Arm NN 19.08 Improvements

Sadik Armagan, Jim Flynn

Agenda

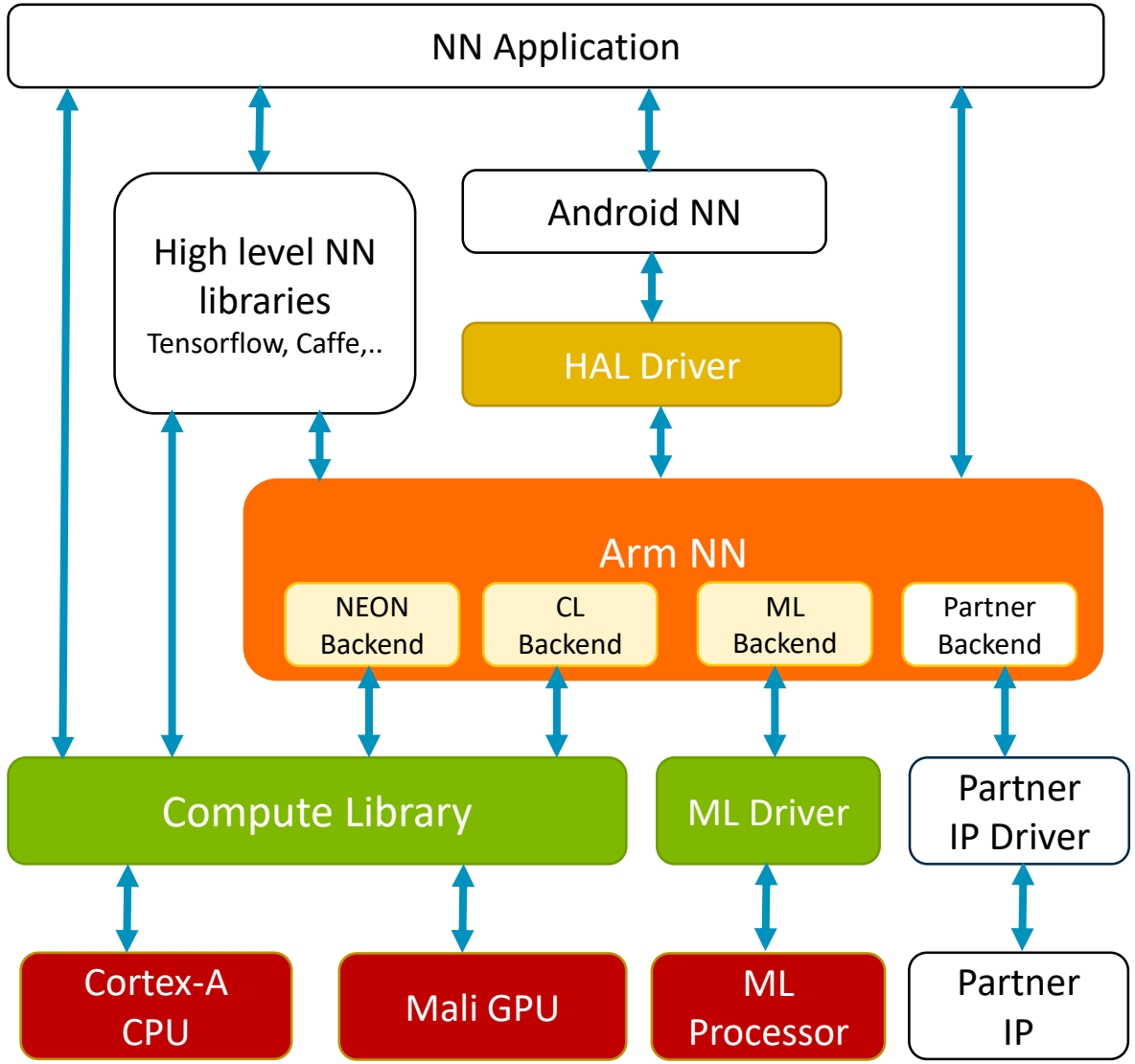
- Arm NN Overview
- Dynamic Backend Loading
- Arm Android NN HAL Driver Improvements
- External Profiling Support First Phase
- Questions

Arm NN Overview

arm

Arm NN Overview

- NN applications and high level libraries can use Arm NN as a single API to access many NN accelerated devices
- Arm Android NN HAL driver provides access to Arm NN for Android applications
- Arm NN provides the backends for the lower level libraries
  - Third-party partners can add their own backends for Arm NN
  - Backends can be dynamically loaded to Arm NN during the runtime's startup
- Some applications use the Compute Library directly, they can still do so
- NN Inference only (training is not supported)



arm

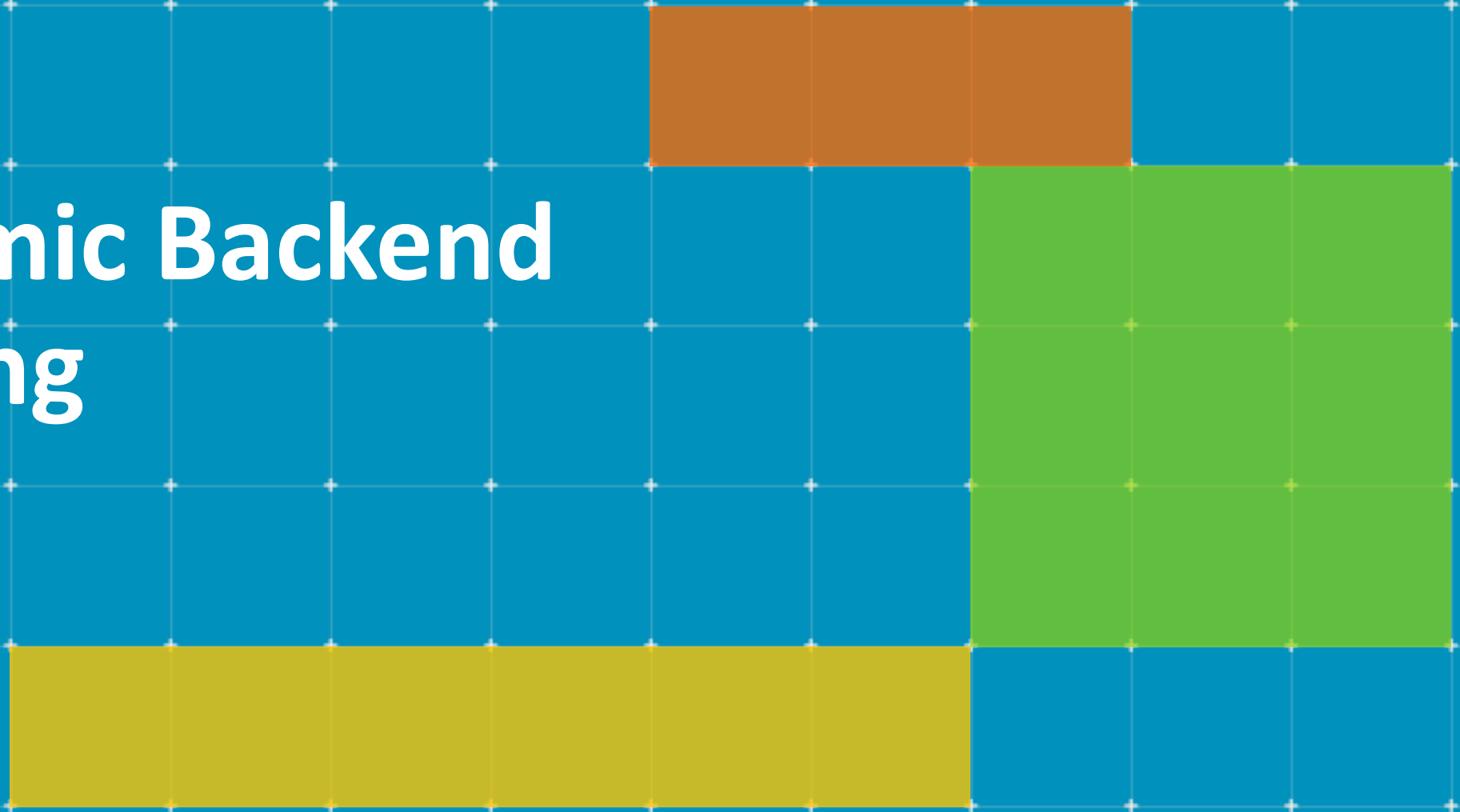
Arm NN Overview (continued)

- C++ 14 ML inference API for Linux
- Developed as Open Source software
  - Contributions are welcome from anyone and are reviewed before acceptance
- Synchronised release with Compute Library and Android NNAPI driver libraries
  - Compute Library
    - Arm CPU with NEON acceleration (ARMv7 and v8x)
    - Arm Mali GPU with OpenCL acceleration (Midgard and Bifrost architectures)
  - Android NNAPI driver
    - Forwards Android NNAPI HAL calls to the Arm NN API

Arm NN	https://review.mlplatform.org/#/admin/projects/ml/armnn
Arm Compute Library	https://review.mlplatform.org/#/admin/projects/ml/ComputeLibrary
Arm Android NN Driver	https://review.mlplatform.org/#/admin/projects/ml/android-nn-driver

arm

Dynamic Backend Loading



arm

Backends Overview

- A backend is an abstraction that maps the layers of a network graph to the hardware that is responsible for executing those layers
- Backends support one, or more, layers from the graph
- They 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
- They execute the workloads they create
- Backends reside under ‘armnn/src/backends’ in separate subdirectories
- Arm NN allows adding new backends through the Pluggable Backend mechanism
  - my_backend is the example directory for the custom backend
    - You create this directory, add the make files and your source files
    - Choose your own name for this directory (must be unique amongst all backends)

```
.../armnn/src/backends
├── aclCommon
│   └── ...
├── backends.cmake
├── backendsCommon
│   ├── BackendRegistry.hpp
│   ├── IBackendContext.hpp
│   ├── IBackendInternal.hpp
│   ├── IMemoryManager.hpp
│   └── ...
├── cl
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── neon
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── my_backend
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── README.md
└── reference
    └── ...
```


arm

Dynamic Backend Loading

- Arm NN allows statically linked and/or dynamically loaded backends
- The Dynamic Backend object must expose the following interface for Arm NN to handle it correctly:

```
extern "C"
{
    const char* GetBackendId();                                  // must return the unique id of the dynamic backends
    void GetVersion(uint32_t* outMajor, uint32_t* outMinor);     // must indicate the version of the dynamic backend
    void* BackendFactory();                                      // must return a valid instance of the backend
}
```

- Arm NN will scan a given set of paths searching for suitable dynamic backend objects to load during the creation of Runtime
  - A list of absolute paths can be specified at compile-time by setting a define named 'DYNAMIC_BACKEND_PATHS'
    - e.g: -DDYNAMIC_BACKEND_PATHS="PATH_1:PATH_2….:PATH_N"
  - Those paths can be overridden when creating the Runtime object by setting the value of the 'm_DynamicBackendsPath' member in the 'CreationOptions' class
    - Only one path is allowed for the override via the 'CreationOptions' class
    - By setting the value of the 'm_DynamicBackendsPath' to a path in the filesystem, Arm NN will entirely ignore the list of paths passed via the 'DYNAMIC_BACKEND_PATHS' compiler directive

arm

Dynamic Backend Loading (continued)

- Arm NN discovers all the backends available and dynamically loads any it might find during the runtime's startup
  - Arm NN will try to load only the files that match the following accepted naming scheme:
    - **<vendor>_<name>_backend.so[<version>] (e.g. "Arm_GpuAcc_backend.so" or "Arm_GpuAcc_backend.so.1.2.3")**
  - Symlinks to other files are allowed to support the standard linux shared object versioning:
    - Arm_GpuAcc_backend.so -> Arm_GpuAcc_backend.so.1.2.3
    - Arm_GpuAcc_backend.so.1 -> Arm_GpuAcc_backend.so.1.2.3
    - Arm_GpuAcc_backend.so.1.2 -> Arm_GpuAcc_backend.so.1.2.3
    - Arm_GpuAcc_backend.so.1.2.3
- To be loaded properly, a dynamic backend must declare a version that is compatible with the current version of the Arm NN Backend API
  - A backend is guaranteed to be compatible when it has been compiled with the same major version of Arm NN's Backend API, and equal to or greater than minor version
    - Dynamic backend version 2.4 (i.e. built with Backend API version 2.4) is compatible with Arm NN's Backend API version 2.4 (same version, backend built against the same Backend API)
    - Dynamic backend version 2.1 (i.e. built with Backend API version 2.1) is compatible with Arm NN's Backend API version 2.4 (same major version, backend built against earlier compatible API)
    - Dynamic backend version 2.0 (i.e. build with Backend API version 2.0) is **not** compatible with Arm NN' Backend API version 1.0 (backend requires a completely new API version)

Dynamic Backend Loading (continued)

- Arm NN will try to load the dynamic backends in the same order as they are parsed from the file system
- A dynamic implementation of the reference backend is also provided.
  - The source files are:
    - **[RefDynamicBackend.hpp](dynamic/reference/RefDynamicBackend.hpp)**
    - **[RefDynamicBackend.cpp](dynamic/reference/RefDynamicBackend.cpp)**
  - The makefile used for building the reference dynamic backend is:
    - **[CMakeLists.txt](dynamic/reference/CMakeLists.txt)**
  - An example mock dynamic backend is also provided in the source code for testing purposes:
    - **[TestDynamicBackend.hpp](backendsCommon/test/TestDynamicBackend.hpp)**
    - **[TestDynamicBackend.cpp](backendsCommon/test/TestDynamicBackend.cpp)**
  - More information about Arm NN Backend Mechanism can be found at 'src/backends/README.md'

```
.../armnn/src/backends
├── aclCommon
│   └── ...
├── backends.cmake
├── backendsCommon
│   ├── BackendRegistry.hpp
│   ├── IBackendContext.hpp
│   ├── IBackendInternal.hpp
│   ├── IMemoryManager.hpp
│   ├── ...
│   └── test
│       ├── TestDynamicBackend.hpp
│       ├── TestDynamicBackend.cpp
│       └── ...
├── dynamic
│   └── reference
│       ├── CMakeLists.txt
│       ├── RefDynamicBackend.cpp
│       └── RefDynamicBackend.hpp
├── cl
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── neon
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── my_backend
│   ├── backend.cmake
│   ├── backend.mk
│   ├── CMakeLists.txt
│   └── ...
├── README.md
└── reference
    └── ...
```


arm

Arm Android NN HAL Driver Improvements

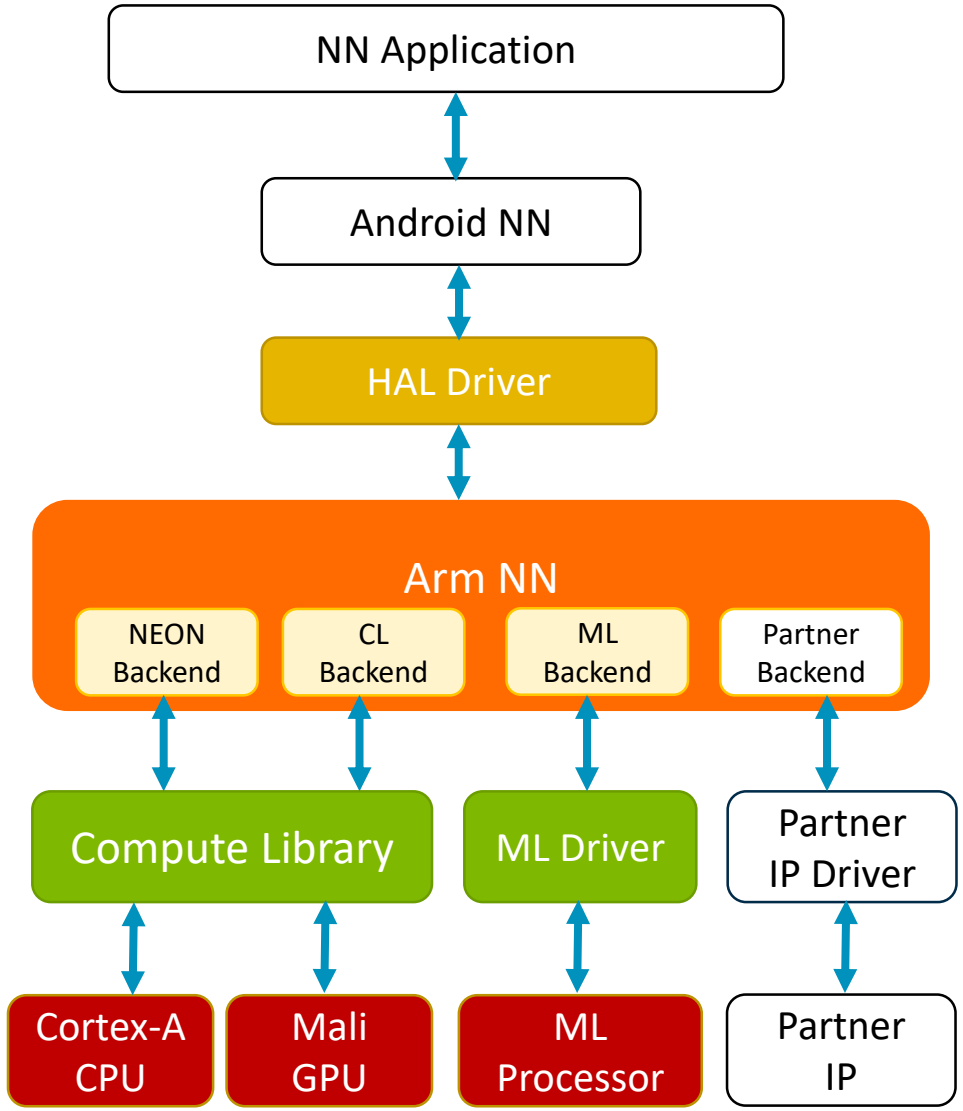
arm

Arm Android NN HAL Driver Improvements

- Arm Android NN HAL Driver for Android NN API
- Android Q fsk-2 support added
  - Currently supporting Android P and Q
  - Added new operator support

CONV_2D with Dilation
DEPTHWISE_CONV2D with Dilation
DEQUANTIZE
LSTM with Normalization Support
MAXIMUM
MINIMUM
PAD_V2
PRELU
RESIZE_BILINEAR
RESIZE_NEAREST_NEIGHBOR
SPACE_TO_DEPTH
QUANTIZE
QUANTIZED_16BIT_LSTM
TRANSPOSE_CONV_2D

- 40 operators are supported in total



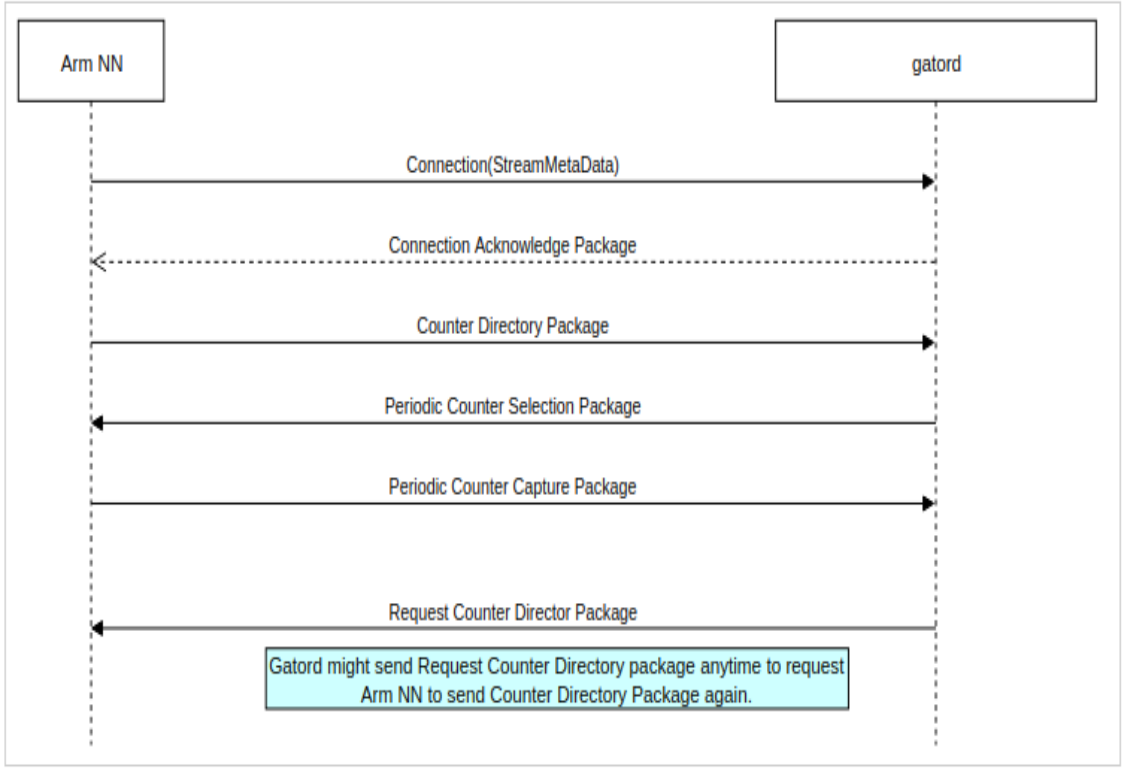
arm

External Profiling Support First Phase

arm

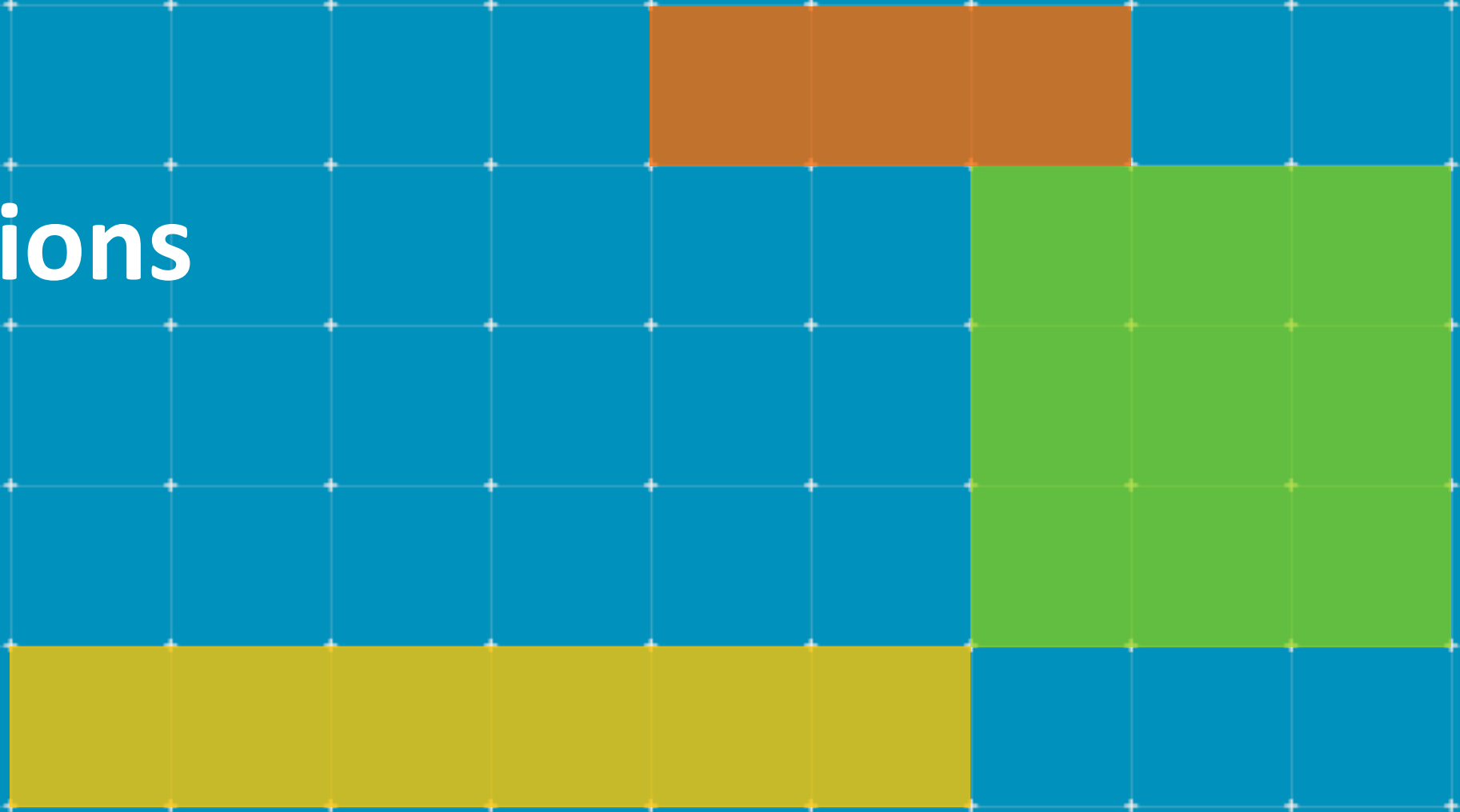
External Profiling Support First Phase

- Provide a good integration with the Arm Development Studio tool and Streamline so that profiling information can be visualized in Streamline with enhanced Machine Learning Awareness
- Defines a new protocol related to the existing **swtrace** and **packet** encoding standards but designed to be better at managing backward compatibility
- The first phase concentrates on providing the infrastructure for communication and exchange of counter metadata, values and processing of send counter metadata and stop/start counter collection commands from Arm Development Studio



arm

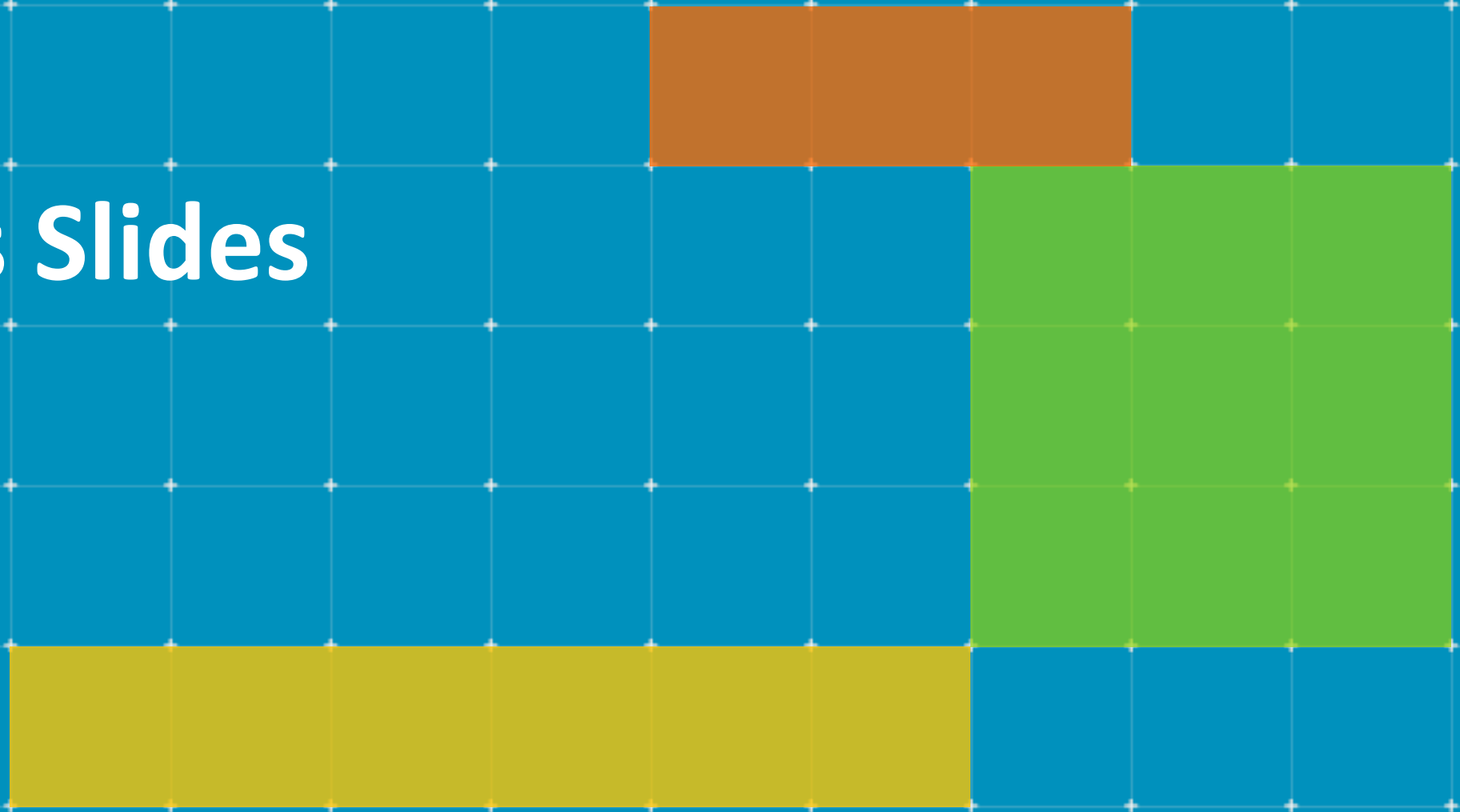
Questions



arm

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

Bonus Slides



arm

ArmnnConverter Application

```
./ArmnnConverter --help
Convert a neural network model from provided file to ArmNN format.

Options:
 --help                                Display usage information
 -f [ --model-format ] arg             Format of the model file, caffe-binary,
                                       caffe-text, onnx-binary, onnx-text,
                                       tensorflow-binary, tensorflow-text, tflite-binary.
 -m [ --model-path ] arg               Path to model file.
 -i [ --input-name ] arg               Identifier of the input tensors in the
                                       network, separated by whitespace.
 -s [ --input-tensor-shape ] arg       The shape of the input tensor in the network
                                       as a flat array of integers, separated by
                                       comma. Multiple shapes are separated by
                                       whitespace. This parameter is optional,
                                       depending on the network.
 -o [ --output-name ] arg              Identifier of the output tensor in the network. If the network
                                       Has multiple outputs this parameter is repeated.
 -p [ --output-path ] arg              Path to serialize the network to.
```

ArmnnQuantizer Application

- Quantizes a Float-32 model, using a representative set of inputs, to produce a QuantizedAsymm-8 or QuantizedSymm-16 model

```
./ArmnnQuantizer

Options:
 --help                                Display usage information
 -f [ --infile ] arg                   Input file containing Float-32 Arm NN Input Graph
 -s [ --scheme ] arg                   Quantization scheme, QAsymm8 or QSymm16, default value QAsymm8.
 -c [ --csvfile ] arg                  CSV file containing paths for RAW input tensors.
 -d [ --outdir ] arg                   Directory that output file will be written to.
 -o [ --outfile ] arg                  Output file name.
 -p [ --preserve-data-type ] arg       Indicate whether to preserve the input and output data types.
```

- The quantized model can be loaded using the Arm NN Seralize/Deserialize API
- Primarily intended as an offline tool

Building the converter and quantizer applications

- To build the ArmnnConverter and ArmnnQuantizer applications you must turn on some options in the build control file `armnn/cmake/GlobalConfig.cmake`

```
./ArmnnConverter
      * option (BUILD_ARMNN_SERIALIZER “Build Armnn Serializer” ON)
      * Plus, at least one of:
         * option (BUILD_CAFFE_PARSER “Build Caffe parser” ON)
         * option (BUILD_TF_PARSER “Build Tensorflow parser” ON)
         * option (BUILD_TF_LITE_PARSER “Build Tensorflow Lite parser” ON)
         * option (BUILD_ONNX_PARSER “Build Onnx parser” ON)

./ArmnnQuantizer
      * option (BUILD_ARMNN_QUANTIZER “Build Armnn quantizer” ON)
```


arm

The Reference Backend

- The reference backend can now be built optionally as all the other backends, it's enabled by default in the global makefile `armnn/cmake/GlobalConfig.cmake`
- To enable/disable it, use the new ARMNNREF CMake option (e.g: add "-DARMNNREF=0" to disable it)
- Alternatively, to make the any change "permanent", change ArmNN's global makefile (`armnn/cmake/GlobalConfig.cmake`) accordingly:
  - By default it is enabled : `option(ARMNNREF "Build with ArmNN reference support" ON)`
  - To disable the reference backend: `option(ARMNNREF "Build with ArmNN reference support" OFF)`
    - Disabling the reference backend will impact some of the unit tests that are built with ArmNN, as many of them use the reference backend as a way to perform cross-verification and end-to-end tests
    - Follow the usage of ARMNNREF through the makefiles and ARMNNREF_ENABLED in the code to know which unit tests may be excluded if the reference backend is disabled

Useful Links

https://review.mlplatform.org/#/admin/projects/ml/armnn

https://review.mlplatform.org/#/admin/projects/ml/ComputeLibrary

https://review.mlplatform.org/#/admin/projects/ml/android-nn-driver

https://github.com/ARM-software/armnn/blob/branches/armnn_19_08/src/armnnCaffeParser/README.md

https://github.com/ARM-software/armnn/blob/branches/armnn_19_08/src/armnnCaffeParser/CaffeSupport.md

https://github.com/ARM-software/armnn/blob/branches/armnn_19_08/src/armnnOnnxParser/README.md

https://github.com/ARM-software/armnn/blob/branches/armnn_19_08/src/armnnOnnxParser/OnnxSupport.md

https://github.com/ARM-software/armnn/blob/branches/armnn_19_08/src/armnnTfLiteParser/README.md

https://github.com/ARM-software/armnn/blob/branches/armnn_19_08/src/armnnTfLiteParser/TensorFlowLiteSupport.md

https://github.com/ARM-software/armnn/blob/branches/armnn_19_08/src/armnnTfParser/README.md

https://github.com/ARM-software/armnn/blob/branches/armnn_19_08/src/armnnTfParser/TensorFlowSupport.md