IMXTFLUG

i.MX TensorFlow Lite on Android User's Guide

Rev. android-10.0.0_2.5.0 — 21 October 2020 User's Guide

Contents

1 Neural Network Runtime Overview

The chapter describes 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. Additional backend can be also implemented to expand support for other frameworks.

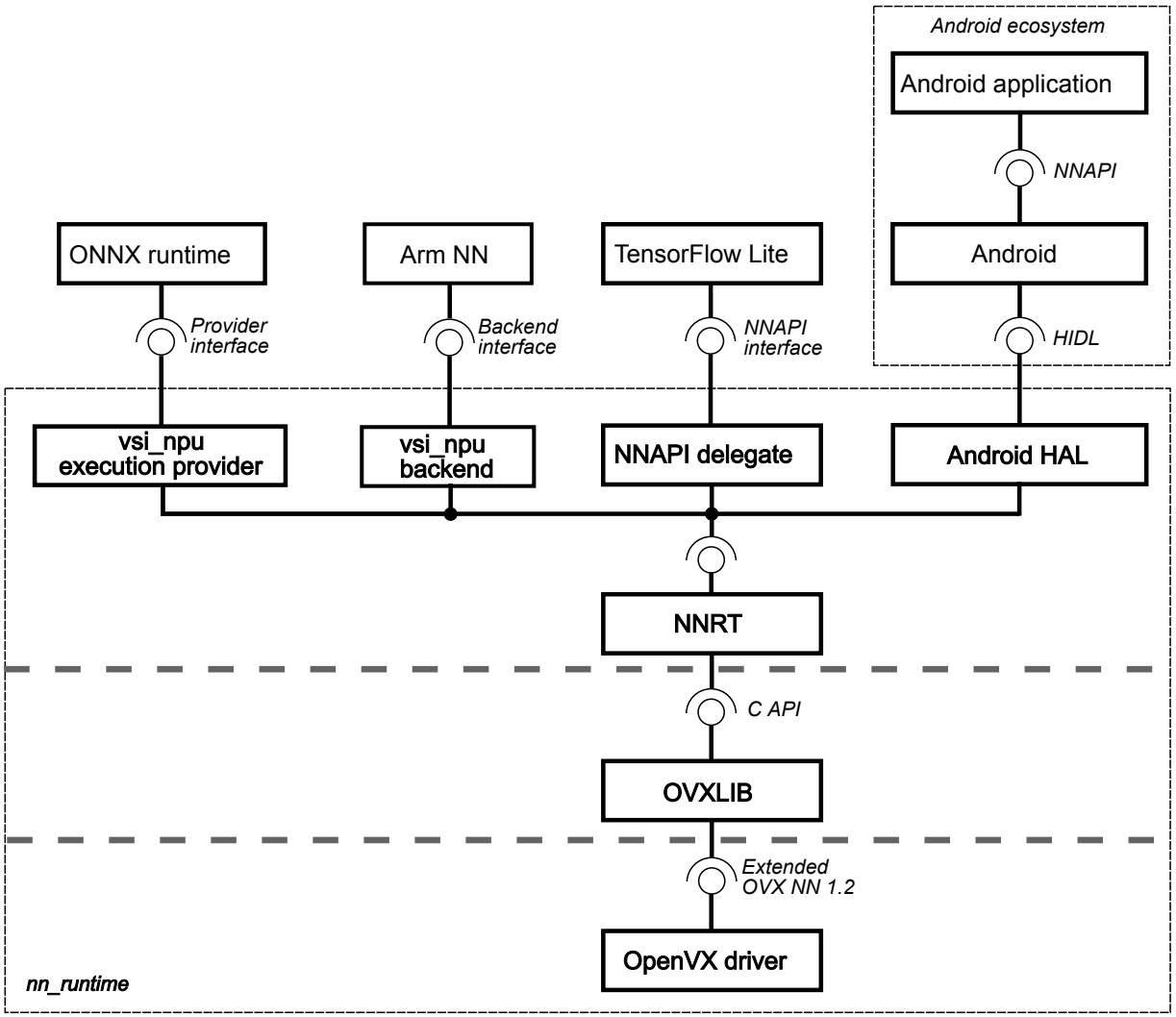

Figure 1. NNRT software architecture

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 v1.2 HAL interface
- For TensorFlow Lite, the NNRT supports NNAPI Delegate. It supports most operations in Android NNAPI v1.2
- For ArmNN, the NNRT registers itself as a compute backend
- For ONNX Runtime, the NNRT registers itself as an execution provider

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.MX8 compute devices, such as NPU, GPU and CPU.

NOTE

Both the OpenCV and PyTorch inference enginees are currently not supported for running on the NXP NN accelerators. Therefore, both frameworks are not included in the above NXP-NN architecture diagram.

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 v2.2.0
- 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

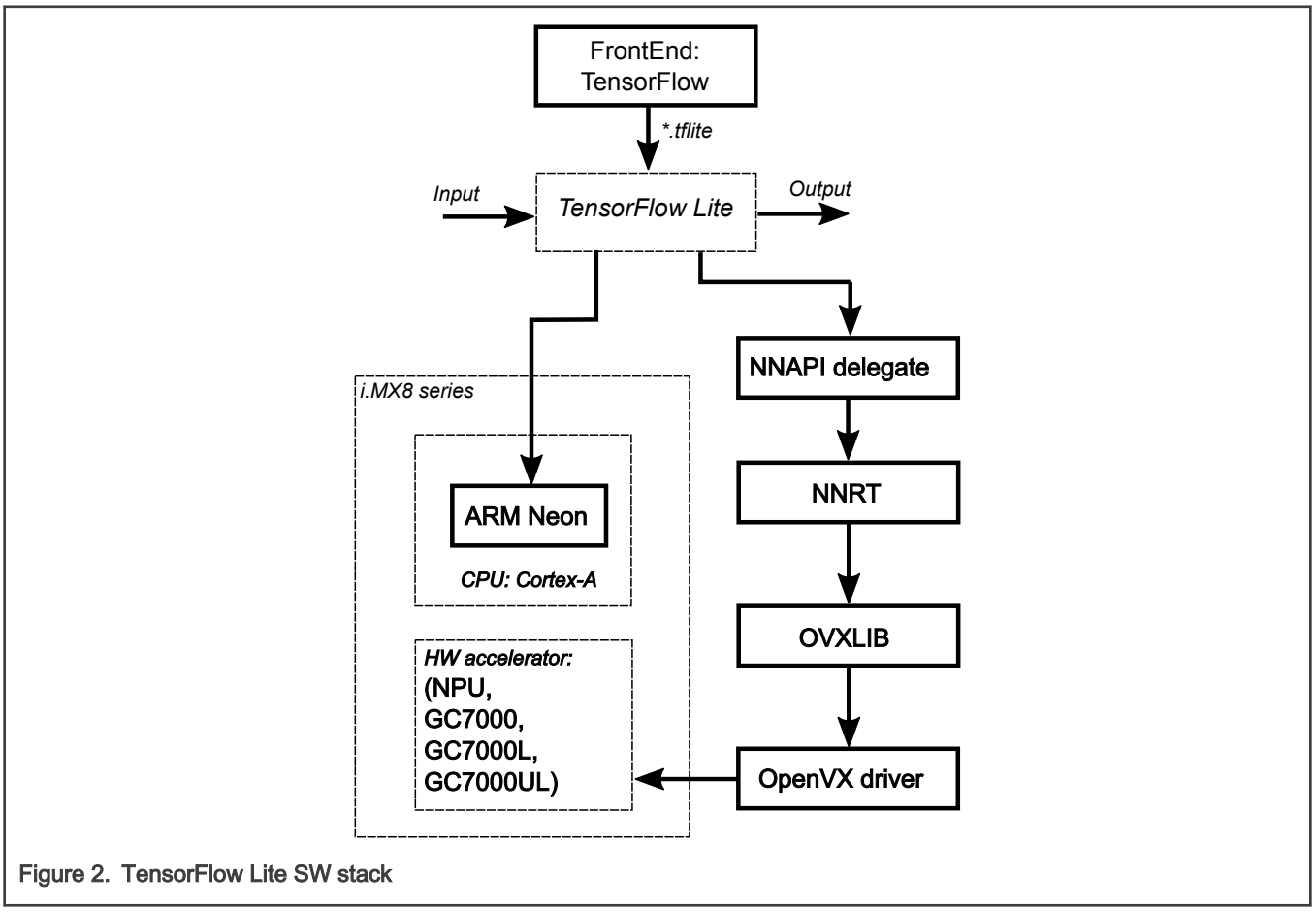

Figure 2. TensorFlow Lite SW stack

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 1.2, 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.

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/r2.2/tensorflow/lite/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/r2.2. The branch is r2.2.
2. See https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android to edit the WORKSPACE to configure the Android NDK/SDK.
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, 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
```

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:

```
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, ---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	760.164 ms	2592.89 ms	42.5733 ms
mobilenet_v1_0_25_224_quant	6.47518 ms	18.39060 ms	4.87023 ms
mobilenet_v1_0_5_224_quant	14.1409 ms	46.9417 ms	5.56093 ms
mobilenet_v1_0_75_224_quant	25.6791 ms	92.7497 ms	6.08879 ms
mobilenet_v1_1_0_224_quant	40.4651 ms	147.368 ms	7.76786 ms
mobilenet_v2_1_0_224_quant	32.4053 ms	111.880 ms	7.04306 ms

NOTE

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

2.3 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 app 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, GPU, or via NNAPI.

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

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.

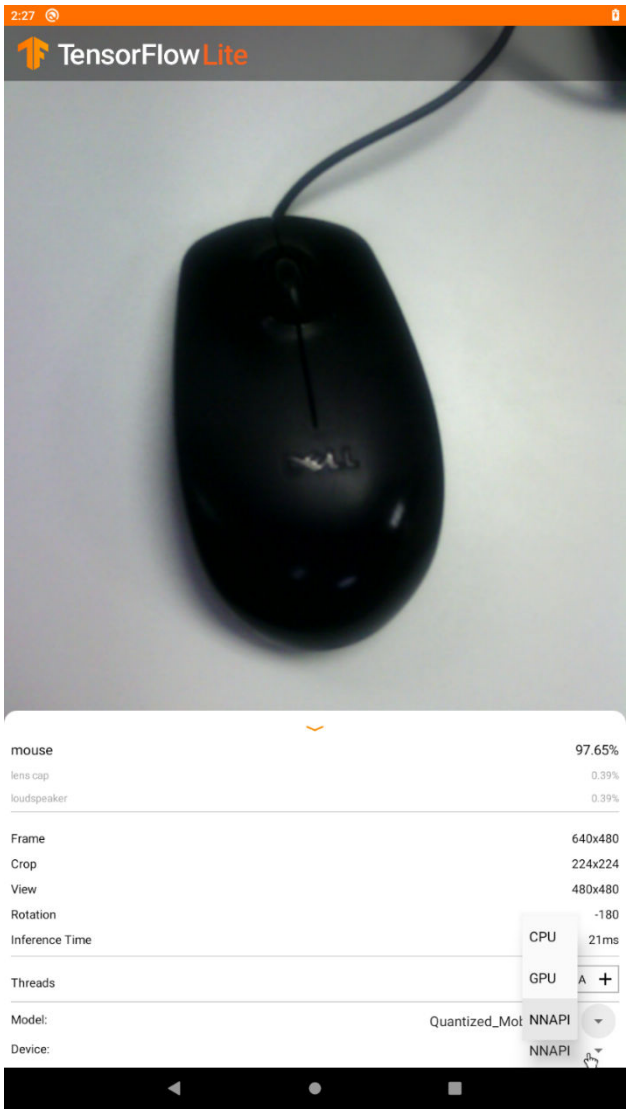

Figure 3. Using NPU accelerator

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

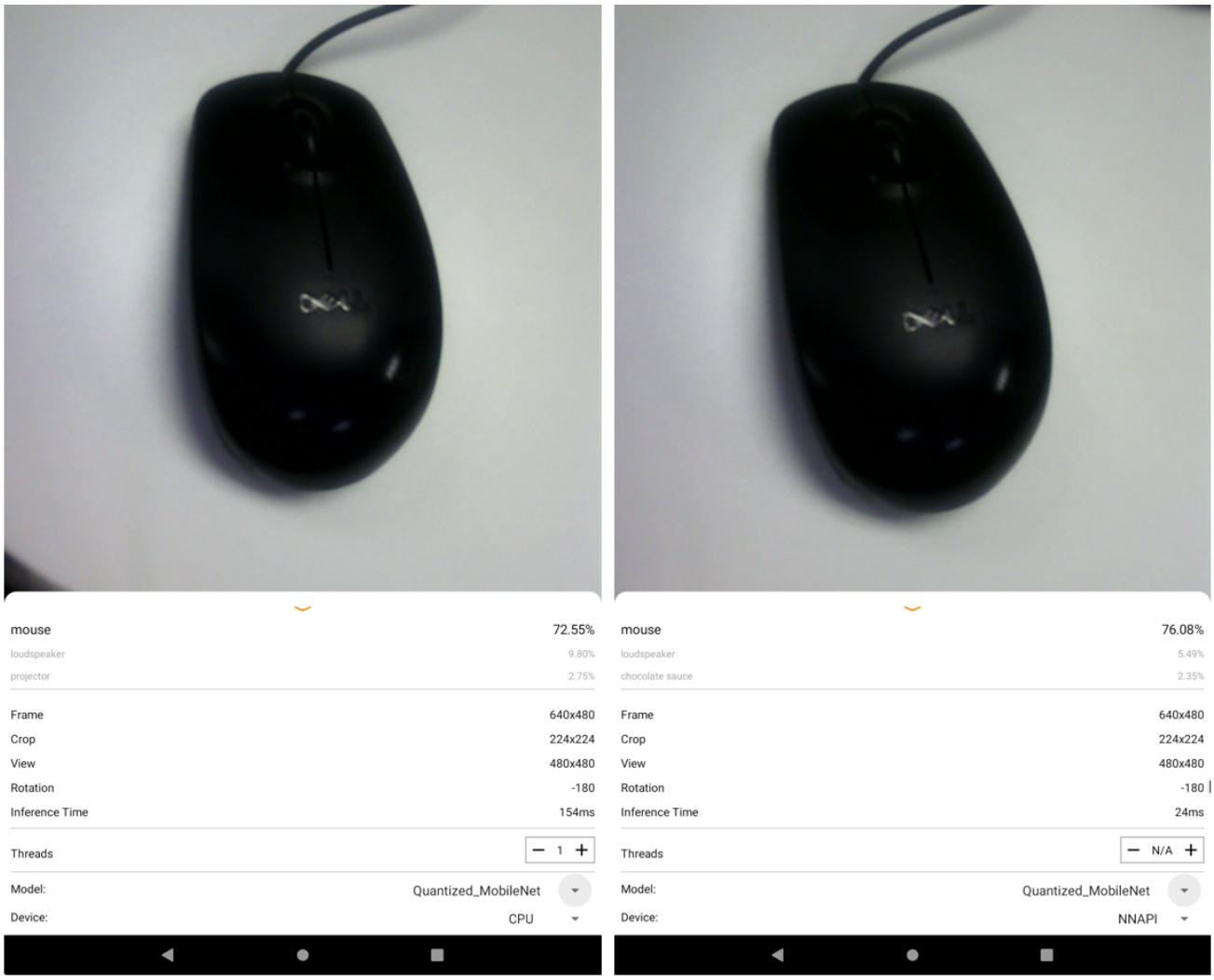

Figure 4. Using CPU or NNAPI as device

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 Revision History

This table provides the revision history.

Table 2. Revision history

Revision number	Date	Substantive changes
android-10.0.0_2.5.0	10/2020	Initial release

A Neural network API reference

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

Table 3. Neural-network operations and supported API functions

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.2.0	Arm NN 20.02	ONNX 1.1.2
Activation					
elu	vx_kernel (ELU)	-	-	-	Elu
floor	vxTensorRounding Node	ANEURALNETWORKS_FLOOR	FLOOR	Floor	Floor
leaky_relu	vxLeakyReluLayer	-	-	Activation/ LeakyReLu	LeakyReLu
prelu	vx_kernel (PRELU)	ANEURALNETWORKS_PRELU	PRELU	PreLu	PreLu
relu	vxActivationLayer	ANEURALNETWORKS_RELU	RELU	Activation/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	Activation/Sigmoid	Sigmoid
softmax	vxSoftmaxLayer	ANEURALNETWORKS_SOFTMAX	SOFTMAX	Softmax	Softmax
softrelu	vxActivationLayer	-	-	Activation/ SoftReLu	-

Table continues on the next page...

Table 3. Neural-network operations and supported API functions (continued)

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.2.0	Arm NN 20.02	ONNX 1.1.2
sqrt	vxActivationLayer	ANEURALNETWORKS_SQRT	SQRT	Activation/Sqrt	Sqrt
tanh	vxActivationLayer	ANEURALNETWORKS_TANH	TANH	Activation/TanH	TanH
bounded	-	-	-	Activation/ BoundedReLu	-
linear	-	-	-	Activation/Linear	-
Dense Layers					
convolution_relu	vxConvolutionReluLayer	-	-	-	-
convolution_relu_pool	vxConvolutionReluPoolingLayer	-	-	-	-
fullyconnected_relu	vxFullyConnectedReluLayer	-	-	-	-
Element Wise					
abs	vxLeakyReluLayer	-	ABS	Activation/Abs	Abs
add	vxTensorAddNode	ANEURALNETWORKS_ADD	ADD	Addition	Add
add_n	vx_kernel (ADDN)	-	ADD_N	-	-
clip_by_value	vx_kernel (CLIP)	-	-	-	Clip
div	vxTensorDivideNode	ANEURALNETWORKS_DIV	DIV	Division	Div
equal	vx_kernel (EQUAL)	-	EQUAL	-	Equal
exp	vx_kernel (EXP)	-	EXP	-	Exp
log	vx_kernel (LOG)	-	-	-	Log
floor_div	vx_kernel (FLOOR_DIV)	-	FLOOR_DIV	-	-
greater	vx_kernel (GREATER)	ANEURALNETWORKS_GREATER	GREATER	-	Greater
greater_equal	vx_kernel (GREATER_EQUAL)	ANEURALNETWORKS_GREATER_EQUAL	GREATER_EQUAL	-	-
less	vx_kernel (LESS)	ANEURALNETWORKS_LESS	LESS	-	Less
less_equal	vx_kernel (LESS_EQUAL)	ANEURALNETWORKS_LESS_EQUAL	LESS_EQUAL	-	-

Table continues on the next page...

Table 3. Neural-network operations and supported API functions (continued)

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.2.0	Arm NN 20.02	ONNX 1.1.2
logical_and	vx_kernel (LOGICAL_AND)	ANEURALNETWORKS_LOGICAL_AND	LOGICAL_AND	-	And
logical_or	vx_kernel (LOGICAL_OR)	ANEURALNETWORKS_LOGICAL_OR	LOGICAL_OR	-	Or
minimum	vx_kernel (MINIMUM)	ANEURALNETWORKS_MINIMUM	MINIMUM	Minimum	Min
maximum	vx_kernel (MAXIMUM)	ANEURALNETWORKS_MAXIMUM	MAXIMUM	Maximum	Max
mul	vxTensorMultiplyNode	ANEURALNETWORKS_MUL	MUL	Multiplication	Mul
negative	vx_kernel (NEG)	ANEURALNETWORKS_NEG	NEG	-	Neg
not_equal	vx_kernel (NOT_EQUAL)	ANEURALNETWORKS_NOT_EQUAL	NOT_EQUAL	-	-
pow	vx_kernel (POW)	ANEURALNETWORKS_POW	POW	-	POW
real_div	vxTenorDivideNode	-	-	-	-
select	vx_kernel (SELECT)	ANEURALNETWORKS_SELECT	SELECT	-	-
square	vxActivationLayer	-	SQUARE	Activation/Square	-
sub	vxTensorSubtractNode	ANEURALNETWORKS_SUB	SUB	Substraction	Sub
where	vx_kernel (SELECT)	-	WHERE	-	Where
Image Processing					
resize_bilinear	vxTensorScaleNode	ANEURALNETWORKS_RESIZE_BILINEAR	RESIZE_BILINEAR	-	Unsample
resize_nearestneighbor	vxTensorScaleNode	ANEURALNETWORKS_RESIZE_NEAREST_NEIGHBOR	RESIZE_NEAREST_NEIGHBOR	-	-
yuv_rgb_scale	vxYUV2RGBScaleNode	-	-	-	-
Matrix Multiplication					

Table continues on the next page...

Table 3. Neural-network operations and supported API functions (continued)

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.2.0	Arm NN 20.02	ONNX 1.1.2
fullyconnected	vxFullyConnectedLayer	ANEURALNETWORKS_FULLY_CONNECTED	FULLY_CONNECTED	FullyConnected	-
matrix_mul	vx_kernel (MATRIXMUL)	-	-	-	-
Normalization					-
batch_normalize	vxBatchNormalizationLayer	-	-	BatchNormalization	-
instance_normalize	vx_kernel (INSTANCE_NORM)	-	-	Normalization	-
l2_normalize	vxL2NormalizeLayer	ANEURALNETWORKS_L2_NORMALIZATION	L2_NORMALIZATION	L2Normalization	-
layer_normalize	vx_kernel (LAYER_NORM)	-	-	-	-
local_response_normalize	vxNormalizationLayer	ANEURALNETWORKS_LOCAL_RESPONSE_NORMALIZATION	LOCAL_RESPONSE_NORMALIZATION	-	LRN
Reshape					
batch_to_space	vxReorgLayer2	ANEURALNETWORKS_BATCH_TO_SPACE_ND	BATH_TO_SPACE_ND	BatchToSpaceNd	-
concat	vxCreateTensorView	ANEURALNETWORKS_CONCATENATION	CONCATENATION	Concat	Concat
crop	vx_kernel (CROP)	-	-	-	-
depth_to_space	vxReorgLayer2	ANEURALNETWORKS_DEPTH_TO_SPACE	DEPTH_TO_SPACE	-	DepthToSpace
expand_dims	vxReshapeTensor	-	EXPAND_DIMS	-	-
flatten	vxReshapeTensor	ANEURALNETWORKS_RESHAPE	RESHAPE	-	-
gather	vx_kernel (GATHER)	-	GATHER	-	-
pad	vxTensorPadNode	ANEURALNETWORKS_PAD	PAD	Pad	Pad

Table continues on the next page...

Table 3. Neural-network operations and supported API functions (continued)

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.2.0	Arm NN 20.02	ONNX 1.1.2
permute	vxTensorPermuteNode	ANEURALNETWORKS_TRANSPOSE	TRANSPOSE	Permute	-
reduce_mean	vxTensorMeanNode	ANEURALNETWORKS_MEAN	MEAN	Mean	ReduceMean
reduce_sum	vxTensorReduceSumNode	-	-	-	ReduseSum
reorg	vxReorgLayer	-	-	-	-
reshape	vxReshapeTensor	ANEURALNETWORKS_RESHAPE	RESHAPE	Reshape	Reshape
reverse	vxTensorReverse	-	-	-	ReverseSequence
reverse_squeeze	vxTensorReverse	-	-	-	-
slice	vxCreateTensorView	-	SLICE	-	Slice
space_to_batch	vxReorgLayer2	ANEURALNETWORKS_SPACE_TO_BATCH_ND	SPACE_TO_BATCH_ND	SpaceToBatchNd	-
space_to_depth	vxReorgLayer2	ANEURALNETWORKS_SPACE_TO_DEPTH	SPACE_TO_DEPTH	SpaceToDepth	SpaceToDepth
split	vxCreateTensorView	ANEURALNETWORKS_SPLIT	SPLIT	-	Split
squeeze	vxReshapeTensor	ANEURALNETWORKS_SQUEEZE	SQUEEZE	-	Squeeze
stack	vx_kernel (STACK)	-	-	-	-
strided_slice	vxTensorStrideSliceNode	ANEURALNETWORKS_STRIDED_SLICE	STRIDED_SLICE	StridedSlice	-
tensor_stack_concat	vx_kernel (TENSORSTACKCONCAT)	-	-	-	-
unstack	vx_kernel (UNSTACK)	-	-	-	-
RNN					
gru	vx_kernel (GRU_OVXLIB)	-	-	-	GRU
gru_cell	vx_kernel (GRUCELL_OVXLIB)	-	-	-	-

Table continues on the next page...

Table 3. Neural-network operations and supported API functions (continued)

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.2.0	Arm NN 20.02	ONNX 1.1.2
lstm_layer	vxLSTMLayer	-	-	-	-
lstm_unit	vxLSTMUnitLayer	ANEURALNETWORKS_LSTM	LSTM	LstmUnit	-
rnn	vxRNNLayer	ANEURALNETWORKS_RNN	RNN	-	-
Sliding Window					
avg_pool	vxPoolingLayer	ANEURALNETWORKS_AVERAGE_POOL	AVERAGE_POOL_2D	Pooling2D/avg	AveragePool
convolution	vxConvolutionLayer	ANEURALNETWORKS_CONV_2D	CONV_2D	Convolution2D	Conv
deconvolution	vxDeconvolutionLayer	ANEURALNETWORKS_TRANSPOSE_CONV_2D	TRANSPOSE_CONV	-	-
depthhwise_convolution	vxConvolutionLayer	ANEURALNETWORKS_DEPTHWISE_CONV_2D	DEPTHWISE_CONV_2D	Depthwise Convolution	-
depthwise_conv1d	vx_kernel (DEPTHWISE_CONV1D)	-	-	-	-
group_conv1d	vx_kernel (CONV1D)	-	-	-	-
Log_softmax	vx_kernel (LOG_SOFTMAX)	ANEURALNETWORKS_LOG_SOFTMAX	LOG_SOFTMAX	-	Logsoftmax
dilated_convolution	vxConvolutionLayer	-	-	-	-
l2_pool	vxPoolingLayer	ANEURALNETWORKS_L2_POOL	L2_POOL_2D	Pooling2D/L2	-
max_pool	vxPoolingLayer	ANEURALNETWORKS_MAX_POOL	MAX_POOL_2D	Pooling2D/max	MaxPool
max_pool_with_argmax	vx_kernel (POOLWITHARGMAX)	-	-	-	-
max_unpool	vx_kernel (UPSAMPLE)	-	-	-	-
Others					
argmax	vx_kernel (ARGMAX)	ANEURALNETWORKS_ARGMAX	ARGMAX	-	ArgMax

Table continues on the next page...

Table 3. Neural-network operations and supported API functions (continued)

Op Category/Name	OpenVX API 1.2	Android NNAPI 1.2	TensorFlow Lite 2.2.0	Arm NN 20.02	ONNX 1.1.2
argmin	vx_kernel (ARGMIN)	ANEURALNETWORKS_ARGMIN	ARGMIN	-	ArgMin
dequantize	vxTensorCopyNode	ANEURALNETWORKS_DEQUANTIZE	DEQUANTIZE	Dequantize	-
quantize	-	ANEURALNETWORKS_QUANTIZE	QUANTIZE	Quantize	-
image_process	vx_kernel (IMAGE_PROCESS)	-	-	-	-
region_proposal	vxRPNLayer	-	-	-	-
roi_pool	vxROIPoolingLayer	-	-	-	-
shuffle_channel	vx_kernel (SHUFFLE_CHANNEL)	-	-	-	-

B 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 4. 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 continues on the next page...

Table 4. 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	✓	✓

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	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	✓	✓
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	✓	✓

Table continues on the next page...

Table 4. 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_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	✓	✓
	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	✓	✓

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
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	✓	✓
	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	✓	✓

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	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	✓	✓
	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	✓	✓

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

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

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
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	✓	✓
	asym-i8		asym-i8	✓	✓

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

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

Table continues on the next page...

Table 4. 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_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	✓	✓

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
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	✓	✓
	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	✓	✓

Table continues on the next page...

Table 4. 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_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	✓	
	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	✓	✓

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	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	✓	✓
	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	✓	✓

Table continues on the next page...

Table 4. OVXLIB operation support with GPU (continued)

OVXLIB Operations	Tensors			Execution Engine	
	Input	Kernel	Output	OpenVX	OpenCL
	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	✓	✓

C 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**: **N**eural-**N**etwork Engine
- **PPU**: **P**arallel **P**rocessing **U**nit

- **TP**: **T**ensor **P**rocessor

Table 5. 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			✓
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			✓

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
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		✓	
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			✓

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	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			✓
	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			✓

Table continues on the next page...

Table 5. 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_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	✓		
	asym-i8		asym-i8	✓		
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_MULTIPLY	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
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			✓
	bool8		bool8			✓
VSI_NN_OP_LOGICAL_OPS	bool8		bool8			✓

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
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			✓

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
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		✓	

Table continues on the next page...

Table 5. 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_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		✓	
	bool8		bool8			

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
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			✓
	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			✓

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	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		✓	✓
	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		✓	✓

Table continues on the next page...

Table 5. 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		✓	✓
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			✓
	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			✓

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
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			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓
	h		h			✓
VSI_NN_OP_TILE	asym-u8		asym-u8			✓
	asym-i8		asym-i8			✓
	fp32		fp32			✓

Table continues on the next page...

Table 5. OVXLIB operation support with NPU (continued)

OVXLIB Operations	Tensors			Execution Engine (NPU)		
	Input	Kernel	Output	NN	TP	PPU
	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			✓

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 21 October 2020

Document identifier: IMXTFLUG

arm