arm AI

Arm AI Virtual Tech Talk

Improve PyTorch App Performance with Android NNAPI Support

Koki Mitsunami

December 14th, 2021

arm

Welcome!

Tweet us: @ArmSoftwareDev -> #AIVTT

Check out our Arm Software Developers YouTube channel

Signup now for our next AI Virtual Tech Talk: www.arm.com/techtalks

Our upcoming Arm AI Tech Talks

Date	Title	Host
December 14th	Improve PyTorch App Performance with Android NNAPI Support	Arm
January 11, 2022	Qeexo AutoML on Arm Cortex-M0+: Bringing TinyML to the tiniest Arm MCUs	Qeexo
January 25, 2022	Peaks, Valleys and Thresholds: The art of segmenting real-time sensor data for tinyML classification, regression and anomaly detection	Reality AI
February 8, 2022	Faster time-to-production for computer vision AI on Arm powered edge devices, but just how fast is fast?	Deeplite
February 22, 2022	*** New Arm ML Quarterly Research Special *** Federated Learning Based on Dynamic Regularization to Debias Model Updates	Arm ML Research

Visit: www.arm.com/techtalks

Presenter

Koki Mitsunami

Staff Engineer @ Arm

Cambridge, UK

- Contribute to enabling a successful developer experience with Arm-based technology
- Demonstrated multiple applications using deep learning focusing computer vision
- Enjoy working on hardware, software, and algorithms

Agenda

- Introduction
  - Purpose of this talk
  - PyTorch
  - Android NNAPI
- PyTorch Mobile with NNAPI
  - Workflow
  - Benchmark Conditions
  - Execution Time Comparison
  - Profiling by Streamline
- Summary

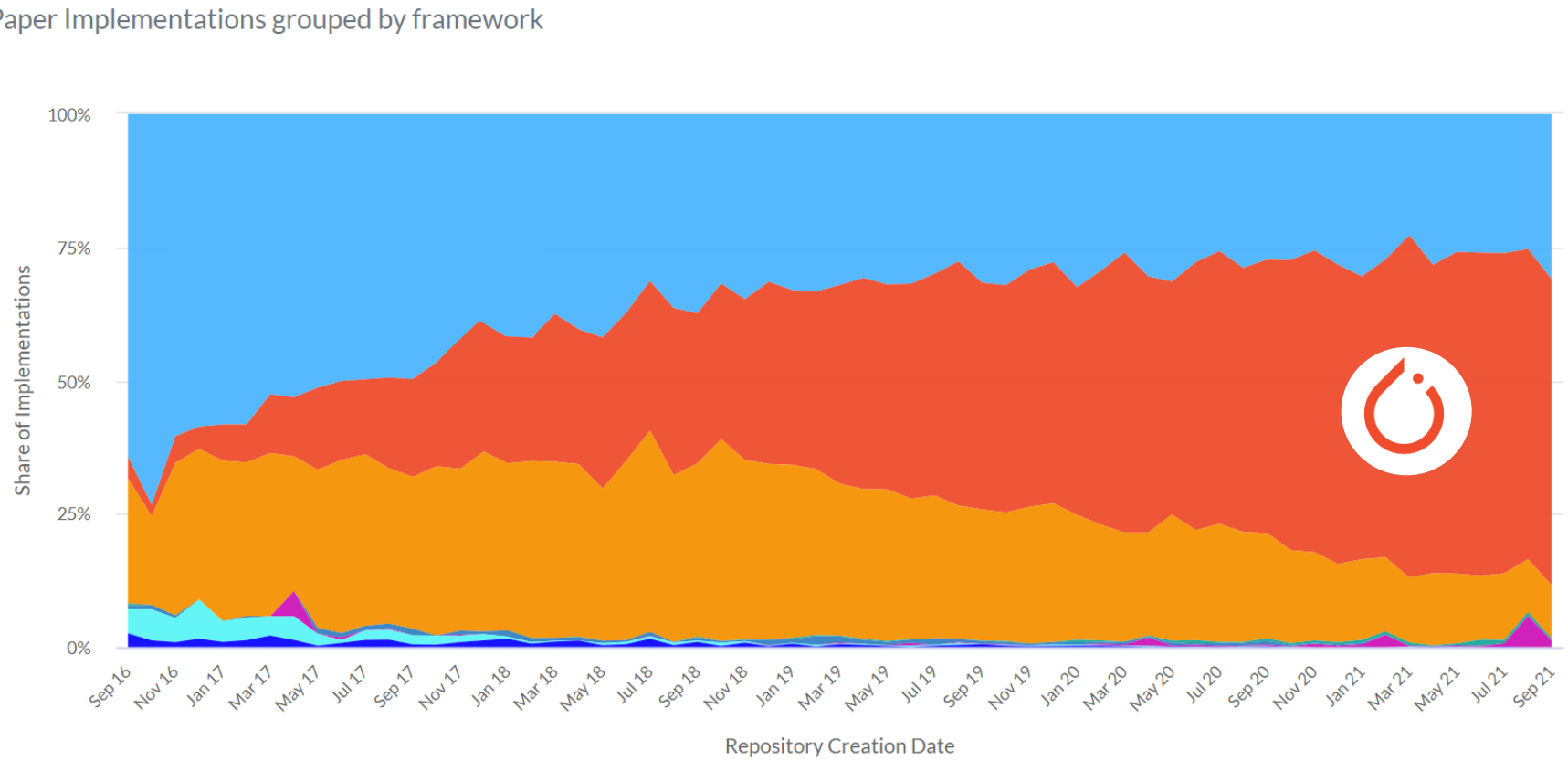
Purpose of this talk

- Introduce PyTorch Mobile with Android NNAPI
  - One way of deploying ML models into mobile devices

- Provide examples on how ML models are executed on mobile devices through PyTorch with NNAPI

PyTorch

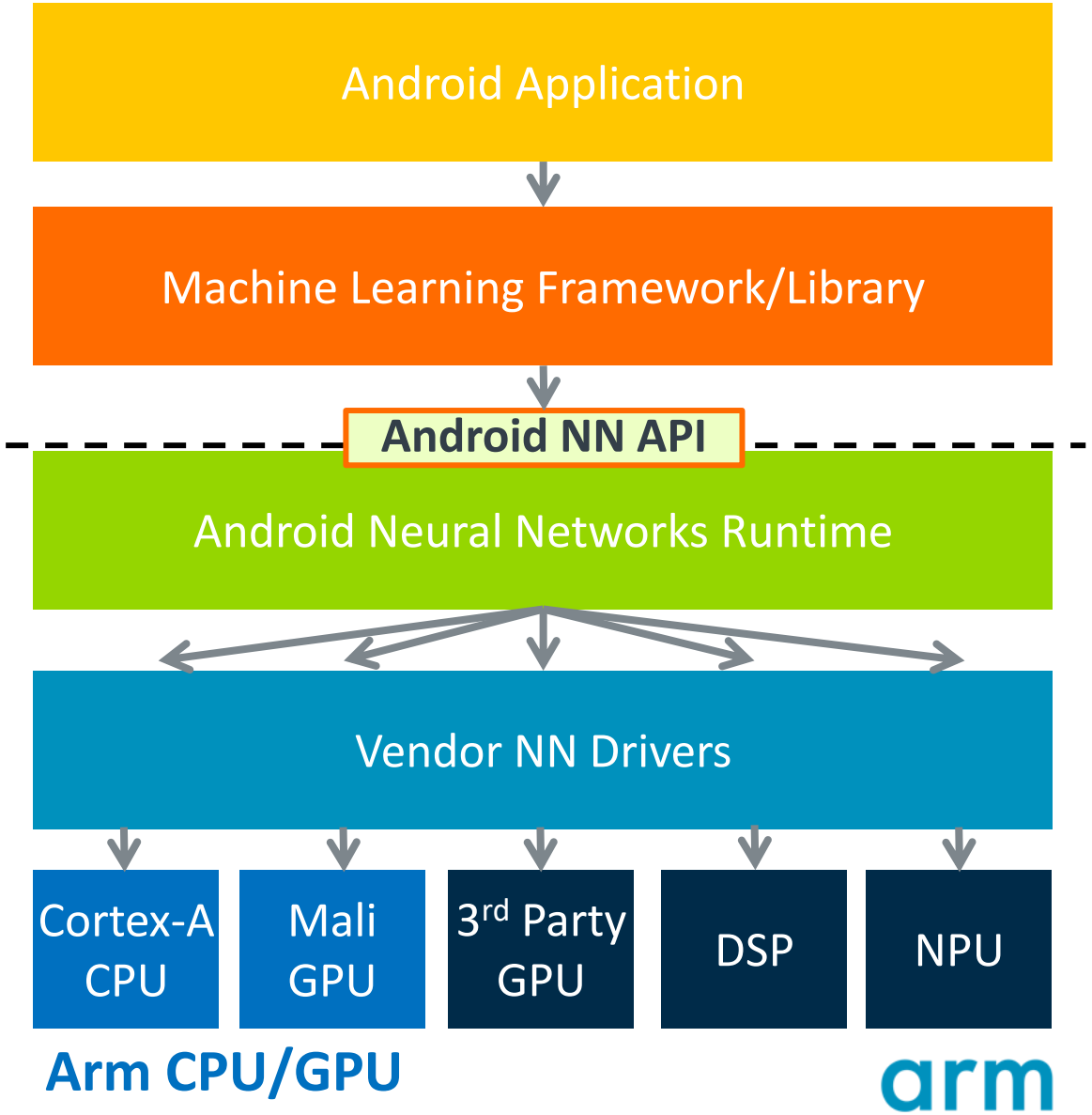
- PyTorch has become popular Deep Learning framework
  - Wide range of supported operators
  - Ease of writing
  - A lot of activity for production



Source: https://paperswithcode.com/trends

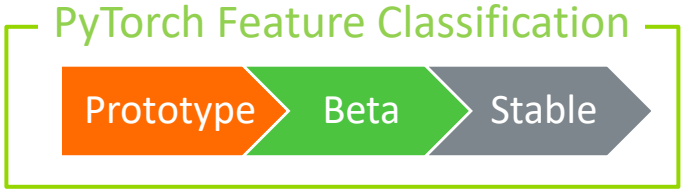
Android Neural Networks API (NNAPI)

- C API designed for running ML models
  - Provide a base layer of functionality
  - Perform hardware-accelerated inference operations on supported devices
- Typically used by ML frameworks
  - Apps would not use NNAPI directly
  - Apps would use ML frameworks



Android NNAPI support is now available in Beta

Announced in PyTorch 1.10 Release

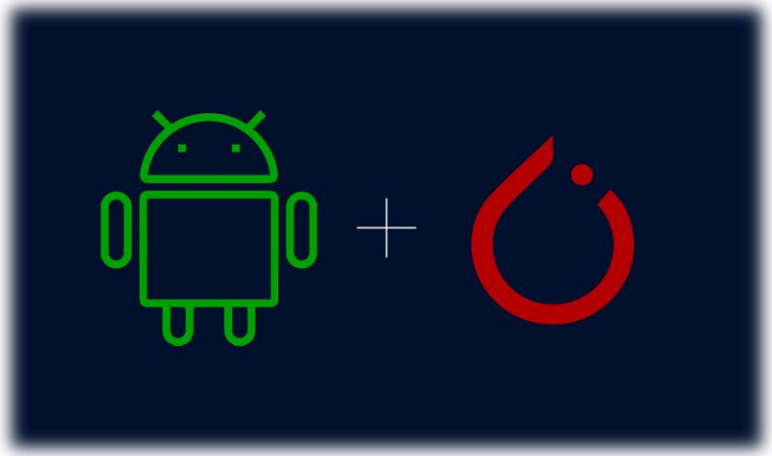


Prototype (Nov 2020)

- NNAPI with the PyTorch framework
- Android 10+
- Linear Convolution models
- Multi-Layer Perceptron models

Beta (Oct 2021)

- Additional operator types
- Support for load-time flexible shapes
- Run models on host for testing



https://pytorch.org/blog/prototype-features-now-available-apis-for-hardware-accelerated-mobile-and-arm64-builds/
https://pytorch.org/blog/pytorch-1.10-released/#beta-android-nnapi-support-in-beta

PyTorch Mobile

- Mechanism to run ML models on mobile
  - Provides end-to-end workflow within PyTorch ecosystem
  - Support for Arm CPU/GPU and hardware accelerators via XNNPACK/QNNPACK, Vulkan, and NNAPI

- TorchScript format
  - Intermediate representation (IR) of a PyTorch model
    - includes code, parameters, attributes, and debug information
  - Can be run in a high-performance environment such as C++

TorchScript example

```
graph(%0 : Float(3, 10), %1 : Float(3, 20), %2 : Float(3, 2
      %3 : Float(80, 10) %4 : Float(80, 20),
      %5 : Float(80), %6 : Float(80)) {
  %7 : Float(10!, 80!) = aten::t(%3)
  %10 : int[] = prim::ListConstruct(3, 80)
  %12 : Float(3!, 80) = aten::expand(%5, %10, 1)
  %15 : Float(3, 80) = aten::addmm(%12, %0, %7, 1, 1)
  %16 : Float(20!, 80!) = aten::t(%4)
  %19 : int[] = prim::ListConstruct(3, 80)
  %21 : Float(3!, 80) = aten::expand(%6, %19, True)
  %24 : Float(3, 80) = aten::addmm(%21, %1, %16, 1, 1)
  %26 : Float(3, 80) = aten::add(%15, %24, 1)
  %29 : Dynamic[] = aten::chunk(%26, 4, 1)
  %30 : Float(3!, 20), %31 : Float(3!, 20),
    %32 : Float(3!, 20), %33 : Float(3!, 20) = prim::ListUnp
  %34 : Float(3, 20) = aten::sigmoid(%30)
  %35 : Float(3, 20) = aten::sigmoid(%31)
  %36 : Float(3, 20) = aten::tanh(%32)
```

Workflow for PyTorch Mobile with NNAPI

1. Create/prepare a model in PyTorch
2. Quantization (optional)
3. Convert to TorchScript
4. Optimize for Mobile (optional)
5. Convert to NNAPI-compatible model
6. Save the model
   - Treat as a TorchScript model
   - For applications already using PyTorch Mobile, no code changes are required

Example)

Prepare a pre-trained quantized model

```
model = torchvision.models.quantization. \
        mobilenet.mobilenet_v2 \
        (pretrained=True, quantize=True)
```

Convert to TorchScript

```
input_tensor = input_tensor.contiguous\
           (memory_format=torch.channels_last)
input_tensor.nnapi_nhwc = True
traced = torch.jit.trace(model, input_tensor)
```

Convert to NNAPI model

```
nnapi_model = torch.backends._nnapi.prepare. \
   convert_model_to_nnapi(traced, input_tensor)
```

Save the model

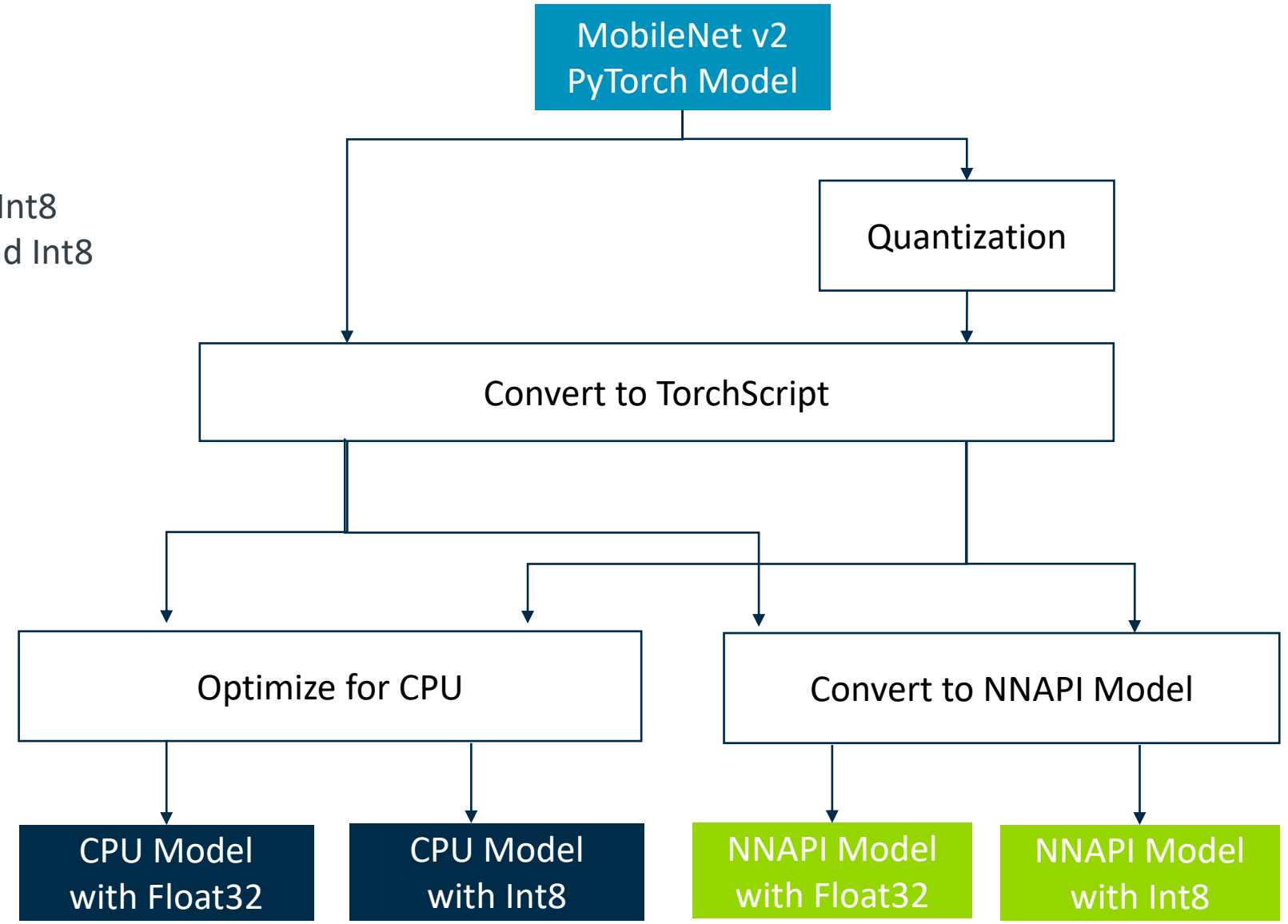
```
nnapi_model._save_for_lite_interpreter\
                ("mobilenetv2-nnapi.pt")
```

Deploy the model to mobile

Experiments - PyTorch Mobile with NNAPI

- ML Model
  - MobileNet v2
    - CPU models with Float32 and Int8
    - NNAPI models with Float32 and Int8
- Mobile devices
  - 8 mobile devices
    - 2 for Android 11
    - 6 for Android 10
- Execution time
  - Avg time over 200 iter

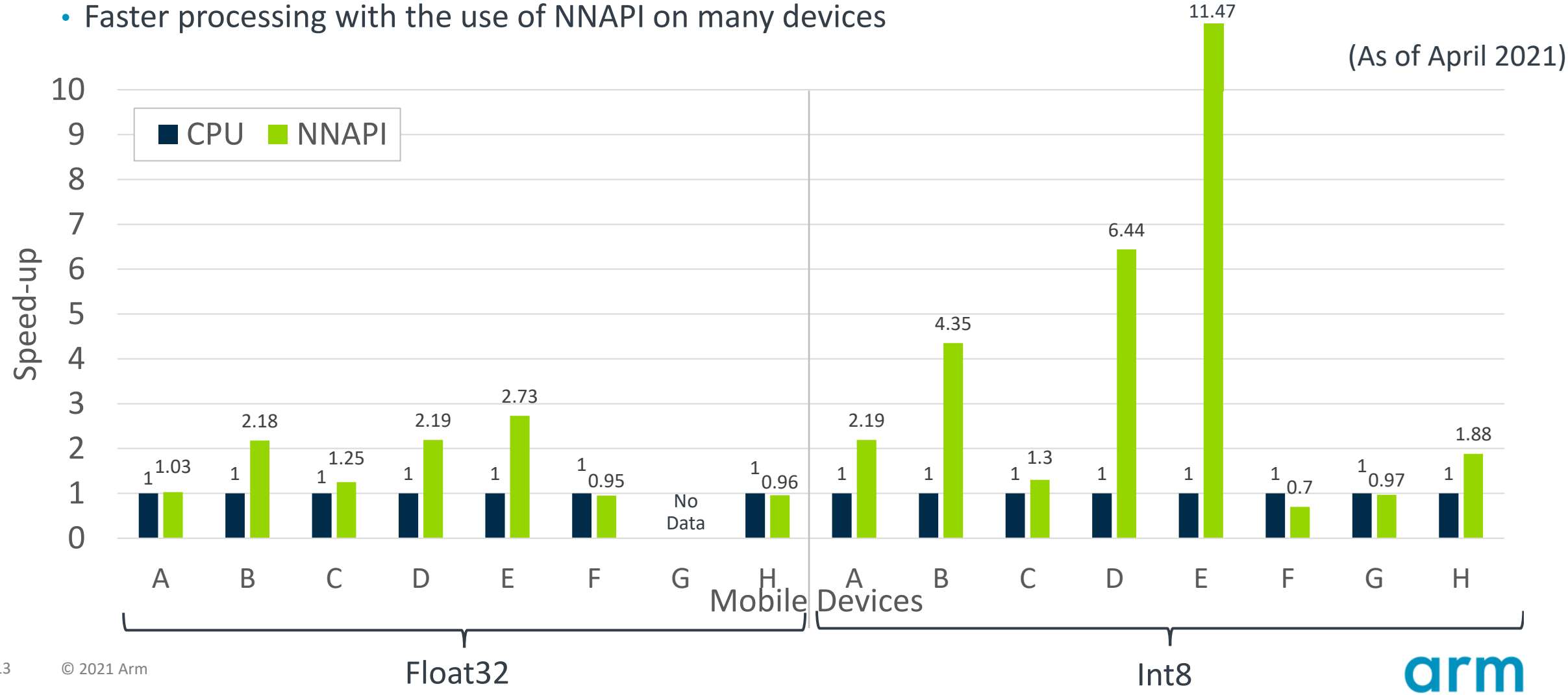
Model Generation Flow



(PyTorch 1.8.0.dev + torchvision 0.9.0.dev)

NNAPI Speed-up on Various Devices

- Speed increase varies from device to device
  - Faster processing with the use of NNAPI on many devices



Profiling with Streamline Performance Analyzer

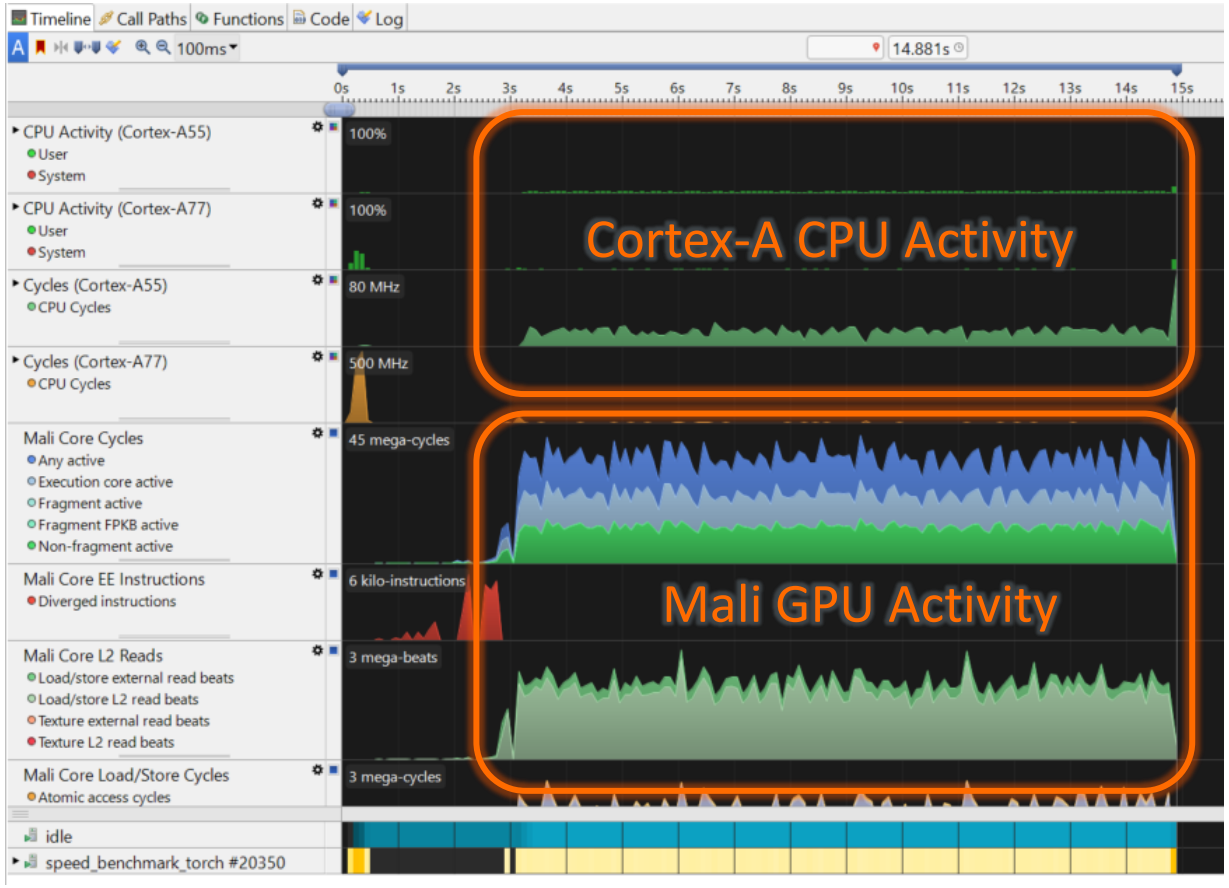
- Arm Streamline allows you to see CPU and GPU activity inside device

Screenshot from Streamline

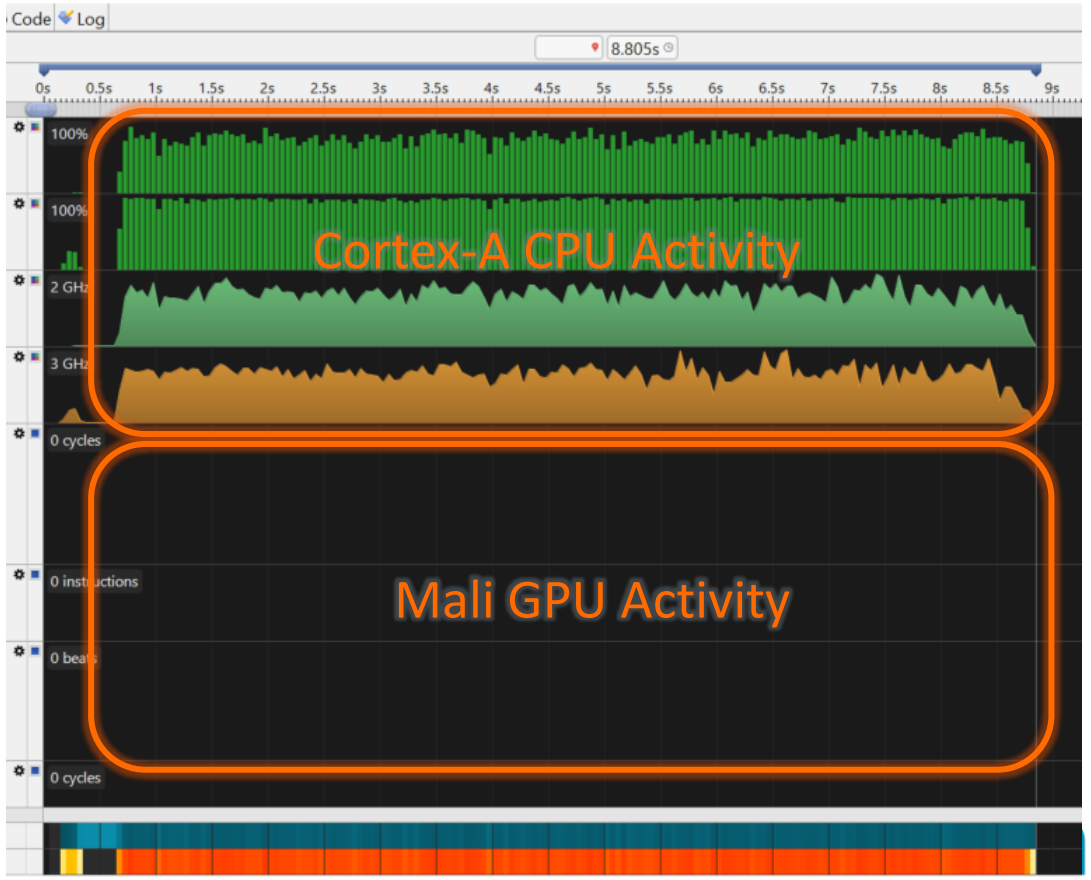
Streamline Profiling with NNAPI Models (Device 1)

- NNAPI selects most performant hardware
  - GPU is used for Float32, while multi-core CPU is used for Int8

NNAPI Model with Float32



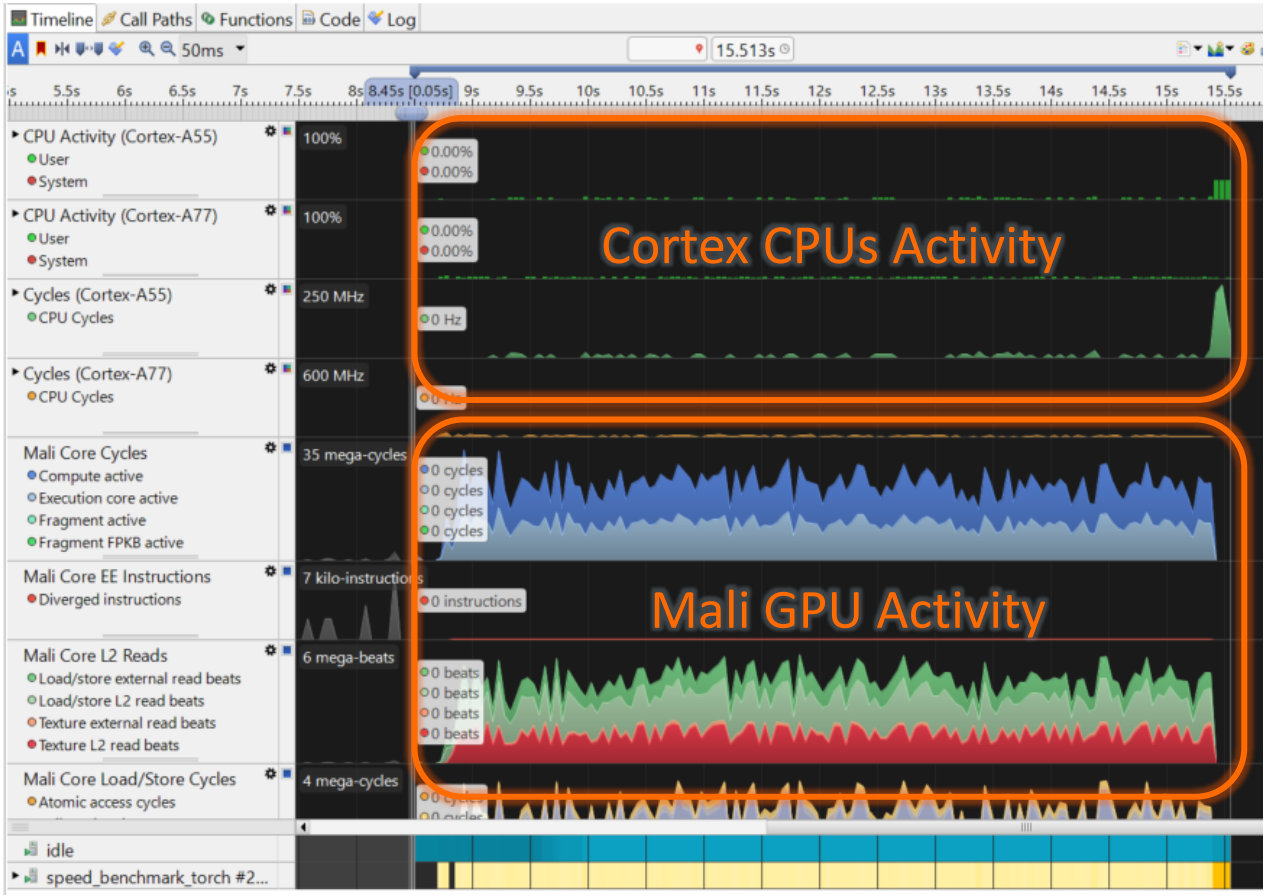
NNAPI Model with Int8



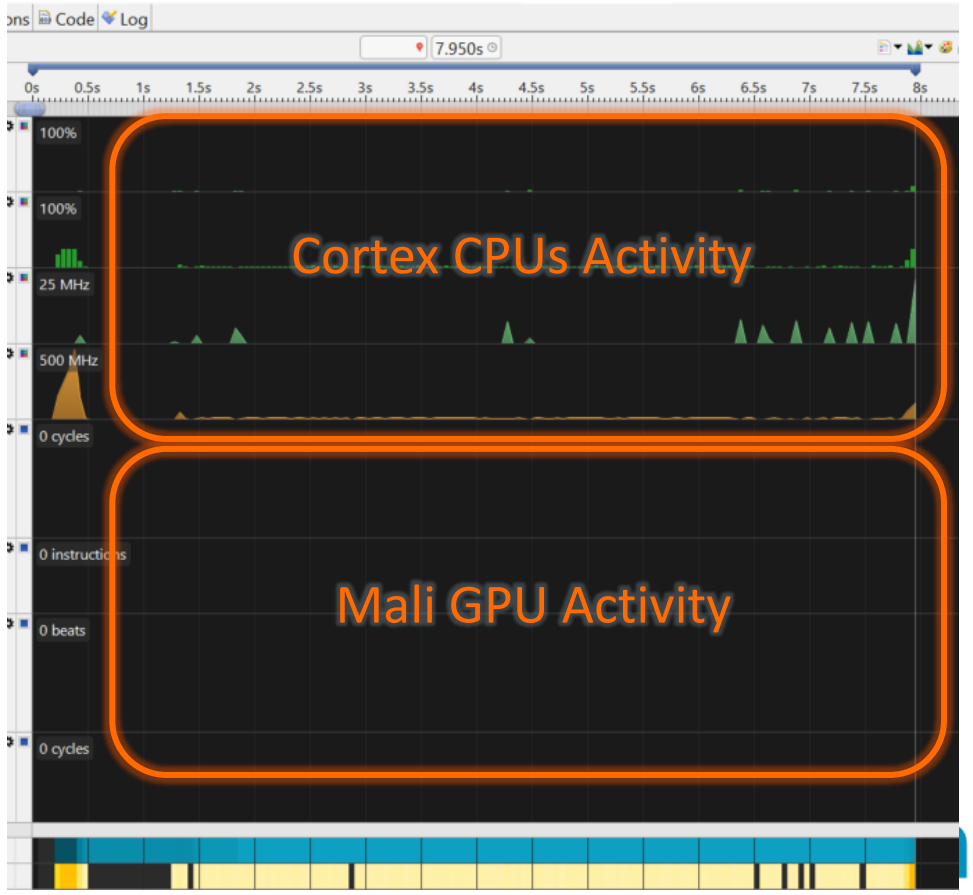
Streamline Profiling with NNAPI Models (Device 2)

- The same model can work differently on different devices by using NNAPI
  - NPU/DSP is used with Int8, which is not visible with Streamline

NNAPI Model with Float32



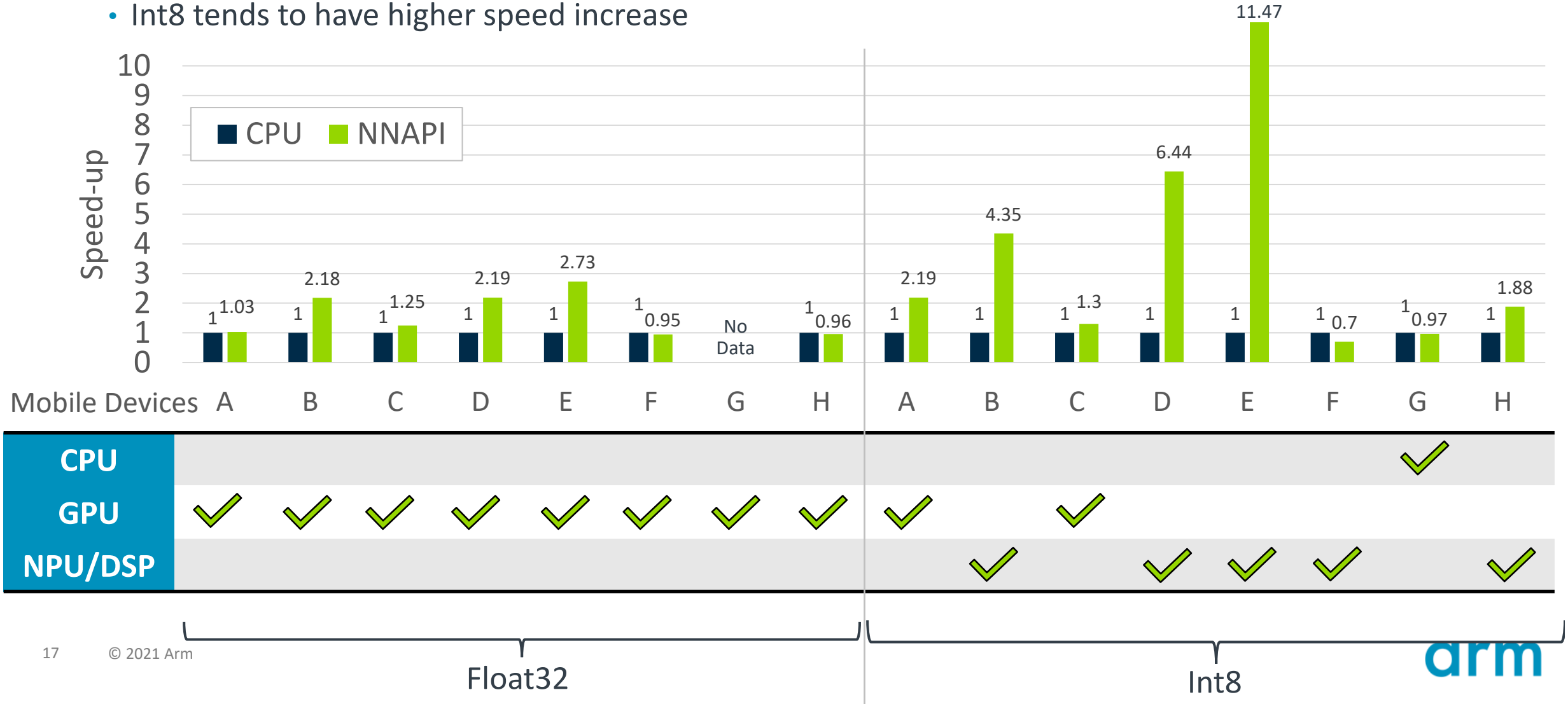
NNAPI Model with Int8



Hardware selected by NNAPI

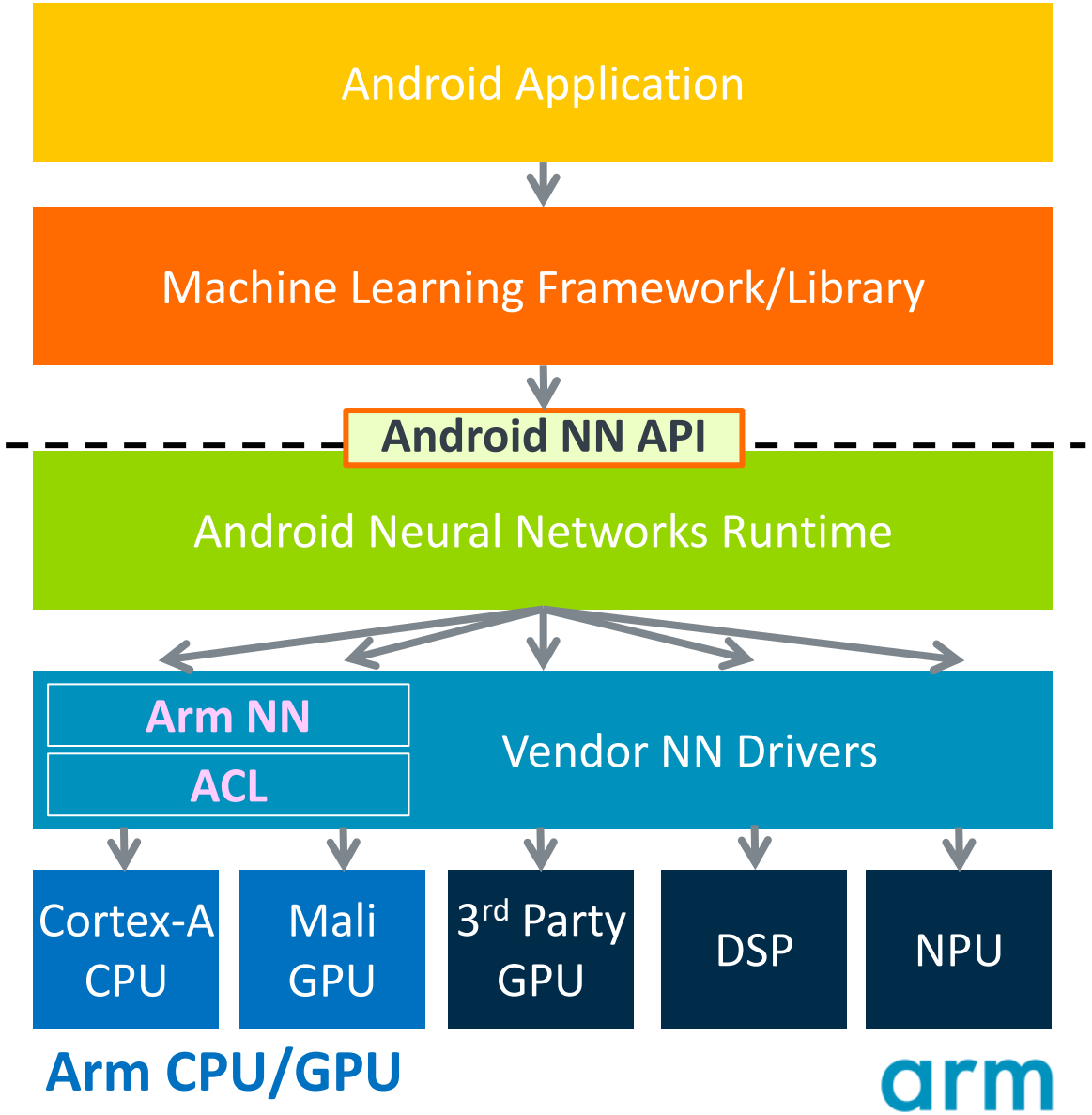
(As of April 2021)

- Quantized models are more likely to benefit from specialized processors
  - Int8 tends to have higher speed increase



Arm NN and Arm Compute Library (ACL)

- Many Android devices have ML models accelerated by underlying Arm NN and ACL
- Superior Performance & Arm Specific Optimization
  - Uses advanced network optimization techniques
  - Quick adoption of new Arm technologies e.g. Armv9-A features



Faster Machine Learning in Android 12

- More than halved inference call overhead by introducing improvements
  - Such as padding, sync fences and reusable execution objects

- Made ML accelerator drivers updatable outside of platform releases
  - Make it easier for developers to take advantage of the latest drivers
  - ML performance improvements and bug fixes reach users faster than ever before

https://android-developers.googleblog.com/2021/04/android-12-developer-preview-3.html

Summary

- Trends around PyTorch
  - Increasing popularity, especially in research field
  - Working to bridge the gap between research and production

- Advantage of NNAPI
  - Provides a single set of APIs
  - Select most performant hardware for running ML models for each device
    - Including GPUs, DSPs, and NPUs
  - Expected to support more and more in the future

- PyTorch with Android NNAPI
  - Moved forward from Prototype phase to Beta phase
  - Provides workflow that simplifies research to production within PyTorch ecosystem

Blog: https://community.arm.com/arm-community-blogs/b/ai-and-ml-blog/posts/improve-pytorch-app-performance-with-android-nnapi-support-386430784

arm AI

AI Virtual Tech Talks Series

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה

arm

Welcome!

Tweet us: @ArmSoftwareDev -> #AIVTT

Check out our Arm Software Developers YouTube channel

Signup now for our next AI Virtual Tech Talk: www.arm.com/techtalks