-with-executorch-and-arm-sme2/

Tack
ధన్యవాదగళు
Merci
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Thank you
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు
Köszönöm