VIM4 KSNN User Usage v1.4.1

1. KSNN Convert

Get the KSNN convert tool.

```
git lfs install
git lfs clone https://gitlab.com/khadas/vim4_npu_sdk.git
```

The tool is vim4_npu_sdk/adla-toolkit-binary-3.1.7.4/python/convert. Here are the arguments of the tool.

--model-name	The model conversion name you want to use. Required option.
-- model-type	Model type used in the conversion. Optional: ‘pytorch’, ‘caffe’, ‘tensorflow’, ‘tflite’, ‘darknet’, ‘onnx’, ’keras’. Required option.
--model	The path of model file. Required option.
--weights	Weights filename. Only Caffe, DarkNet and MxNet model must set this.
--channel-mean-values	Mean values for quant models. Required option.
--inputs	The input name of model.
--inputs-shape	The input shape of model.
--outputs	Output points of graph. Only TensorFlow model must set this.
--quantized-dtype	Quantization type. Currently, it supports "int8", "int16" and "uint8".
--qtype	Describe the type of input.
--source-files	The path of txt which is written quantized image paths. Required option.
--kboard	Choose khadas board. Optional: VIM4.
--print-level	Information log level. Default: 0. Optional: 0, 1.
--outdir	The path of folder which saves converted model and dynamic library.
--batch-size	Quantify batch size of each iteration. This argument is used together with ‘--iterations’. If it is omitted, system will use the value of input shape[0].
--iterations	The number of quantitation iteration. Default: 1. Use together with ‘--batch-size’. batch-size × iterations = the number of quantified images be used.
--inference-input-type	After converting to adla, the input data type of the model. Now only supports float32.
--inference-output-type	After converting to adla, the output data type of the model. Now only supports float32.

--disable-per-channel	Disable per-channel quantitation. Default True.

2. KSNN (Object)

2.1 Initialize KSNN

The initialization function of KSNN API. It must be called before using API interfaces.

API name	KSNN	/
parameters	board	Choose the board you use. Optional: ‘VIM4’.

The example is as follows.

```
model = KSNN('VIM4')
```

2.2 get_nn_version

Show the version of KSNN.

API name	get_nn_version	/
parameters	/	/
return	version	/

The example is as follows.

```
model = KSNN('VIM4')

version = model.get_nn_version()
print('KSNN version is', version)
```

2.3 nn_init

Build neural network.

API name	nn_init	/
parameters	library	The path of static library. Required option.
	model	The path of model file. Required option.
	level	Information log level. Default: 0. Optional: 0, 1.
return	ksnn_stat	/

The example is as follows.

```
model = KSNN('VIM4')

model.nn_init(library='model.so', model='model.adla', level=0)
```

2.4 nn_set_inputs

Convert image to data and set it into neural network.

API name	nn_set_inputs	/
parameters	input_tensor	The input image.
	input_shape	The input shape of model.
	input_type	Choose “RGB” input or “RAW” input. “RGB” input means input_tensor is a numpy.array type np.uint8 and you do not need to do normalization. “RAW” input means input_tensor is a numpy.array type np.float32 and you need to do normalization. If you do not set --inference-input-type, you should choose “RGB”. Set --inference-input-type float32, you should choose “RAW”.
return	ksnn_stat	/

Now KSNN only supports single input model.

2.5 nn_run

Run neural networks.

API name	nn_run	/
parameters	output_type	Choose output type, "FLOAT" or "RAW". If you do not set --inference-output-type, you should choose “FLOAT”. Set --inference-output-type float32, you should choose “RAW”.
return	ksnn_stat	/

2.6 nn_get_outputs

Get outputs data after running neural network.

API name	nn_get_outputs	/
parameters	output_shape	The outputs shape of model. List for each output shape. Example: Single output [(80, 80, 144),] multiply output [(80, 80, 144), (40, 40, 144), (20, 20, 144)] You can set --print-level 1 to print quantification model output.
return	list	/

The including nn_set_inputs, nn_run and nn_get_outputs example is as follows.

RGB example.

```
model = KSNN('VIM4')

model.nn_init(library='model.so', model='model.adla', level=0)

orig_img = cv.imread(picture, cv.IMREAD_COLOR)
img = cv.resize(orig_img, (640, 640))

model.nn_set_inputs(img, input_shape=(640, 640, 3), input_type="RGB")
model.nn_run(output_type="FLOAT")
outputs = model.nn_get_outputs(output_shape=[(80, 80, 144), (40, 40, 144), (20, 20, 144)])
```

RAW example.

```
model = KSNN('VIM4')

model.nn_init(library='model.so', model='model.adla', level=0)

orig_img = cv.imread(picture, cv.IMREAD_COLOR)
img = cv.resize(orig_img, (640, 640)).astype(np.float32)
img[:, :, 0] = img[:, :, 0] - 0
img[:, :, 1] = img[:, :, 1] - 0
img[:, :, 2] = img[:, :, 2] - 0
img = img / 255

model.nn_set_inputs(img, input_shape=(640, 640, 3), input_type="RAW")
model.nn_run(output_type="RAW")
outputs = model.nn_get_outputs(output_shape=[(80, 80, 144), (40, 40, 144), (20, 20, 144)])
```

API nn_inference contains nn_set_inputs, nn_run and nn_get_outputs. So we suggest that you have better use nn_inference instead of them.

2.7 nn_inference

Unify interfaces from input to output.

API name	nn_inference	/
parameters	Input_array	The input image.

	input_shape	The input shape of model.
	input_type	Choose “RGB” input or “RAW” input. “RGB” input means input_tensor is a numpy.array type np.uint8 and you do not need to do normalization. “RAW” input means input_tensor is a numpy.array type np.float32 and you need to do normalization. If you do not set --inference-input-type, you should choose “RGB”. Set --inference-input-type float32, you should choose “RAW”.
	output_type	Choose output type, "FLOAT" or "RAW". If you do not set --inference-output-type, you should choose “FLOAT”. Set --inference-output-type float32, you should choose “RAW”.
	output_shape	The outputs shape of model. List for each output shape. Example: Single output [(80, 80, 144),] multiply output [(80, 80, 144), (40, 40, 144), (20, 20, 144)] You can set --print-level 1 to print quantification model output.
return	list	/

The example is as follows.

RGB example.

```
model = KSNN('VIM4')

model.nn_init(library='model.so', model='model.nb', level=0)

orig_img = cv.imread(picture, cv.IMREAD_COLOR)
img = cv.resize(orig_img, (640, 640))

outputs = model.nn_inference(img, input_shape=(640, 640, 3), input_t
ype="RGB", output_shape=[(80, 80, 144), (40, 40, 144), (20, 20, 144)
], output_type="FLOAT")
```

RAW example.

```
model = KSNN('VIM4')

model.nn_init(library='model.so', model='model.nb', level=0)

orig_img = cv.imread(picture, cv.IMREAD_COLOR)
img = cv.resize(orig_img, (640, 640)).astype(np.float32)
img[:, :, 0] = img[:, :, 0] - 0
img[:, :, 1] = img[:, :, 1] - 0
```

```
img[:, :, 2] = img[:, :, 2] - 0
img = img / 255

outputs = model.nn_inference(img, input_shape=(640, 640, 3), input_type="RAW", output_shape=[(80, 80, 144), (40, 40, 144), (20, 20, 144)], output_type="RAW")
```

2.8 nn_destory_network

Release model from NPU.

API name	nn_destory_network	/
parameters	/	/
return	ksnn_stat	/

The example is as follows.

```
model = KSNN('VIM4')

model.nn_init(library='model.so', model='model.adla', level=0)

model.nn_destory_network()
```

3. KSNN types (Enum)

3.1 ksnn_stat

Meural Network stat Enum class.

STAT_SUCCESS	0
STAT_FAIL	1

3.2 ksnn_board

Support Board List.

STAT_UNKNOWN	0
STAT_VIM4	1

3.3 output_format

Support output format

STAT_FORMAT_UINT8	0

STAT_FORMAT_INT8	1
STAT_FORMAT_INT16	2
STAT_FORMAT_FLOAT32	3