IMXTFLUG

i.MX TensorFlow Lite on Android User's Guide

Rev. automotive-12.1.0_1.1.0 — 15 December 2022 User guide

Document information

Information	Content
Keywords	Android, i.MX, automotive-12.1.0_1.1.0
Abstract	This document provides the i.MX TensorFlow Lite information on the Android platform.

1 Neural Network Runtime Overview

The chapter provides an overview of the NXP eIQ software stack for use with the NXP Neural Network Accelerator IPs (GPU or NPU). The following figure shows the data flow between each element. The key part of this diagram is the Neural Network Runtime (NNRT), which is a middleware bridging various inference frameworks and the NN accelerator driver. The NNRT supplies different backends for Android NN HAL, Arm NN, ONNX, and TensorFlow Lite allowing quick application deployment. The NNRT also empowers an application-oriented framework for use with i.MX8 processors. Application frameworks such as Android NN, TensorFlow Lite, and Arm NN can be speed up by NNRT directly benefiting from its built-in backend plugins. This documents focuses on Android NN framework only. Additional backend can be also implemented to expand support for other frameworks.

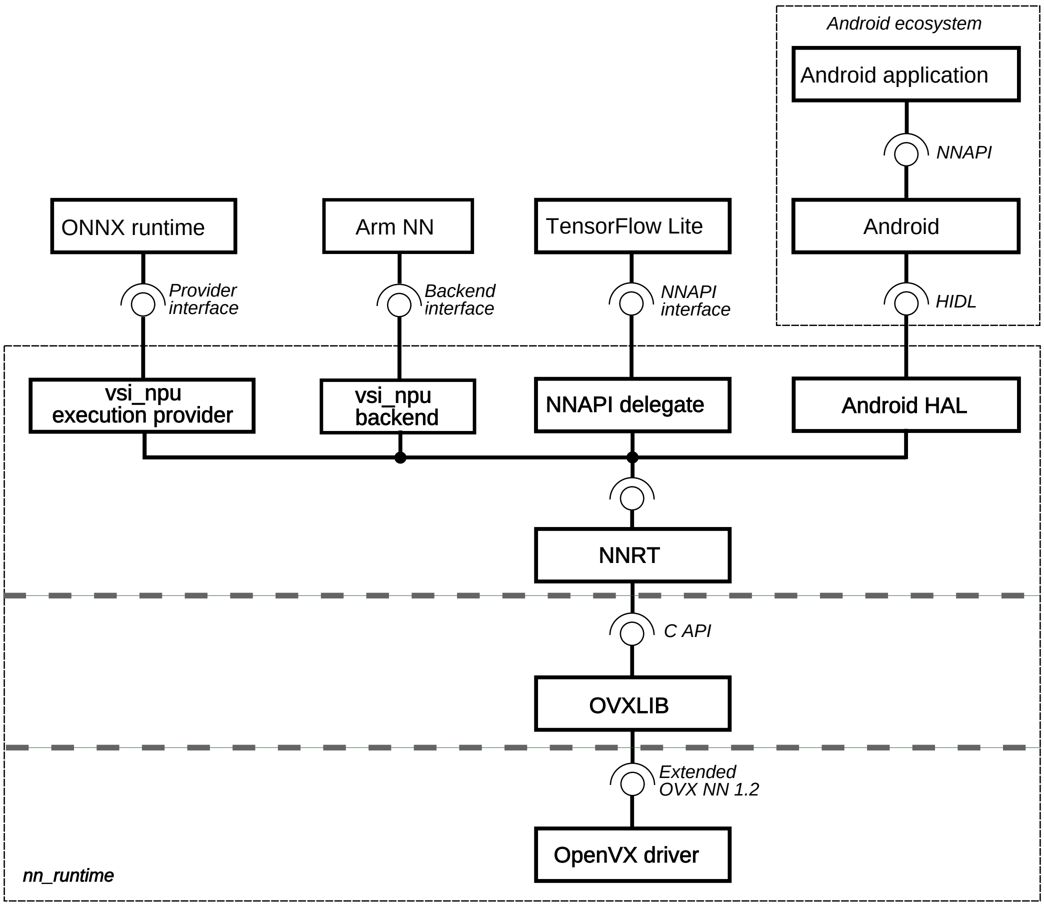

Figure 1. NNRT software architecture

NNRT supports different Machine Learning frameworks by registering itself as a compute backend. Because each framework defines a different backend API, a lightweight backend layer is designed for each. For Android NN, the NNRT follows the Android HIDL definition. It is compatible with v1.3 HAL interface.

In doing so, NNRT unifies application framework differences and provides an universal runtime interface into the driver stack. At the same time, NNRT also acts as the heterogeneous compute platform for further distributing workloads efficiently across i.MX 8 series compute devices, such as NPU, GPU and CPU.

2 TensorFlow Lite

TensorFlow Lite is a light-weight version of and a next step from TensorFlow. TensorFlow Lite is an open-source software library focused on running machine learning models on mobile and embedded devices (available at www.tensorflow.org/lite). It enables on-device machine learning inference with low latency and small binary size. TensorFlow Lite also supports hardware acceleration using Android OS Neural Networks API (NNAPI).

Features:

- TensorFlow Lite v2.4.0
- Multithreaded computation with acceleration using Arm Neon SIMD instructions on Cortex-A cores
- Parallel computation using GPU/NPU hardware acceleration (on shader or convolution units)
- C++ and Python API (supported Python version 3)
- Per-tensor and Per-channel quantized models support

2.1 TensorFlow Lite software stack

The TensorFlow Lite software stack is shown on the below picture. The TensorFlow Lite supports computation on the following HW units:

- CPU Arm Cortex-A core
- GPU/NPU hardware accelerator using the Android NN API driver

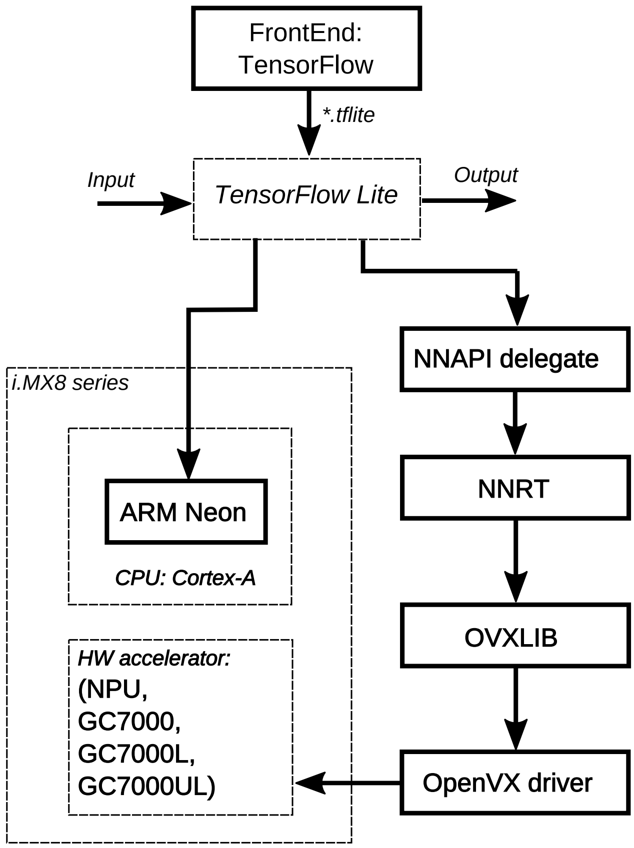

Figure 2. TensorFlow Lite SW stack

Note:

The TensorFlow Lite library uses the Android NN API driver implementation from the GPU/NPU driver for running inference using the GPU/NPU hardware accelerator. The implemented NN API version is 1.3, which has limitations in supported tensor data types and operations, compared to the feature set of TensorFlow Lite. Therefore, some models may work without acceleration enabled, but may fail when using the NN API. For the full list of supported features, see the NN HAL versions section of the NN API documentation: https://source.android.com/devices/neural-networks#hal-versions.

The first execution of model inference using the NN API always takes many times longer, because of model graph initialization needed by the GPU/NPU hardware accelerator. The iterations following the graph initialization are performed many times faster.

The NN API implementation uses the OpenVX library for model graph execution acceleration on the GPU/NPU hardware accelerator. Therefore, OpenVX library support must be available for the selected device to be able to use the acceleration. For more details on the OpenVX library availability, see the i.MX Graphics User's Guide (IMXGRAPHICUG).

The GPU/NPU hardware accelerator driver support both per-tensor and per-channel quantized models. In case of per-channel quantized models, performance degradation varies from slight differences, depending on the model used. This is caused by a hardware limitation, which is designed for per-tensor quantized models.

The TensorFlow Lite Converter V2 uses Quantize and Dequantize nodes to encapsulate some operation during quantization. Typically, the input and output tensor are followed/preceded by these nodes. As they are not fully supported by NN API their execution falls back to CPU. This causes the computational graph segmentation, leading to performance degradation. Because Android Tensorflow-lite benchmark program needs to interact with Vsi_Npu through HAL and RPC call, it introduces additional overhead for inference time while comparing with the Tensorflow-lite benchmark on the Linux OS.

2.2 Running benchmark applications

Benchmark application performs a simple TensorFlow Lite model inference and prints benchmarking information. For details, see https://github.com/tensorflow/tensorflow/tree/v2.4.0/tensorflow/tools/benchmark.

To build, install and run it, follow the steps below:

1. Download the TensorFlow source code from https://github.com/tensorflow/tensorflow/tree/v2.4.0.
2. See https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android to edit the WORKSPACE to configure the Android NDK/SDK.
   - Bazel version: 3.1.0 or later version
   - Android SDK API level: 31 (compatible with Android 12)
   - Android build tools version: 30.0.0 or later version
   - Android NDK API level 21
3. Build for your specific platform, for example,

```
bazel build -c opt --config=android_arm64 tensorflow/lite/
tools/benchmark:benchmark_model
```

4. Use a USB cable to connect the board with the machine. Push the binary to the board with ADB push (make the directory if required):

```
adb push bazel-bin/tensorflow/lite/tools/benchmark/
benchmark_model /data/local/tmp
```

5. Make the binary executable.

```
adb shell chmod +x /data/local/tmp/benchmark_model
```

6. Push the compute graph that you need to test. For example,

```
adb push mobilenet_quant_v1_224.tflite /data/local/tmp
```

7. Run the benchmark. For example,

```
adb shell /data/local/tmp/benchmark_model --graph=/data/
local/tmp/mobilenet_quant_v1_224.tflite --num_threads=4 adb
 shell /data/local/tmp/benchmark_model --graph=/data/local/
tmp/mobilenet_quant_v1_224.tflite --use_nnapi=true
```

Benchmarking instructions:

To run the benchmark with computation on CPU with XNNPACK, use the following command line arguments:

```
adb shell /data/local/tmp/benchmark_model --graph=/data/
local/tmp/mobilenet_v2_1.0_224_quant.tflite --num_threads=4 --
use_xnnpack=true
```

The output of the benchmarking application should be similar to:

```
STARTING!
Min num runs: [50]
Min runs duration (seconds): [1]
Max runs duration (seconds): [150]
Inter-run delay (seconds): [-1]
Num threads: [4]
Benchmark name: []
Output prefix: []
Min warmup runs: [1]
Min warmup runs duration (seconds): [0.5]
Graph: [/data/local/tmp/mobilenet_v2_1.0_224_quant.tflite]
Input layers: []
Input shapes: []
Input value ranges: []
Input layer values files: []
Use legacy nnapi : [0]
Allow fp16 : [0]
Require full delegation : [0]
Enable op profiling: [0]
Max profiling buffer entries: [1024]
CSV File to export profiling data to: []
Max number of delegated partitions : [0]
Use gpu : [0]
Allow lower precision in gpu : [1]
Use Hexagon : [0]
Hexagon lib path : [/data/local/tmp]
Hexagon Profiling : [0]
Use nnapi : [0]
Use xnnpack : [0]
```

```
Loaded model /data/local/tmp/mobilenet_v2_1.0_224_quant.tflite
INFO: Initialized TensorFlow Lite runtime.
The input model file size (MB): 3.57776
Initialized session in 2.975 ms.
Running benchmark for at least 1 iterations and at least 0.5
 seconds but terminate if exceeding 150 seconds.
count=16 first=35481 curr=32384 min=32038 max=35481 avg=32465.5
 std=787
Running benchmark for at least 50 iterations and at least 1
 seconds but terminate if exceeding 150 seconds.
count=50 first=32623 curr=32300 min=32056 max=32895 avg=32283.4
 std=136
Average inference timings in us: Warmup: 32465.5, Init: 2975,
 Inference: 32283.4
Note: as the benchmark tool itself affects memory footprint,
 the following is only APPROXIMATE to the actual memory
 footprint of the model at runtime. Take the information at
 your discretion.
Peak memory footprint (MB): init=2.46484 overall=5.73047
```

To run the inference using the GPU/NPU hardware accelerator, add the `--use_nnapi=true` command line argument:

```
For NPU:
adb shell /data/local/tmp/benchmark_model
--graph=/data/local/tmp/mobilenet_v2_1.0_224_quant.tflite
--use_nnapi=true --nnapi_accelerator_name=vsi-npu
For GPU:
adb root adb shell setprop vendor.USE_GPU_INFERENCE 1 adb
 shell /data/local/tmp/benchmark_model
--graph=/data/local/tmp/mobilenet_v2_1.0_224_quant.tflite --
use_nnapi=true
```

The output with GPU/NPU module acceleration enabled should be similar to:

```
STARTING!
Min num runs: [50]
Min runs duration (seconds): [1]
Max runs duration (seconds): [150]
Inter-run delay (seconds): [-1]
Num threads: [1]
Benchmark name: []
Output prefix: []
Min warmup runs: [1]
Min warmup runs duration (seconds): [0.5]
Graph: [/data/local/tmp/mobilenet_v2_1.0_224_quant.tflite]
Input layers: []
Input shapes: []
Input value ranges: []
Input layer values files: []
Use legacy nnapi : [0]
Allow fp16 : [0]
Require full delegation : [0]
Enable op profiling: [0]
Max profiling buffer entries: [1024]
CSV File to export profiling data to: []
Max number of delegated partitions : [0]
Use gpu : [0]
Allow lower precision in gpu : [1]
```

```
Use Hexagon : [0]
Hexagon lib path : [/data/local/tmp]
Hexagon Profiling : [0]
Use nnapi : [1]
nnapi accelerator name: [] (Available: vsi-npu,nnapi-reference)
Use xnnpack : [0]
Loaded model /data/local/tmp/mobilenet_v2_1.0_224_quant.tflite
INFO: Initialized TensorFlow Lite runtime.
INFO: Created TensorFlow Lite delegate for NNAPI.
Applied NNAPI delegate.
The input model file size (MB): 3.57776
Initialized session in 204.722 ms.
Running benchmark for at least 1 iterations and at least 0.5
 seconds but terminate if exceeding 150 seconds.
count=1 curr=9329095
Running benchmark for at least 50 iterations and at least 1
 seconds but terminate if exceeding 150 seconds.
count=126 first=6650 curr=8304 min=6558 max=17570 avg=7878.48
 std=1078
Average inference timings in us: Warmup: 9.3291e+06, Init:
 204722, Inference: 7878.48
Note: as the benchmark tool itself affects memory footprint,
 the following is only APPROXIMATE to the actual memory
 footprint of the model at runtime. Take the information at
 your discretion.
Peak memory footprint (MB): init=4.58203 overall=5.01172
```

The benchmark application is also useful to check the optional segmentation of the models if accelerated on GPU/NPU hardware accelerator. For this purpose, the `---enable_op_profiling=true` option can be used.

```
adb shell /data/local/tmp/benchmark_model --graph=/data/local/
tmp/mobilenet_v2_1.0_224_quant.tflite --use_nnapi=true --
enable_op_profiling=true
```

In addition to output presented above, this detailed profiling info is available:

```
Profiling Info for Benchmark Initialization:
============================== Run Order ==============================
[node type]            [start]  [first] [avg  ms]     [%]    [cdf%]   [mem KB]  [times called]
 [Name]
ModifyGraphWithDelegate  0.000   194.276  194.276  62.600%   62.600%    2972.000  1
 ModifyGraphWithDelegate/0
AllocateTensors        136.266  116.058   58.034  37.400%   100.000%    0.000    2
 AllocateTensors/0
============================== Top by Computation Time ==============================
[node type]            [start]  [first]  [avg  ms]    [%]    [cdf%]   [mem KB] [times called]
 [Name]
ModifyGraphWithDelegate  0.000   194.276  194.276  62.600%   62.600%   2972.000   1
 ModifyGraphWithDelegate/0
AllocateTensors        136.266  116.058  58.034   37.400%   100.000%   0.000     2
 AllocateTensors/0
Number of nodes executed: 2
============================== Summary by node type ==============================
[Node type]           [count]  [avg  ms] [avg %]   [cdf %]     [mem KB]  [times called]
ModifyGraphWithDelegate   1     194.276   62.600%   62.600%     2972.000        1
AllocateTensors           1     116.068   37.400%   100.000%     0.000          2
Timings (microseconds): count=1 curr=310344
Memory (bytes): count=0
2 nodes observed
Operator-wise Profiling Info for Regular Benchmark Runs:
============================== Run Order ==============================
[node type]        [start]   [first]   [avg  ms]    [%]      [cdf%]  [mem KB]  [times called]
 [Name]
```

```
TfLiteNnapiDelegate  0.000     6.694     8.080   100.000%   100.000%    0.000        1
 [output]:65
============================== Top by Computation Time ==============================
[node type]         [start]   [first]    [avg  ms]     [%]      [cdf%]    [mem KB]  [times called]
 [Name]
TfLiteNnapiDelegate    0.000    6.694     8.080   100.000%   100.000%   0.000         1
 [output]:65
Number of nodes executed: 1
============================== Summary by node type ==============================
[Node type]         [count]  [avg  ms]  [avg %]     [cdf %]    [mem KB]    [times called]
TfLiteNnapiDelegate     1       8.079   100.000%     100.000%    0.000           1
Timings (microseconds): count=120 first=6694 curr=8315 min=6496 max=17352 avg=8079.66 std=1026
Memory (bytes): count=0
1 nodes observed
```

Using this tool, we do benchmark tests on different hardware delegate of i.MX 8M Plus. The following table shows the results.

Table 1. Comparison of inference time between CPU and NPU on i.MX 8M Plus EVK

Model Name	4 X A53	1 X A53	NPU
inception_v4_299_quant	744.424 ms	2507 ms	36.163 ms
mobilenet_v1_0_25_224_quant	6.73327 ms	17.9476 ms	2.42682 ms
mobilenet_v1_0_5_224_quant	14.1138 ms	45.0554 ms	2.82396 ms
mobilenet_v1_0_75_224_quant	25.3732 ms	86.2732 ms	3.31951 ms
mobilenet_v1_1_0_224_quant	40.0481 ms	138.407 ms	3.99012 ms
mobilenet_v2_1_0_224_quant	32.3174 ms	104.026 ms	4.53579 ms

Note:

All models above were downloaded from: https://www.tensorflow.org/lite/guide/hosted_models#quantized_model.

2.3 VSI profiling on hardware accelerators

This section describes how to enable profiler on VSI NPU, and how to capture logs.

1. Stop the EVK board in U-Boot by pressing **Enter**.
2. Disable selinux through the U-Boot command lines.

```
u-boot=> setenv append_bootargs
 androidboot.selinux=permissive
u-boot=> boot
```

3. To enable the OEM to unlock the Android device (EVK board), perform the following steps on both the Host and Target:
   a. On the Target, from **Settings** -> **About Tablet**, click 10 times on **Build Number**. This enables **Developer options**.
   b. In **Developer options**, select the **OEM Unlocking** check box to enable OEM unlocking.
   c. On the Android terminal (UART terminal), execute the following command:

```
$ reboot bootloader
```

d. On the Host, execute the following command:

```
$ sudo fastboot oem unlock
```

After a while, the system prompts that the OEM unlocks successfully. Then you can restart the board.

4. Set properties on the Android terminal. First, execute `adb root` to enter root mode. Then, on the Android terminal, execute the following commands:

```
setprop vendor.CNN_PERF 1
setprop vendor.NN_EXT_SHOW_PERF 1
setprop vendor.VIV_VX_DEBUG_LEVEL 1
setprop vendor.VIV_VX_PROFILE 1
setprop vendor.VSI_NN_LOG_LEVEL 5
```

5. On the Android terminal, find service `/vendor/bin/hw/android.neural.network***vsi-npu***`, and rename it to other name.
6. Use `ps -ef | grep vsi-npu` to find the current service and then terminate it.
7. Restart the service by executing `./<Name you used when renaming it>`, and then keep this terminal for VSI profiler use later. At this time, if you use another Android terminal to do the NPU benchmark test or other tasks, the previous terminal displays the detailed VSI profile.

2.4 Running image classification applications

This image classification is with a pre-trained model that can recognize 1000 different types of items from input frames on a mobile camera.

This application uses image classification to continuously classify whatever it sees from the device's back camera. Inference is performed using the TensorFlow Lite Java API. The demo application classifies frames in real time, displaying the top most probable classifications. It allows the user to choose between a floating point or quantized model, select the thread count, and decide whether to run on CPU, both GPU and NPU via NNAPI.

To build, install and run it, see https://github.com/tensorflow/examples/blob/master/lite/examples/image_classification/android/README.md .

In Android studio, find "Tools – SDK Manager". Find "Appearance & Behavior – System Settings – Android SDK". Choose the SDK Version corresponding to the system version, which is used on the board. Click "OK", and then SDK starts to be installed.

When the TFL Classify app is opened, choose "Quantized_MobileNet" from the drop-down menu for Model. In the drop-down menu for Device, choose "CPU" or "NNAPI" to use NPU accelerator as follows.

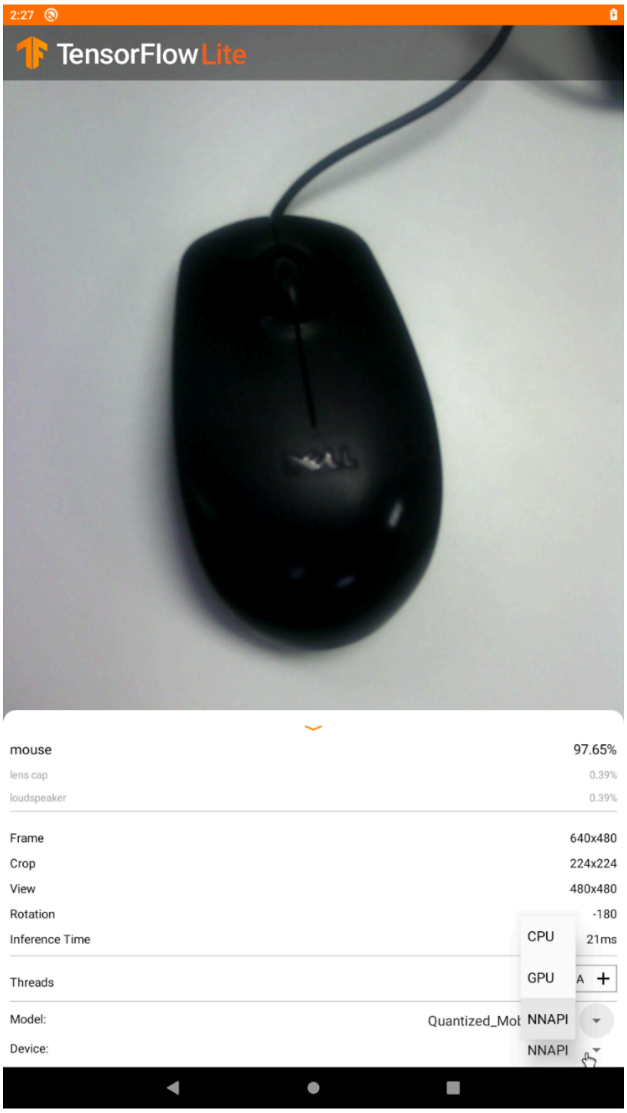

Figure 3. Using NPU accelerator

The following are two pictures using CPU or NNAPI as device. The inference time is different.

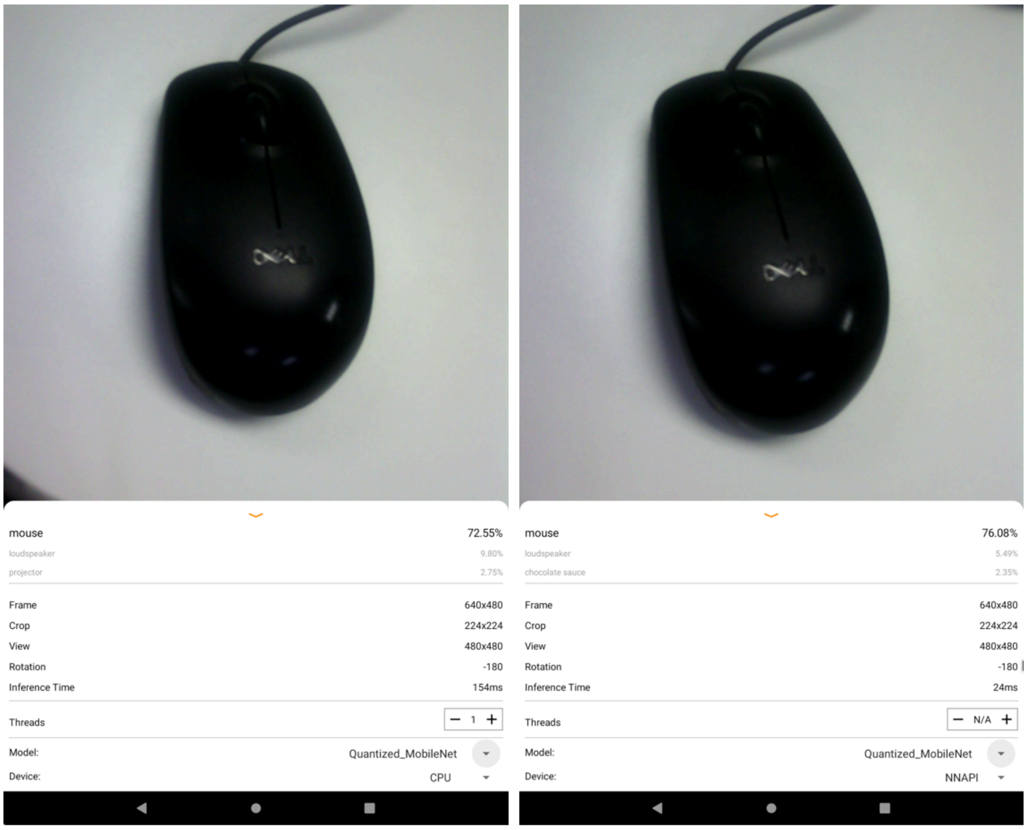

Figure 4. Using CPU or NNAPI as device

The output using logcat should be similar to:

```
1970-01-01 08:00:22.841 375-375/? I/ServiceManagement:
 Registered android.hardware.neuralnetworks@1.2::IDevice/vsi-
npu (start delay of 1600ms)
……
2020-06-16 22:01:24.986 2755-2789/
org.tensorflow.lite.examples.classification D/tensorflow:
 ClassifierActivity: Closing classifier.
2020-06-16 22:01:24.990 2755-2789/
org.tensorflow.lite.examples.classification D/
tensorflow: ClassifierActivity: Creating classifier
 (model=QUANTIZED_MOBILENET, device=NNAPI, numThreads=4)
2020-06-16 22:01:24.996 2755-2789/
org.tensorflow.lite.examples.classification I/tflite: Created
 TensorFlow Lite delegate for NNAPI.
2020-06-16 22:01:24.998 2755-2789/
org.tensorflow.lite.examples.classification I/Manager:
 DeviceManager::DeviceManager
2020-06-16 22:01:24.998 2755-2789/
org.tensorflow.lite.examples.classification I/Manager:
 findAvailableDevices
2020-06-16 22:01:24.998 2755-2789/
org.tensorflow.lite.examples.classification I/Manager: Found
 interface vsi-npu
```

3 Neural network API reference

The neural-network operations and corresponding supported API functions are listed in the following table.

Table 2. Neural-network operations and supported API functions

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.4.0
Activation			
elu	vx_kernel (ELU)	-	-
floor	vxTensorRoundingNode	ANEURALNETWORKS_ FLOOR	FLOOR
leaky_relu	vxLeakyReluLayer	-	-
prelu	vx_kernel (PRELU)	ANEURALNETWORKS_ PRELU	PRELU
relu	vxActivationLayer	ANEURALNETWORKS_ RELU	RELU
relu1	vxActivationLayer	ANEURALNETWORKS_ RELU1	RELU_N1_TO_1
relu6	vxActivationLayer	ANEURALNETWORKS_ RELU6	RELU6
relun	vxActivationLayer	-	-
swish	-	-	-
Hard_swish	-	ANEURALNETWORKS_ HARD_SWISH	HARD_SWISH
rsqrt	vxActivationLayer	ANEURALNETWORKS_ RSQRT	RSQRT
sigmoid	vxActivationLayer	ANEURALNETWORKS_ LOGISTIC	LOGISTIC
softmax	vxSoftmaxLayer	ANEURALNETWORKS_ SOFTMAX	SOFTMAX
softrelu	vxActivationLayer	-	-
sqrt	vxActivationLayer	ANEURALNETWORKS_ SQRT	SQRT
tanh	vxActivationLayer	ANEURALNETWORKS_ TANH	TANH
bounded	-	-	-
linear	-	-	-
Dense Layers			
convolution_relu	vxConvolutionReluLayer	-	-
convolution_relu_pool	vxConvolutionReluPooling Layer	-	-
fullyconnected_relu	vxFullyConnectedReluLayer	-	-
Element Wise			
abs	vxLeakyReluLayer	-	ABS

Table 2. Neural-network operations and supported API functions*...continued*

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.4.0
add	vxTensorAddNode	ANEURALNETWORKS_ ADD	ADD
add_n	vx_kernel (ADDN)	-	ADD_N
clip_by_value	vx_kernel (CLIP)	-	-
div	vxTensorDivideNode	ANEURALNETWORKS_DIV	DIV
equal	vx_kernel (EQUAL)	-	EQUAL
exp	vx_kernel (EXP)	-	EXP
log	vx_kernel (LOG)	-	-
floor_div	vx_kernel (FLOOR_DIV)	-	FLOOR_DIV
greater	vx_kernel (GREATER)	ANEURALNETWORKS_ GREATER	GREATER
greater_equal	vx_kernel (GREATER_ EQUAL)	ANEURALNETWORKS_ GREATER_EQUAL	GREATER_EQUAL
less	vx_kernel (LESS)	ANEURALNETWORKS_ LESS	LESS
less_equal	vx_kernel (LESS_EQUAL)	ANEURALNETWORKS_ LESS_EQUAL	LESS_EQUAL
logical_and	vx_kernel (LOGICAL_AND)	ANEURALNETWORKS_ LOGICAL_AND	LOGICAL_AND
logical_or	vx_kernel (LOGICAL_OR)	ANEURALNETWORKS_ LOGICAL_OR	LOGICAL_OR
minimum	vx_kernel (MINIMUM)	ANEURALNETWORKS_ MINIMUM	MINIMUM
maximum	vx_kernel (MAXIMUM)	ANEURALNETWORKS_ MAXIMUM	MAXIMUM
mul	vxTensorMultiplyNode	ANEURALNETWORKS_ MUL	MUL
negative	vx_kernel (NEG)	ANEURALNETWORKS_ NEG	NEG
not_equal	vx_kernel (NOT_EQUAL)	ANEURALNETWORKS_ NOT_EQUAL	NOT_EQUAL
pow	vx_kernel (POW)	ANEURALNETWORKS_ POW	POW
real_div	vxTenorDivideNode	-	-
select	vx_kernel (SELECT)	ANEURALNETWORKS_ SELECT	SELECT
square	vxActivationLayer	-	SQUARE
sub	vxTensorSubtractNode	ANEURALNETWORKS_SUB	SUB
where	vx_kernel (SELECT)	-	WHERE
Image Processing			
resize_bilinear	vxTensorScaleNode	ANEURALNETWORKS_ RESIZE_BILINEAR	RESIZE_BILINEAR

Table 2. Neural-network operations and supported API functions*...continued*

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.4.0
resize_nearestneighbor	vxTensorScaleNode	ANEURALNETWORKS_ RESIZE_NEAREST_ NEIGHBOR	RESIZE_NEAREST_ NEIGHBOR
yuv_rgb_scale	vxYUV2RGBScaleNode	-	-
Matrix Multiplication			
fullyconnected	vxFullyConnectedLayer	ANEURALNETWORKS_ FULLY_CONNECTED	FULLY_CONNECTED
matrix_mul	vx_kernel (MATRIXMUL)	-	-
Normalization			
batch_normalize	vxBatchNormalizationLayer	-	-
instance _normalize	vx_kernel (INSTANCE_ NORM)	-	-
l2_normalize	vxL2NormalizeLayer	ANEURALNETWORKS_L2_ NORMALIZATION	L2_NORMALIZATION
layer_normalize	vx_kernel (LAYER_NORM)	-	-
local_response_normalize	vxNormalizationLayer	ANEURALNETWORKS_ LOCAL_RESPONSE_ NORMALIZATION	LOCAL_RESPONSE_ NORMALIZATION
Reshape			
batch_to_space	vxReorgLayer2	ANEURALNETWORKS_ BATCH_TO_SPACE_ND	BATH_TO_SPACE_ND
concat	vxCreateTensorView	ANEURALNETWORKS_ CONCATENATION	CONCATENATION
crop	vx_kernel (CROP)	-	-
depth_to_space	vxReorgLayer2	ANEURALNETWORKS_ DEPTH_TO_SPACE	DEPTH_TO_SPACE
expand_dims	vxReshapeTensor	-	EXPAND_DIMS
flatten	vxReshapeTensor	ANEURALNETWORKS_ RESHAPE	RESHAPE
gather	vx_kernel (GATHER)	-	GATHER
pad	vxTensorPadNode	ANEURALNETWORKS_PAD	PAD
permute	vxTensorPermuteNode	ANEURALNETWORKS_ TRANSPOSE	TRANSPOSE
reduce_mean	vxTensorMeanNode	ANEURALNETWORKS_ MEAN	MEAN
reduce_sum	vxTensorReduceSumNode	-	-
reorg	vxReorgLayer	-	-
reshape	vxReshapeTensor	ANEURALNETWORKS_ RESHAPE	RESHAPE
reverse	vxTensorReverse	-	-
reverse_squeeze	vxTensorReverse	-	-

Table 2. Neural-network operations and supported API functions*...continued*

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.4.0
slice	vxCreateTensorView	-	SLICE
space_to_batch	vxReorgLayer2	ANEURALNETWORKS_ SPACE_TO_BATCH_ND	SPACE_TO_BATCH_ND
space_to_depth	vxReorgLayer2	ANEURALNETWORKS_ SPACE_TO_DEPTH	SPACE_TO_DEPTH
split	vxCreateTensorView	ANEURALNETWORKS_ SPLIT	SPLIT
squeeze	vxReshapeTensor	ANEURALNETWORKS_ SQUEEZE	SQUEEZE
stack	vx_kernel (STACK)	-	-
strided_slice	vxTensorStrideSliceNode	ANEURALNETWORKS_ STRIDED_SLICE	STRIDED_SLICE
tensor_stack_concat	vx_kernel (TENSORSTACKCONCAT)	-	-
unstack	vx_kernel (UNSTACK)	-	-
RNN			
gru	vx_kernel (GRU_OVXLIB)	-	-
gru_cell	vx_kernel (GRUCELL_ OVXLIB)	-	-
lstm_layer	vxLSTMLayer	-	-
lstm_unit	vxLSTMUnitLayer	ANEURALNETWORKS_ LSTM	LSTM
rnn	vxRNNLayer	ANEURALNETWORKS_ RNN	RNN
Sliding Window			
avg_pool	vxPoolingLayer	ANEURALNETWORKS_ AVERAGE_POOL	AVERAGE_POOL_2D
convolution	vxConvolutionLayer	ANEURALNETWORKS_ CONV_2D	CONV_2D
deconvolution	vxDeconvolutionLayer	ANEURALNETWORKS_ TRANSPOSE_CONV_2D	TRANSPOSE_CONV
depthhwise_convolution	vxConvolutionLayer	ANEURALNETWORKS_ DEPTHWISE_CONV_2D	DEPTHWISE_CONV_2D
depthwise_conv1d	vx_kernel (DEPTHWISE_ CONV1D)	-	-
group_conv1d	vx_kernel (CONV1D)	-	-
Log_softmax	vx_kernel (LOG_SOFTMAX)	ANEURALNETWORKS_ LOG_SOFTMAX	LOG_SOFTMAX
dilated_convolution	vxConvolutionLayer	-	-
l2_pool	vxPoolingLayer	ANEURALNETWORKS_L2_ POOL	L2_POOL_2D

Table 2. Neural-network operations and supported API functions*...continued*

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.4.0
max_pool	vxPoolingLayer	ANEURALNETWORKS_ MAX_POOL	MAX_POOL_2D
max_pool_with_argmax	vx_kernel (POOLWITHARGMAX)	-	-
max_unpool	vx_kernel (UPSAMPLE)	-	-
Others			
argmax	vx_kernel (ARGMAX)	ANEURALNETWORKS_ ARGMAX	ARGMAX
argmin	vx_kernel (ARGMIN)	ANEURALNETWORKS_ ARGMIN	ARGMIN
dequantize	vxTensorCopyNode	ANEURALNETWORKS_ DEQUANTIZE	DEQUANTIZE
quantize	-	ANEURALNETWORKS_ QUANTIZE	QUANTIZE
image_process	vx_kernel (IMAGE_ PROCESS)	-	-
region_proposal	vxRPNLayer	-	-
roi_pool	vxROIPoolingLayer	-	-
shuffle_channel	vx_kernel (SHUFFLE_ CHANNEL)	-	-

4 OVXLIB Operation Support with GPU

This section provides a summary of the neural network OVXLIB operations supported by the NXP Graphics Processing Unit (GPU) IP with hardware support for OpenVX and OpenCL and a compatible Software stacks. OVXLIB operations are listed in the following table.

The following abbreviations are used for format types:

- `asym-u8`: `asymmetric_affine-uint8`
- `asym-i8`: `asymmetric_affine-int8`
- `fp32`: `float32`
- `pc-sym-i8`: `perchannel_symmetric_int8`
- `h`: `half`
- `bool8`: `bool8`
- `int16`: `int16`
- `int32`: `int32`

Table 3. OVXLIB operation support with GPU

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
Basic Operations					
VSI_NN_OP_CONV2D	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	p8	asym-i8	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
VSI_NN_OP_CONV1D	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	p8	asym-i8	✓	✓
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
VSI_NN_OP_DEPTHWISE_CONV1D	asym-u8	asym-u8	asym-u8	✓	
	asym-i8	asym-i8	asym-i8	✓	
VSI_NN_OP_DECONVOLUTION	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	p8	asym-i8	✓	✓
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
VSI_NN_OP_FCL	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	p8	asym-i8	✓	✓
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
Activation Operations					
VSI_NN_OP_ELU	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_HARD_SIGMOID	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SWISH	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_LEAKY_RELU	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_PRELU	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓

Table 3. OVXLIB operation support with GPU...*continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_RELU	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_RELUN	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_RSQRT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SIGMOID	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SOFTRELU	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SQRT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_TANH	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_ABS	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_CLIP	asym-u8		asym-u8	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_EXP	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_LOG	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_NEG	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_MISH	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SOFTMAX	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_LOG_SOFTMAX	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SQUARE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SIN	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
Elementwise Operations					
VSI_NN_OP_ADD	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SUBTRACT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_MULTIPLY	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_DIVIDE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_MAXIMUN	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_MINIMUM	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_POW	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_FLOORDIV	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_MATRIXMUL	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_RELATIONAL_OPS	asym-u8		bool8	✓	✓
	asym-i8		bool8	✓	✓
	fp32		bool8	✓	✓
	h		bool8	✓	✓
	bool8		bool8	✓	✓
VSI_NN_OP_LOGICAL_OPS	bool8		bool8	✓	✓
VSI_NN_OP_SELECT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
	bool8		bool8	✓	✓
VSI_NN_OP_ADDN	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
Normalization Operations					
VSI_NN_OP_BATCH_NORM	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_LRN	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_LRN2	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_L2_NORMALIZE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_L2NORMALZESCALE	asym-u8		asym-u8	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_LAYER_NORM	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_INSTANCE_NORM	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_BATCHNORM_SINGLE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_MOMENTS	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
Reshape Operations					
VSI_NN_OP_SLICE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SPLIT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_CONCAT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_STACK	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	h		h	✓	✓
VSI_NN_OP_UNSTACK	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_RESHAPE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SQUEEZE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_PERMUTE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_REORG	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SPACE2DEPTH	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_DEPTH2SPACE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_BATCH2SPACE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SPACE2BATCH	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_PAD	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_REVERSE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_STRIDED_SLICE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_CROP	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_REDUCE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_ARGMX	asym-u8		asym-u8/ int16/int32	✓	✓
	asym-i8		asym-u8/ int16/int32	✓	✓
	fp32		int32	✓	✓
	h		asym-u8/ int16/int32	✓	✓
VSI_NN_OP_ARGMIN	asym-u8		asym-u8/ int16/int32	✓	✓
	asym-i8		asym-u8/ int16/int32	✓	✓
	fp32		int32	✓	✓
	h		asym-u8/ int16/int32	✓	✓
VSI_NN_OP_SHUFFLECHANNEL	asym-u8		asym-u8	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
RNN Operations					
VSI_NN_OP_LSTMUNIT_OVXLIB	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	p8	asym-i8	✓	✓
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
VSI_NN_OP_LSTM	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	pc-sym-i8	asym-i8	✓	✓
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
VSI_NN_OP_GRUCELL_OVXLIB	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	p8	asym-i8	✓	✓
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
VSI_NN_OP_GRU_OVXLIB	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	p8	asym-i8	✓	✓
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
VSI_NN_OP_SVDF	asym-u8	asym-u8	asym-u8	✓	✓
	asym-i8	p8	asym-i8	✓	✓
	fp32	fp32	fp32	✓	✓
	h	h	h	✓	✓
Pooling Operations					
VSI_NN_OP_ROI_POOL	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_POOLWITHARGMAX	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_UPSAMPLE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	fp32		fp32	✓	✓
	h		h	✓	✓
Miscellaneous Operations					
VSI_NN_OP_PROPOSAL	asym-u8		asym-u8	✓	
	asym-i8		asym-i8	✓	
	fp32		fp32	✓	
	h		h	✓	
VSI_NN_OP_VARIABLE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_DROPOUT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_RESIZE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_DATACONVERT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_A_TIMES_B_PLUS_C	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_FLOOR	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_EMBEDDING_LOOKUP	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
VSI_NN_OP_GATHER	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_GATHER_ND	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_TILE	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_RELU_KERAS	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_ELTWISEMAX	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_INSTANCE_NORM	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_FCL2	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_POOL	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓
VSI_NN_OP_SIGNAL_FRAME	asym-u8		asym-u8	✓	
	asym-i8		asym-i8	✓	
	fp32		fp32	✓	

Table 3. OVXLIB operation support with GPU*...continued*

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	h		h	✓	
VSI_NN_OP_CONCATSHIFT	asym-u8		asym-u8	✓	✓
	asym-i8		asym-i8	✓	✓
	fp32		fp32	✓	✓
	h		h	✓	✓

5 OVXLIB Operation Support with NPU

This section provides a summary of the neural network OVXLIB operations supported by the NXP Neural Processor Unit (NPU) IP and a compatible Software stacks. OVXLIB operations are listed in the following table.

The following abbreviations are used for format types:

- `asym-u8`: asymmetric_affine-uint8
- `asym-i8`: asymmetric_affine-int8
- `fp32`: float32
- `pc-sym-i8`: perchannel_symmetric-int8
- `h`: half
- `bool8`: bool8
- `int16`: int16
- `int32`: int32

The following abbreviations are used to reference key Execution Engines (NPU) in the hardware:

- NN: Neural-Network Engine
- PPU: Parallel Processing Unit
- TP: Tensor Processor

Table 4. OVXLIB operation support with NPU

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
Basic Operations						
VSI_NN_OP_CONV2D	asym-u8	asym-u8	asym-u8	✓		
	asym-i8	pc-sym-i8	asym-i8	✓		✓
	fp32	fp32	fp32			✓
	h	h	h			✓
VSI_NN_OP_CONV1D	asym-u8	asym-u8	asym-u8	✓		
	asym-i8	pc-sym-i8	asym-i8	✓		✓
	fp32	fp32	fp32			✓
	h	h	h			✓

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
VSI_NN_OP_DEPTHWISE_CONV1D	asym-u8	asym-u8	asym-u8			✓
	asym-i8	asym-i8	asym-i8			✓
VSI_NN_OP_DECONVOLUTION	asym-u8	asym-u8	asym-u8	✓		
	asym-i8	pc-sym-i8	asym-i8	✓		✓
	fp32	fp32	fp32			✓
	h	h	h			✓
VSI_NN_OP_FCL	asym-u8	asym-u8	asym-u8		✓	
	asym-i8	pc-sym-i8	asym-i8		✓	✓
	fp32	fp32	fp32			✓
	h	h	h		✓	
Activation Operations						
VSI_NN_OP_ELU	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_HARD_SIGMOID	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_SWISH	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_LEAKY_RELU	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_PRELU	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_RELU	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
VSI_NN_OP_RELUN	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_RSQRT	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_SIGMOID	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_SOFTRELU	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_SQRT	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_TANH	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_ABS	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_CLIP	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_EXP	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	h		h			✓
VSI_NN_OP_LOG	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_NEG	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_MISH	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_SOFTMAX	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_LOG_SOFTMAX	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_SQUARE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_SIN	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
Elementwise Operations						
VSI_NN_OP_ADD	asym-u8		asym-u8	✓		
	asym-i8		asym-i8	✓		
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_SUBTRACT	asym-u8		asym-u8	✓		

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	asym-i8		asym-i8	✓		
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_MULTIPLY	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_DIVIDE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_MAXIMUN	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_MINIMUM	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_POW	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_FLOORDIV	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_MATRIXMUL	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_RELATIONAL_OPS	asym-u8		bool8			✓
	asym-i8		bool8			✓
	fp32		bool8			✓
	h		bool8			✓

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	bool8		bool8			✓
VSI_NN_OP_LOGICAL_OPS	bool8		bool8			✓
VSI_NN_OP_SELECT	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
	bool8		bool8			✓
VSI_NN_OP_ADDN	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
Normalization Operations						
VSI_NN_OP_BATCH_NORM	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_LRN	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_LRN2	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_L2_NORMALIZE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_L2NORMALZESCALE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_LAYER_NORM	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	h		h			✓
VSI_NN_OP_INSTANCE_NORM	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_BATCHNORM_SINGLE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_MOMENTS	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
Reshape Operations						
VSI_NN_OP_SLICE	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_SPLIT	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_CONCAT	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_STACK	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_UNSTACK	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_RESHAPE	asym-u8		asym-u8		✓	

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_SQUEEZE	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_PERMUTE	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_REORG	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_SPACE2DEPTH	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_DEPTH2SPACE	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
	bool8		bool8			
VSI_NN_OP_BATCH2SPACE	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_SPACE2BATCH	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_PAD	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	h		h		✓	
VSI_NN_OP_REVERSE	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_STRIDED_SLICE	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_CROP	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_REDUCE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_ARGMAX	asym-u8		asym-u8/ int16/int32			✓
	asym-i8		asym-u8/ int16/int32			✓
	fp32		int32			✓
	h		asym-u8/ int16/int32			✓
VSI_NN_OP_ARGMIN	asym-u8		asym-u8/ int16/int32			✓
	asym-i8		asym-u8/ int16/int32			✓
	fp32		int32			✓
	h		asym-u8/ int16/int32			✓
VSI_NN_OP_SHUFFLECHANNEL	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
RNN Operations						
VSI_NN_OP_LSTMUNIT_OVXLIB	asym-u8	asym-u8	asym-u8		✓	✓

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	asym-i8	pc-sym-i8	asym-i8		✓	✓
	fp32	fp32	fp32			✓
	h	h	h		✓	✓
VSI_NN_OP_LSTM	asym-u8	asym-u8	asym-u8		✓	✓
	asym-i8	pc-sym-i8	asym-i8		✓	✓
	fp32	fp32	fp32			✓
	h	h	h		✓	✓
VSI_NN_OP_GRUCELL_OVXLIB	asym-u8	asym-u8	asym-u8		✓	✓
	asym-i8	pc-sym-i8	asym-i8		✓	✓
	fp32	fp32	fp32			✓
	h	h	h		✓	✓
VSI_NN_OP_GRU_OVXLIB	asym-u8	asym-u8	asym-u8		✓	✓
	asym-i8	pc-sym-i8	asym-i8		✓	✓
	fp32	fp32	fp32			✓
	h	h	h		✓	✓
VSI_NN_OP_SVDF	asym-u8	asym-u8	asym-u8		✓	✓
	asym-i8	pc-sym-i8	asym-i8		✓	✓
	fp32	fp32	fp32			✓
	h	h	h		✓	✓
Pooling Operations						
VSI_NN_OP_ROI_POOL	asym-u8		asym-u8		✓	✓
	asym-i8		asym-i8		✓	✓
	fp32		fp32			✓
	h		h		✓	✓
VSI_NN_OP_POOLWITHARGMAX	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_UPSAMPLE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
Miscellaneous Operations						
VSI_NN_OP_PROPOSAL	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓

Table 4. OVXLIB operation support with NPU...*continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_VARIABLE	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_DROPOUT	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_RESIZE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_DATACONVERT	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_A_TIMES_B_PLUS_C	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_FLOOR	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_EMBEDDING_LOOKUP	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_GATHER	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_GATHER_ND	asym-u8		asym-u8			✓

Table 4. OVXLIB operation support with NPU*...continued*

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_TILE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_RELU_KERAS	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_ELTWISEMAX	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_INSTANCE_NORM	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_FCL2	asym-u8		asym-u8		✓	
	asym-i8		asym-i8		✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_POOL	asym-u8		asym-u8	✓	✓	
	asym-i8		asym-i8	✓	✓	
	fp32		fp32			✓
	h		h		✓	
VSI_NN_OP_SIGNAL_FRAME	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_CONCATSHIFT	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓

6 Revision History

This table provides the revision history.

Revision history

Revision number	Date	Substantive changes
android-10.0.0_2.5.0	10/2020	Initial release
android-11.0.0_1.0.0	12/2020	i.MX 8M Plus EVK Beta release, and all the other i.MX 8 GA release
android-11.0.0_1.1.0-AUTO	01/2021	i.MX 8QuadXPlus/8QuadMax MEK GA release
android-11.0.0_1.2.0	03/2021	i.MX 8M Plus EVK GA release
android-11.0.0_1.2.1	06/2021	i.MX 8M Plus EVK GA release
android-11.0.0_2.2.0	07/2021	i.MX 8M Mini, i.MX 8M Nano, i.MX 8M Plus, and i.MX 8M Quad GA release
android-11.0.0_2.4.0	10/2021	i.MX 8ULP EVK Alpha release, i.MX 8M Mini, i.MX 8M Nano, i.MX 8M Plus, and i.MX 8M Quad GA release
android-11.0.0_2.6.0	01/2022	i.MX 8ULP EVK Beta release, i.MX 8M Mini, i.MX 8M Nano, i.MX 8M Plus, and i.MX 8M Quad GA release
android-12.0.0_1.0.0	03/2022	i.MX 8ULP EVK Beta release, i.MX 8M Mini, i.MX 8M Nano, i.MX 8M Plus, and i.MX 8M Quad GA release
android-12.0.0_2.0.0	07/2022	i.MX 8ULP EVK Beta release, i.MX 8M Mini, i.MX 8M Nano, i.MX 8M Plus, and i.MX 8M Quad GA release
android-12.1.0_1.0.0	10/2022	i.MX 8ULP EVK Beta release, i.MX 8M Mini, i.MX 8M Nano, i.MX 8M Plus, i.MX 8M Quad, i.MX 8QuadMax, and i.MX 8QuadXPlus GA release
automotive-12.1.0_1.1.0	12/2022	i.MX 8QuadXPlus/8QuadMax MEK (Silicon Revision B0, C0) GA release

7 Legal information

7.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

7.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

Date of release: 15 December 2022
Document identifier: IMXTFLUG
Document number: