arm AI

AI Virtual Tech Talks Series

Supercharge your development time with the Arm NN and Arm Compute Library

Ronan Naughton, Jim Flynn, Gian Marco Iodice

9th February 2021

AI Virtual Tech Talks Series

Date	Title	Host
February 9th	Supercharge your development time with the new Arm NN 20.11 release	Arm
February 23rd	Hands-on with PyArmNN for object detection	Arm Workshop
March 9th	Automate tinyML Development & Deployment with Qeexo AutoML	Qeexo

Visit: developer.arm.com/solutions/machine-learning-on-arm/ai-virtual-tech-talks

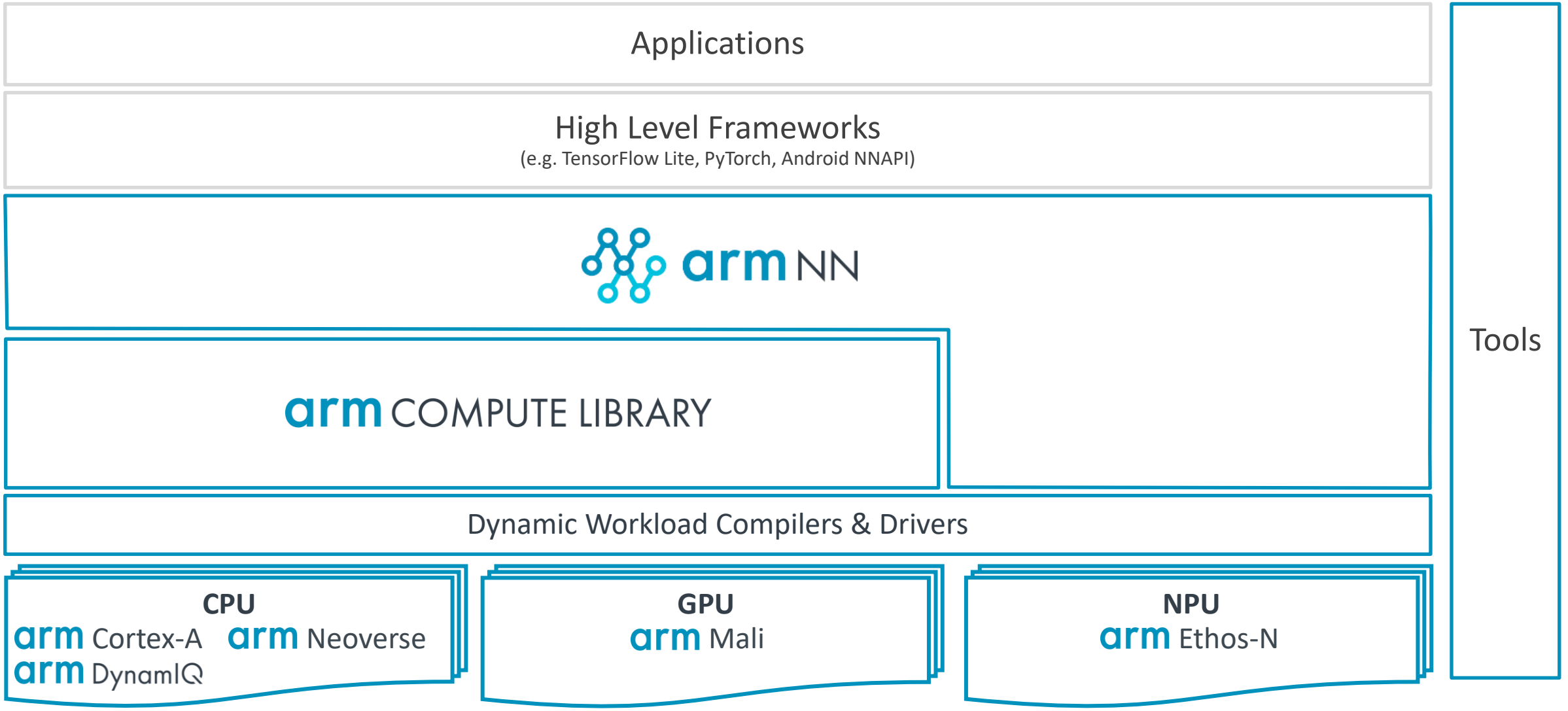
Presenters

Ronan Naughton
Senior Product Manager,
MLG, Arm

Jim Flynn
Arm NN Engineering Manager,
MLG, Arm

Gian Marco Iodice
ACL Staff Software Engineer,
MLG, Arm

Arm Software Stack



Why Use the Arm NN and Arm Compute Library?

Versatile and Portable:

- Easily target multiple platforms (CPU, GPU and NPU) from a single code base
- Reduce overall development time, keep using existing framework and tools
- Deployable for Android, Linux and ‘bare metal’ applications

Superior Performance:

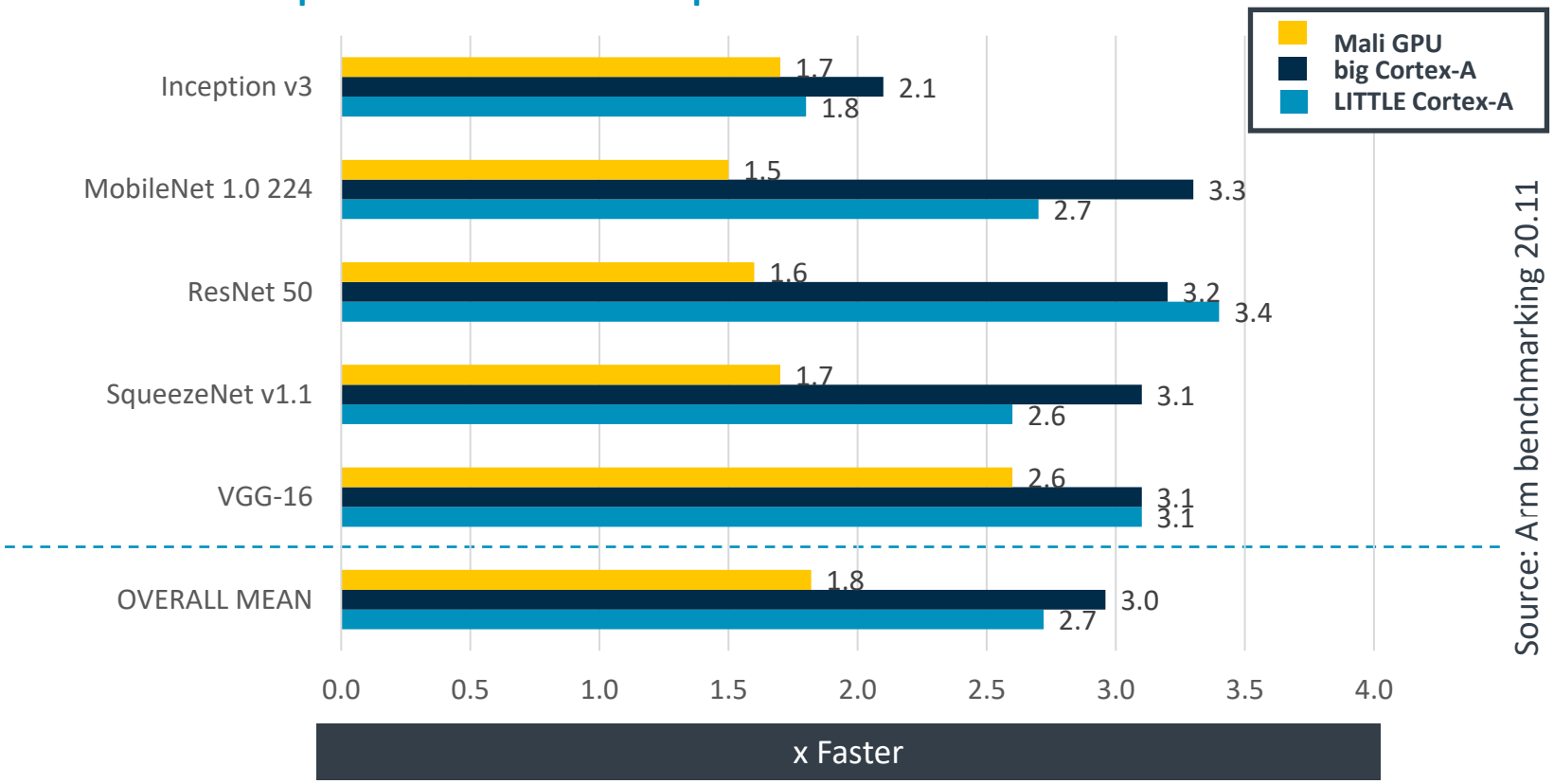
- Best in class across a wide range of popular networks
- Uses advanced network optimization techniques, workload tuning and GEMM heuristics

Arm Specific Optimizations:

- Outperforms generic math and ML libraries due to Arm specific optimization
- Specific architectures (e.g. dot product for Armv8.2A) and micro architecture optimizations (e.g. Cortex-A53)
- Quick adoption of new Arm technologies e.g. SVE, SVE2

Relentless Performance Optimization

Arm NN performance improvements*

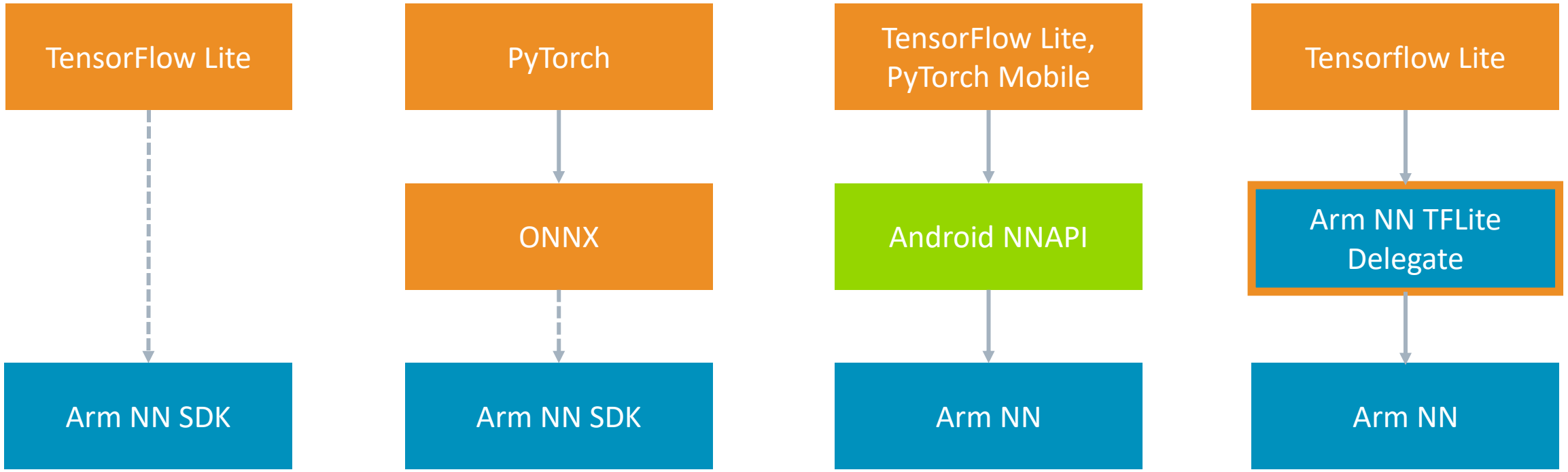


*Mean performance improvements of Arm NN relative to up to six different industry software libraries

- Continuous optimization of the key operators in the most popular networks
- New operators added to support each release of Android
- Advanced network manipulation techniques used to improve performance

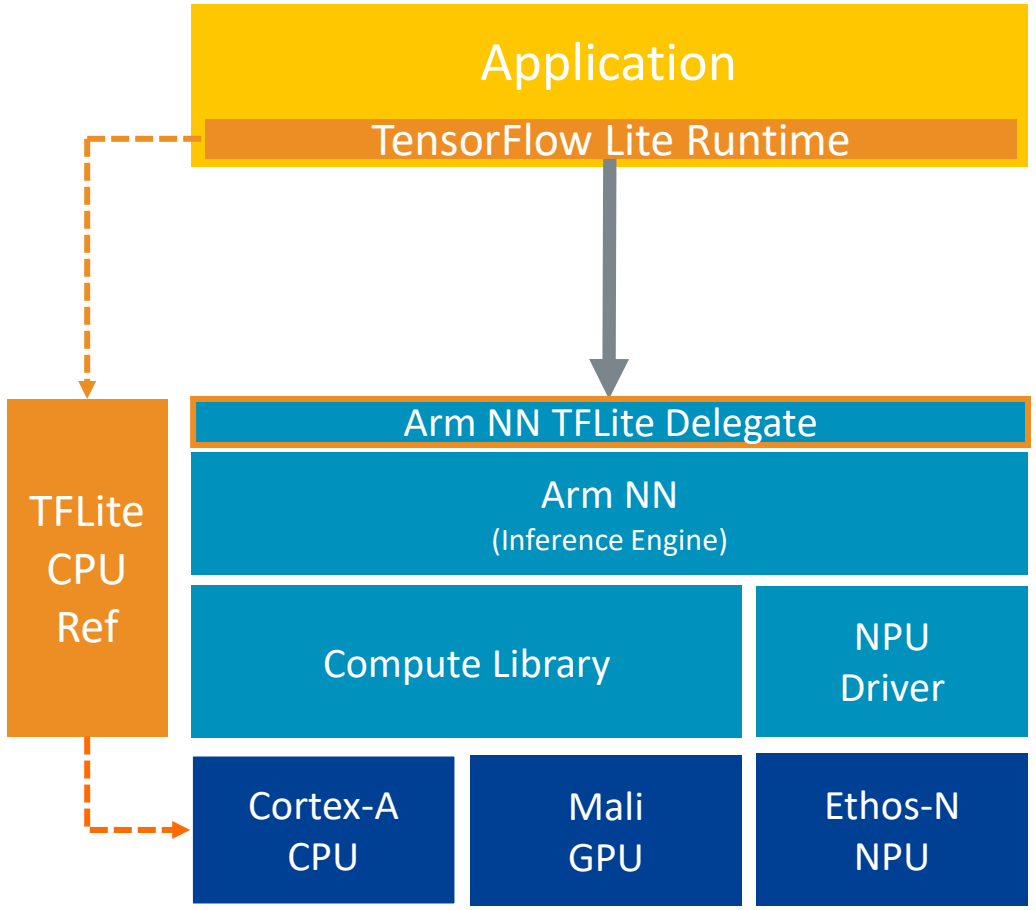
Arm NN Integration Options with Neural Network Frameworks

- Multiple framework integration options for Linux and Android



Arm NN TFLite Delegate

Accelerating inference on Android and Linux



- Unlocks Arm specific CPU and GPU optimizations for TFLite users
- All TFLite models can be accelerated through Arm NN with unsupported ops handled by the TFLite runtime
- Enables performance and great operator coverage

Debian Packages for Arm NN and ACL

- 20.08 Release of Arm SW stack available on Ubuntu Launchpad PPA. Formal release in Bullseye (Debian 11)
- Benefits:
  - Reliable: All build dependencies are taken care of by the robust Debian packaging infrastructure
  - Quick setup: Ready for prototyping with just a few ‘apt-get’ commands
  - Accessible: Ubuntu ‘Groovy’ now available for Raspberry Pi with full Aarch64 desktop experience

PyArmNN

Rapid developing and prototyping

- Python APIs for Arm NN inference engine
- Compatible with Numpy arrays
- Minimal overhead – thin C++ API
- Easy to install using Debian packages

Create a parser object and load your model file.

```
import pyarmnn as ann
import imageio
parser = ann.ITfLiteParser()
network = parser.CreateNetworkFromBinaryFile('./model.tflite')
```

Get the input binding information by using the name of the input layer.

```
input_binding_info = parser.GetNetworkInputBindingInfo(0, 'model/input')
options = ann.CreationOptions()
runtime = ann.IRuntime(options)
```

Choose preferred backends for execution and optimize the network.

```
preferredBackends = [ann.BackendId('CpuAcc'), ann.BackendId('CpuRef')]
opt_network, messages = ann.Optimize(network, preferredBackends,
runtime.GetDeviceSpec(),                                    ann.OptimizerOptions())
net_id, _ = runtime.LoadNetwork(opt_network)
```

Make workload tensors using input and output binding information.

```
img = imageio.imread('./image.png')
input_tensors = ann.make_input_tensors([input_binding_info], [img])
output_binding_info = parser.GetNetworkOutputBindingInfo(0, 'model/output')
output_tensors = ann.make_output_tensors([output_binding_info])
```

Perform inference and get the results back into a numpy array.

```
runtime.EnqueueWorkload(net_id, input_tensors, output_tensors)
results = ann.workload_tensors_to_ndarray(output_tensors)
print(results)
```

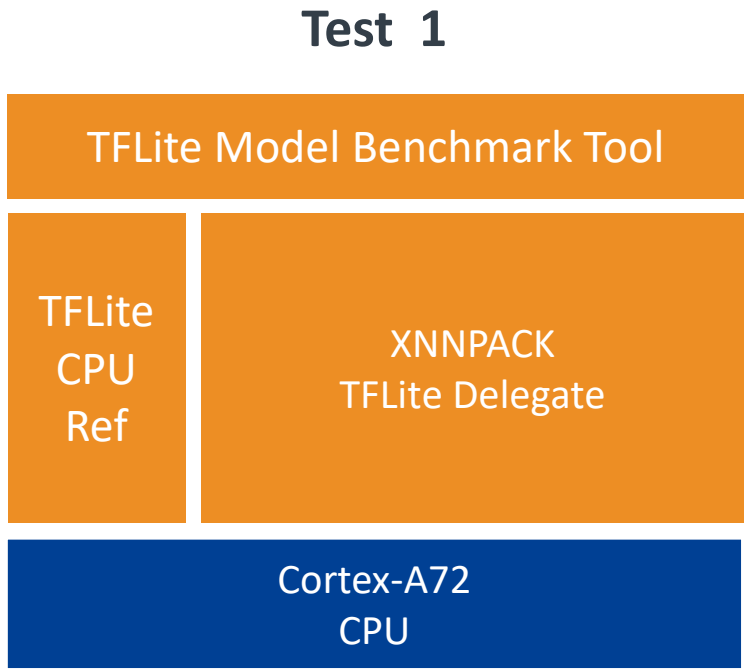
Arm Public Model Zoo

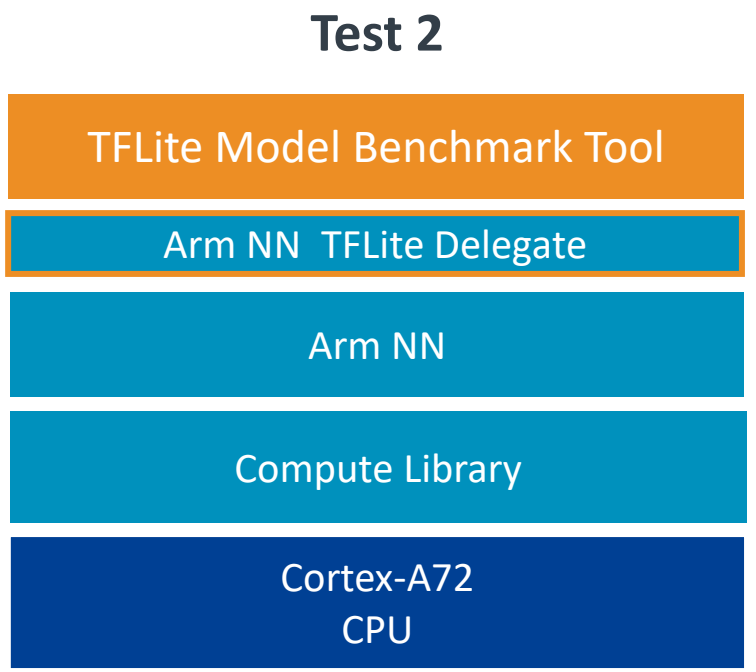
- Highly optimized networks specific to Arm architectures
- Clustering, pruning and quantization aware training used to produce the most efficient models
- Model, meta data and test data available
- Supported by code samples and How-To guides

Model	Improvement
SSD MobileNet v1	Revised for 30% Arm NN performance uplift
MobileNet v2	Clustering and pruning for better accuracy at the same sized
Wav2Letter	10x faster, tuned to mobile / embedded voice UI
DS-CNN	Revised for updated tooling and clustering
Yolo V3	Simplified classes and BB, detector quantization compatible

Demo 1: Inference on Linux using Raspberry Pi4

Examining the performance on Cortex-A72 CPU using TFLite Delegates



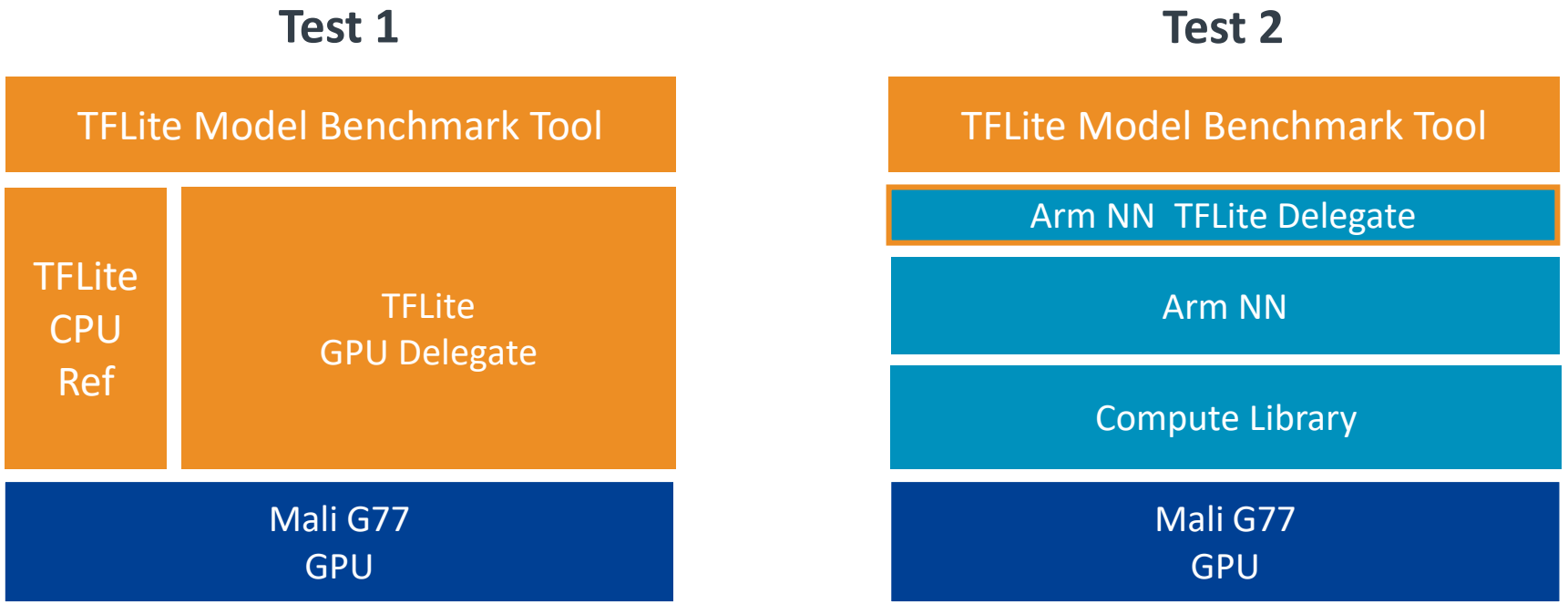


	Yolo V3 (Arm - fp32)	Inception V3 (TFLite - fp32)
Accelerator	Min Exec. time - ms	
TFLite CPU Ref (Test 1a)	505	793.5
TFLite CPU XNNPACK (Test 1b)	372.2	600.9
TFLite CPU ArmNN (Test 2)	246	385.3

~ 1.5x - 1.6x Perf Speed Up over XNNPACK

Demo 2: Inference on Android

Examining the performance on Mali G77 GPU using TFLite Delegates



	Yolo V3 (Arm - fp32)	Inception V3 (TFLite - fp32)
Accelerator	Min. Exec. time - ms	
TFLite GPU delegate (Test 1)	44.8	107.8
TFLite GPU ArmNN w. tuner (Test 2)	26.5	49.2

~ 1.7x - 2.2x Perf Speed Up over TFLite GPU Delegate

Useful Links

- Arm NN HAL for Android: https://github.com/ARM-software/android-nn-driver
- Arm NN TFLite Delegate: https://github.com/ARM-software/armnn/tree/branches/armnn_20_11/delegate
- Arm Model Zoo: https://github.com/ARM-software/ML-zoo/tree/master/models
- Arm ML Examples: https://github.com/ARM-software/ML-examples/
- How to Guides: https://developer.arm.com/solutions/machine-learning-on-arm/
- Ubuntu for Raspberry Pi: https://ubuntu.com/raspberry-pi
- Raspberry Pi OS: https://www.raspberrypi.org/forums/viewtopic.php?t=275370
- Contributions: https://www.mlplatform.org/contributing/

SHA versions

- TensorFlow: 1b215470642efd86e927d1c15ba026b4ff45dfa7
- ArmNN: 97bf84f6e162307fc3e8c53045ef0bc60a3e3289
- ACL: b309fc249e4383b4d40ae03e377c3cbad3f9f5f7

arm

Thank you!

Tweet us: @ArmSoftwareDev

Check out our Arm YouTube channel and our Arm Software Developers YouTube channel

Signup now for our next AI Virtual Tech Talk here

Attendees: don’t forget to fill out the survey to be in with a chance of winning an Arduino Nano 33 BLE board

arm AI

AI Virtual Tech Talks Series

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