everyday genius

NeuroPilot v1.07

Revision History

Revision	Date	Description
v1.0 function only	Jan. 2018	NeuroPilot release
v1.05 performance improvement	March 2018	NeuroPilot release
v1.07 performance improvement	March 2018	NeuroPilot release

Table of Contents

Table of Contents 3

1 NeuroPilot Limitation 5

2 NeuroPilot SDK Overview 6

2.1 Purpose of Release 6

2.2 Capabilities 6

2.3 Architecture 6

3 Supported Operation List 8

4 Tensorflow Lite Toco Model Convertor 9

5 Environment Setup SOP 11

6 Sample Code 14

6.1 Android NN Sample Code 14

6.2 NeuroPilot Extension APIs 14

6.3 Profiler 15

6.3.1 ANeuralNetworksCompilation_enableProfiler 15

6.3.2 ANeuralNetworksExecution_getProfilerInfoCount 15

6.3.3 Tutorial 15

6.3.4 Summary 17

6.4 Debugger 18

6.4.1 ANeuralNetworksExecution_keepOperations 18

6.4.2 ANeuralNetworksExecution_getOperationOutput 18

6.4.3 Tutorial 18

6.4.4 Summary 19

6.5 Operation Binding 20

6.5.1 ANeuralNetworks_listRegDevice 20

6.5.2 ANeuralNetworksModel_setRuntime 20

6.5.3 Tutorial 20

6.5.4 Summary 21

6.6 TensorFlow Lite Interpreter 22

6.6.1 ANeuralNetworksTFLite_create 22

6.6.2 ANeuralNetworksTFLite_getTensor 22

6.6.3 ANeuralNetworksTFLite_invoke 23

6.6.4 ANeuralNetworksTFLite_free 23

6.6.5 Tutorial 23

6.7 Force CPU to Execute All Operations 25

6.7.1 ANeuralNetworks_setCpuOnly 25

7 GPU NN Operation Specification 26

8 VPU NN Operation Specification 27

9 Custom Operation 28

9.1 Customized Operation Solution 28

9.2 Add Custom OP by splitting model 28

9.3 Add Custom Op by registering a new HAL device 29

9.3.1 Application Flow with Android NN Operations 29

9.3.2 Application Flow with Customized Operations 29

9.3.3 Steps for Adding a Custom Operation 30

9.3.4 The mechanism of Android NN HAL 31

1 NeuroPilot Limitation

Category	Limitation
Supported Operations	Not all operations of Android NN are supported. Details in chapter **Supported Operation List.**
GPU Operations	For performance consideration, some GPU operations are implemented using FP16 computation and precision will be reduced.

2 NeuroPilot SDK Overview

2.1 Purpose of Release

NeuroPilot SDK release is for developers to pre-integrate their application on Mediatek MT8183 platform. Developers can:

- Convert pre-trained TensorFlow model (with limited operations) into TensorFlow Lite model, and deploy on Mediatek MT8183 platform.
- Deploy pre-converted TensorFlow Lite model on Mediatek MT8183 platform.
- Develop neural network applications by directly programming with Android Neural Network API and NeuroPilot extension API.

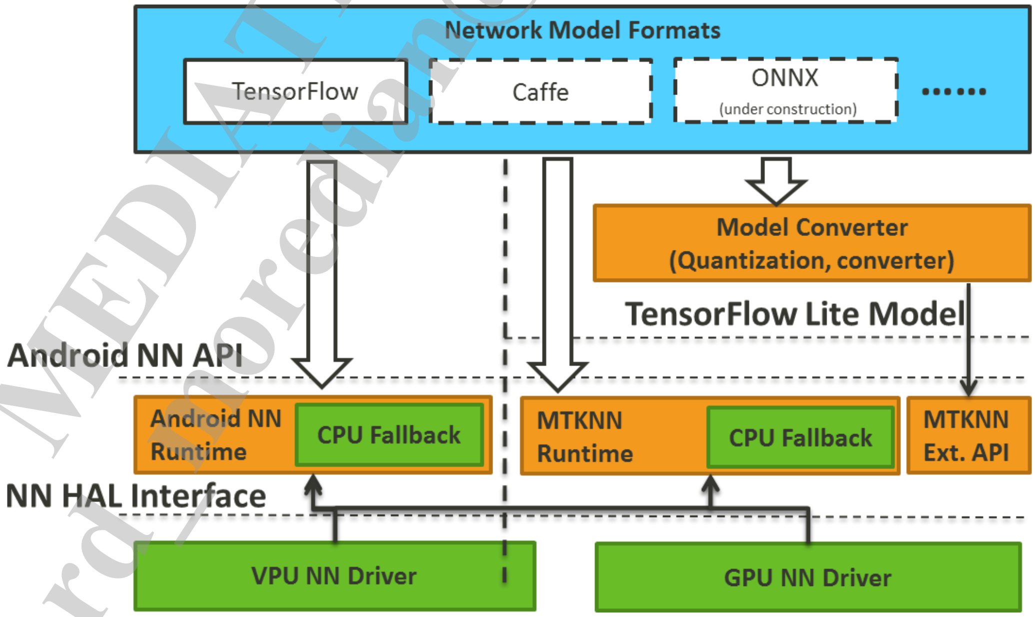
Performance optimization of supported operations is still on going. Supported list of operations will be extended in future release version.

2.2 Capabilities

- Convert TensorFlow model into TensorFlow Lite model
- Dispatch operations of a network model into CPU, GPU, and VPU
- Integrate pre-trained neural network models with C++ programming API
- Profile and debug neural network applications

2.3 Architecture

NeuroPilot Runtime is compatible with Android Neural Network API. The implementation of both VPU and GPU HAL devices is in conformity with the Android NN HAL interface definition. On the top of NeuroPilot Runtime, TensorFlow Lite based interpreter is integrated to interpret TensorFlow Lite model to Android Neural Network API. Model converter turns TensorFlow model into TensorFlow Lite model for running on TensorFlow Lite based interpreter.



- Model Convertor

TOCO which is part of the TensorFlow project is also privoded in the NeuroPilot SDK release.TOCO is an offline tool to convert pre-trained TensorFlow model into TensorFlow model. Currently only small subset of TensorFlow operations can be converted into TensorFlow Lite format. Pre-trained TensorFlow model with operations not in the supported list can't be converted now.

- TensorFlow Lite Interpreter
  TensorFlow Lite is the light weight solution of TensorFlow for mobile and embedded device. It supports hardware acceleration with Android Neural Network API. TensorFlow uses protocol buffer based file format, while TensorFlow Lite uses FlatBuffer based file format. NeuroPilot SDK extends the list of supported operations of TensorFlow Lite. TensorFlow Lite interpreter is already integrated into NeuroPilot SDK. Developers can easily use extension API to load a .tflite model converted by the Model convertor.

- NeuroPilot NN Runtime
  NeuroPilot Runtime is compatible with Android Neural Network API, so program developed with Android NN API can be linked to NeuroPilot Runtime with no effort. NeuroPilot Extended API provides facilities to profile and debug neural network applications and also provides interface to pin target operation on a specific hardware accelerator, like GPU or VPU.

- VPU and GPU NN Driver
  VPU and GPU NN Drivers provide implementation of hardware accelerated operations for NeuroPilot runtime. Currently VPU NN Driver supports only quant8 data type and GPU NN Driver supports only FP32 data type operations. The operations defined by Android Neural Network API could be accelerated by the corresponding device according to the data type in NeuroPilot.

3 Supported Operation List

(*): Limitation

Operation	CPU		GPU		VPU		TFLite Interpreter	Model Convertor
	FP32	Quant8	FP32	Quant8	FP32	Quant8		
ADD	V	V	V(*)			V	V	V
AVERAGE_POOL_2D	V	V	V(*)			V(*)	V	V
CONCATENATION	V	V	V(*)			V(*)	V	V
CONV_2D	V	V	V			V(*)	V	V
DEPTH_TO_SPACE	V	V	V			V	V	V
DEPTHWISE_CONV_2D	V	V	V(*)			V(*)	V	V
DEQUANTIZE		V					V	V
EMBEDDING_LOOKUP	V	V	V					
FLOOR	V		V				V	V
FULLY_CONNECTED	V	V	V(*)			V(*)	V	V
HASHTABLE_LOOKUP	V	V	V					
L2_NORMALIZATION	V	V	V				V	V
L2_POOL_2D	V		V(*)				V	V
LOCAL_RESPONSE_NORMALIZATION	V		V				V	V
LOGISTIC	V	V	V			V	V	V
LSH_PROJECTION	V	V	V					
LSTM	V	V						
MAX_POOL_2D	V	V	V(*)			V(*)	V	V
MUL	V	V	V(*)			V	V	V
RELU1	V	V	V			V		
RELU6	V	V	V			V	V	V
RELU	V	V	V			V	V	V
RESHAPE	V	V	V(*)			V	V	V
RESIZE_BILINEAR	V		V			V(*)	V	V
RNN	V							
SOFTMAX	V	V	V			V	V	V
SPACE_TO_DEPTH	V	V	V			V	V	V
SVDF	V	V						
TANH	V		V				V	V

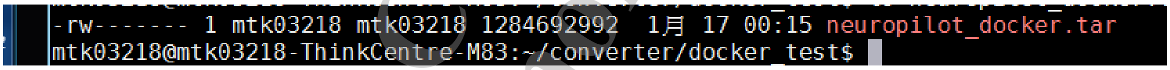
4 TensorFlow Lite Toco Model Convertor

In NeuroPilot, TOCO, The TensorFlow Lite Optimizing Converter, is provided to convert TensorFlow model into TensorFlow Lite model. TOCO originally comes from TensorFlow project and it is extended in NeuroPilot to provide capability to covert more types neural network operations from TensorFlow model to TensorFlow Lite model. For more detail about TOCO, please check the TensorFlow project on GitHub:
https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/lite/toco

TOCO supports only the conversion from frozen TensorFlow graph to TensorFlow Lite graph. To avoid library dependency issues, extended TOCO is provided alone with a Docker image. There are 4 sample models, mnist, MobileNet, InceptionV3 2015, and SqueezeNet V1.1, in the Docker image for TOCO usage demostration. Please follow the steps to setup the environment and convert sample models:

1. Please follow Docker's doc https://docs.docker.com/engine/installation/ to install Docker in your Linux host machine. If Docker was already installed, please skip this step.

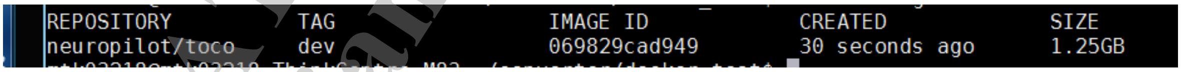
2. Copy the compressed Docker image file into your Linux host machine and decompress it
   # gunzip neuropilot_docker.tar.gz
   # ls neuropilot_docker.tar -l



```
-rw------- 1 mtk03218 mtk03218 1284692992  1月 17 00:15 neuropilot_docker.tar
mtk03218@mtk03218-ThinkCentre-M83:~/converter/docker_test$
```

3. Import Docker image:
   # docker import neuropilot_docker.tar neuropilot/toco:dev

   And you can check the image was successfully imported:
   # docker images



```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
neuropilot/toco     dev                 069829cad949        30 seconds ago      1.25GB
```

4. Launch container with the imported image:
   # docker run -t -i -w /home/tflite neuropilot/toco:dev bash

   And now you're in a Docker container environment with the prompt:
   root@8dab642c7653:/home/tflite#

   In the /home/tflite folder, you will see a script file "convert_sample_models.sh" and a sub-folder "sample_models" which contains the frozen sample model: "frozen_mnist-nodropout.pb" for mnist, "frozen_mobilenet_1.0-nodropout.pb" for MobileNet, "frozen_inceptionv3_2015.pb" for InceptionV3 2015, and "frozen_squeezenet11.pb" for SqueezNet V1.1.

5. Now you can use the script to convert frozen TensorFlow sample models into TensorFlow Lite model by:

```
# ./convert_sample_models.sh mnist
```

The output file “minst.tflite” is the converted TensorFlow Lite model.

Similar for MobileNet. Run:

```
# ./convert_sample_models.sh mobilenet
```

The output file “mobilenet.tflite” is the converted TensorFlow Lite model.

For InceptionV3 2015:

```
# ./convert_sample_models.sh inceptionv3
```

The output file “inceptionv3_2015.tflite” is the converted TensorFlow Lite model.

For SqueezeNet V1.1:

```
# ./convert_sample_models.sh squeezenet
```

The output file “squeezenet11.tflite” is the converted TensorFlow Lite model.

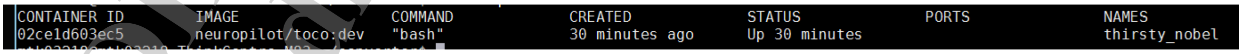
6. You can also use toco command directly to convert the sample models or your target model. The detail model conversion instruction is:

```
# toco --input_file=<Frozen TensorFlow Model File> \
            --input_format=TENSORFLOW_GRAPHDEF \
            --output_format=TFLITE\
            --output_file=<Output TensorFlow Lite Model File Name> \
            --inference_type=FLOAT \
            --inference_input_type=FLOAT \
            --input_arrays=<INPUT_ARRAYS> \
            --output_arrays=<OUTPUT_ARRAYS> \
            --input_shapes=<INPUT_SHAPES>
```

The parameters input_arrays, output_arrays, and input_shapes of the target model can be checked in TensorBoard tool comes with TensorFlow.

7. To copy files to/from Docker container, you can do:
First, check the container name run with the Docker image:

```
# docker ps
```



```
CONTAINER ID        IMAGE                  COMMAND             CREATED             STATUS              PORTS               NAMES
02ce1d603ec5        neuropilot/toco:dev    "bash"              30 minutes ago      Up 30 minutes                           thirsty_nobel
```

To copy files to container:

```
# docker cp <filename> <container_name>:<target path>
```

In this example:

```
# docker cp <filename> thirsty_nobel:/home/tflite/
```

To copy files from container:

```
# docker cp <container_name>:<target path>/<filename> <local path>
```

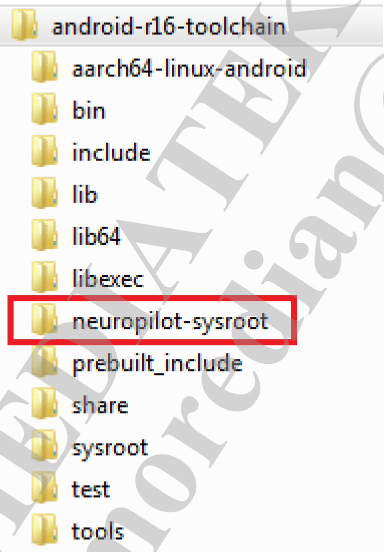
In this example:

```
# docker cp thirsty_nobel:/home/tflite/<filename> <local path>
```

5 Environment Setup SOP

The Mediatek MT8183 platform already provides NeuroPilot runtime execution environment. Developers can use this SOP to develop your own Neural Networks application.

1. Download standard Android r16b NDK
   https://developer.android.com/ndk/downloads/index.html
2. Prepare standalone toolchain
   a. $ cd [toolchain]/build/tools/
   b. $ ./make-standalone-toolchain.sh --arch=[Arch.] --platform=android-27 --install-dir=[NDK Install Path]
      E.g. for arm64
      $ ./make-standalone-toolchain.sh --arch=arm64 --platform=android-27 --install-dir=/XXX/android-r16-toolchain
      E.g. for arm
      $ ./make-standalone-toolchain.sh --arch=arm --platform=android-27 --install-dir=/XXX/android-r16-toolchain-arm32
3. Copy neuropilot-sysroot folder in NDK/[arm64/arm]/ to your own standalone toolchain folder
   E.g.

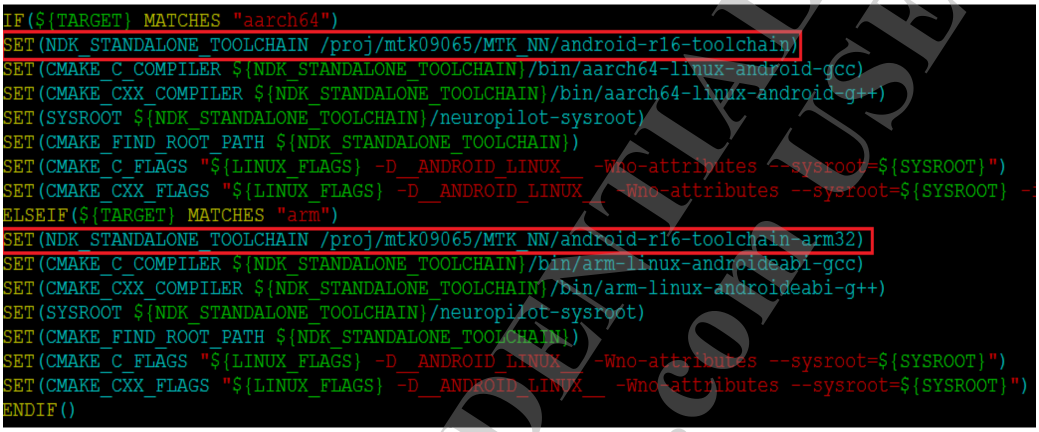


```
android-r16-toolchain
    aarch64-linux-android
    bin
    include
    lib
    lib64
    libexec
    neuropilot-sysroot
    prebuilt_include
    share
    sysroot
    test
    tools
```

4. Change directory to samples
5. Build sample code

a. Edit CMakeLists.txt

Change NDK_STANDALONE_TOOLCHAIN path to yours

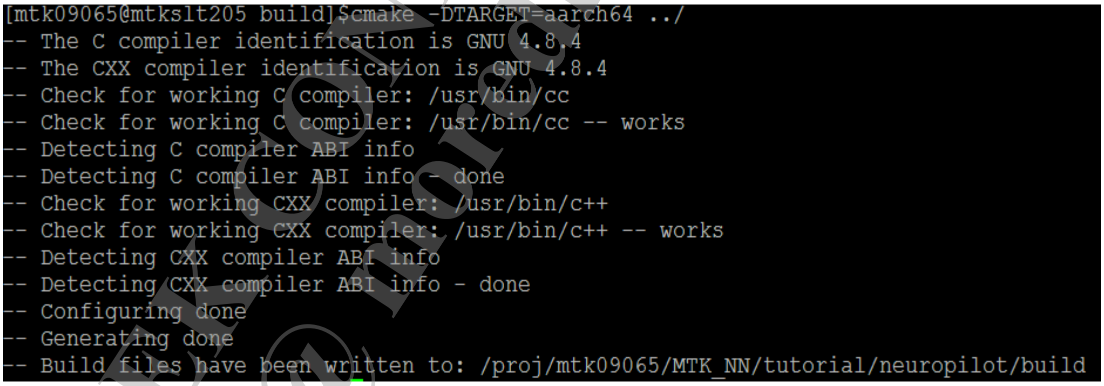


```
IF(${TARGET} MATCHES "aarch64")
SET(NDK_STANDALONE_TOOLCHAIN /proj/mtk09065/MTK_NN/android-r16-toolchain)
SET(CMAKE_C_COMPILER ${NDK_STANDALONE_TOOLCHAIN}/bin/aarch64-linux-android-gcc)
SET(CMAKE_CXX_COMPILER ${NDK_STANDALONE_TOOLCHAIN}/bin/aarch64-linux-android-g++)
SET(SYSROOT ${NDK_STANDALONE_TOOLCHAIN}/neuropilot-sysroot)
SET(CMAKE_FIND_ROOT_PATH ${NDK_STANDALONE_TOOLCHAIN})
SET(CMAKE_C_FLAGS "${LINUX_FLAGS} -D__ANDROID_LINUX__ -Wno-attributes --sysroot=${SYSROOT}")
SET(CMAKE_CXX_FLAGS "${LINUX_FLAGS} -D__ANDROID_LINUX__ -Wno-attributes --sysroot=${SYSROOT} -
ELSEIF(${TARGET} MATCHES "arm")
SET(NDK_STANDALONE_TOOLCHAIN /proj/mtk09065/MTK_NN/android-r16-toolchain-arm32)
SET(CMAKE_C_COMPILER ${NDK_STANDALONE_TOOLCHAIN}/bin/arm-linux-androideabi-gcc)
SET(CMAKE_CXX_COMPILER ${NDK_STANDALONE_TOOLCHAIN}/bin/arm-linux-androideabi-g++)
SET(SYSROOT ${NDK_STANDALONE_TOOLCHAIN}/neuropilot-sysroot)
SET(CMAKE_FIND_ROOT_PATH ${NDK_STANDALONE_TOOLCHAIN})
SET(CMAKE_C_FLAGS "${LINUX_FLAGS} -D__ANDROID_LINUX__ -Wno-attributes --sysroot=${SYSROOT}")
SET(CMAKE_CXX_FLAGS "${LINUX_FLAGS} -D__ANDROID_LINUX__ -Wno-attributes --sysroot=${SYSROOT}")
ENDIF()
```

b. $ mkdir build

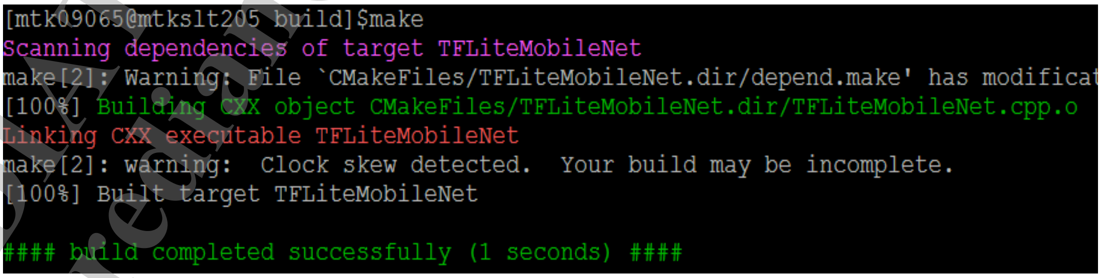
c. $ cd build

d. $ cmake –DTARGET=aarch64 ../ (-DTARGET=arm for 32 bits if you want)



```
[mtk09065@mtkslt205 build]$cmake -DTARGET=aarch64 ../
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: /proj/mtk09065/MTK_NN/tutorial/neuropilot/build
```

e. $ make



```
[mtk09065@mtkslt205 build]$make
Scanning dependencies of target TFLiteMobileNet
make[2]: Warning: File `CMakeFiles/TFLiteMobileNet.dir/depend.make' has modificat
[100%] Building CXX object CMakeFiles/TFLiteMobileNet.dir/TFLiteMobileNet.cpp.o
Linking CXX executable TFLiteMobileNet
make[2]: warning:  Clock skew detected.  Your build may be incomplete.
[100%] Built target TFLiteMobileNet

#### build completed successfully (1 seconds) ####
```

6. Run sample

a. $ adb push TFLiteMobileNet /data/local/tmp

b. $ adb push mobilenet.tflite /data/local/tmp

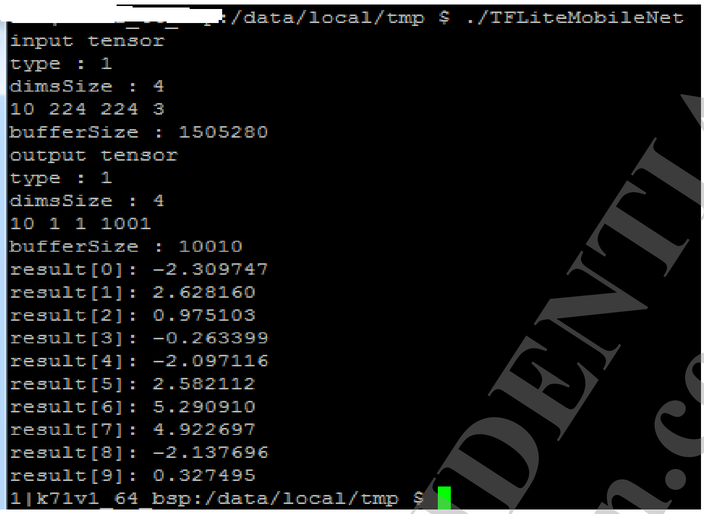
c. $ adb push mobilenet_input.bin /data/local

d. $ adb shell

e. $ cd /data/local/tmp

f. $ chmod +x TFLiteMobileNet

g. $./TFLiteMobileNet



```
[redacted]:/data/local/tmp $ ./TFLiteMobileNet
input tensor
type : 1
dimsSize : 4
10 224 224 3
bufferSize : 1505280
output tensor
type : 1
dimsSize : 4
10 1 1 1001
bufferSize : 10010
result[0]: -2.309747
result[1]: 2.628160
result[2]: 0.975103
result[3]: -0.263399
result[4]: -2.097116
result[5]: 2.582112
result[6]: 5.290910
result[7]: 4.922697
result[8]: -2.137696
result[9]: 0.327495
1|k71v1_64_bsp:/data/local/tmp $
```

6 Sample Code

6.1 Android NN Sample Code

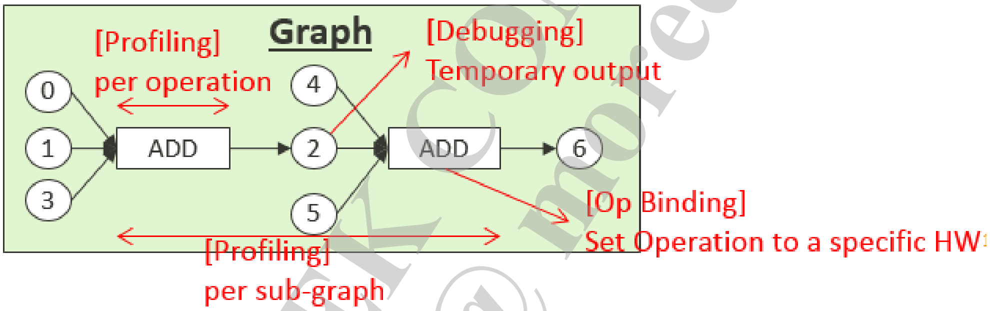
Developers can reference https://developer.android.com/ndk/guides/neuralnetworks/index.html

6.2 NeuroPilot Extension APIs

NeuroPilot provides three categories extension APIs which are important for development while they are not in the current version of Android NN framework. They are

- Profiling
  - Measure timing of per sub-graph
  - Measure timing of per operation
- Debugging
  - Monitor the specific node's output
- Operation Binding
  - Force an operation to execute on specific hardware

Here is a graph to explain the roles they play in the neural network runtime



Currently, the debugging mechanism is default on for developing, and we also provide a method to turn on/off this extension functions at running time.

- Profiling & Debugging
  - On: adb shell setprop nn.profiler.supported 1
  - Off: adb shell setprop nn.profiler.supported 0
- Operation Binding
  - On: adb shell setprop nn.bindoperation.supported 1
  - Off: adb shell setprop nn.bindoperation.supported 0

6.3 Profiler

6.3.1 ANeuralNetworksCompilation_enableProfiler

```
int ANeuralNetworksCompilation_enableProfiler(ANeuralNetworksCompilation *compilation,
                                              uint32_t type);
```

Enable profiler for NN runtime. There is no solution to monitor the execution time per operation or per subgraph in Android NN. Mediatek provides an extension API for such purpose. There are two different level profiling options. One is ANEURALNETWORKS_PROFILER_GRAPH and the other is ANEURALNETWORKS_PROFILER_OPERATION

Mode	Description
ANEURALNETWORKS_PROFILER_GRAPH	Turn on profiler for each sub-graph. In this mode, the profiler will keep the origin Android NN partition work policy.
ANEURALNETWORKS_PROFILER_OPERATION	Turn on profiler for each operation. In this mode, profiler will change partition work behavior. It splits the graph to sub-graphs in operation basis

Choose per operation or per sub-graph. If using per operation, an input graph will be split into sub-graphs, where each sub-graph only contains one operation. This profiling policy maybe impact the performance

6.3.2 ANeuralNetworksExecution_getProfilerInfoCount

```
int ANeuralNetworksExecution_getProfilerInfoCount(ANeuralNetworksExecution *execution,
                                                  uint32_t *count);
```

Get total count of recorded profiling information.

6.3.3 Tutorial

1. Include headers, including standard Android Neural Network and Mediatek proprietary API.

```
#include <android/NeuralNetworks.h>
#include <mtk/NeuralNetworksMTK.h>
```

2. Create Model as Android NN did.

```
ANeuralNetworksModel *model = nullptr;
if (ANeuralNetworksModel_create(&model) != ANEURALNETWORKS_NO_ERROR){
   return 0;
}
…………
if (ANeuralNetworksModel_finish(model) != ANEURALNETWORKS_NO_ERROR){
}
```

3. Create Compilation class

```
ANeuralNetworksCompilation* compilation = nullptr;
if (ANeuralNetworksCompilation_create(model, &compilation) != ANEURALNETWORKS_NO_ERROR){
}
```

4. Enable Profiler

```
if (ANeuralNetworksCompilation_enableProfiler(compilation,
                                              ANEURALNETWORKS_PROFILER_OPERATION)
                    != ANEURALNETWORKS_NO_ERROR){
}
```

5. Finish to compilation

```
if (ANeuralNetworksCompilation_finish(compilation) != ANEURALNETWORKS_NO_ERROR){
}
```

6. Create Execution

```
ANeuralNetworksExecution* execution = nullptr;
if (ANeuralNetworksExecution_create(compilation, &execution) !=
                                         ANEURALNETWORKS_NO_ERROR){
}
```

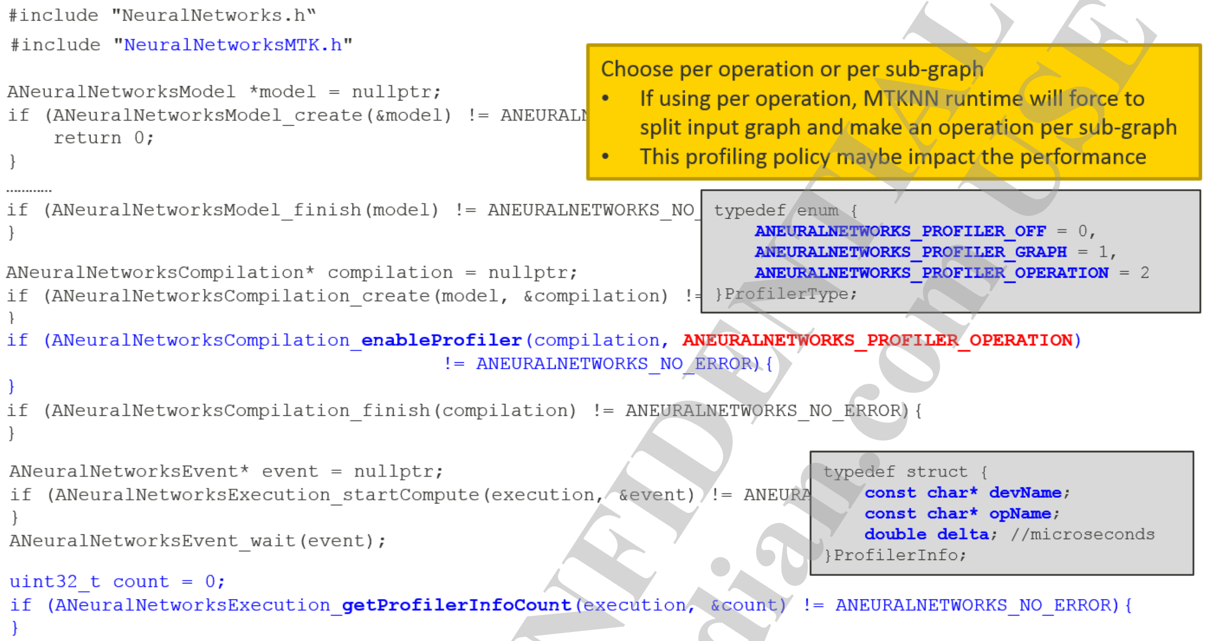
7. Start to compute

```
ANeuralNetworksEvent* event = nullptr;
if (ANeuralNetworksExecution_startCompute(execution, &event) !=
              ANEURALNETWORKS_NO_ERROR){
}
ANeuralNetworksEvent_wait(event);
```

8. Get profiling information

```
ANeuralNetworksEvent_wait(event);
uint32_t count = 0;
if (ANeuralNetworksExecution_getProfilerInfoCount(execution, &count) !=
              ANEURALNETWORKS_NO_ERROR){
}
for (uint32_t i=0 ; i<count ; i++) {
  ProfilerInfo info;
  if (ANeuralNetworksExecution_getProfilerInfo(execution, i, &info) !=
              ANEURALNETWORKS_NO_ERROR){
  }
  printf("\tdeviceName : %s, opName : %s, delta : %.0f us\n",
           info.devName, info.opName, info.delta);
}
```

6.3.4 Summary



```
#include "NeuralNetworks.h"
#include "NeuralNetworksMTK.h"

ANeuralNetworksModel *model = nullptr;
if (ANeuralNetworksModel_create(&model) != ANEURALN
    return 0;
}
…………
if (ANeuralNetworksModel_finish(model) != ANEURALNETWORKS_NO_
}

ANeuralNetworksCompilation* compilation = nullptr;
if (ANeuralNetworksCompilation_create(model, &compilation) !=
}
if (ANeuralNetworksCompilation_enableProfiler(compilation, ANEURALNETWORKS_PROFILER_OPERATION)
                                            != ANEURALNETWORKS_NO_ERROR){
}
if (ANeuralNetworksCompilation_finish(compilation) != ANEURALNETWORKS_NO_ERROR){
}

ANeuralNetworksEvent* event = nullptr;
if (ANeuralNetworksExecution_startCompute(execution, &event) != ANEURA
}
ANeuralNetworksEvent_wait(event);

uint32_t count = 0;
if (ANeuralNetworksExecution_getProfilerInfoCount(execution, &count) != ANEURALNETWORKS_NO_ERROR){
}
```

Choose per operation or per sub-graph

- If using per operation, MTKNN runtime will force to split input graph and make an operation per sub-graph
- This profiling policy maybe impact the performance

```
typedef enum {
    ANEURALNETWORKS_PROFILER_OFF = 0,
    ANEURALNETWORKS_PROFILER_GRAPH = 1,
    ANEURALNETWORKS_PROFILER_OPERATION = 2
}ProfilerType;
```

```
typedef struct {
    const char* devName;
    const char* opName;
    double delta; //microseconds
}ProfilerInfo;
```

6.4 Debugger

6.4.1 ANeuralNetworksExecution_keepOperations

```
int ANeuralNetworksExecution_keepOperations(ANeuralNetworksExecution *execution,
                                            uint32_t* list,
                                            uint32_t listSize);
```

Developers can choose which operation will be kept in memory and dump the output of kept operation after ANeuralNetworksExecution_startComput

6.4.2 ANeuralNetworksExecution_getOperationOutput

```
int ANeuralNetworksExecution_getOperationOutput(ANeuralNetworksExecution *execution,
                                                uint32_t keepingIndex,
                                                uint32_t outputIndex,
                                                uint8_t** buffer);
```

Get output data of specific operation

6.4.3 Tutorial

1. Include headers, including standard Android Neural Network and Mediatek proprietary API.

```
#include <android/NeuralNetworks.h>
#include <mtk/NeuralNetworksMTK.h>
```

2. Create Model as Android NN did.

```
ANeuralNetworksModel *model = nullptr;
if (ANeuralNetworksModel_create(&model) != ANEURALNETWORKS_NO_ERROR){
  return 0;
}
…………
if (ANeuralNetworksModel_finish(model) != ANEURALNETWORKS_NO_ERROR){
}
```

3. Create Compilation

```
ANeuralNetworksCompilation* compilation = nullptr;
if (ANeuralNetworksCompilation_create(model, &compilation) != ANEURALNETWORKS_NO_ERROR){
}
…………
if (ANeuralNetworksCompilation_finish(compilation) != ANEURALNETWORKS_NO_ERROR){
}
```

4. Create Execution

```
ANeuralNetworksExecution* execution = nullptr;
if (ANeuralNetworksExecution_create(compilation, &execution) !=
                                    ANEURALNETWORKS_NO_ERROR){
}
```

5. Keep a specific operation’s output

```
uint32_t list[2] = {1, 2};
if (ANeuralNetworksExecution_keepOperations(execution, list, 2) != ANEURALNETWORKS_NO_ERROR){
}
```

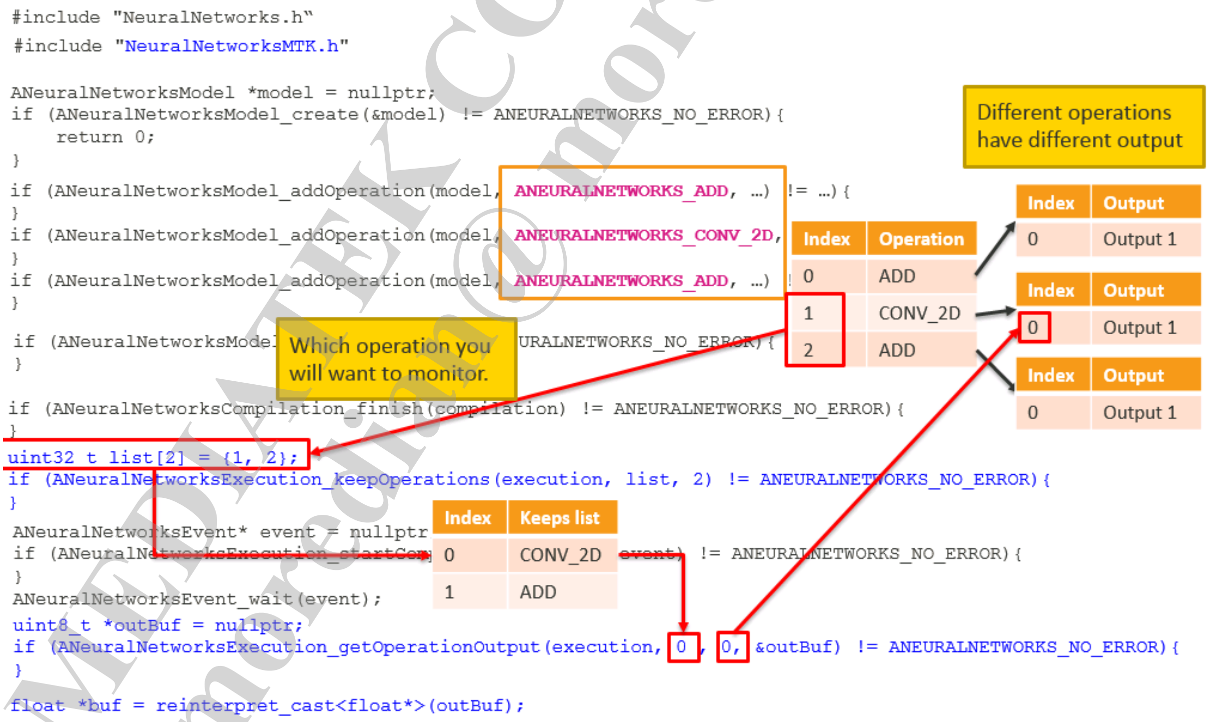
6. Start to compute

```
ANeuralNetworksEvent* event = nullptr;
if (ANeuralNetworksExecution_startCompute(execution, &event) !=
                                                    ANEURALNETWORKS_NO_ERROR){
}
ANeuralNetworksEvent_wait(event);
```

7. Get output buffer’s address of monitoring operation

```
uint8_t *outBuf = nullptr;
if (ANeuralNetworksExecution_getOperationOutput(execution, 0 , 0, &outBuf) !=
                                                    ANEURALNETWORKS_NO_ERROR){
}
float *buf = reinterpret_cast<float*>(outBuf);
```

6.4.4 Summary



```
#include "NeuralNetworks.h"
#include "NeuralNetworksMTK.h"

ANeuralNetworksModel *model = nullptr;
if (ANeuralNetworksModel_create(&model) != ANEURALNETWORKS_NO_ERROR){
    return 0;
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_ADD, …) != …){
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_CONV_2D,
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_ADD, …) !
}

if (ANeuralNetworksModel                    URALNETWORKS_NO_ERROR){
}

if (ANeuralNetworksCompilation_finish(compilation) != ANEURALNETWORKS_NO_ERROR){
}
uint32 t list[2] = {1, 2};
if (ANeuralNetworksExecution_keepOperations(execution, list, 2) != ANEURALNETWORKS_NO_ERROR){
}
ANeuralNetworksEvent* event = nullptr
if (ANeuralNetworksExecution_startCom                event) != ANEURALNETWORKS_NO_ERROR){
}
ANeuralNetworksEvent_wait(event);
uint8_t *outBuf = nullptr;
if (ANeuralNetworksExecution_getOperationOutput(execution, 0 , 0, &outBuf) != ANEURALNETWORKS_NO_ERROR){
}
float *buf = reinterpret_cast<float*>(outBuf);
```

Different operations have different output

Which operation you will want to monitor.

Index	Operation
0	ADD
1	CONV_2D
2	ADD

Index	Output
0	Output 1

Index	Output
0	Output 1

Index	Output
0	Output 1

Index	Keeps list
0	CONV_2D
1	ADD

6.5 Operation Binding

6.5.1 ANeuralNetworks_listRegDevice

```
const char* ANeuralNetworks_listRegDevice(void);
```

List all registered HAL device(s)

6.5.2 ANeuralNetworksModel_setRuntime

```
int ANeuralNetworksModel_setRuntime(ANeuralNetworksModel *model,
                                    uint32_t operation,
                                    const char* device);
```

Developers can specify an operation to run in a specific hardware by using ANeuralNetworksModel_setRuntime, where the candidate hardware list could be retrieved by using ANeuralNetworks_listRegDevice API.

6.5.3 Tutorial

1. Include headers, including standard Android Neural Network and Mediatek proprietary API.

```
#include <android/NeuralNetworks.h>
#include <mtk/NeuralNetworksMTK.h>
```

2. List all registered devices
   Developers can retrieve a string composed of all devices' name, for example, "**cpunn:armnn:vpunn:**". It represents that there are three different HAL devices already registered to Android's device manager, including CPU, GPU, and VPU.

```
const char* devs = ANeuralNetworks_listRegDevice();
```

3. Create Model

```
ANeuralNetworksModel *model = nullptr;
if (ANeuralNetworksModel_create(&model) != ANEURALNETWORKS_NO_ERROR){
  return 0;
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_ADD, ...) != ...){
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_CONV_2D, ...) != ...){
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_ADD, ...) != ...){
}
```

4. **Before finish model**, you can bind an operation to a specific hardware using the extension API, **ANeuralNetworksModel_setRuntime**. The device name is chosen from **ANeuralNetworks_listRegDevice()**

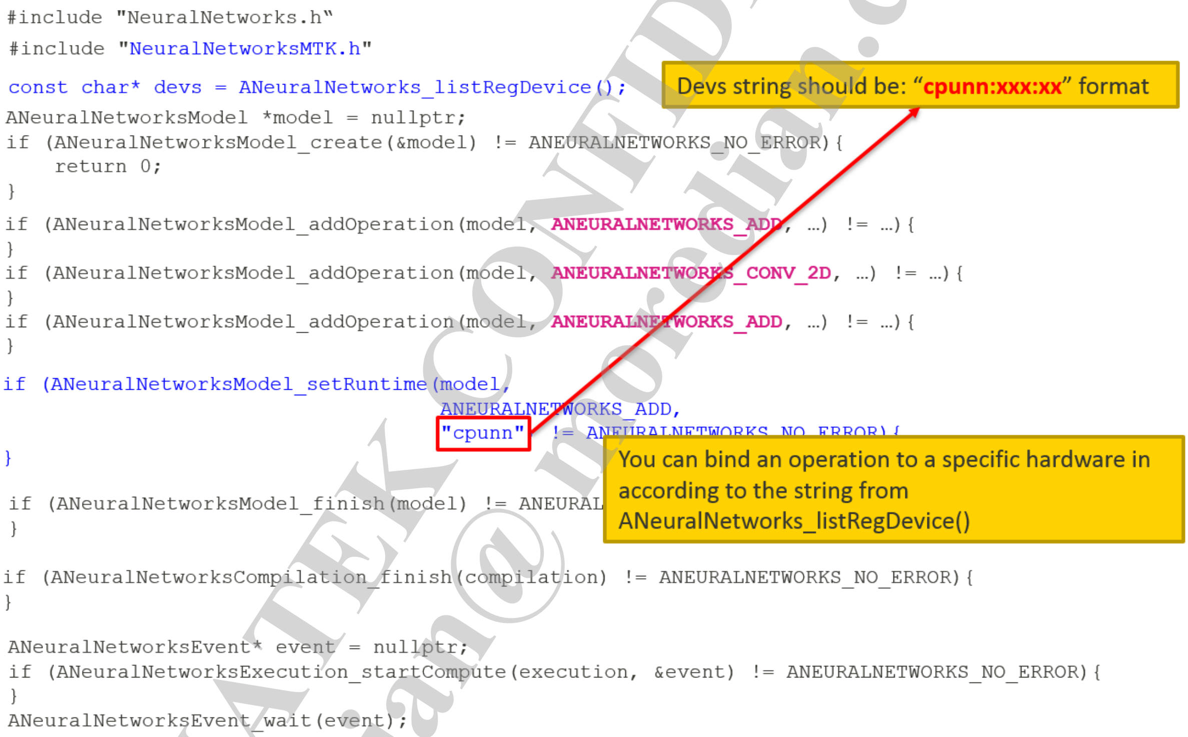
```
if (ANeuralNetworksModel_setRuntime(model,
                                    ANEURALNETWORKS_ADD,
                                    "cpunn") != ANEURALNETWORKS_NO_ERROR){
}
```

5. Finish to create model

```
if (ANeuralNetworksModel_finish(model) != ANEURALNETWORKS_NO_ERROR){
}
```

6. After that, all you can do is following the standard Android NN API to develop.

6.5.4 Summary



```
#include "NeuralNetworks.h"
#include "NeuralNetworksMTK.h"
const char* devs = ANeuralNetworks_listRegDevice();
ANeuralNetworksModel *model = nullptr;
if (ANeuralNetworksModel_create(&model) != ANEURALNETWORKS_NO_ERROR){
    return 0;
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_ADD, …) != …){
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_CONV_2D, …) != …){
}
if (ANeuralNetworksModel_addOperation(model, ANEURALNETWORKS_ADD, …) != …){
}
if (ANeuralNetworksModel_setRuntime(model,
                                    ANEURALNETWORKS_ADD,
                                    "cpunn"  != ANEURALNETWORKS_NO_ERROR){
}
 if (ANeuralNetworksModel_finish(model) != ANEURAL
 }

if (ANeuralNetworksCompilation_finish(compilation) != ANEURALNETWORKS_NO_ERROR){
}

ANeuralNetworksEvent* event = nullptr;
if (ANeuralNetworksExecution_startCompute(execution, &event) != ANEURALNETWORKS_NO_ERROR){
}
ANeuralNetworksEvent_wait(event);
```

Devs string should be: "cpunn:xxx:xx" format

You can bind an operation to a specific hardware in according to the string from ANeuralNetworks_listRegDevice()

6.6 TensorFlow Lite Interpreter

6.6.1 ANeuralNetworksTFLite_create

```
int ANeuralNetworksTFLite_create(ANeuralNetworksTFLite** tflite, const char* modelPath);
```

Developer can get a TensorFlow Lite (TFLite) model from TensorFlow model by using TFLite Toco convertor. NeuroPilot provides an extension API to interpret TFLite model directly on the fly by using ANeuralNetworksTFLite_create. This API will create an ANeuralNetworksTFLite instance and need a specific TFLite model path.

6.6.2 ANeuralNetworksTFLite_getTensor

```
int ANeuralNetworksTFLite_getTensor(ANeuralNetworksTFLite* tflite,
                                  TFLiteBufferType btype,
                                  TFLiteTensor *tfliteTensor);
```

Before inferencing, developers SHOULD set the input data of your own model and after inferencing, developers can retrieve an output. You can get the buffer address of input and output by using ANeuralNetworksTFLite_getTensor.

Developers can choose which kind of buffer you want to get in accordance with TfliteBufferType.

```
typedef enum {
  TFLITE_BUFFER_TYPE_INPUT = 0,
  TFLITE_BUFFER_TYPE_OUTPUT = 1
} TFLiteBufferType;
```

Here is an explanation of TFLiteTensor structure.

Field	Description

type	typedef enum {
	  TFLITE_TENSOR_TYPE_NONE = 0,
	  TFLITE_TENSOR_TYPE_FLOAT = 1,
	  TFLITE_TENSOR_TYPE_INT32 = 2
	} TFLiteTensorType;

dimsSize	The dimension of tensor.
dims[];	The value of each dimension. The supported max dimension currently is 4.
buffer	Buffer address of input or output

bufferSize	Buffer size.
	E.g.
	TFLiteTensor tensor = {
	  .type = TFLITE_TENSOR_TYPE_FLOAT,
	  .dimsSize = 4,
	  .dims = {1,3,3,1},
	  .buffer = 0xAABBCCDD,
	  .bufferSize = (1 x 3 x 3 x 1)
	};

Field	Description
	The actual buffer size is 36 bytes (1 x 3 x 3 x 1 x sizeof(float)) but the value of bufferSize is 9.

6.6.3 ANeuralNetworksTFLite_invoke

```
int ANeuralNetworksTFLite_invoke(ANeuralNetworksTFLite* tflite);
```

Start to inference by using ANeuralNetworksTFLite_invoke. Once finishing to invoke, you can get the output data by using ANeuralNetworksTFLite_getTensor API.

6.6.4 ANeuralNetworksTFLite_free

```
void ANeuralNetworksTFLite_free(ANeuralNetworksTFLite* tflite);
```

Free ANeuralNetworksTFLite instance.

6.6.5 Tutorial

1. Include headers, including standard Android Neural Network and Mediatek proprietary API

```
#include <android/NeuralNetworks.h>
#include <mtk/NeuralNetworksMTK.h>
```

2. Specific the paths of TFLite model and input

```
  const char* model_path = "mobilenet.tflite";
  const char* input_path = "mobilenet_input.bin";
```

3. Create ANerualNetworksTFLite instance

```
ANeuralNetworksTFLite* tflite = nullptr;
if (ANeuralNetworksTFLite_create(&tflite, model_path) !=
                        ANEURALNETWORKS_NO_ERROR){
}
```

4. Get input buffer and fill it.

```
TFLiteTensor inputTensor;
if (ANeuralNetworksTFLite_getTensor(tflite,
                  TFLITE_BUFFER_TYPE_INPUT,
                  &inputTensor) != ANEURALNETWORKS_NO_ERROR){
  ANeuralNetworksTFLite_free(tflite);
}
```

5. Start to invoke

```
if (ANeuralNetworksTFLite_invoke(tflite) != ANEURALNETWORKS_NO_ERROR){
  ANeuralNetworksTFLite_free(tflite);
}
```

6. Get output buffer

```
TFLiteTensor outputTensor;
if (ANeuralNetworksTFLite_getTensor(tflite,
                          TFLITE_BUFFER_TYPE_OUTPUT,
                          &outputTensor) != ANEURALNETWORKS_NO_ERROR){
    ANeuralNetworksTFLite_free(tflite);
}
```

7. Free the instance of ANeuralNetworksTFLite

```
ANeuralNetworksTFLite_free(tflite);
```

6.7 Force CPU to Execute All Operations

6.7.1 ANeuralNetworks_setCpuOnly

```
void ANeuralNetworks_setCpuOnly(bool onlyCPU);
```

Developers can force CPU to execute all operations by using this API with onlyCPU setting to true. If onlyCPU is false, NeuroPilot NN runtime will dispatch the operations to each NN HAL device in accordance with its capabilities.

7 GPU NN Operation Specification

Operation	GPU		Has Limitation
	FP32	Quant8	
ADD	V		V
AVERAGE_POOL_2D	V		V
CONCATENATION	V		V
CONV_2D	V		
DEPTH_TO_SPACE	V		
DEPTHWISE_CONV_2D	V		V
DEQUANTIZE			
EMBEDDING_LOOKUP	V		
FLOOR	V		
FULLY_CONNECTED	V		V
HASHTABLE_LOOKUP	V		
L2_NORMALIZATION	V		
L2_POOL_2D	V		V
LOCAL_RESPONSE_NORMALIZATION	V		
LOGISTIC	V		
LSH_PROJECTION	V		
LSTM			
MAX_POOL_2D	V		V
MUL	V		V
RELU1	V		
RELU6	V		
RELU	V		
RESHAPE	V		V
RESIZE_BILINEAR	V		
RNN			
SOFTMAX	V		
SPACE_TO_DEPTH	V		
SVDF			
TANH	V		

Item	Limitation
*_POOL_2D	pool width = 4, 5, 6 currently not supported pool width != height (non-square) currently not supported
DEPTHWISE_CONV_2D	depth_multiplier > 1 currently not supported
RESHAPE	Support non-4D or flat 4D tensors (all dim but one being 1).
BORADCAST is not supported	Currently two input tensors have to be in same dimensions
Padding limitation	Asymmetric padding currently not supported.
Parameter limitation	Trained parameters could not be identified as input or output of model. E.g. weights and bias in convolution 2D.
CONCATENATION limitation	Concatenation axis value only support for 3 (channel), other values (0, 1, 2) may have unpredictable results.

8 VPU NN Operation Specification

Operation	VPU		Has Limitation
	FP32	Quant8	
ADD		V	
AVERAGE_POOL_2D		V	V
CONCATENATION		V	V
CONV_2D		V	V
DEPTH_TO_SPACE		V	
DEPTHWISE_CONV_2D		V	V
DEQUANTIZE			
EMBEDDING_LOOKUP			
FLOOR			
FULLY_CONNECTED		V	V
HASHTABLE_LOOKUP			
L2_NORMALIZATION			
L2_POOL_2D			
LOCAL_RESPONSE_NORMALIZATION			
LOGISTIC		V	
LSH_PROJECTION			
LSTM			
MAX_POOL_2D		V	V
MUL		V	
RELU1		V	
RELU6		V	
RELU		V	
RESHAPE		V	
RESIZE_BILINEAR		V	V
RNN			
SOFTMAX		V	
SPACE_TO_DEPTH		V	
SVDF			
TANH			

Item	Limitation
AVG_POOL_2D	ksize : [1-16] stride : [1, 2]
MAX_POOL_2D	ksize : [1-16] stride : [1, 2]
CONV_2D	ksize : [1-16] stride : [1, 2, 4]
DEPTHWISE_CONV_2D	ksize : [1-16] stride : [1, 2, 4] Cannot use weight as input
FULLY_CONNECTED	Kernel width / height: [1-255]
CONCATENATION	Max 6 input
RESIZE_BILINEAR	Zero point & scale of input/output should be the same

9 Custom Operation

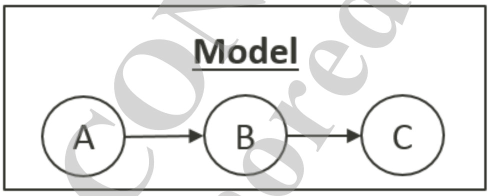
9.1 Customized Operation Solution

Solution	Pros	Cons
Add custom operation by registering a new HAL device	NN runtime can select the better HAL device to run if this OP is supported by multiple devices	1. New HAL implementation efforts 2. Tightly coupled with Neural Networks framework
Add Custom OP by splitting model	Easier porting to different NN framework	More memory copy (i.e. more BW) may hurt performance

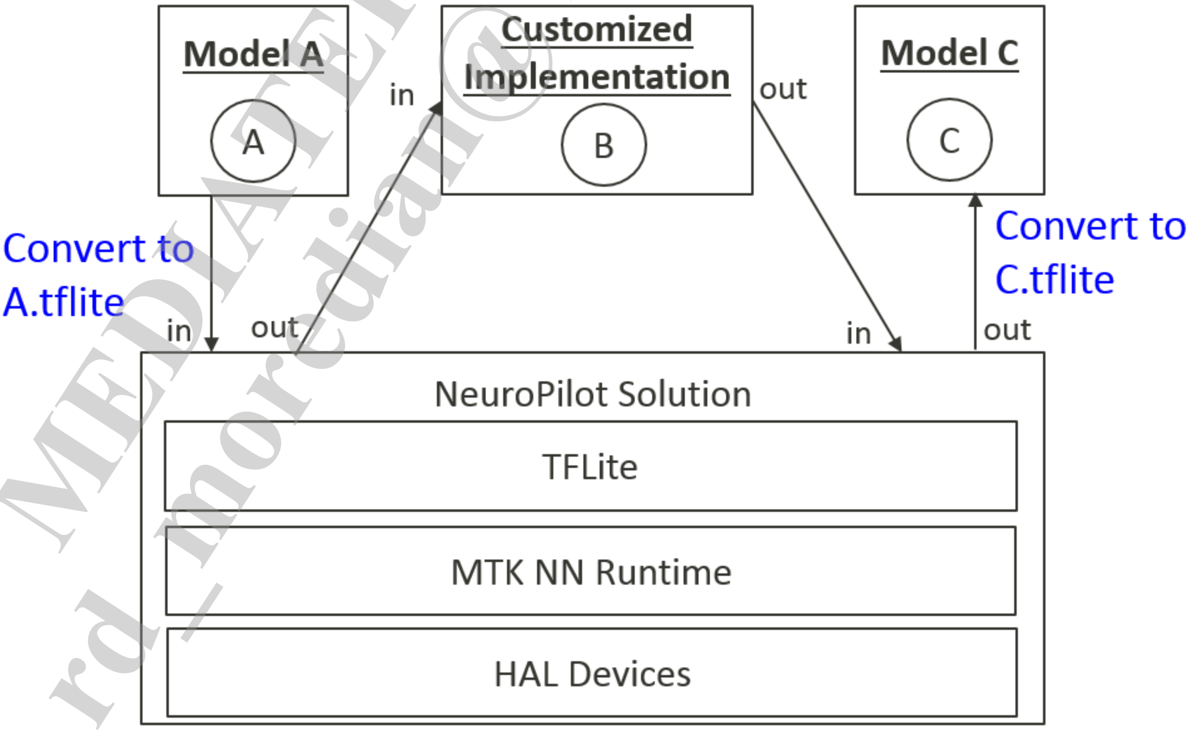
9.2 Add Custom OP by splitting model

Assume that there is a model

- A and C are supported by NeuroPilot Runtime
- B is a specific operation by customers



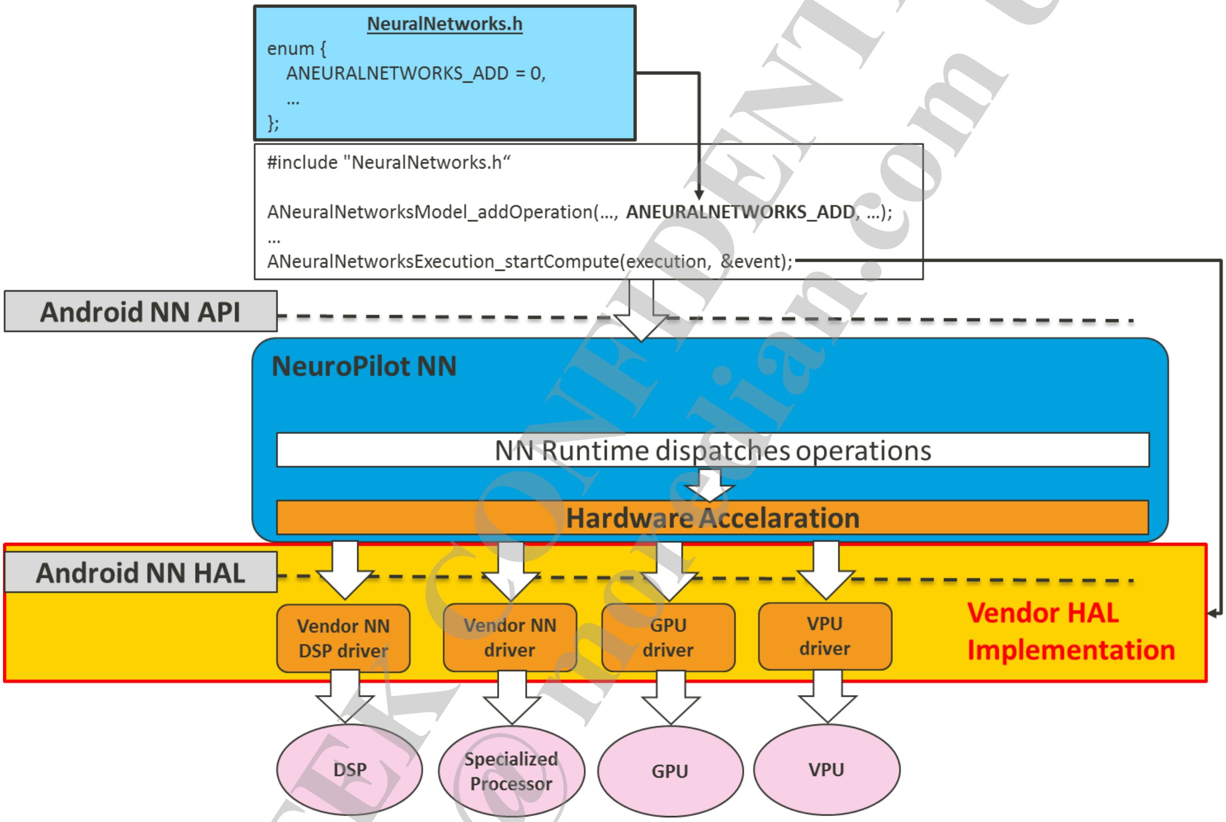
Developers can split model into Model-A, Model-B, and Model-C. Model-A and Model-C still can be converted by using TFLite Toco convertor.



9.3 Add Custom Op by registering a new HAL device

9.3.1 Application Flow with Android NN Operations

Developers use Android NN API to deploy their own application and NN runtime dispatches operations to different HAL devices according to per device's capabilities. Here is a simple flow of application with Android NN.



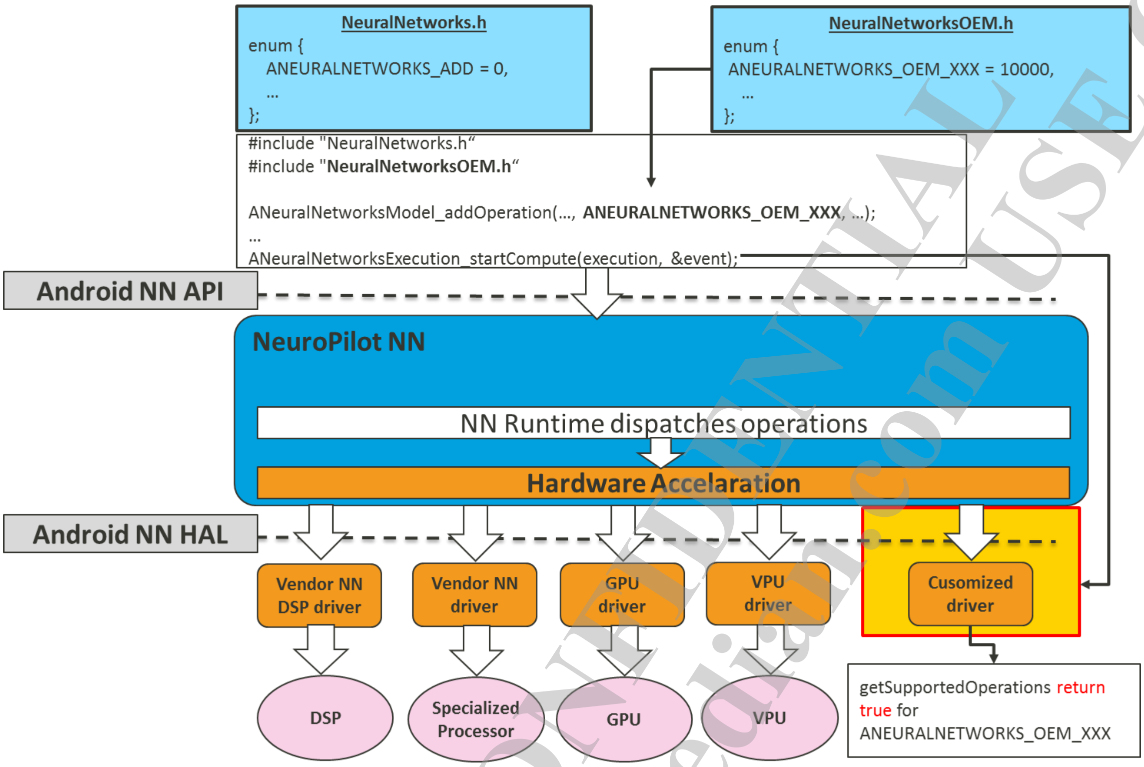
NeuralNetworks.h

```
enum {
   ANEURALNETWORKS_ADD = 0,
   ...
};
```

```
#include "NeuralNetworks.h“

ANeuralNetworksModel_addOperation(..., ANEURALNETWORKS_ADD, ...);
...
ANeuralNetworksExecution_startCompute(execution, &event);
```

9.3.2 Application Flow with Customized Operations

All operations supported by Android NN are mapped to an unique enumeration. The first step to add a customized operation is to add a new enumeration for it. At runtime, the NueroPilot NN will query which device supports the new enumeration. If a supported device is found, NeuroPilot will dispatch this new operation to the specific HAL device.



```
NeuralNetworks.h
enum {
   ANEURALNETWORKS_ADD = 0,
   …
};
```

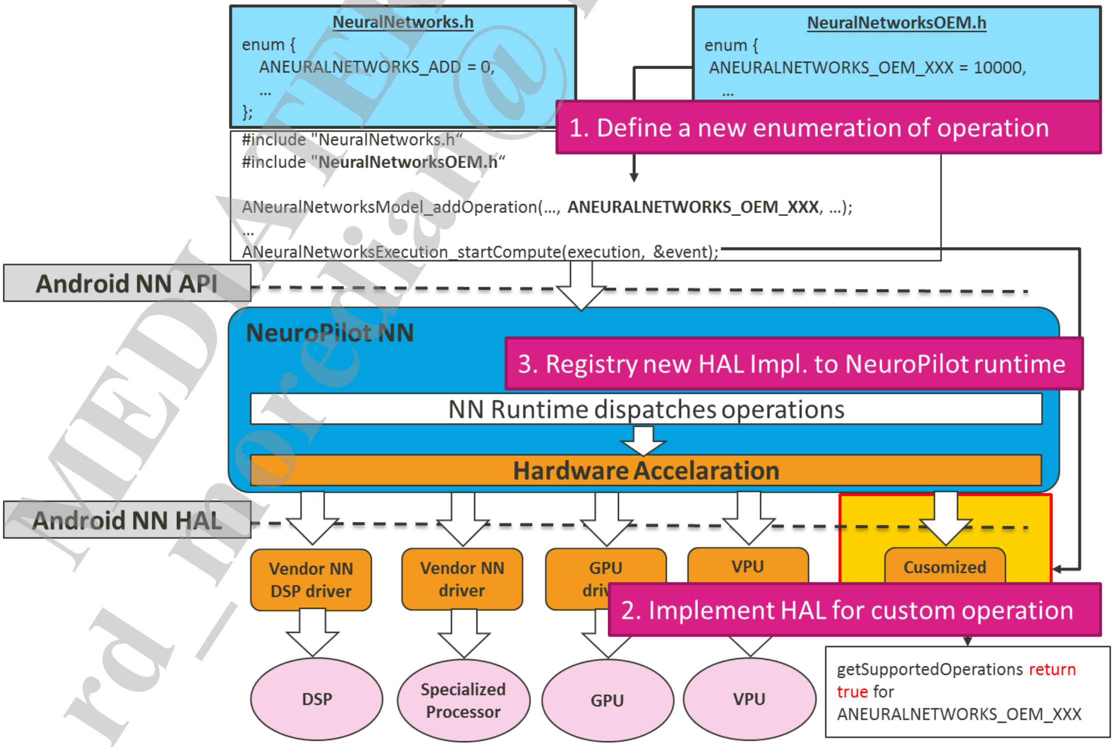
```
NeuralNetworksOEM.h
enum {
 ANEURALNETWORKS_OEM_XXX = 10000,
   …
};
```

```
#include "NeuralNetworks.h“
#include "NeuralNetworksOEM.h“

ANeuralNetworksModel_addOperation(…, ANEURALNETWORKS_OEM_XXX, …);
…
ANeuralNetworksExecution_startCompute(execution, &event);
```

getSupportedOperations return true for ANEURALNETWORKS_OEM_XXX

9.3.3 Steps for Adding a Custom Operation

1. Define a new enumeration of operation
2. Implement HAL for custom operation
3. Registry new HAL Implement to Device Manager



```
NeuralNetworks.h
enum {
   ANEURALNETWORKS_ADD = 0,
   …
};
```

```
NeuralNetworksOEM.h
enum {
 ANEURALNETWORKS_OEM_XXX = 10000,
   …
```

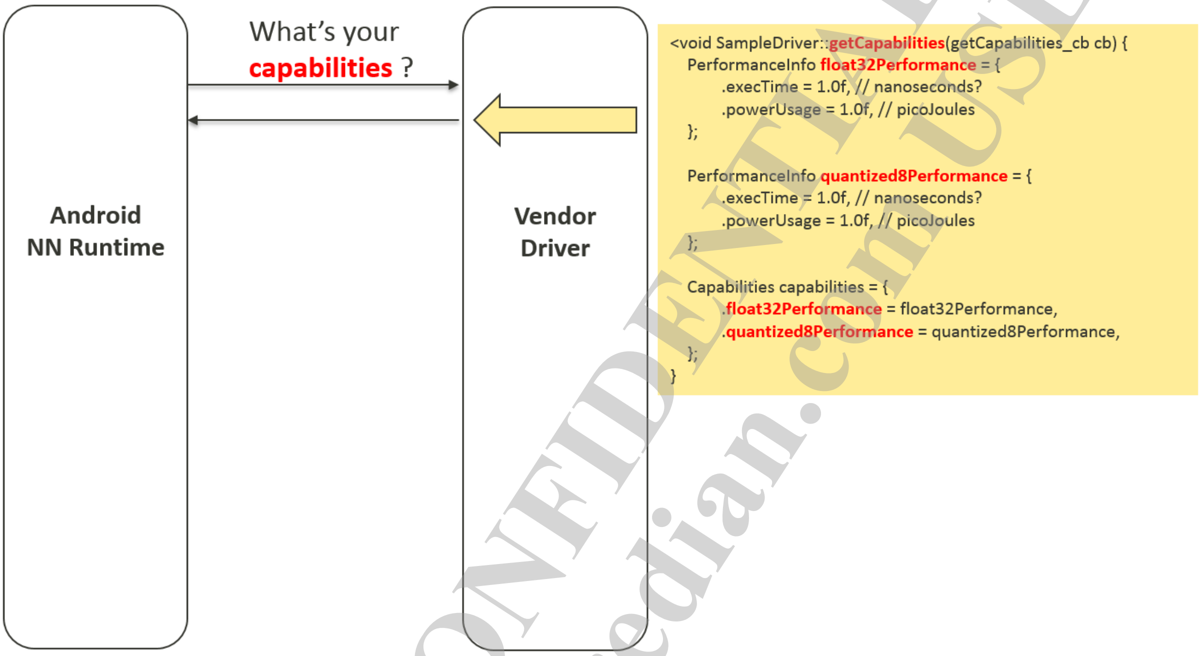
```
#include "NeuralNetworks.h“
#include "NeuralNetworksOEM.h“

ANeuralNetworksModel_addOperation(…, ANEURALNETWORKS_OEM_XXX, …);
…
ANeuralNetworksExecution_startCompute(execution, &event);
```

1. Define a new enumeration of operation
Android NN API
NeuroPilot NN
3. Registry new HAL Impl. to NeuroPilot runtime
NN Runtime dispatches operations
Hardware Accelaration
Android NN HAL
Vendor NN DSP driver
Vendor NN driver
GPU driv
VPU
Cusomized
2. Implement HAL for custom operation
DSP
Specialized Processor
GPU
VPU

getSupportedOperations return true for ANEURALNETWORKS_OEM_XXX

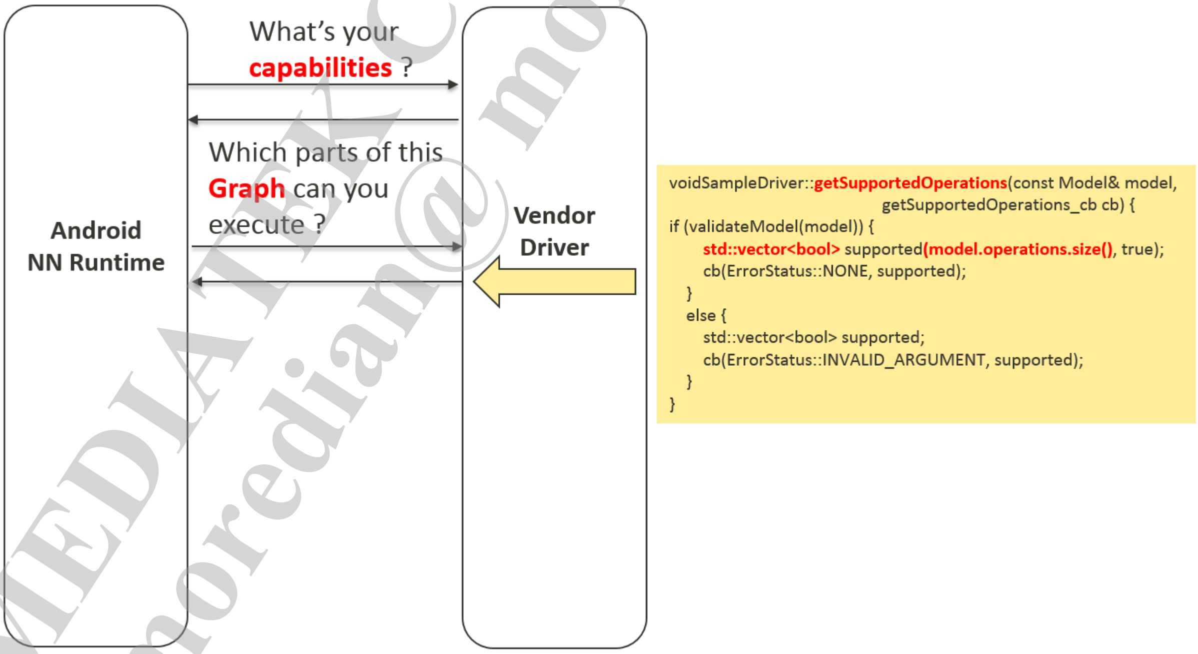
9.3.4 The mechanism of Android NN HAL

1. Query capabilities of HAL device



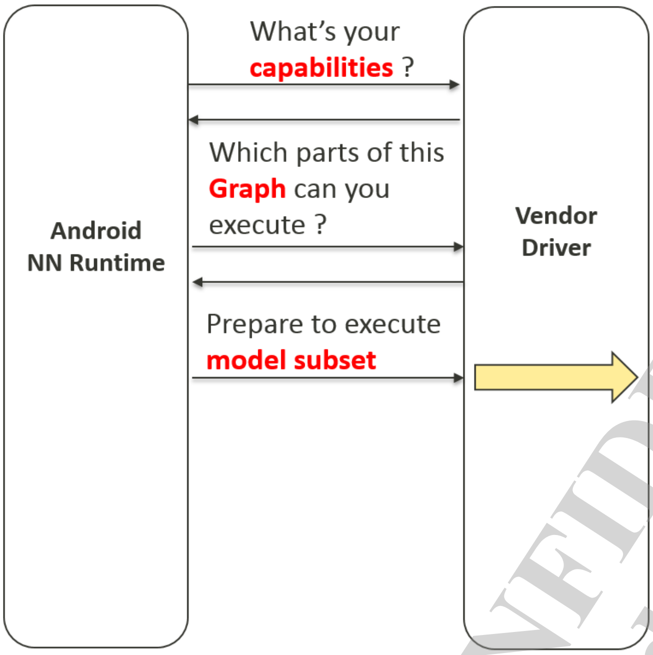
```
<void SampleDriver::getCapabilities(getCapabilities_cb cb) {
   PerformanceInfo float32Performance = {
         .execTime = 1.0f, // nanoseconds?
         .powerUsage = 1.0f, // picoJoules
   };

   PerformanceInfo quantized8Performance = {
         .execTime = 1.0f, // nanoseconds?
         .powerUsage = 1.0f, // picoJoules
   };

   Capabilities capabilities = {
         .float32Performance = float32Performance,
         .quantized8Performance = quantized8Performance,
   };
}
```

2. Query device for supporting operations



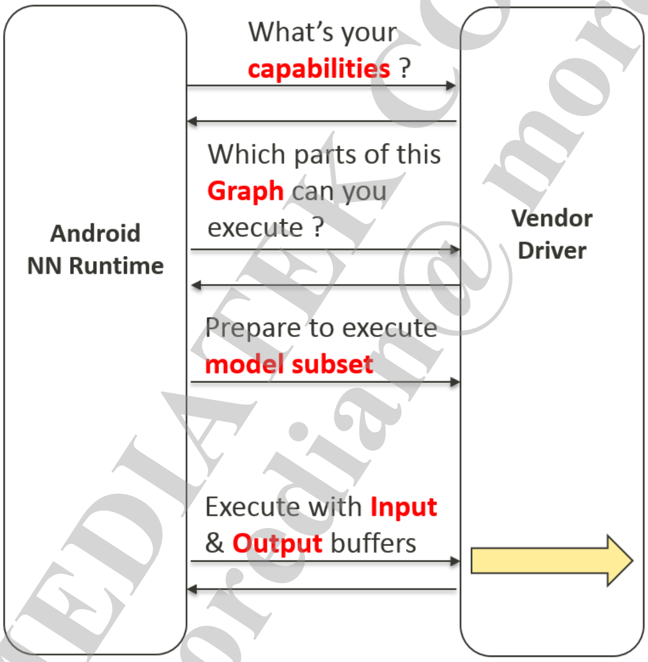
```
voidSampleDriver::getSupportedOperations(const Model& model,
                                        getSupportedOperations_cb cb) {
if (validateModel(model)) {
      std::vector<bool> supported(model.operations.size(), true);
      cb(ErrorStatus::NONE, supported);
   }
   else {
      std::vector<bool> supported;
      cb(ErrorStatus::INVALID_ARGUMENT, supported);
   }
}
```

3. Prepare execution model


Android
NN Runtime
What's your capabilities ?
Which parts of this Graph can you execute ?
Prepare to execute model subset
Vendor
Driver

```
void SampleDriver::prepareModel(const Model& model, const
sp<IEvent>& event,
                     prepareModel_cb cb) {
if (validateModel(model)) {
      // TODO: make asynchronous later
      cb(ErrorStatus::NONE, new SamplePreparedModel(model));
   }
   else {
      cb(ErrorStatus::INVALID_ARGUMENT, nullptr);
   }

   // TODO: notify errors if they occur
   event->notify(ErrorStatus::NONE);
}
```

4. Execution


Android
NN Runtime
What's your capabilities ?
Which parts of this Graph can you execute ?
Prepare to execute model subset
Execute with Input & Output buffers
Vendor
Driver

HAL Implementation Sample code
frameworks/ml/nn/driver/sample

```
int CpuExecutor::executeOperation(const Operation& operation) {
   switch (operation.opTuple.operationType) {
      case OperationType::ADD: {
      }break;
      case OperationType::MUL: {
      }break;
      ..
   }
}
```

10 Android NN CTS report for CPU

Here is the testing result:

```
[==========] Running 150 tests from 7 test cases.
[----------] Global test environment set-up.
[----------] 2 tests from MemoryTest
[ RUN      ] MemoryTest.TestASharedMemory
[       OK ] MemoryTest.TestASharedMemory (10 ms)
[ RUN      ] MemoryTest.TestFd
[       OK ] MemoryTest.TestFd (6 ms)
[----------] 2 tests from MemoryTest (18 ms total)

[----------] 4 tests from TrivialTest
[ RUN      ] TrivialTest.AddTwo
[       OK ] TrivialTest.AddTwo (3 ms)
[ RUN      ] TrivialTest.AddThree
[       OK ] TrivialTest.AddThree (6 ms)
[ RUN      ] TrivialTest.BroadcastAddTwo
[       OK ] TrivialTest.BroadcastAddTwo (3 ms)
[ RUN      ] TrivialTest.BroadcastMulTwo
[       OK ] TrivialTest.BroadcastMulTwo (3 ms)
[----------] 4 tests from TrivialTest (15 ms total)

[----------] 1 test from ValidationTest
[ RUN      ] ValidationTest.CreateModel
[       OK ] ValidationTest.CreateModel (1 ms)
[----------] 1 test from ValidationTest (1 ms total)

[----------] 6 tests from ValidationTestModel
[ RUN      ] ValidationTestModel.AddOperand
[       OK ] ValidationTestModel.AddOperand (0 ms)
[ RUN      ] ValidationTestModel.SetOperandValue
[       OK ] ValidationTestModel.SetOperandValue (0 ms)
[ RUN      ] ValidationTestModel.AddOperation
[       OK ] ValidationTestModel.AddOperation (1 ms)
[ RUN      ] ValidationTestModel.SetInputsAndOutputs
[       OK ] ValidationTestModel.SetInputsAndOutputs (0 ms)
[ RUN      ] ValidationTestModel.Finish
[       OK ] ValidationTestModel.Finish (0 ms)
[ RUN      ] ValidationTestModel.CreateCompilation
[       OK ] ValidationTestModel.CreateCompilation (1 ms)
[----------] 6 tests from ValidationTestModel (2 ms total)

[----------] 3 tests from ValidationTestCompilation
[ RUN      ] ValidationTestCompilation.SetPreference
[       OK ] ValidationTestCompilation.SetPreference (0 ms)
[ RUN      ] ValidationTestCompilation.CreateExecution
[       OK ] ValidationTestCompilation.CreateExecution (0 ms)
[ RUN      ] ValidationTestCompilation.Finish
[       OK ] ValidationTestCompilation.Finish (0 ms)
[----------] 3 tests from ValidationTestCompilation (0 ms total)

[----------] 132 tests from GeneratedTests
[ RUN      ] GeneratedTests.add_broadcast_quant8
[       OK ] GeneratedTests.add_broadcast_quant8 (5 ms)
[ RUN      ] GeneratedTests.add
[       OK ] GeneratedTests.add (3 ms)
[ RUN      ] GeneratedTests.add_quant8
[       OK ] GeneratedTests.add_quant8 (3 ms)
[ RUN      ] GeneratedTests.avg_pool_float_1
[       OK ] GeneratedTests.avg_pool_float_1 (3 ms)
[ RUN      ] GeneratedTests.avg_pool_float_2
[       OK ] GeneratedTests.avg_pool_float_2 (270 ms)
[ RUN      ] GeneratedTests.avg_pool_float_3
[       OK ] GeneratedTests.avg_pool_float_3 (255 ms)
[ RUN      ] GeneratedTests.avg_pool_float_4
[       OK ] GeneratedTests.avg_pool_float_4 (226 ms)
[ RUN      ] GeneratedTests.avg_pool_quant8_1
[       OK ] GeneratedTests.avg_pool_quant8_1 (5 ms)
[ RUN      ] GeneratedTests.avg_pool_quant8_2
[       OK ] GeneratedTests.avg_pool_quant8_2 (67 ms)
[ RUN      ] GeneratedTests.avg_pool_quant8_3
[       OK ] GeneratedTests.avg_pool_quant8_3 (18 ms)
[ RUN      ] GeneratedTests.avg_pool_quant8_4
[       OK ] GeneratedTests.avg_pool_quant8_4 (3 ms)
[ RUN      ] GeneratedTests.concat_float_1
[       OK ] GeneratedTests.concat_float_1 (4 ms)
[ RUN      ] GeneratedTests.concat_float_2
[       OK ] GeneratedTests.concat_float_2 (334 ms)
[ RUN      ] GeneratedTests.concat_float_3
[       OK ] GeneratedTests.concat_float_3 (290 ms)
[ RUN      ] GeneratedTests.concat_quant8_1
[       OK ] GeneratedTests.concat_quant8_1 (3 ms)
[ RUN      ] GeneratedTests.concat_quant8_2
[       OK ] GeneratedTests.concat_quant8_2 (413 ms)
[ RUN      ] GeneratedTests.concat_quant8_3
[       OK ] GeneratedTests.concat_quant8_3 (544 ms)
[ RUN      ] GeneratedTests.conv_float_channels
[       OK ] GeneratedTests.conv_float_channels (4 ms)
[ RUN      ] GeneratedTests.conv_float_channels_weights_as_inputs
[       OK ] GeneratedTests.conv_float_channels_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.conv_float_large
[       OK ] GeneratedTests.conv_float_large (4 ms)
[ RUN      ] GeneratedTests.conv_float_large_weights_as_inputs
[       OK ] GeneratedTests.conv_float_large_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.conv_float
[       OK ] GeneratedTests.conv_float (2 ms)
[ RUN      ] GeneratedTests.conv_float_weights_as_inputs
[       OK ] GeneratedTests.conv_float_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.conv_quant8_channels
[       OK ] GeneratedTests.conv_quant8_channels (3 ms)
[ RUN      ] GeneratedTests.conv_quant8_channels_weights_as_inputs
[       OK ] GeneratedTests.conv_quant8_channels_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.conv_quant8_large
[       OK ] GeneratedTests.conv_quant8_large (4 ms)
[ RUN      ] GeneratedTests.conv_quant8_large_weights_as_inputs
[       OK ] GeneratedTests.conv_quant8_large_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.conv_quant8
[       OK ] GeneratedTests.conv_quant8 (3 ms)
[ RUN      ] GeneratedTests.conv_quant8_overflow
[       OK ] GeneratedTests.conv_quant8_overflow (4 ms)
[ RUN      ] GeneratedTests.conv_quant8_overflow_weights_as_inputs
[       OK ] GeneratedTests.conv_quant8_overflow_weights_as_inputs (3 ms)
[ RUN      ] GeneratedTests.conv_quant8_weights_as_inputs
[       OK ] GeneratedTests.conv_quant8_weights_as_inputs (3 ms)
[ RUN      ] GeneratedTests.depth_to_space_float_1
[       OK ] GeneratedTests.depth_to_space_float_1 (4 ms)
[ RUN      ] GeneratedTests.depth_to_space_float_2
[       OK ] GeneratedTests.depth_to_space_float_2 (3 ms)
[ RUN      ] GeneratedTests.depth_to_space_float_3
[       OK ] GeneratedTests.depth_to_space_float_3 (4 ms)
[ RUN      ] GeneratedTests.depth_to_space_quant8_1
[       OK ] GeneratedTests.depth_to_space_quant8_1 (5 ms)
[ RUN      ] GeneratedTests.depth_to_space_quant8_2
[       OK ] GeneratedTests.depth_to_space_quant8_2 (4 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_float_large_2
[       OK ] GeneratedTests.depthwise_conv2d_float_large_2 (7 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_float_large_2_weights_as_inputs
[       OK ] GeneratedTests.depthwise_conv2d_float_large_2_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_float_large
[       OK ] GeneratedTests.depthwise_conv2d_float_large (5 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_float_large_weights_as_inputs
[       OK ] GeneratedTests.depthwise_conv2d_float_large_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_float
[       OK ] GeneratedTests.depthwise_conv2d_float (5 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_float_weights_as_inputs
[       OK ] GeneratedTests.depthwise_conv2d_float_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_quant8_large
[       OK ] GeneratedTests.depthwise_conv2d_quant8_large (5 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_quant8_large_weights_as_inputs
[       OK ] GeneratedTests.depthwise_conv2d_quant8_large_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_quant8
[       OK ] GeneratedTests.depthwise_conv2d_quant8 (4 ms)
[ RUN      ] GeneratedTests.depthwise_conv2d_quant8_weights_as_inputs
[       OK ] GeneratedTests.depthwise_conv2d_quant8_weights_as_inputs (5 ms)
[ RUN      ] GeneratedTests.dequantize
[       OK ] GeneratedTests.dequantize (3 ms)
[ RUN      ] GeneratedTests.embedding_lookup
[       OK ] GeneratedTests.embedding_lookup (4 ms)
[ RUN      ] GeneratedTests.floor
[       OK ] GeneratedTests.floor (5 ms)
[ RUN      ] GeneratedTests.fully_connected_float_large
[       OK ] GeneratedTests.fully_connected_float_large (4 ms)
[ RUN      ] GeneratedTests.fully_connected_float_large_weights_as_inputs
[       OK ] GeneratedTests.fully_connected_float_large_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.fully_connected_float
[       OK ] GeneratedTests.fully_connected_float (4 ms)
[ RUN      ] GeneratedTests.fully_connected_float_weights_as_inputs
[       OK ] GeneratedTests.fully_connected_float_weights_as_inputs (5 ms)
[ RUN      ] GeneratedTests.fully_connected_quant8_large
[       OK ] GeneratedTests.fully_connected_quant8_large (4 ms)
[ RUN      ] GeneratedTests.fully_connected_quant8_large_weights_as_inputs
[       OK ] GeneratedTests.fully_connected_quant8_large_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.fully_connected_quant8
[       OK ] GeneratedTests.fully_connected_quant8 (4 ms)
[ RUN      ] GeneratedTests.fully_connected_quant8_weights_as_inputs
[       OK ] GeneratedTests.fully_connected_quant8_weights_as_inputs (3 ms)
[ RUN      ] GeneratedTests.hashtable_lookup_float
[       OK ] GeneratedTests.hashtable_lookup_float (3 ms)
[ RUN      ] GeneratedTests.hashtable_lookup_quant8
[       OK ] GeneratedTests.hashtable_lookup_quant8 (3 ms)
[ RUN      ] GeneratedTests.l2_normalization_large
[       OK ] GeneratedTests.l2_normalization_large (3 ms)
[ RUN      ] GeneratedTests.l2_normalization
[       OK ] GeneratedTests.l2_normalization (2 ms)
[ RUN      ] GeneratedTests.l2_pool_float_large
[       OK ] GeneratedTests.l2_pool_float_large (2 ms)
[ RUN      ] GeneratedTests.l2_pool_float
[       OK ] GeneratedTests.l2_pool_float (2 ms)
[ RUN      ] GeneratedTests.local_response_norm_float_1
[       OK ] GeneratedTests.local_response_norm_float_1 (3 ms)
[ RUN      ] GeneratedTests.local_response_norm_float_2
[       OK ] GeneratedTests.local_response_norm_float_2 (2 ms)
[ RUN      ] GeneratedTests.local_response_norm_float_3
[       OK ] GeneratedTests.local_response_norm_float_3 (2 ms)
[ RUN      ] GeneratedTests.local_response_norm_float_4
[       OK ] GeneratedTests.local_response_norm_float_4 (2 ms)
[ RUN      ] GeneratedTests.logistic_float_1
[       OK ] GeneratedTests.logistic_float_1 (3 ms)
[ RUN      ] GeneratedTests.logistic_float_2
[       OK ] GeneratedTests.logistic_float_2 (119 ms)
[ RUN      ] GeneratedTests.logistic_quant8_1
[       OK ] GeneratedTests.logistic_quant8_1 (4 ms)
[ RUN      ] GeneratedTests.logistic_quant8_2
[       OK ] GeneratedTests.logistic_quant8_2 (7 ms)
[ RUN      ] GeneratedTests.lsh_projection_2
[       OK ] GeneratedTests.lsh_projection_2 (4 ms)
[ RUN      ] GeneratedTests.lsh_projection
[       OK ] GeneratedTests.lsh_projection (3 ms)
[ RUN      ] GeneratedTests.lsh_projection_weights_as_inputs
[       OK ] GeneratedTests.lsh_projection_weights_as_inputs (4 ms)
[ RUN      ] GeneratedTests.lstm2
[       OK ] GeneratedTests.lstm2 (6 ms)
[ RUN      ] GeneratedTests.lstm2_state2
[       OK ] GeneratedTests.lstm2_state2 (7 ms)
[ RUN      ] GeneratedTests.lstm2_state
[       OK ] GeneratedTests.lstm2_state (6 ms)
[ RUN      ] GeneratedTests.lstm3
[       OK ] GeneratedTests.lstm3 (11 ms)
[ RUN      ] GeneratedTests.lstm3_state2
[       OK ] GeneratedTests.lstm3_state2 (11 ms)
[ RUN      ] GeneratedTests.lstm3_state3
[       OK ] GeneratedTests.lstm3_state3 (9 ms)
[ RUN      ] GeneratedTests.lstm3_state
[       OK ] GeneratedTests.lstm3_state (12 ms)
[ RUN      ] GeneratedTests.lstm
[       OK ] GeneratedTests.lstm (8 ms)
[ RUN      ] GeneratedTests.lstm_state2
[       OK ] GeneratedTests.lstm_state2 (8 ms)
[ RUN      ] GeneratedTests.lstm_state
[       OK ] GeneratedTests.lstm_state (8 ms)
[ RUN      ] GeneratedTests.max_pool_float_1
[       OK ] GeneratedTests.max_pool_float_1 (4 ms)
[ RUN      ] GeneratedTests.max_pool_float_2
[       OK ] GeneratedTests.max_pool_float_2 (10 ms)
[ RUN      ] GeneratedTests.max_pool_float_3
[       OK ] GeneratedTests.max_pool_float_3 (11 ms)
[ RUN      ] GeneratedTests.max_pool_quant8_1
[       OK ] GeneratedTests.max_pool_quant8_1 (4 ms)
[ RUN      ] GeneratedTests.max_pool_quant8_2
[       OK ] GeneratedTests.max_pool_quant8_2 (9 ms)
[ RUN      ] GeneratedTests.max_pool_quant8_3
[       OK ] GeneratedTests.max_pool_quant8_3 (8 ms)
[ RUN      ] GeneratedTests.mobilenet_quantized
[       OK ] GeneratedTests.mobilenet_quantized (231 ms)
[ RUN      ] GeneratedTests.mul_broadcast_quant8
[       OK ] GeneratedTests.mul_broadcast_quant8 (1 ms)
[ RUN      ] GeneratedTests.mul
[       OK ] GeneratedTests.mul (2 ms)
[ RUN      ] GeneratedTests.mul_quant8
[       OK ] GeneratedTests.mul_quant8 (2 ms)
[ RUN      ] GeneratedTests.mul_relu
[       OK ] GeneratedTests.mul_relu (1 ms)
[ RUN      ] GeneratedTests.relu1_float_1
[       OK ] GeneratedTests.relu1_float_1 (1 ms)
[ RUN      ] GeneratedTests.relu1_float_2
[       OK ] GeneratedTests.relu1_float_2 (45 ms)
[ RUN      ] GeneratedTests.relu1_quant8_1
[       OK ] GeneratedTests.relu1_quant8_1 (3 ms)
[ RUN      ] GeneratedTests.relu1_quant8_2
[       OK ] GeneratedTests.relu1_quant8_2 (248 ms)
[ RUN      ] GeneratedTests.relu6_float_1
[       OK ] GeneratedTests.relu6_float_1 (5 ms)
[ RUN      ] GeneratedTests.relu6_float_2
[       OK ] GeneratedTests.relu6_float_2 (65 ms)
[ RUN      ] GeneratedTests.relu6_quant8_1
[       OK ] GeneratedTests.relu6_quant8_1 (6 ms)
[ RUN      ] GeneratedTests.relu6_quant8_2
[       OK ] GeneratedTests.relu6_quant8_2 (154 ms)
[ RUN      ] GeneratedTests.relu_float_1
[       OK ] GeneratedTests.relu_float_1 (4 ms)
[ RUN      ] GeneratedTests.relu_float_2
[       OK ] GeneratedTests.relu_float_2 (154 ms)
[ RUN      ] GeneratedTests.relu_quant8_1
[       OK ] GeneratedTests.relu_quant8_1 (6 ms)
[ RUN      ] GeneratedTests.relu_quant8_2
[       OK ] GeneratedTests.relu_quant8_2 (118 ms)
[ RUN      ] GeneratedTests.reshape
[       OK ] GeneratedTests.reshape (3 ms)
[ RUN      ] GeneratedTests.reshape_quant8
[       OK ] GeneratedTests.reshape_quant8 (3 ms)
[ RUN      ] GeneratedTests.reshape_quant8_weights_as_inputs
[       OK ] GeneratedTests.reshape_quant8_weights_as_inputs (3 ms)
[ RUN      ] GeneratedTests.reshape_weights_as_inputs
[       OK ] GeneratedTests.reshape_weights_as_inputs (3 ms)
[ RUN      ] GeneratedTests.resize_bilinear
[       OK ] GeneratedTests.resize_bilinear (3 ms)
[ RUN      ] GeneratedTests.rnn
[       OK ] GeneratedTests.rnn (4 ms)
[ RUN      ] GeneratedTests.rnn_state
[       OK ] GeneratedTests.rnn_state (5 ms)
[ RUN      ] GeneratedTests.softmax_float_1
[       OK ] GeneratedTests.softmax_float_1 (3 ms)
[ RUN      ] GeneratedTests.softmax_float_2
[       OK ] GeneratedTests.softmax_float_2 (3 ms)
[ RUN      ] GeneratedTests.softmax_quant8_1
[       OK ] GeneratedTests.softmax_quant8_1 (3 ms)
[ RUN      ] GeneratedTests.softmax_quant8_2
[       OK ] GeneratedTests.softmax_quant8_2 (3 ms)
[ RUN      ] GeneratedTests.space_to_depth_float_1
[       OK ] GeneratedTests.space_to_depth_float_1 (3 ms)
[ RUN      ] GeneratedTests.space_to_depth_float_2
[       OK ] GeneratedTests.space_to_depth_float_2 (3 ms)
[ RUN      ] GeneratedTests.space_to_depth_float_3
[       OK ] GeneratedTests.space_to_depth_float_3 (4 ms)
[ RUN      ] GeneratedTests.space_to_depth_quant8_1
[       OK ] GeneratedTests.space_to_depth_quant8_1 (3 ms)
[ RUN      ] GeneratedTests.space_to_depth_quant8_2
[       OK ] GeneratedTests.space_to_depth_quant8_2 (3 ms)
[ RUN      ] GeneratedTests.svdf
[       OK ] GeneratedTests.svdf (5 ms)
[ RUN      ] GeneratedTests.svdf_state
[       OK ] GeneratedTests.svdf_state (5 ms)
[ RUN      ] GeneratedTests.tanh
[       OK ] GeneratedTests.tanh (3 ms)
[ RUN      ] GeneratedTests.conv_1_h3_w2_SAME
[       OK ] GeneratedTests.conv_1_h3_w2_SAME (8 ms)
[ RUN      ] GeneratedTests.conv_1_h3_w2_VALID
[       OK ] GeneratedTests.conv_1_h3_w2_VALID (7 ms)
[ RUN      ] GeneratedTests.conv_3_h3_w2_SAME
[       OK ] GeneratedTests.conv_3_h3_w2_SAME (12 ms)
[ RUN      ] GeneratedTests.conv_3_h3_w2_VALID
[       OK ] GeneratedTests.conv_3_h3_w2_VALID (13 ms)
[ RUN      ] GeneratedTests.depthwise_conv
[       OK ] GeneratedTests.depthwise_conv (7 ms)
[ RUN      ] GeneratedTests.mobilenet
[       OK ] GeneratedTests.mobilenet (79 ms)
[----------] 132 tests from GeneratedTests (4156 ms total)

[----------] 2 tests from PartitioningTest
[ RUN      ] PartitioningTest.SimpleModel
[       OK ] PartitioningTest.SimpleModel (2 ms)
[ RUN      ] PartitioningTest.Cpu
[       OK ] PartitioningTest.Cpu (5 ms)
[----------] 2 tests from PartitioningTest (8 ms total)

[----------] Global test environment tear-down
[==========] 150 tests from 7 test cases ran. (4201 ms total)
[  PASSED  ] 150 tests.
```