arm AI

AI Virtual Tech Talks Series

arm

Running accelerated ML applications on mobile and embedded devices using Arm NN

Arm

James Conroy

8 September 2020

Presenter and Moderator

James Conroy
Software Engineer
Galway, Ireland

Jim Flynn
Arm NN Engineering Manager
Galway, Ireland

arm

AI Virtual Tech Talks Series

Date	Title	Host
September 20, 2020	How To Reduce AI Bias with Synthetic Data for Edge Applications	Dori AI
October 22, 2020	Optimizing Power and Performance For Machine Learning at the Edge - Model Deployment Overview	Arm

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

arm

Agenda

- Why Arm NN?
- Arm NN Overview
- Arm NN Flows
  - Direct to Arm NN
  - Parsers e.g. TF Lite, ONNX, PyTorch
  - Android NN API
- Backends
- PyArmNN
- Future Development
- How to get involved

arm AI

AI Virtual Tech Talks Series

arm

Why Arm NN?

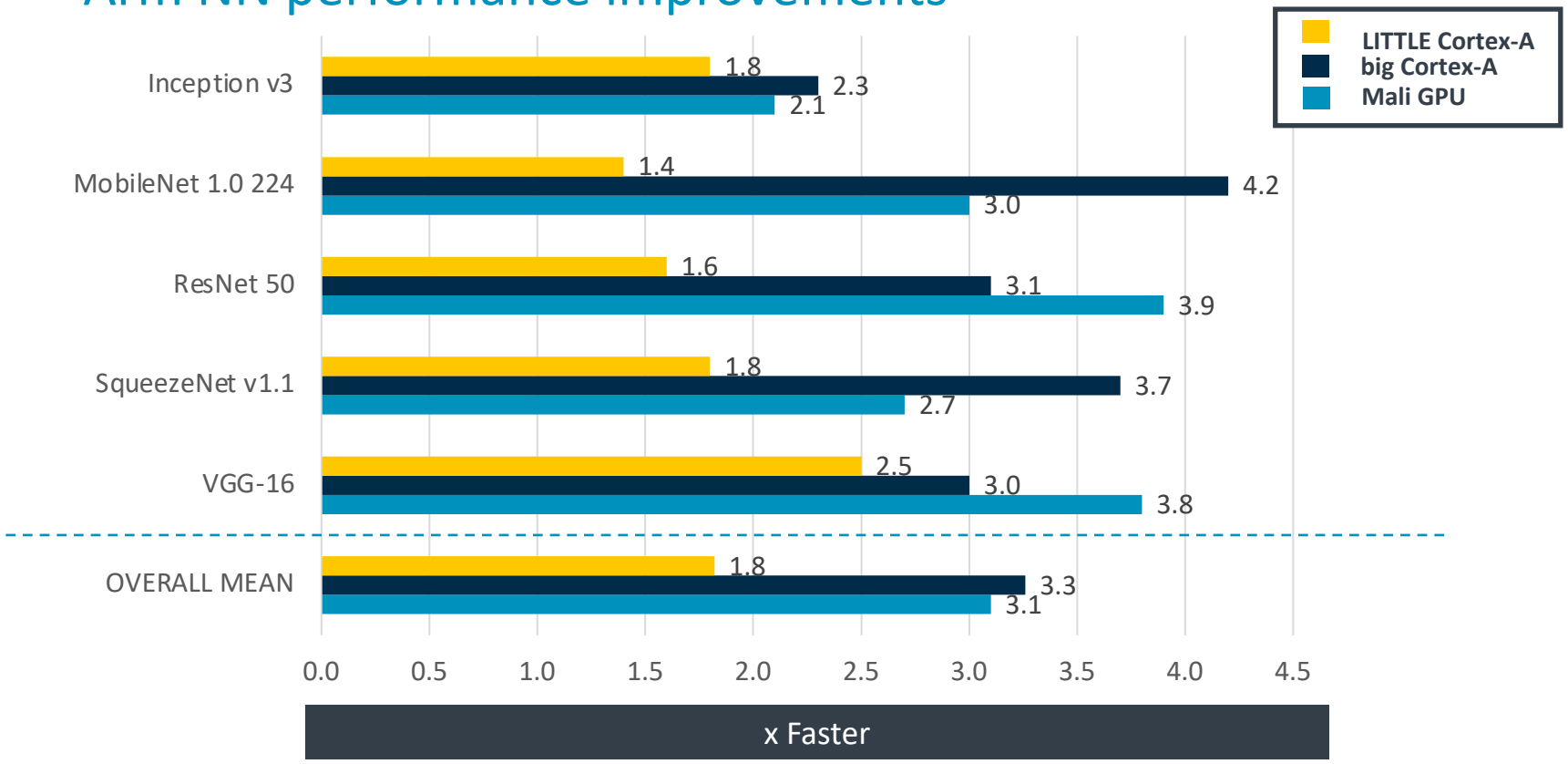
Why Arm NN?

- NN Framework Translation
  - Parsers enable rapid application development through the support of commonly used frameworks such as TF Lite, ONNX, PyTorch and Caffe
- Android NNAPI Support
  - Arm NN interfaces with Google's Android NN using the HAL Driver to target Arm IP
- Efficient Targeting of Arm IP
  - Using Arm Compute Library and Ethos-N Driver Stack

Why Arm NN?

Performance Improvements on Arm IP

Arm NN performance improvements*



- Arm NN shows what's possible for inference network optimizations on Arm IP
- Arm works with Partners to improve Arm hardware performance for other software libraries

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

arm

arm AI

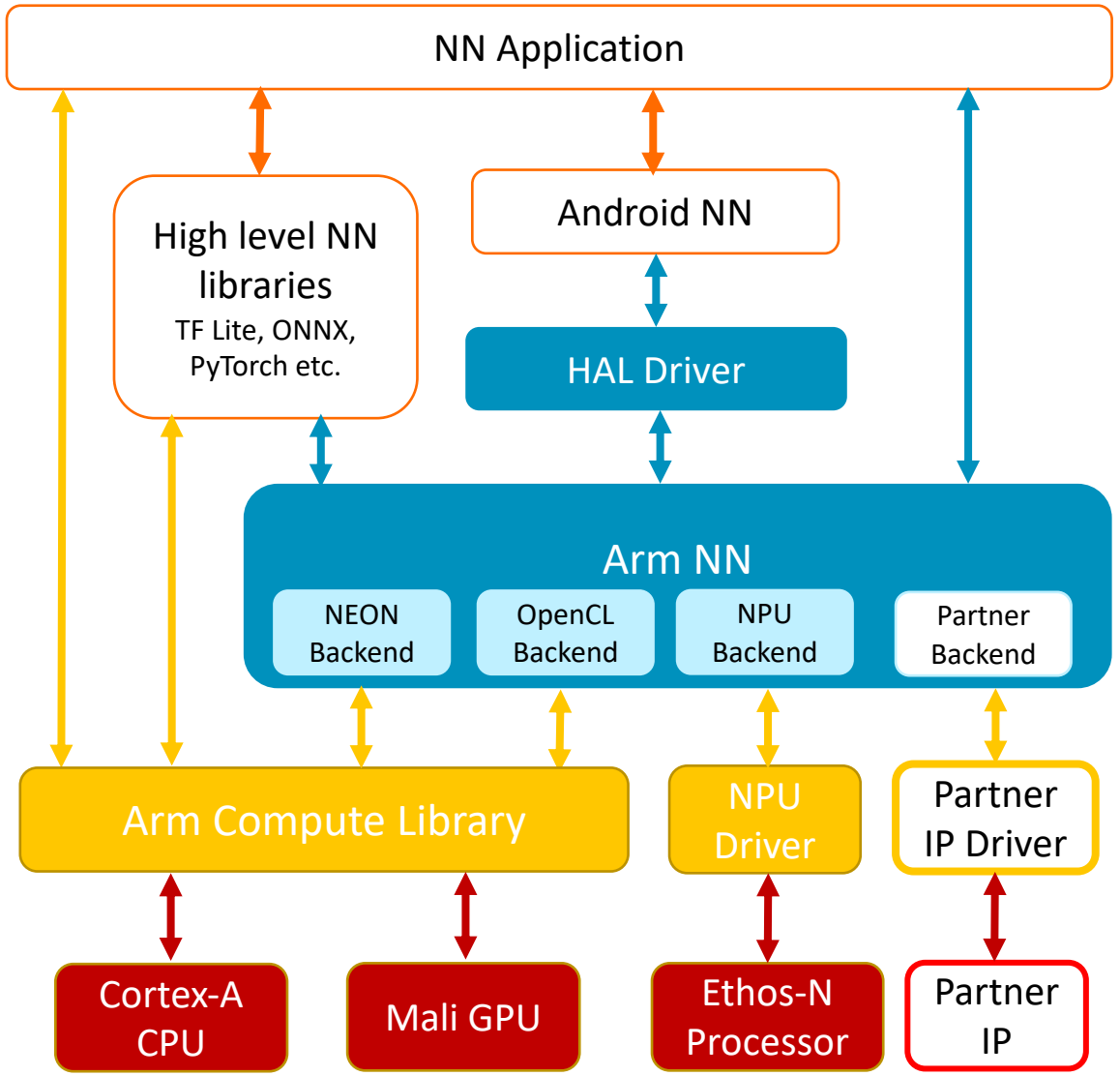
AI Virtual Tech Talks Series

arm

Arm NN Overview

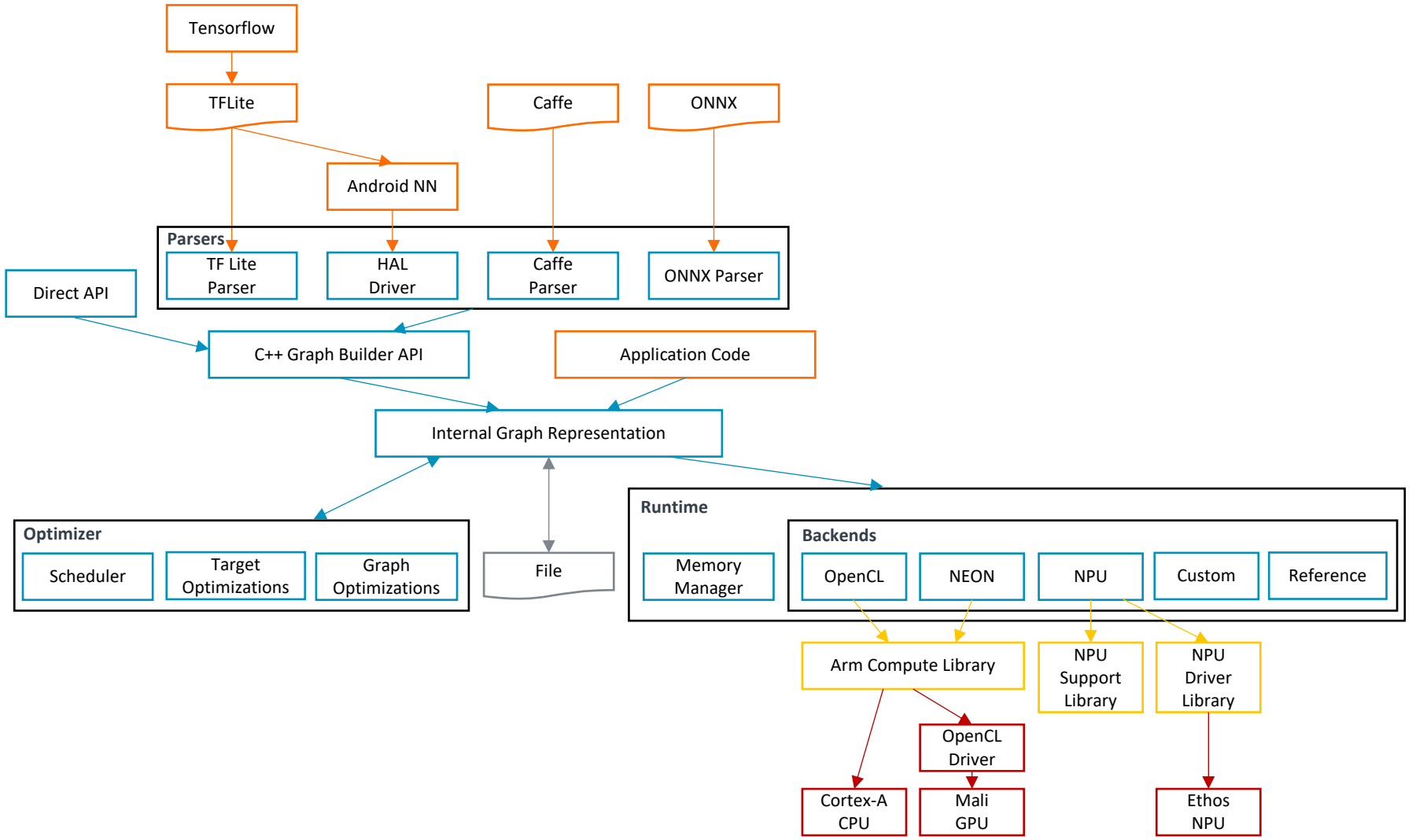
Arm NN Overview

- ML inference API for Linux written in **C++ 14**
- API to access many different NN accelerated devices
- Developed as **open source** and external contributions are always welcome
- Android NN **HAL driver** provides access to Arm NN for Android applications
- **Arm NN** provides the backends for the lower level libraries and hardware drivers
  - Third-party partners can add their own backends for Arm NN
  - Backends can be **dynamically loaded** to Arm NN during the runtime's start-up
- **Arm Compute Library (ACL)**
  - **Arm Cortex-A CPU** with NEON acceleration (ARMv7 and v8x)
  - **Arm Mali GPU** with OpenCL acceleration (Midgard and Bifrost architectures)



Arm NN Components

- Parsers
  - TensorFlow Lite
  - ONNX
  - PyTorch via ONNX
  - Caffe
- Android NN API
  - HAL Driver
- Core
  - Graph Builder API
  - Optimizer
  - Runtime



Arm NN Use Cases

- Image Classification
- Object Recognition
- Super Resolution
- Speech Recognition

Pre-processing(250px x 300px)

ML Algorithm

Post-processing (500 x 600)

arm

Arm NN is Open Source

- Currently released under an **MIT license**
- **Arm NN** started as **quarterly** release, to Arm NN and Android NN Driver on **GitHub**
  - Release schedule, from late 2018, at end of Feb, May, Aug, Nov each year
- **Arm Compute Library** (ACL) releases at same time, to Compute Library on **GitHub**.
  - ACL provides hardware acceleration for Arm CPU and Mali GPU
- Moved to **continuously integrated** development model for both **Arm NN** and **ACL**
  - When accepted into **Linaro** as part the official Artificial Intelligence Initiative
  - Every commit to master is publicly available on review.mlplatform.org
- Note: we are planning to change license from **MIT** to **Apache 2.0** to provide clarity over patents to both contributors and users

Release Process

- The master branch from review.mlplatform.org is automatically mirrored to GitHub

- Releases are still made to **GitHub** every quarter as follows:
  1. Release candidate **branch** created on **mlplatform.org**
  2. Release **testing** and qualification done on **mlplatform.org** until candidate is **certified**
  3. Once certified, the **release process** is kicked-off to generate **documentation**, **release notes** etc.
  4. Final stage of release is to **mirror** the release **branch** to **GitHub** and **announce** public availability

arm AI

AI Virtual Tech Talks Series

arm

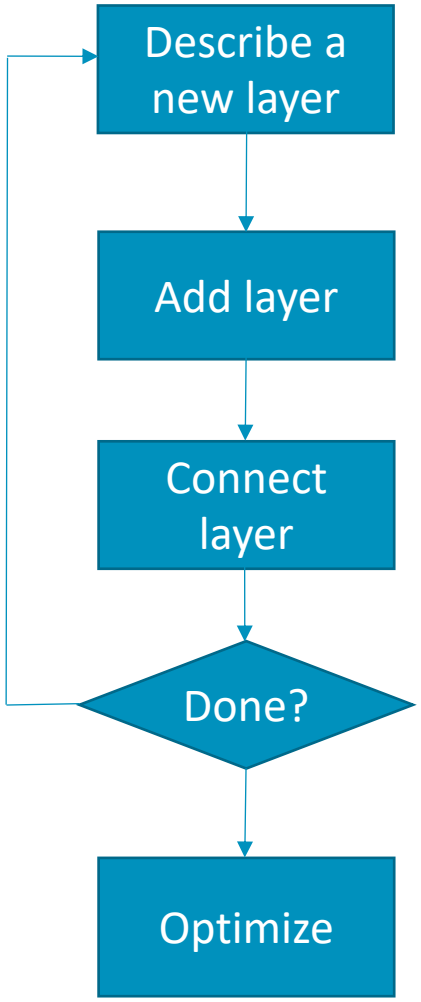
Arm NN Flows

Use Arm NN interfaces directly

Arm NN Parsers

Android NN API Integration

Use Arm NN interfaces directly



- Describe network and optimize it
  - Using the *INetwork* interface, **construct the layers** and **connections** between them
  - Call the network optimizer to allow Arm NN to apply a known set of static **optimizations** to network
- Load the network and execute inference
  - Create an Arm NN *IRuntime* interface
  - Set options on it such as **preferred backends**
  - Load the optimized network
  - Construct input and output tensors
  - Run the inference

arm

Arm NN Parsers

- The parsers are a way to integrate with common ML frameworks

```
ITfLiteParser::TfLiteParserOptions options;
ITfLiteParserPtr parser(ITfLiteParser::Create(armnn::Optional<ITfLiteParser::TfLiteParserOptions>(options)));
armnn::INetworkPtr network = parser->CreateNetworkFromBinaryFile("TestFile.tflite");
```

- The created network can now be **optimized**, and inferences executed

- We deliver a test tool, *ExecuteNetwork*, to load and execute models

```
./ExecuteNetwork -model-format tflite-binary -model-path TestModel.tflite -input-name my_input -output-name
output_layer -input-tensor-data input_data.snpy -compute CpuAcc,CpuRef -write-outputs-to-file results.snpy
```

Android NN API Integration

- The Arm NN HAL Driver interfaces with Google's Android Neural Networks API

- Once the driver is loaded and registered with Android, requests will be translated through the Arm NN interface to execute code optimized for Arm IP

```
/vendor/bin/hw/android.hardware.neuralnetworks@1.2-service-armnn -v -c CpuAcc,GpuAcc -n arm-armnn &
```

- In user code the NN API must be enabled. For TF Lite this is a flag on the interpreter.

```
std::unique_ptr<tflite::Interpreter> interpreter;
interpreter->UseNNAPI(1);
```

arm AI

AI Virtual Tech Talks Series

arm

Backends

Backend Overview

- A backend is an **abstraction** that maps the layers of a network graph to the **hardware** that is responsible for executing those layers
- Support one, or more, layers from the graph
- Create backend-specific workloads for the layers they support
  - Each layer will be executed using a workload
  - A workload is used to enqueue a layer for computation
- Execute the workloads they create
- Arm NN allows statically linked and/or dynamically loaded backends

Custom Backends

- Used to accelerate using specific hardware
- Arm NN allows adding custom backends through the **pluggable backend mechanism**
- All backends must be uniquely identified by a *BackendId*
- Each backend can have backend optimization
- Implement **memory management** to optimize memory usage
- *Backend Context* notifies when a network is loaded or unloaded
- Custom backends can also be loaded at runtime through the **dynamic backend interface**

More information:

https://github.com/ARM-software/armnn/blob/branches/armnn_20_08/src/backends/README.md

https://github.com/ARM-software/armnn/blob/branches/armnn_20_08/src/dynamic/README.md

arm

arm AI

AI Virtual Tech Talks Series

arm

PyArmNN

PyArmNN

- A Python extension for Arm NN SDK
- Interface is similar to Arm NN C++ API
- Built around public headers of Arm NN
- All operations are delegated to the Arm NN library
- Uses SWIG to generate the Arm NN python shadow classes and C wrapper

Available at:

https://github.com/ARM-software/armnn/tree/branches/armnn_20_08/python/pyarmnn

PyArmNN

- More information on PyArmNN:

https://github.com/ARM-software/armnn/blob/branches/armnn_20_08/python/pyarmnn/README.md

- Examples can be found at:

https://github.com/ARM-software/armnn/tree/branches/armnn_20_08/python/pyarmnn/examples

arm AI

AI Virtual Tech Talks Series

arm

Future Development

Planned for End Of November Release (20.11)

- **Usability**
  - Debian Package (verified for deployment on the Odroid N2+ board, Rasberry Pi 4)
  - TensorFlow Lite Delegate
  - Updated website and documentation with many more examples across a range of problem domains

- **Performance**
  - Fastmath option (Winograd Convolution)
  - Operator Fusing/Folding

arm AI

AI Virtual Tech Talks Series

arm

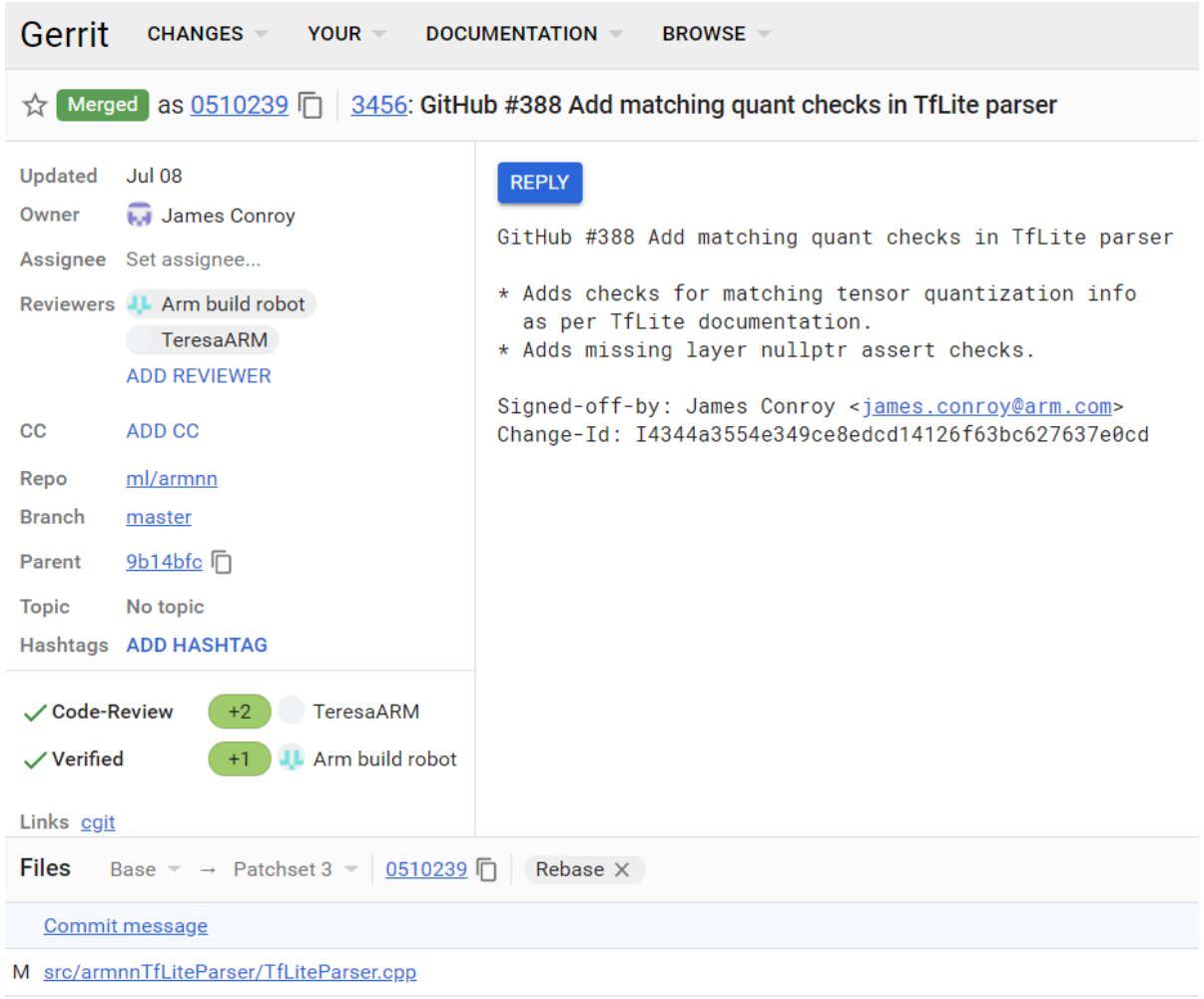
How to get involved

Contributing code to Arm NN

- All code reviews are performed on Linaro ML Platform Gerrit
- GitHub account credentials are required for creating an account on ML Platform
- Setup Arm NN git repo
  - git clone https://review.mlplatform.org/ml/armnn
  - cd armnn
  - git config user.name "FIRST_NAME SECOND_NAME"
  - git config user.email your@email.address
- Commit using sign-off and push patch for code review
  - git commit **-s**
  - git push origin HEAD:refs/for/master
- Patch will appear on ML Platform Gerrit here
- The Contributor Guide contains details of copyright notice and developer certificate of origin sign off

arm

Reviewing code

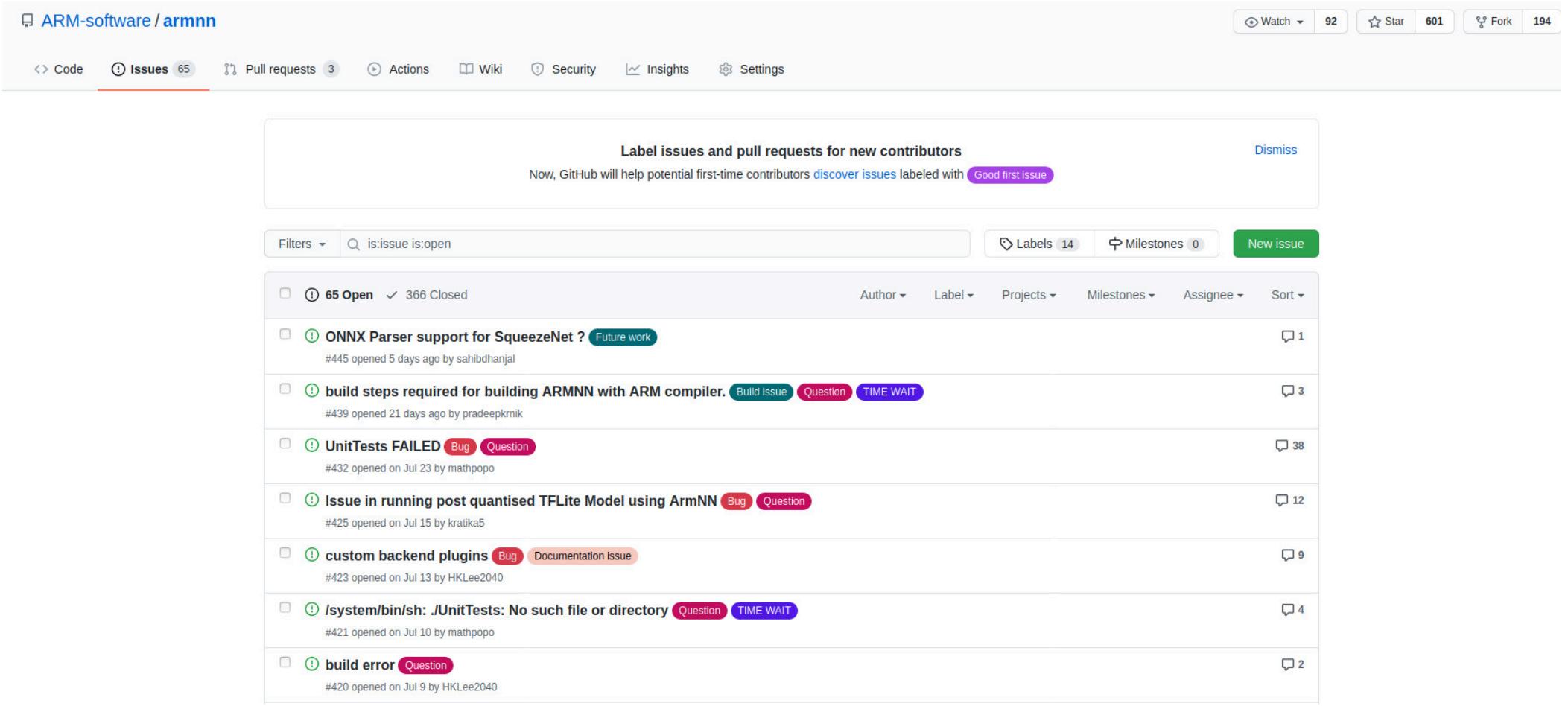


- Core contributors can give *+2/-2* reviews and submit code
- All contributors can give *+1/0/-1* code reviews
- All patches require *+1 Verified* from CI testing and verification
- See Gerrit Review UI docs for more information

arm

Reporting Issues

- Issues are reported via GitHub at https://github.com/ARM-software/armnn/issues



arm

Arm NN Tutorials

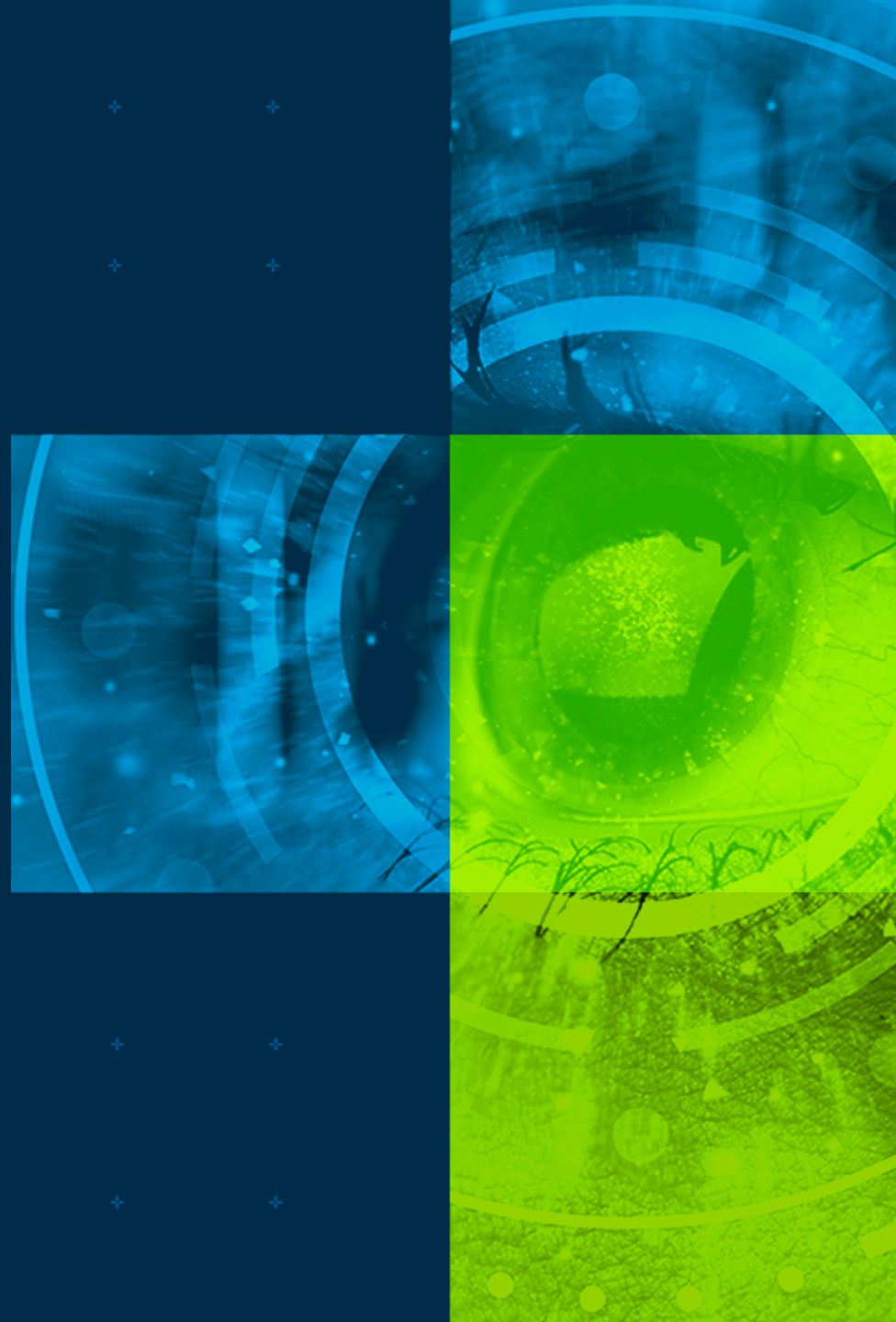
- Configuring Arm NN
  - TF Lite: Configuring the Arm NN SDK build environment for TensorFlow Lite
  - ONNX: Configuring the Arm NN SDK build environment for ONNX
- Deployment Examples
  - Quantized TF Lite: Deploying a quantized TensorFlow Lite MobileNet v1 model
  - Style Transfer on Android: Implementing a neural style transfer on Android
  - Text-to-Speech: Creating a Text-to-speech engine with Tesseract and Arm NN on Raspberry Pi
  - PyArmNN: Accelerating ML Inference on Raspberry Pi with PyArmNN
- Customization
  - Custom backends: Building Arm NN custom backend plugins

arm DEV SUMMIT

Join us at Arm DevSummit

Oct 6 - 8 l Virtual Conference

Register here https://devsummit.arm.com/arm-ai-ml



arm AI

AI Virtual Tech Talks Series

arm

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

תודה

arm AI

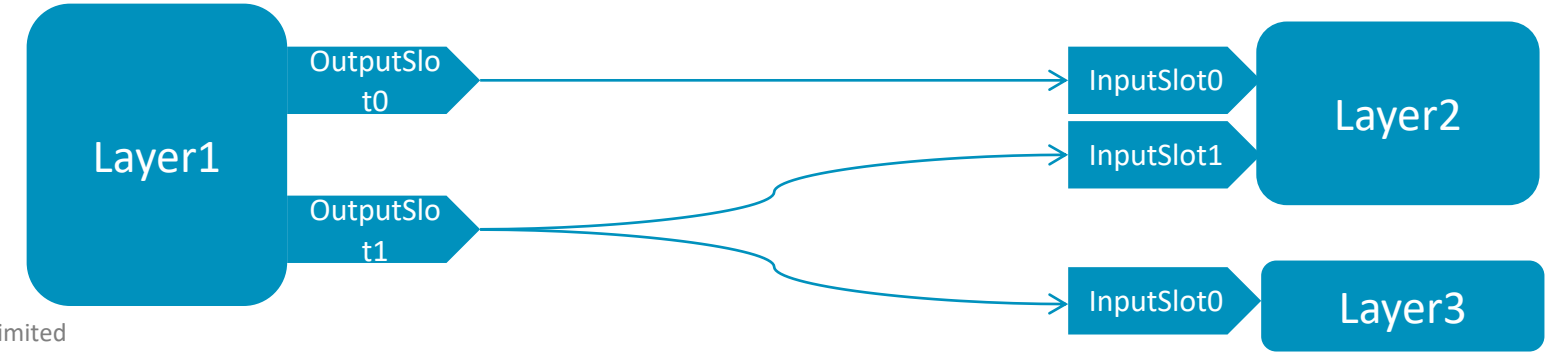
AI Virtual Tech Talks Series

arm

Backup Slides

Graph Builder

- Provides step-by-step API for building a complete model
  - Sometimes also called the model builder
- Layers are connected via the *IConnectableLayer* interface using a slot mechanism
- Each layers has *[0-n]* input slots and *[0-n]* output slots
  - Number of input/output slots depends on the layer type
  - Most are fixed but some can be variable (e.g. Merger/Splitter layer)
- Output slots are connected to 1 or more input slots
- Input slots are connected to only 1 output slot



Optimizer

- Performs basic validation of the input network
- Modifies the network graph
  - Inserts FP32/FP16 conversion layers if necessary (specified in `OptimizerOptions`)
  - Adds debug layers, if required (specified in `OptimizerOptions`)
- Performs backend-independent optimizations
  - Removes redundant operations
  - Optimizes permutes/reshapes where possible (inverse permutes, permutes as reshapes, consecutive reshapes, …)

Optimizer

- Decides which backend to assign to each layer
  - If the layer is not supported, it asks the next preferred backend, and so on...
- Runs backend-specific optimizations
  - For each selected backend, extracts the subgraphs that can be executed on that backend
  - For each subgraph, call OptimizeSubGraph on the selected backend
- To ensure the backend is assigned to a layer, it must be the first *BackendId* in the list of preferred backends

Runtime

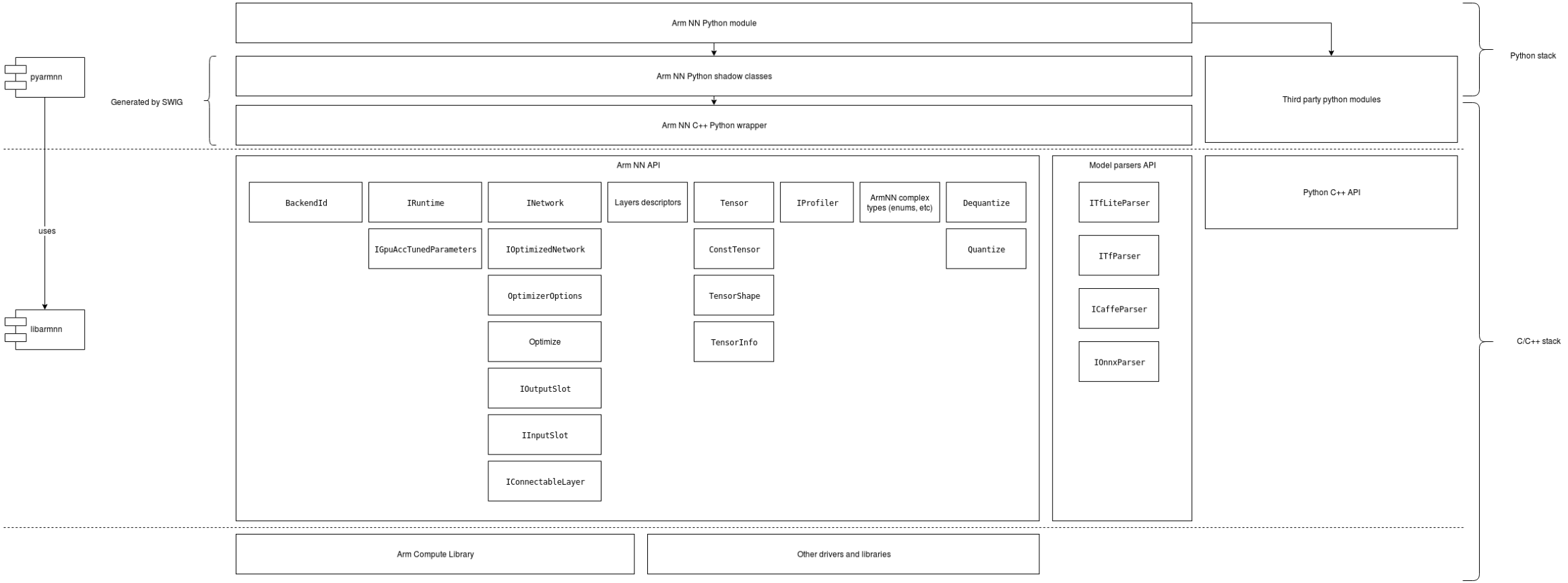
- Loads an optimized network
- Creates the input and output tensors
- Manages runtime memory
- Executes inference/predictions through backends

Sample app: armnn/samples/SimpleSample.cpp

Workloads

- Each layer will be executed using a workload
- A workload is used to enqueue a layer for computation
- Each workload is created by a *WorkloadFactory*
- Each backend needs its own *WorkloadFactory*
  - Creates workloads specific to each layer

Conceptual Architecture of PyArmNN



arm AI

AI Virtual Tech Talks Series

arm

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

תודה