VIM3 KSNN User Usage v1.4

1. KSNN Convert

Get the KSNN convert tool.

```
git clone --recursive https://github.com/khadas/aml_npu_sdk.git
```

The tool is aml_npu_sdk/acuity-toolkit/python/convert. Here are the arguments of the tool.

--model-name	The model conversion name you want to use. Required option.
--platform	Choose you platform. Optional: ‘pytorch’, ‘caffe’, ‘tensorflow’, ‘tflite’, ‘darknet’, ‘onnx’, ’keras’. Required option.
--model	The path of model file. Required option.
--mean-values	Mean values for quant models. Required option.
--input-size-list	Inputs size list for correspoding input points. Only TensorFlow model must set this.
--inputs	Inputs points of graph. Only TensorFlow model must set this.
--outputs	Output points of graph. Only TensorFlow model must set this.
--quantized-dtype	Quant type. Optional: ‘asymmetric_affine’, ‘dynamic_fixed_point’. Required option.
--qtype	Quant type. Optional: ‘int8’, ‘int16’. If ‘--quantized-dtype’ choose ‘dynamic_fixed_point’, It must be set.
--source-files	The path of txt which is written quantized image paths. Required option.
--kboard	Choose khadas board. Optional: ‘VIM3’, ‘VIM3L’.
--print-level	Information log level. Default: 0. Optional: 0, 1.
--weights	Weights filename. Only Caffe and DarkNet model must set this.
--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.
--device	Default: ‘CPU’. Optional: ‘CPU’, ‘GPU’.

The convert example you can find in https://github.com/khadas/ksnn.git, ksnn/examples. Each platform has a README.md which has convert example.

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. Default: 'VIM3'. Optional: 'VIM3', 'VIM3L'.

The example is as follows.

```
model = KSNN('VIM3')
```

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('VIM3')

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, 2.
return	ksnn_stat	/

The example is as follows.

```
model = KSNN('VIM3')

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

2.4 nn_set_inputs

Convert image to data and set it into neural network.

API name	nn_set_inputs	/
parameters	img	The image list, e.g. [image]. This image should be done

		resize and normalized before. If it has more than one inputs, img should be [image_1, image_2, ..]. Required option.
	platform	The platform of origin model. Optional: 'TENSORFLOW', 'KERAS', 'TFLITE', 'CAFFE', 'PYTORCH', 'DARKNET', 'ONNX'. Required option.
	reorder	Channel order. Default: '0 1 2'. Optional: '0 1 2', '2 1 0'. Required option.
	tensor	The input number. Default: 1. Required option.
return	ksnn_stat	/

2.5 nn_run

Run neural networks.

API name	nn_run	/
parameters	/	/
return	ksnn_stat	/

2.6 nn_get_outputs

Get outputs data after running neural network.

API name	nn_get_outputs	/
parameters	tensor	The output number. Default: 1. Required option.
	Output_format	Output data format. Default: output_format.OUT_FORMAT_FLOAT32. Optional: output_format.OUT_FORMAT_UINT8, output_format.OUT_FORMAT_INT8, output_format.OUT_FORMAT_INT16, output_format.OUT_FORMAT_FLOAT32. Required option.
return	list	/

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

```
model = KSNN('VIM3')

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

cv_img =  list()
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

img = img.transpose(2, 0, 1)
cv_img.append(img)

model.nn_set_inputs(cv_img, platform='ONNX', reorder='2 1 0', tensor
=1)
model.nn_run()
outputs = model.nn_get_outputs(tensor=3, output_format=output_format
.OUT_FORMAT_FLOAT32)
```

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	cv_img	The image list, e.g. [image]. This image should be done resize and normalized before. If it has more than one inputs, img should be [image_1, image_2, ..]. Required option.
	platform	The platform of origin model. Optional: 'TENSORFLOW', 'KERAS', 'TFLITE', 'CAFFE', 'PYTORCH', 'DARKNET', 'ONNX'. Required option.
	reorder	Channel order. Default: '0 1 2'. Optional: '0 1 2', '2 1 0'. Required option.
	input_tensor	The input number. Default: 1. Required option.
	output_tensor	The output number. Default: 1. Required option.
	output_format	Output data format. Default: output_format.OUT_FORMAT_FLOAT32. Optional: output_format.OUT_FORMAT_UINT8, output_format.OUT_FORMAT_INT8, output_format.OUT_FORMAT_INT16, output_format.OUT_FORMAT_FLOAT32. Required option.
return	list	/

The example is as follows.

```
model = KSNN('VIM3')
```

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

cv_img =  list()
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

img = img.transpose(2, 0, 1)
cv_img.append(img)

outputs = model.nn_inference(cv_img, platform='ONNX', reorder='2 1 0
', output_tensor=3, output_format=output_format.OUT_FORMAT_FLOAT32)
```

2.8 nn_get_output_tensor_info

Get output tensor info.

API name	nn_get_output_tensor_info	/
parameters	num	Output layer index. Required option.
return	tensor	/

The example is as follows.

```
model = KSNN('VIM3')

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

cv_img =  list()
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

img = img.transpose(2, 0, 1)
cv_img.append(img)

model.nn_set_inputs(cv_img, platform='ONNX', reorder='2 1 0', tensor
=1)
```

```
model.nn_run()
output_1 = model.nn_get_output_tensor_info(num=1)
```

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_VIM3	1
STAT_VIM3L	2

3.3 output_format

Support output format

STAT_FORMAT_UINT8	0
STAT_FORMAT_INT8	1
STAT_FORMAT_INT16	2
STAT_FORMAT_FLOAT32	3