User's Manual

DRP-AI Translator i8 V1.01

User's Manual

Renesas Electronics
www.renesas.com

Rev.1.01 Feb. 2024

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intendeds or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

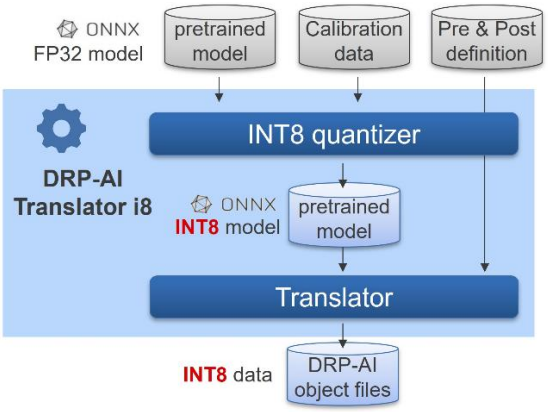
For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Table of Contents

1. About this Manual

1.1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of the function and execution procedure for the "DRP-AI Translator i8". The manual consists of an overview of the tool, how to install and uninstall, how to define input files, how to run, and error messages.



For the details about INT8 quantizer, please refer to User's Manual of INT8 Quantizer.

1.2. Required Role

This manual is intended for the following User/User Roles:

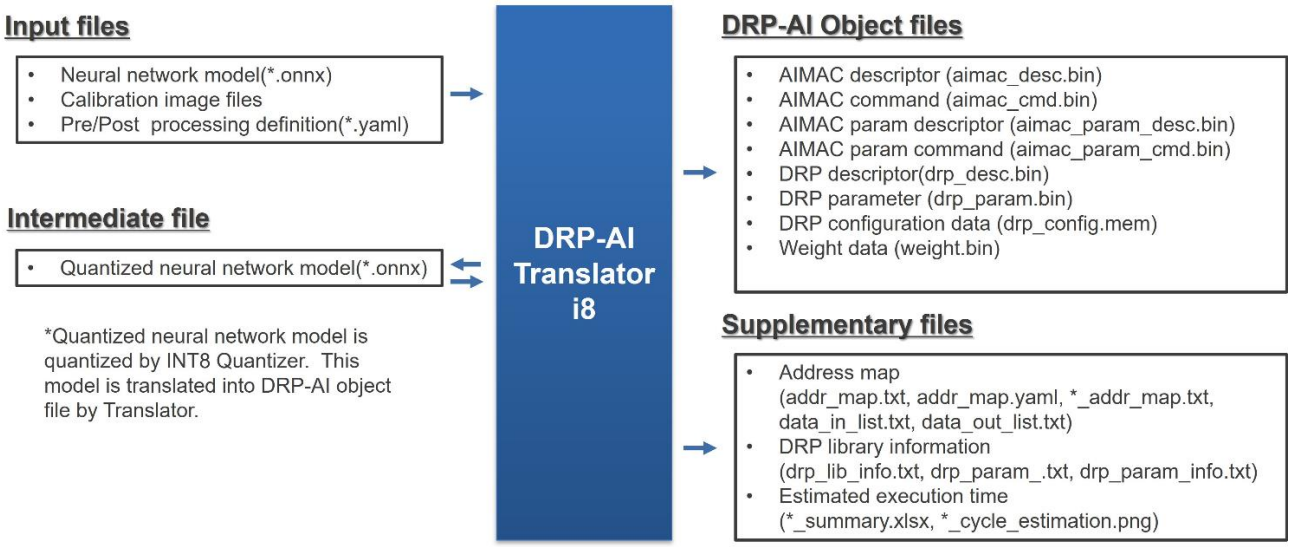
- Knowledge of command line I/F for Ubuntu OS and Python.
- Basic knowledge of the structure of convolutional neural networks and their input/output I/F.
- Basic knowledge about preprocessing and postprocessing of neural networks and applications including them.
- Knowledge of the ONNX format and its operators.
- Basic knowledge of the yaml format.
- Knowledge of target device hardware specifications, especially DRP-AI and address space.

1.3. List of Abbreviations and Acronyms

Abbreviation	Full Form
DRP-AI	Name of the Renesas architecture that accelerates the inference of neural networks
AIMAC	Components of the DRP-AI. Mainly used for the operation of the convolutional layer
DRP	Components of the DRP-AI. Mainly used for the operation of layers other than the above and pre and postprocessing
ONNX	A data format for representing neural network models. In this tool, it is used as an input format for neural networks. ONNX: Open Neural Network Exchange
YAML	A data format for serializing structured data and objects into strings. In this tool, it is used to define pre and postprocessing

2. Overview

The DRP-AI Translator i8 is a software tool for translating neural network models of ONNX format into DRP-AI object files for target LSI.
The input and output files of the DRP-AI translator are as follows:



- Input files
  - Neural network model (*.onnx)
  - Calibration image files
  - Pre and Postprocessing definition (*.yaml)
- Intermediate file
  - Quantized Neural network model (*.onnx)
- Output files
  - DRP-AI object files
    - AIMAC descriptor and command files (aimac_cmd.bin/aimac_desc.bin /aimac_param_cmd.bin/aimac_param_desc.bin)
    - DRP descriptor (drp_desc.bin)
    - DRP parameter (drp_param.bin)
    - DRP configuration data (drp_config.mem)
    - Weight data (weight.bin)
  - Supplementary files
    - Address map (addr_map.yaml, addr_map.txt, data_out_list.txt, data_in_list.txt)
    - DRP Library information (drp_lib_info.txt, drp_param.txt and drp_param_info.txt)
    - Estimated execution time (*_summary.xlsx, *_cycle_estimation.png)

2.1. Input Files

Please refer to 4. Input Files for the specifications of supported ONNX operators and Pre and Post-processing definition files. For the details of images for calibration, please refer to the User's Manual of INT8 Quantizer.

2.2. Output Files

DRP-AI object files

- AIMAC descriptor (aimac_cmd.bin/aimac_desc.bin/aimac_param_cmd.bin /aimac_param_desc.bin)
  - Descriptors for DRP-AI's MAC (AIMAC) unit
  - Location address: Start address of **aimac_cmd**/**aimac_desc**/**aimac_param_cmd** /**aimac_param_desc** in addr_map.yaml
- DRP descriptor (drp_desc.bin)
  - Descriptor for DRP-AI's DRP unit
  - Location address: Start address of **drp_desc** in addr_map.yaml
- DRP parameter (drp_param.bin)
  - Parameter referenced by the DRP library
  - Location address: Start address of **drp_param** in addr_map.yaml
- Weight data (weight.bin)
  - Weight data referenced by AIMAC
  - Location address: Start address of **weight** in addr_map.yaml
- DRP configuration data (drp_config.mem)
  - Data to configure the DRP
  - Location address: Start address of **drp_config** in addr_map.yaml

Supplementary files

- Address map (addr_map.yaml, addr_map.txt)
  - Address map including size information of DRP-AI object files, input/output data area,

etc.

- Estimated execution time (*_summary.xlsx)
  - Execution time summary sheet
- Internal files (These are not user related)
  - DRP library mapped information (drp_lib_info.txt)
  - DRP parameters information (drp_param.txt and drp_param_info.txt)

3. Installation

3.1. Prerequisites

Item	Requirement
CPU	x86_64
Memory Size	>= 6GB
OS	Ubuntu 20.04
Python	3.8

3.2. Installation

3.2.1. Installation Procedure

Launch the DRP-AI Translator package as following procedure:

```
$ apt update && DEBIAN_FRONTEND=noninteractive \
  apt install -y git wget unzip\
  curl libboost-all-dev libeigen3-dev \
  build-essential python3-pip libgl1-mesa-dev
$ chmod +x DRP-AI_Translator_i8-v1.xx-Linux-x86_64-Install
$ ./DRP-AI_Translator_i8-v1.xx-Linux-x86_64-Install
```

Add environment variables to INT8 Quantizer:

```
$ export PYTHONPATH=<Install dir>/DRP-AI_Translator_i8/drpAI_Quantizer:${PYTHONPATH}
```

The following directory structure will be generated in the current directory:

```
DRP-AI_Translator_i8
 ├──── GettingStarted
 │     ├──── README.md
 │     ├──── tutorials
 │     ├──── how-to
 │     ├──── ...
 ├──── onnx_models
 │     ├──── ...
 ├──── drpAI_Quantizer
 │     ├──── ...
 └──── translator
       ├──── DRP-AI_Translator
       ├──── UserConfig
       ├──── output (Generated after translation)
       └──── run_Translator_v2h.sh
```

Directory name, File name	Contents
GettingStarted	Guide for DRP-AI Translator i8
GettingStarted/README.md	Overview of Getting Started
GettingStarted/tutorials	Translation tutorial

GettingStarted/how-to	Guide for other onnx models
onnx_models/	Sample ONNX models
drpAI_Quantizer/	Root directory of INT8 Quantizer
translator/	Root directory of Translator
translator/DRP-AI_translator	Contains the DRP-AI translator program
translator/UserConfig	Sample Pre and Postprocessing definition file
translator/output	The output of the DRP-AI translator is stored in this directory Note: This directory will be generated after translation
translator/run_Translator_v2h.sh	Shell scripts for DRP-AI translator

Getting Started helps you learn how to use DRP-AI Translator i8. If you use Translator i8 for the first time, please refer to *GettingStarted/README.md*.

3.2.2. Dependencies

Dependencies are automatically installed by the installer.
The following is a list of main dependencies:

Package	Requirement Ver.
libgl1-mesa-dev	>=20.0.8-0ubuntu1
numpy	1.24.4
pillow	10.1.0
matplotlib	3.7.4
protobuf	4.25.1
onnx	1.14.0
packaging	23.2
pandas	2.0.3
openpyxl	3.1.2
pyyaml	6.0.1
networkx	3.1
onnxoptimizer	0.3.13
onnxruntime	1.14.1
opencv-python	4.8.1.78
sympy	1.12
scipy	1.10.1

4. Input Files

4.1. Neural Network Model

The DRP-AI Translator translates neural network models of ONNX format into DRP-AI object files for target LSI.

4.1.1. ONNX Operators

- The DRP-AI translator supports the following ONNX operators. See the “Supported Values” column for limitation on each.
- The following ONNX versions are supported:
  - ONNX version : 1.12.0
  - Opset version : 13

Supported ONNX Operator List

Operator	Category	Name	Supported Values
Add	-	-	Support same size addition
ArgMax	Inputs	Inputs: din	Support 4D input shape. shape(din) : (1, ich, H, W)
	Attributes	keepdifms	1
	Attributes	axis	1/2/3
	Attributes	select_last_index	Support only "0"
	Outputs	Outputs: dout	Support 4D output shape. dtype : uint16 (int64 is automatically changed to uint16)
ArgMin	Inputs	Inputs: din	Support 4D input shape. shape(din) : (1, ich, H, W)
	Attributes	keepdifms	1
	Attributes	axis	1/2/3
	Attributes	select_last_index	Support only "0"
	Outputs	Outputs: dout	Support 4D output shape. dtype : uint16 (int64 is automatically changed to uint16)
AveragePool	Attributes Inputs	Attributes kernel_shape strides pads Inputs X : (1, ich, H, W)	See section 4.1.2 for the detail
	Attributes	auto_pads	default | “NOTSET”
	Attributes	pads	all 0
	Attributes	ceil_mode	(default) | 0
	Attributes	count_include_pad	(default) | 0 | 1
	-	-	Support GlobalAveragePool equivalent case, i.e. where kernel_shape is the same as the shape of the feature map
BatchNormalization	-	-	-
Cast	-	-	Support if this operator can be removed by constant folding optimization. Please use graph optimizer. See section 6.7 for the detail
Clip	Attributes	max	6
	Attributes	min	0

Concat	-	-	Support handling of constant parameters embedded in ONNX
	Inputs Attributes	Attributes : axis Inputs : input0, input1, ...	- axis == 1 - input0: (1, ch0, H, W) input1: (1, ch1, H, W) ...
Constant	-	-	Support if this operator can be removed by constant folding optimization. Please use graph optimizer. See section 6.7 for the detail
ConstantOfShape	-	-	Support if this operator can be removed by constant folding optimization. Please use graph optimizer. See section 6.7 for the detail
Conv	Attributes	kernel_shape	See section 4.1.2 for the detail
Div	-	-	Support the case that are equivalent to BatchNormalization in combination with Add or Sub. Support the case that can be fused to previous BatchNormalization.
Dropout	-	-	Please use graph optimizer. See section 6.7 for the detail
Flatten	-	-	Support handling of constant parameters embedded in ONNX
	Inputs Attributes	Inputs: input Attributes: axis	Support reshaping feature map from 4dim to 2dim just before fully connected layer - shape(input) : (1, ch, H, W) - axis : 1 or (default)
Gather	-	-	Support if this operator can be removed by constant folding optimization. Please use graph optimizer. See section 6.7 for the detail
Gemm	Attributes	alpha	(default) | 1.0
	Attributes	beta	(default) | 1.0
	Attributes	transA	(default) | 0
	Attributes	transB	(default) | 0 | 1
	-	-	Support fully connected layer equivalent case
GlobalAveragePool	Inputs	X: (1, ich, H, W)	–
(HardSwish)	-	-	See section 4.1.2 for the detail
Identity	-	-	Please use graph optimizer. See section 6.7 for the detail
LeakyRelu	-	-	-
MatMul	-	-	Support fully connected layer equivalent case Please use graph optimizer. See section 6.7 for the detail
MaxPool	Attributes Inputs	Attributes kernel_shape strides	See section 4.1.2 for the detail

		pads Inputs X : (1, ich, H, W)	
(Mish)	-	-	See section 4.1.2 for the detail
Mul	-	-	Support the case that are equivalent to BatchNormalization in combination with Add or Sub. e.g. Mul > Add
	-	-	Work as a part of “Swish(Sigmoid > Mul)” operation. See section 4.1.2 for the detail
Pad	-	-	Support the case that can be fused to the next Conv, MaxPool and AveragePool. Note: Must meet the constraints of the pad attribute on the destination to be fused
PRelu	-	-	-
ReduceMean	-	-	Support GlobalAveragePool equivalent case. Please use graph optimizer. See section 6.7 for the detail
Relu	-	-	-
(Relu6)	-	-	See section 4.1.2 for the detail
Reshape	-	-	Support handling of constant parameters embedded in ONNX Support Flatten equivalent case
	Inputs Attributes	Inputs : data Attributes: perm	Support reorg(Darknet impl) component case
Resize	Inputs	X: (1, ich, H, W)	W(in) <= 4096 && W(out) <= 4096 H(in) <= 4096 && H(out) <= 4096
	Inputs	scales: (1, ich, H, W)	[1, 1, y, x]
	Inputs	size: (1, ich, H, W)	[1, CH(out), H(out), W(out)]
	Attributes	coordinate_transformation_mode	(default) | half_pixel | pytorch_half_pixel | align_corners | asymmetric
	Attributes	exclude_outside	(defalut) | 0
	Attributes	mode	(defalut) | nearest | linear
	Attributes	nearest_mode	floor
Selu	-	-	-
Sigmoid	-	-	-
Slice	-	-	Support only "Focus" operation See section 4.1.3 for the detail
Softmax	Inputs	X: (1, ch) or X : (1, ch, H, W)	axis1<=ch<=16384

		== 1 (ch direction only)	
Sofplus	-	-	-
SpaceToDepth	Attributes	blocksize	blocksize == 2
Squeeze	-	-	Support if this operator can be removed by constant folding optimization. Please use graph optimizer. See section 6.7 for the detail
Sub	-	-	Support the case that are equivalent to BatchNormalization in combination with Mul or Div. e.g. Div > Sub
Sum	Inputs	Inputs: data_0, data_1, …	Support same size summation. shape(data_0): (1, ch, H, W) shape(data_1): (1, ch, H, W) ... Please use graph optimizer. See section 6.7 for the detail
(Swish)	-	-	See section 4.1.2 for the detail
Tanh	-	-	-
Transpose	-	-	Support handling of constant parameters embedded in ONNX
	Inputs Attributes	Inputs : data Attributes: perm	Support reorg(Darknet impl) component case
Unsqueeze	-	-	Support if this operator can be removed by constant folding optimization. Please use graph optimizer. See section 6.7 for the detail
Upsample	-	-	Support condition is the same as "Resize" operator Please use graph optimizer. See section 6.7 for the detail

Note: pads = [left, right, top, bottom]

4.1.2.Detail of supported attributes

The DRP-AI translator supports the following attributes/functions.

Function	Category	Onnx operator(s)	Note/Supported Values
Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [1, 1] strides : [1, 1] or [2, 2] pads : [0, 0, 0, 0] group : 1 dilation : 1
Convolution	Attributes of	Conv	Attributes :

	Conv operator		kernel_shape : [3, 3] strides : [1, 1] or [2, 2] pads : [l,r,t,b] (pad size <=1) group : 1 dilation : 1
Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [4, 4] strides : [1, 1] pads : [l,r,t,b] (pad size <=2) group : 1 dilation : 1
Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [5, 5] strides : [1, 1] or [2, 2] pads : [l,r,t,b] (pad size <=2) group : 1 dilation : 1
Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [6, 6] strides : [1, 1] pads : [l,r,t,b] (pad size <=3) group : 1 dilation : 1
Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [7, 7] strides : [1, 1] or [2, 2] pads : [l,r,t,b] (pad size <=3) group : 1 dilation : 1
Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [9, 9] strides : [1, 1] or [2, 2] pads : [l,r,t,b] (pad size <=4) group : 1 dilation : 1
Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [11, 11] strides : [1, 1]

			pads : [l,r,t,b] (pad size <=5) group : 1 dilation : 1
Depthwise Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [3, 3] strides : [1, 1] or [2, 2] pads : [l,r,t,b] (pad size <=1) group : n dilation : 1 * n = ich = och
Depthwise Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [5, 5] strides : [1, 1] or [2, 2] pads : [l,r,t,b] (pad size <=2) group : n dilation : 1 * n = ich = och
Depthwise Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [7, 7] strides : [1, 1] or [2, 2] pads : [l,r,t,b] (pad size <=3) group : n dilation : 1 * n = ich = och
Dilated Convolution	Attributes of Conv operator	Conv	Attributes : kernel_shape : [3, 3] strides : [1, 1] pads : [l,r,t,b] (pad size <=1) group : 1 dilation : n * 1 < n <= 36
MaxPooling	Attributes of MaxPool operator	MaxPool	Attributes : kernel : [1,1], strides : [2,2], pads : [0,0,0,0] kernel : [2,2], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<1) kernel : [3,3], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<1) kernel : [4,4], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<2) kernel : [5,5], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<2) kernel : [6,6], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<3)

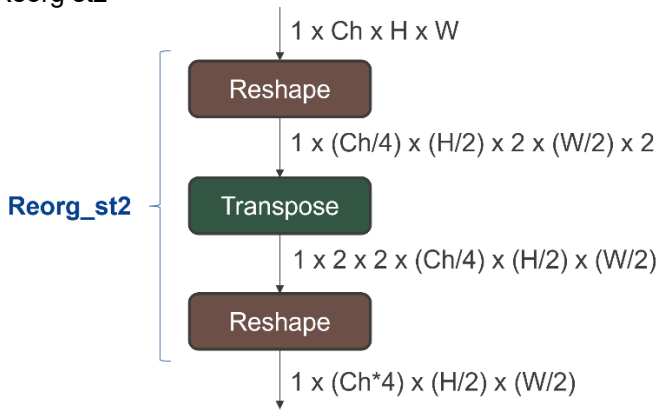
			kernel : [7,7], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<3) kernel : [8,8], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<4) kernel : [9,9], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<4) kernel : [10,10], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<5) kernel : [11,11], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<5) kernel : [12,12], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<6) kernel : [13,13], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<6) kernel : [14,14], strides : [1,1]or[2,2], pads : [l,r,t,b] (pad size =<7) kernel : [15,15] strides : [1,1]or[2,2] pads : [l,r,t,b] (pad size =<7)
AveragePooling	Attributes of AveragePool operator	AveragePool	Attributes : kernel_shape : [n, n] * 2 <= n <= 15 strides : [1, 1] or [2, 2] pads : 0, 0, 0, 0 count_include_pad=0 or 1
AveragePooling	Attributes of AveragePool operator	AveragePool	Attributes : kernel_shape : [2, 2] strides : [2, 2] pads : 0, 1, 0, 1 count_include_pad=1
AveragePooling	Attributes of AveragePool operator	AveragePool	Attributes : kernel_shape : [3, 3] strides : [1, 1] pads : 0, 1, 0, 1 or 1, 0, 1, 0 or 1, 1, 1, 1 count_include_pad=1
AveragePooling	Attributes of AveragePool operator	AveragePool	Attributes : kernel_shape : [7, 7] strides : [1, 1] pads : 0, 3, 0, 3 or 3, 0, 3, 0 or 3, 3, 3, 3 count_include_pad=1
HardSwish	Group of operator	Add, Clip, Div, Mul	Note: HardSwish formula is f(x) = x * ReLu6(x+3)/6, and in the onnx graph it is represented by combination of Add, Clip, Div and and Mul operators
Mish	Group of operator	Softplus, Tanh, Mul	Note: Mish formula is f(x) = x * Tanh(Softplus(x)), and in the onnx graph it is represented by combination of Softplus, Tanh and Mul operators
Relu6	Specific attributes	Clip	Attributes: min = 0 , max = 6
Swish	Group of operator	Sigmoid, Mul	Note: Swish formula is f(x) = x * sigmoid(x), and in the onnx graph it is represented by combination of Sigmoid and Mul operators

Reorg_st2	Group of operator	Reshape, Transpose	See section 4.1.3 for the detail
Focus	Group of operator	Slice	See section 4.1.3 for the detail

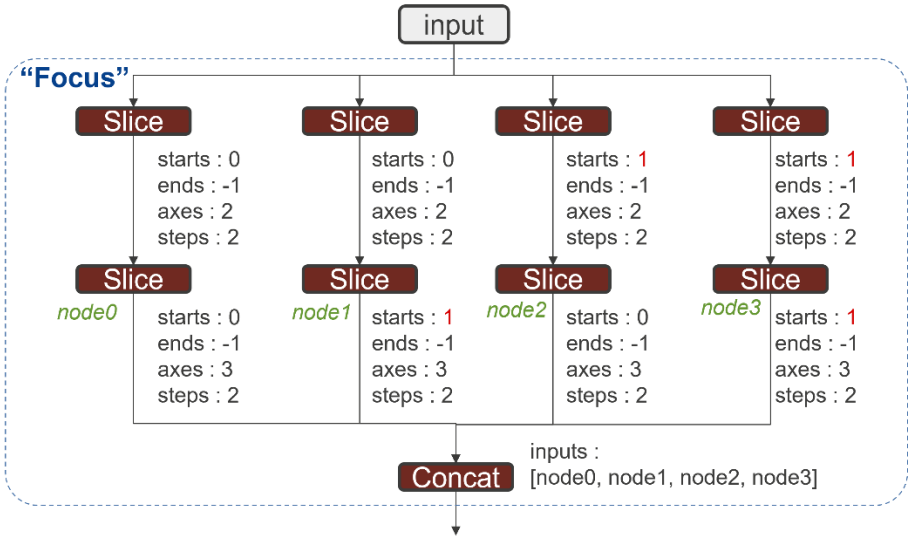
Note: pads = [left, right, top, bottom]

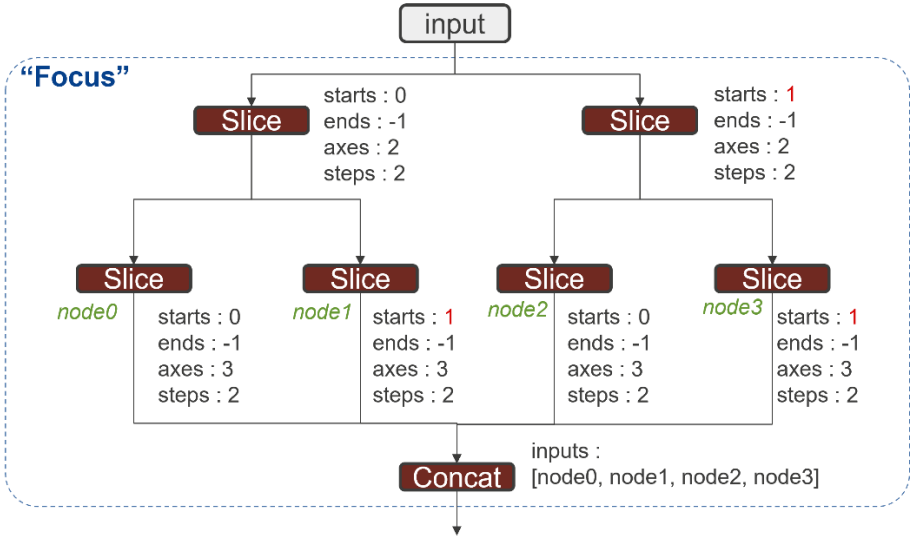
4.1.3.Supported special graph structure

- Reorg st2



- Focus


input
"Focus"
Slice
Slice
Slice
Slice
starts : 0
ends : -1
axes : 2
steps : 2
starts : 0
ends : -1
axes : 2
steps : 2
starts : 1
ends : -1
axes : 2
steps : 2
starts : 1
ends : -1
axes : 2
steps : 2
Slice
Slice
Slice
Slice
node0
node1
node2
node3
starts : 0
ends : -1
axes : 3
steps : 2
starts : 1
ends : -1
axes : 3
steps : 2
starts : 0
ends : -1
axes : 3
steps : 2
starts : 1
ends : -1
axes : 3
steps : 2
Concat
inputs :
[node0, node1, node2, node3]


input
"Focus"
Slice
Slice
starts : 0
ends : -1
axes : 2
steps : 2
starts : 1
ends : -1
axes : 2
steps : 2
Slice
Slice
Slice
Slice
node0
node1
node2
node3
starts : 0
ends : -1
axes : 3
steps : 2
starts : 1
ends : -1
axes : 3
steps : 2
starts : 0
ends : -1
axes : 3
steps : 2
starts : 1
ends : -1
axes : 3
steps : 2
Concat
inputs :
[node0, node1, node2, node3]

4.2. Pre and Postprocessing Definition

Preprocessing and postprocessing can be added before and after the ONNX model given as an input file, and the whole can be translated by the DRP-AI translator.
The preprocessing and postprocessing supported by the DRP-AI translator are as follows.

4.2.1. Preprocessing Types

- Transpose from CHW to HWC
- Convert from YUV to RGB
- Convert from YUV/RGB/BGR to Grayscale
- Resize
- Cast to fp16
- Normalisation
- Memory copy
- Crop

4.2.2. Postprocessing Types

- Transpose from HWC to CHW
- Softmax
- Cast fp16 and fp32
- Memory copy
- Argmin and Argmax

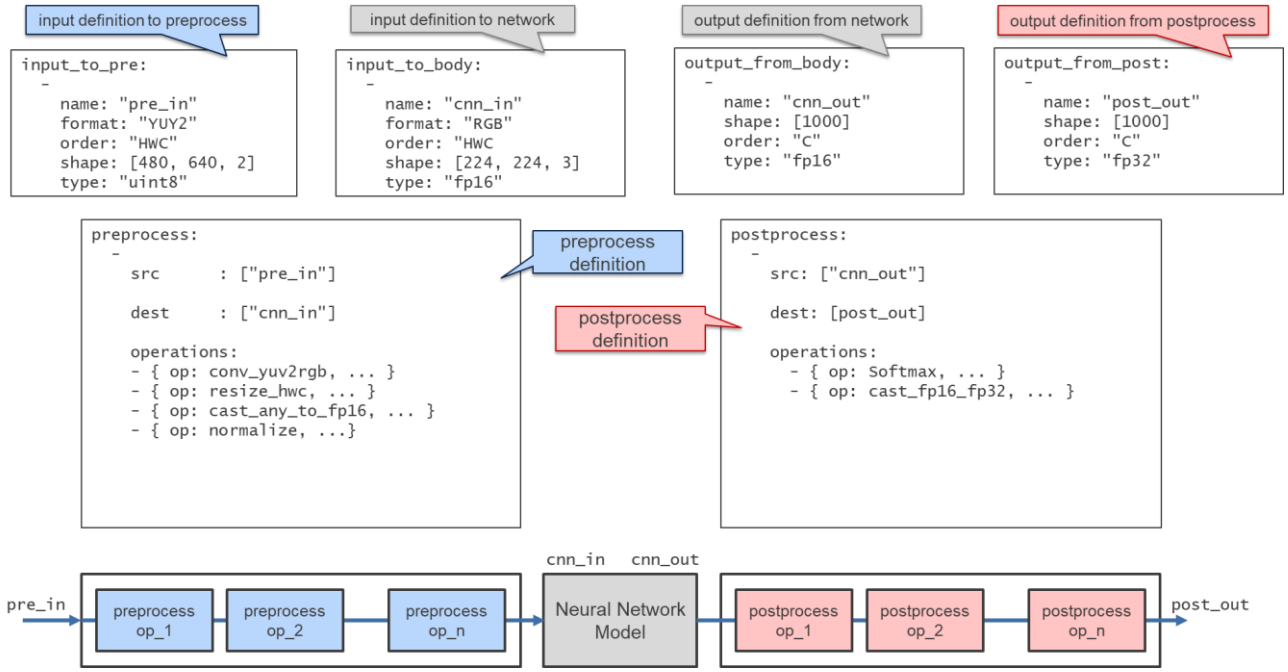
4.2.3. Pre and Postprocessing Definition Format

Pre and Postprocessing are defined in a YAML file. See 5. Details of Pre and Postprocessing Definition for details.

4.3. Address Map Definition

Defines the address to place the DRP-AI object files in memory space of target LSI. Please add the start physical address to a argument of --s_addr. See 7. How to Run for the detail.

5. Details of Pre and Postprocessing Definition



5.1. How to Describe Pre and Postprocessing Definition

- Put the following keys in the top layer with mapping.
  - input_to_pre
  - input_to_body
  - output_from_body
  - output_from_post
  - preprocess
  - postprocess
- Describe the value of each key with sequence
- The order of each key is arbitrary
- The top layer has the following description style:

```
input_to_pre:      [...]
input_to_body:     [...]
output_from_body:  [...]
output_from_post:  [...]
preprocess:        [...]
postprocess:       [...]
```

5.2. How to Describe Keys

input_to_pre

- A key to define the input to the preprocessing part (must)
- In the case of multiple inputs, describe each value with sequence
- Describe the elements that make up the value with mapping as follows:

```
input_to_pre:
```

```
  [{ name: "pre_in", shape: [480, 640, 2], order: "HWC", format: "YUY2", type: "uint8"
 }]
```

or

```
input_to_pre:
  -
    name: "pre_in"
    shape: [480, 640, 2]
    order: "HWC"
    format: "YUY2"
    type: "uint8"
```

- The key of element and possible value are as follows:

Key	Type	Selectable Values
name	str	Any string | (default)
format	str	RGB | BGR | YUY2 | GRAY
order	str	HWC | CHW
shape	int []	...
type	str	uint8 | fp16 | fp32

- name (str)
  - Specify the input data name
- format (str)
  - Specify the input image data format
  - RGB, BGR, YUY2, and GRAY can be specified
- order (str)
  - Specifies the meaning of each axis that corresponds to the shape of the input data
  - HWC : Height, Width, Channel
  - CHW : Channel, Height, Width
- shape (int [])
  - Specifies the input data shape
    e.g. [Height, Width, Channel]
    e.g. [Channel, Height, Width]
  - In case of YUY2, only 2 channels can be specified
- type (str)
  - Specifies the input data format
  - uint8, fp16 and fp32 can be specified

input_to_body

- A key to define the input to the main body of the neural network (must)
- In the case of multiple inputs, describe each value with sequence
- Describe the elements that make up the value with mapping as follows:

```
input_to_body:
  [{ name: "data", shape: [224, 224, 3], order: "HWC", format: "RGB", type: "fp16" }]
```

or

```
input_to_body:
  -
    name: "data"
```

```
    shape: [224, 224, 3]
    order: "HWC"
    format: "RGB"
    type: "fp16"
```

- The key of element and possible value are as follows:

Key	Type	Selectable Values
name	str	Any string | (default)
shape	int []	...
order	str	HWC
format	str	RGB | BGR | GRAY
type	str	fp16

- name (str)
  - Specify the input data name
- shape (int [])
  - Specifies the input data shape
- order (str)
  - Specifies the meaning of each axis that corresponds to the shape of the input data
  - HWC : Height, Width, Channel
- format (str)
  - Specify the input image data format
  - RGB, BGR, and GRAY can be specified
- type (str)
  - Specifies the input data format
  - fp16 can be specified

output_from_body

- A key to define the output from the main body of the neural network (must)
- In the case of multiple outputs, describe each value with sequence
- Describe the elements that make up the value with mapping as follows:

```
output_from_body:
  [{ name: "cnn_out", shape: [1000], order: "C", type: "fp16" }]
```

or

```
output_from_body:
  -
    name: "cnn_out"
    shape: [1000]
    order: "C"
    type: "fp16"
```

- The key of element and possible value are as follows:

Key	Type	Selectable Values
name	str	Any string | (default)
shape	int []	...
shape_post	int []	...

order	str	HWC | C
type	str	fp16

- name (str)
  - Specify the output data name
- shape (int [])
  - Specifies the output data shape
- shape_post (int [])
  - Specify the shape to pass the output data to the postprocessing part. By describing value with sequence, it is possible to pass one output data by dividing it into multiple postprocessing inputs.

```
output_from_body:
  -
    name: "cnn_out"
    order: "HWC"
    shape: [77, 46, 57]
    shape_post:
      - { name: "heatmap", shape: [77, 46, 19]}
      - { name: "paf",     shape: [77, 46, 38]}
```

- order (str)
  - Specifies the meaning of each axis that corresponds to the shape of the output data
    - HWC : Height, Width, Channel
    - C : Channel
- type (str)
  - Specifies the output data format
  - fp16 can be specified

output_from_post

- A key to define the output from the postprocessing part (must)
- In the case of multiple outputs, describe each value with sequence
- Describe the elements that make up the value with mapping as follows:

```
output_from_post:
    [{ name: "post_out", shape: [1000], order "C", type: "fp32" }]
```

or

```
output_from_post:
  -
    name: "post_out"
    shape: [1000]
    order: "C"
    type: "fp32"
```

- The key of element and possible value are as follows:

Key	Type	Selectable Values
name	str	Any string | (default)
shape	int []	...
order	str	HWC | CHW | C

type	str	fp16 | fp32 | uint8

- name (str)
  - Specify the output data name
- shape (int [])
  - Specifies the output data shape
- order (str)
  - Specifies the meaning of each axis that corresponds to the shape of the output data
  - HWC : Height, Width, Channel
  - CHW : Channel, Height, Width
  - C : Channel
- type (str)
  - Specifies the output data format
  - fp16, fp32 and uint8 can be specified

preprocess

- A key to define the preprocessing (must)
- In the case of multiple inputs, describe each value with sequence
- Describe the elements that make up the value with mapping as follows. Describe the contents of the required preprocessing in sequence as the value of "operations" key

```
preprocess:
  -
    src: ["pre_in"]

    dest: ["data"]

   operations:
    - { op: conv_yuv2rgb, ... }
    - { op: resize_hwc, ... }
    - { op: cast_any_to_fp16, ... }
    - { op: normalize, ...}
    - { op: memcopy, ... }
```

- The key of element and possible value are as follows:

Key	Type	Selectable Values
src	List of str	Any string | (default)
dest	List of str	Any string | (default)
operations	List of Mapping	...

- src (List of str)
  - Specify the input data name of the preprocessing part with list
- dest (List of str)
  - Specify the output data name from the preprocessing part in list
- operations (List of Mapping)
  - Specify preprocessing content in a sequence for each process
  - Describe the elements that make up the value of each process with mapping
  - The key of element and possible value are as follows:

Key	Type

op	str
param	Mapping

- op (str)
  - Specify the operation name
- param (Mapping)
  - Specify the parameter for the operation with mapping
  - Describe the parameter name and value in pairs as follows:
    - "parameter1" -> "value1"
    - "parameter2" -> "value2"
    - ...

postprocess

- A key to define the postprocessing (must)
- In the case of multiple outputs, describe each value with sequence
- Postprocessing means:
  1. Read the "output_from_body",
  2. Perform some arithmetic operations, and
  3. Returns the operation result as "output_from_post"
- Describe the elements that make up the value with mapping as follows. Describe the contents of the required postprocessing in sequence as the value of "operations" key

```
postprocess:
  -
    src: ["cnn_out"]

    dest: ["post_out"]

    operations:
    - { op: softmax, ... }
```

- The key of element and possible value are as follows:

Key	Type	Selectable Values
src	List of str	Any string | (default)
dest	List of str	Any string | (default)
operations	List of Mapping	...

- src (List of str)
  - Specify the input data name of the postprocessing part with list
- dest (List of str)
  - Specify the output data name from the postprocessing part in list
- operations (List of Mapping)
  - Specify postprocessing content in a sequence for each process
  - Describe the elements that make up the value of each process with mapping
  - The key of element and possible value are as follows:

Key	Type
op	str
param	Mapping

- op (str)
  - Specify the operation name
- param (Mapping)
  - Specify the parameter for the operation with mapping
  - Describe the parameter name and value in pairs as follows:
    - "parameter1" -> "value1"
    - "parameter2" -> "value2"
    - ...

5.3. How to Describe Preprocessing Definition

5.3.1. Preprocessing List

Name	Input Type	Output Type	Input Order	Output Order	Input Data Number	Output Data Number
transpose	uint8	uint8	CHW	HWC	1	1
conv_yuv2rgb	uint8	uint8	HWC	HWC	1	1
conv_x2gray	uint8	uint8	HWC	HWC	1	1
crop	uint8 fp16	uint8 fp16	HWC CHW	HWC	1	1
resize_hwc	uint8 fp16	<-	HWC	HWC	1	1
cast_any_to_fp16	uint8 fp16 fp32	fp16	HWC	<-	1	1
normalize	fp16	fp16	HWC	HWC	1	1
memcopy	fp16	fp16	Any	<-	1	1

- Preprocessing definitions (*.yaml) must be specified in the order shown in the table above
  Note: Can't specify both **`transpose`** and **`conv_yuv2rgb/conv_x2gray`**
- ON/OFF of each process can be specified. If the process is written in YAML, it means on, otherwise it means off.
  Note: Specify **`memcopy`** if no preprocessing is required.
- The format and the order of dimension of input data in each preprocessing operation must match before and after. That is,
  - The format of the output data of the previous operation and the format of the input data to the next operation must match
  - The dimension order of the output data of the previous operation and the dimension order of the input data to the next operation must match

5.3.2. Preprocessing Parameters

transpose(WORD_SIZE, IS_CHW2HWC)

- Transpose from CHW to HWC
- Parameters:
  - WORD_SIZE (int)
    - Specify the data size
    - 0 (="1Byte")
  - IS_CHW2HWC

        - 1 (="CHW to HWC")
- Limitation:
    - shape_in:
        - width <= 65535
        - height <= 65535
        - ch <= 65535
    - shape_out:
        - width == width of shape_in
        - height == height of shape_in
        - ch == ch of shape_in
- Example

```
op: transpose
param:
  WORD_SIZE: 0    # 1Byte
  IS_CHW2HWC: 1   # CHW to HWC
```

conv_yuv2rgb(DOUT_RGB_FORMAT)

- Convert from YUV to RGB
- Parameters:
    - DIN_YUV_FORMAT (int)
        - Specify input data format

Value	Description	Format to be specifiled in input_to_pre
0	"Y_0UY_1V"("YUY2")	YUY2
1	"Y_0VY_1U"	YUY2
2	"UY_0VY_1"	YUY2
3	"VUY_1Y_0"	YUY2
4096	"Y_0UY_1V"("YUY2")	YUY2
4097	"UY_0VY_1"	YUY2
4098	"YV12"	YUY2
4099	"IYUV"	YUY2
4100	"NV12"	YUY2
4101	"NV21"	YUY2
4102	"IMC1"	RGB
4103	"IMC2"	YUY2
4104	"IMC3"	RGB
4105	"IMC4"	YUY2

        - Default value is 0 (="Y_0UY_1V"("YUY2"))
    - DOUT_RGB_FORMAT (int)
        - Specify output data format
        - 0 (="RGB") or 1 (= "BGR")
        - Default case is 0 (="RGB")
- Limitation:
    - shape_in:
        - width % 2 == 0
        - 4 <= width <= 65535
        - 5 <= height <= 65535
    - shape_out:
        - width == width of shape_in
        - height == height of shape_in

- Example

```
op: conv_yuv2rgb
param:
  DIN_YUV_FORMAT: 0 # "YUY2"
  DOUT_RGB_FORMAT: 0  # "RGB"
```

conv_x2gray(DIN_FORMAT)

- Convert from YUV/RGB/BGR to Grayscale
- Parameters:
  - DIN_FORMAT (int)
    - Specify input data format
    - 0 (="Y_0UY_1V"("YUY2")) or 1 (= "Y_0VY_1U") or 2 (= "UY_0VY_1") or 3 (= "VUY_1Y_0") or 4096 (= "RGB24") or 4097 (= "BGR24")
- Limitation:
  - shape_in:
    - width % 2 == 0 when DIN_FORMAT != 4096 or 4097
- Example

```
op: conv_x2gray
param:
  DIN_YUV_FORMAT: 0 # "YUY2"
```

resize_hwc(RESIZE_ALG, DATA_TYPE, shape_out)

- Resize
- Parameters:
  - RESIZE_ALG (int)
    - Specify the interpolation algorithm
    - 0 (= "Nearest") or 1 (= "Bilinear")
    - Default case is 0 (= "Nearest")
  - DATA_TYPE (int)
    - Specify input data type
    - 0 (= "uint8") or 1 (= "fp16")
    - Default case is 0 (= "uint8")
  - shape_out (List of int)
    - [resized height, resized width]
- Limitation:
  - shape_in:
    - (width > 2) && (height > 2)
    - ch <= 4096
  - shape_out:
    - (width > 2) && (height > 2)
    - ch == ch of shape_in
- Example

```
op: resize_hwc
param:
  RESIZE_ALG: 1    # "Bilinear"
  DATA_TYPE: 0     # "uint8"
  shape_out: [224, 224]
```

cast_any_to_fp16(DIN_FORMAT)

- Cast to fp16
- Parameters:
  - DIN_FORMAT (int)
    - Specify input data format
    - 0 (= "uint8") or 1 (= "fp16") or 2 (= "fp32")
    - Default case is 0 (= "uint8")
- Limitation:
  - shape_in:
    - (width * ch) < 32'hFFFF_FFFF
  - shape_out:
    - width == width of shape_in
    - height == height of shape_in
    - ch == ch of shape_in
- Example

```
op: cast_any_to_fp16
param:
  DIN_FORMAT       : 0    # UINT8
```

normalize(DOUT_RGB_ORDER, cof_add, cof_mul)

- Normalization
- **Dout** is calculated using the parameters **cof_add** and **cof_mul** for the input **Din** as follows:

```
Dout = (Din + cof_add) * cof_mul
```

On the other hand, the following normalization may be used for the training of the neural network by the image data:

```
range = 255
img = img / range
img = (img - mean) / stdev
```

In the above case, find cof_add and cof_mul from mean and stdev using the following conversion formula:

```
cof_add = -(mean * range)
cof_mul = 1/(stdev * range)
```

- Parameters:
  - DOUT_RGB_ORDER (int)
    - Specify output data order
    - 0 : Output RGB order = Input RGB order
    - 1 : Output RGB order = Swapped R and B channels of input RGB order
      - If input order = "RGB", output order = "BGR"
      - If input order = "BGR", output order = "RGB"
      - "cast_any_to_fp16" is required before "normalize" operation. The input data format to "cast_any_to_fp16" must be uint8 only
  - cof_add (float [])

  - cof_mul (float [])
    Note: If DOUT_RGB_ORDER=1, cof_mul and cof_add must be specified in the order after the swap
- Limitation:
  - shape_out:
    - width == width of shape_in
    - height == height of shape_in
    - ch == ch of shape_in
- Example

```
op: normalize
param:
  DOUT_RGB_ORDER: 0    # Output RGB order = Input RGB order
  cof_add: [-123.675,    -116.28,   -103.53]
  cof_mul: [0.01712475, 0.017507, 0.01742919]
```

memcopy(WORD_SIZE)

- Memory copy
- Parameters:
  - WORD_SIZE(int)
  - 2: Number of bytes per word = 2
    Note: The parameter value is fixed at 2.
- Limitation:
  - shape_out:
    - width == width of shape_in
    - height == height of shape_in
    - ch == ch of shape_in
- Example

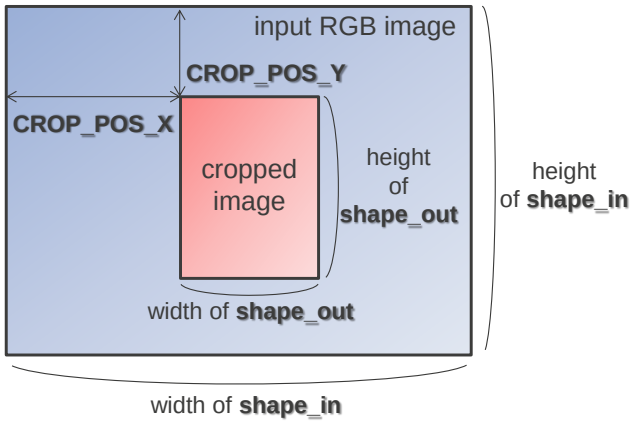
```
op: memcopy
param:
  WORD_SIZE: 2
```

crop(CROP_POS_X, CROP_POS_Y, shape_out, DATA_TYPE, DATA_FORMAT)

- Crop
  (Caution: Note that crop operation is defined differently than other operations. Follow the limitations and example described below.)
- Parameters:
  - CROP_POS_X (int)
    - Top left X coordinate of the image to be cut
  - CROP_POS_Y (int)
    - Top left Y coordinate of the image to be cut
  - shape_out (List of int)
    - [cropped height, cropped width]
  - DATA_TYPE (bool)
    - Specify input/output data type
    - 0 (=1 byte(uint8)) or 1 (=2 byte(uint16, fp16))
  - DATA_FORMAT (int)
    - Specify the order of input data
    - 0 (= "HWC") or 1(=CHW)
- Limitation:

- shape_out:
  - ch == ch of shape_in
- CROP_POS_X:
  - CROP_POS_X <= width of shape_in - 1
- CROP_POS_Y:
  - CROP_POS_Y <= height of shape_in - 1

- Example




```
op: crop
param:
  CROP_POS_X: 60
  CROP_POS_Y: 40
  shape_out : [200,200]
  DATA_TYPE: 0 # 0 : 1Byte, 1 : 2Byte
  DATA_FORMAT : 0 # 0: HWC
```

5.4. How to Describe Postprocessing Definition

5.4.1. Postprocessing List

Name	Input Type	Output Type	Input Order	Output Order	Input Data Number	Output Data Number
transpose	fp16	fp16	HWC	CHW	1	1
softmax	fp16	fp16 fp32	C	C	1	1
cast_fp16_fp32	fp16 fp32	fp32 fp16	Any	Any	1	1
argminmax	fp16	uint8 uint16	HWC CHW	<-	1	1
memcopy	fp16	fp16	Any	<-	1	1

- Postprocessing definitions (*.yaml) must be specified in the order shown in the table above
- ON/OFF of each process can be specified. If the process is written in YAML, it means on, otherwise it means off.
  Note: Specify **memcopy** if no postprocessing is required.
- The format and the order of dimension of input data in each postprocessing operation must match before and after. That is,
  - The format of the output data of the previous operation and the format of the input data to the next operation must match

- The dimension order of the output data of the previous operation and the dimension order of the input data to the next operation must match

5.4.2.Postprocessing Parameters

transpose(WORD_SIZE, IS_CHW2HWC)

- Transpose from HWC to CHW
- Parameters:
  - WORD_SIZE (int)
    - Specify the data size
    - 1 (="2Byte")
  - IS_CHW2HWC
    - 0 (="HWC to CHW")
- Limitation :
  - shape_in :
    - width <= 65535
    - height <= 65535
    - ch <= 65535
  - shape_out :
    - width == width of shape_in
    - height == height of shape_in
    - ch == ch of shape_in
- Example

```
op: transpose
param:
  WORD_SIZE: 1    # 2Byte
  IS_CHW2HWC: 0   # HWC to CHW
```

softmax

- Softmax
  - The softmax function. Consider the input as flatten data in the "C" dimension.
- Parameters:
  - DOUT_FORMAT (int)
    - Specify output data format
    - 0 (= "FP16") or 1 (= "FP32")
    - Default case is 0 (= "FP16")
- Limitation :
  - shape_in :
    - 1 <= ch <= 16384
  - shape_out :
    - ch == ch of shape_in
    - height == height of shape_in
    - width == width of shape_in
- Example

```
op: softmax
param:
  DOUT_FORMAT: 0    # FP16
```

cast_fp16_fp32(CAST_MODE)

- Cast fp16 and fp32
- Parameters:
  - CAST_MODE (int)
    - Specify input/output data format
    - 0 (= "FP16 to FP32") or 1 (= "FP32 to FP16")
    - Default case is 0 (= "FP16 to FP32")
- Limitation :
  - shape_in :
    - (width * height * ch) <=s 32'hFFFF_FFFF
  - shape_out :
    - width == width of shape_in
    - height == height of shape_in
    - ch == ch of shape_in
- Example

```
op: cast_fp16_fp32
param:
  CAST_MODE: 0    # FP16 to FP32
```

memcopy(WORD_SIZE)

- Memory copy
  See 5-3-2. Preprocessing Parameters for details.

argminmax(DIN_FORMAT, DOUT_TYPE, AXIS, ARG_MODE)

- Returns minimum /maximum value of the specified axis (channel/width/height)
- Parameters:
  - DIN_FORMAT
    - Specify input data format
    - 0 (= "HWC") or 1 (= "CHW")
  - DOUT_TYPE
    - Specify output data type
    - 0 (= "uint8") or 1 (=uint16)
  - AXIS
    - Specify the target axis
    - 0 (= "ch") or 1 (= "width") or 2 (= "height")
  - ARG_MODE
    - Specify the arguments mode
    - 0 (= "ARGMAX") or 1 (= "ARGMIN")
- Limitation:
  - shape_in:
    - AXIS == 0 && ch <=256
    - AXIS == 1 && width <=256
    - AXIS == 2 && height <=256
- Example

```
op: argminmax
param:
  DIN_FORMAT: 0    # "HWC"
  DOUT_TYPE: 0     # "uint8"
```

```
  AXIS: 1 # "width"
  ARG_MODE: 1 # "argmin"
```

5.5. Examples of Pre and Postprocessing Definition

Example 1: 1,000 classification models trained by ImageNet

Input = 224x224x3ch, Output = 1,000x1ch

```
########################################
# Input data
########################################
input_to_pre:
  -
    name: "pre_in"
    format: "YUY2"
    order: "HWC"
    shape: [480, 640, 2]
    type: "uint8"

input_to_body:
  -
    name: "cnn_in"
    format: "RGB"
    order: "HWC"
    shape: [224, 224, 3]
    type: "fp16"

########################################
# Output data
########################################
output_from_body:
  -
    name: "cnn_out"
    shape: [1000]
    order: "C"
    type: "fp16"

output_from_post:
  -
    name: "post_out"
    shape: [1000]
    order: "C"
    type: "fp32"

########################################
# Preprocess
########################################
preprocess:
  -
    src: ["pre_in"]

    dest: ["cnn_in"]

    operations:
    -
      op: conv_yuv2rgb
```

```
      param:
        DOUT_RGB_FORMAT: 0 # "RGB"
    -
      op: resize_hwc
      param:
        RESIZE_ALG: 1 # Bilinear"
        DATA_TYPE: 0  # "uint8"
        shape_out: [224, 224]
    -
      op: cast_any_to_fp16
      param:
        DIN_FORMAT: 0 # "uint8"
    -
      op: normalize
      param:
        DOUT_RGB_ORDER: 0 # Output RGB order = Input RGB order
        cof_add: [-123.675, -116.28, -103.53]
        cof_mul: [0.01712475, 0.017507, 0.01742919]

########################################
# Postprocess
########################################
postprocess:
  -
    src: ["cnn_out"]

    dest: ["post_out"]

    operations:
    -
      op: softmax
      param:
        DOUT_FORMAT: 1 # "FP32"
```

Example 2: Object recognition model trained by Pascal VOC with single scale output

Input = 416x416x3ch, Output = 13x13x125ch

```
########################################
# Input data
########################################
input_to_pre:
  -
    name: "pre_in"
    format: "YUY2"
    order: "HWC"
    shape: [480, 640, 2]
    type: "uint8"

input_to_body:
  -
    name: "cnn_in"
    format: "RGB"
    order: "HWC"
    shape: [416, 416, 3]
    type: "fp16"

########################################
# Output data
```

```
########################################
output_from_body:
  -
    name: "cnn_out"
    shape: [13, 13, 125]
    order: "HWC"
    type: "fp16"

output_from_post:
  -
    name: "post_out"
    shape: [125, 13, 13]
    order: "CHW"
    type: "fp32"

########################################
# Preprocess
########################################
preprocess:
  -
    src: ["pre_in"]

    dest: ["cnn_in"]

    operations:
    -
      op: conv_yuv2rgb
      param:
        DOUT_RGB_FORMAT: 0 # "RGB"
    -
      op: resize_hwc
      param:
        RESIZE_ALG: 1   # "Bilinear"
        DATA_TYPE: 0    # "uint8"
        shape_out: [416, 416]
    -
      op: cast_any_to_fp16
      param:
        DIN_FORMAT: 0 # "uint8"
    -
      op: normalize
      param:
        DOUT_RGB_ORDER: 0 # Output RGB order = Input RGB order
        cof_add: [0.0, 0.0, 0.0]
        cof_mul: [0.00392157, 0.00392157, 0.00392157]

########################################
# Postprocess
########################################
postprocess:
  -
    src: ["cnn_out"]

    dest: ["post_out"]

    operations:
      -
        op: transpose
        param:
```

```
          WORD_SIZE: 1  # 2Byte
          IS_CHW2HWC: 0 # HWC to CHW
      -
        op: cast_fp16_fp32
        param:
          CAST_MODE: 0 # FP32
```

Example 3: Object recognition model trained by MS-COCO with multi scale outputs

Input = 416x416x3ch, Output=13x13x255ch, 26x26x255ch, 52x52x255ch

```
########################################
# Input data
########################################
input_to_pre:
  -
    name: "camera_data"
    format: "YUY2"
    order: "HWC"
    shape: [480, 640, 2]
    type: "uint8"

input_to_body:
  -
    name: "images"
    format: "RGB"
    order: "HWC"
    shape: [416, 416, 3]
    type: "fp16"

########################################
# Output data
########################################
output_from_body:
  -
    name: "391"
    shape: [13,13, 255]
    order: "HWC"
    type: "fp16"
  -
    name: "371"
    shape: [26,26,255]
    order: "HWC"
    type: "fp16"
  -
    name: "output"
    shape: [52, 52,255]
    order: "HWC"
    type: "fp16"

output_from_post:
  -
    name: "post_out_13x13"
    shape: [13, 13, 255]
    order: "HWC"
    type: "fp32"
  -
    name: "post_out_26x26"
    shape: [26, 26, 255]
```

```
    order: "HWC"
    type: "fp32"
  -
    name: "post_out_52x52"
    shape: [52, 52, 255]
    order: "HWC"
    type: "fp32"

########################################
# Preprocess
########################################
preprocess:
  -
    src      : ["camera_data"]

    dest     : ["images"]

    operations:
    -
      op: conv_yuv2rgb
      param:
        DOUT_RGB_FORMAT: 0 # "RGB"
    -
      op: resize_hwc
      param:
        RESIZE_ALG: 1 # "Bilinear"
        DATA_TYPE: 0  # "uint8"
        shape_out: [416, 416]

    -
      op: cast_any_to_fp16
      param:
        DIN_FORMAT: 0 # "uint8"

    -
      op: normalize
      param:
        DOUT_RGB_ORDER: 0 # Output RGB order = Input RGB order
        cof_add: [0.0, 0.0, 0.0]
        cof_mul: [0.00392157, 0.00392157, 0.00392157]

########################################
# Postprocess
########################################
postprocess:
  -
    src: ["391"]

    dest: ["post_out_13x13"]

    operations:
      -
        op : cast_fp16_fp32
        param:
          CAST_MODE: 0 # FP16 to FP32

  -
    src: ["371"]
```

```
    dest: ["post_out_26x26"]

    operations:
      -
        op : cast_fp16_fp32
        param:
          CAST_MODE: 0 # FP16 to FP32

  -
    src: ["output"]

    dest: ["post_out_52x52"]

    operations:
      -
        op : cast_fp16_fp32
        param:
          CAST_MODE: 0 # FP16 to FP32
```

Example 4: Model with crop operation

Onnx model shape : Input = 96x96x3ch, Output=96x96x3ch

```
########################################
# Input data
########################################
input_to_pre:
  -
    name: "camera_data"
    format: "YUY2"
    order: "HWC"
    shape: [480, 640, 2]
    type: "uint8"

input_to_body:
  -
    name: "input"
    format: "RGB"
    order: "HWC"
    shape: [96, 96, 3]
    type: "fp16"

########################################
# Output data
########################################
output_from_body:
  -
    name: "output"
    shape: [96, 96, 3]
    order: "HWC"
    type: "fp16"

output_from_post:
  -
    name: "post_out"
    shape: [3,96,96]
    order: "CHW"
    type: "fp16"
```

```
########################################
# Preprocess
########################################
preprocess:
  -
    src      : ["camera_data"]
    dest     : ["input"]

    operations:
    -
      op: conv_yuv2rgb
      param:
        DOUT_RGB_FORMAT: 0 # "RGB"
    -
      op: crop
      param:
        CROP_POS_X : 60
        CROP_POS_Y : 40
        shape_out : [200, 200]
        DATA_TYPE : 0   # 0 : 1Byte, 1 : 2Byte
        DATA_FORMAT : 0 # 0 : "HWC"
    -
      op: resize_hwc
      param:
        RESIZE_ALG: 1 # "Bilinear"
        DATA_TYPE: 0  # "uint8"
        shape_out: [96, 96]
    -
      op: cast_any_to_fp16
      param:
        DIN_FORMAT: 0 # "uint8"

########################################
# Postprocess
########################################
postprocess:
  -
    src: ["output"]
    dest: ["post_out"]

    operations:
    -
      op : transpose
      param:
        WORD_SIZE: 1  # FP16
        IS_CHW2HWC: 0 # HWC to CHW
    -
      op : cast_fp16_fp32
      param:
        CAST_MODE : 0 # FP16 to FP32
```

Example 5: Model with multi inputs

Onnx model shape : Input = [64x64x3ch, 64x64x3ch] , Output=16x16x32ch

```
########################################
# Input data
```

```
########################################
input_to_pre:
  -
    name: "input_preA"
    format: "RGB"
    order: "HWC"
    shape: [64, 64, 3]
    type: "fp16"

  -
    name: "input_preB"
    format: "RGB"
    order: "HWC"
    shape: [64, 64, 3]
    type: "fp16"

input_to_body:
  -
    name: "input_A"
    format: "RGB"
    order: "HWC"
    shape: [64, 64, 3]
    type: "fp16"

  -
    name: "input_B"
    format: "RGB"
    order: "HWC"
    shape: [64, 64, 3]
    type: "fp16"

########################################
# Output data
########################################
output_from_body:
  -
    name: "output"
    shape: [16, 16, 32]
    order: "HWC"
    type: "fp16"

output_from_post:
  -
    name: "output_post"
    shape: [16, 16, 32]
    order: "HWC"
    type: "fp16"

########################################
# Preprocess
########################################
preprocess:
  -
    src      : ["input_preA"]

    dest     : ["input_A"]

    operations:
    -
```

```
      op: memcopy
      param:
          WORD_SIZE : 2 # FP16

  -
    src      : ["input_preB"]

    dest     : ["input_B"]

    operations:
    -
      op: memcopy
      param:
          WORD_SIZE : 2 # FP16

########################################
# Postprocess
########################################
postprocess:
  -
    src      : ["output"]

    dest     : ["output_post"]

    operations:
    -
      op: memcopy
      param:
          WORD_SIZE : 2 # FP16
```

6. How to Run

6.1. Quantization onnx Models

As the first step, DRP-AI Translator i8 converts the ONNX model of float to the INT8 onnx model by using INT8 quantizer. The script to run quantization is described below.

```
$ python3 -m drpai_quantizer.cli_interface \
  --input_model_path ./YoloV2_sparse90.onnx \
  --output_model_path ./YoloV2_sparse90_INT8.onnx \
  --calibrate_dataset  <Install dir>/DRP-AI_Translator_i8/drpAI_Quantizer/calibrate_images \
  --datareader_path  <Install dir>/DRP-AI_Translator_i8/drpAI_Quantizer/nchw_datareader.py \
  --norm_mean [0.0,0.0,0.0] \
  --norm_std [1.0,1.0,1.0]
```

The arguments used in the above example are explained below.

argument	Note
--input_model_path	Input onnx model file name (float model)
--output_model_path	Output quantized onnx model file name (INT8 model)
--calibrate_dataset	Dataset folder name for calibration
--datareader_path	Datareader script
--norm_mean	Mean values for normalization
--norm_std	Standard deviation values for normalization

To achieve higher accuracy, download the dataset used training and apply it as calibration data.
[Note] For more details, please refer to User's Manual of INT8 Quantizer.

6.2. Translation onnx Models

Translator is a tool that generates object files using quantized INT8 onnx as the input. The script to run translation is described below.

```
$ ./run_Translator_v2h.sh \
        yolov2_sparse \
        --onnx ../YoloV2_sparse90_INT8.onnx \
        --prepost ../prepost_yolov2.yaml \
        --s_addr 0x80000000 \
        --sparse
[START DRP-AI Transaltor V2H] ver x.x.x
 Prefix     : yolov2_sparse
 ONNX model : ../YoloV2_sparse90_INT8.onnx
 PrePost    : ../prepost_yolov2.yaml
 Start address : 0x80000000
[Start operator check]
[PASS] The number of operators is 210
----------------------------------------
[Prepost file check] start
  Check prepost yaml and onnx : PASS
[Sparse mode : on]
  >> Start sparsity check
```

```
    Layer : Conv_0, Not Sparse layer
    Layer : Conv_3, Sparsity : 89.58[%]
    Layer : Conv_6, Sparsity : 89.93[%]

  ...

Check address overwrap
  [OK] :  data_in
  [OK] :  data,
  [OK] :  data_out
  [OK] :  work
  [OK] :  weight
  [OK] :  drp_config
  [OK] :  aimac_param_cmd
  [OK] :  aimac_param_desc
  [OK] :  aimac_cmd
  [OK] :  aimac_desc
  [OK] :  drp_param
  [OK] :  drp_desc
Input_node_name: camera_data
         Address: 0x80000000
         Channel: 2
         Width  : 640
         Height : 480

Output_node_name: post_out
         Address: 0x811d2700
         Channel: 125
         Width  : 13
         Height : 13

> Save output files
  Object files are saved to  output/yolov2_sparse
[Finish DRP-AI Transaltor for V2H]
```

The following are options for the shell scripts.

Option	Category	Argment	Description
`--onnx` *file_path*	Must	str	Path to the ONNX model file
`--prepot` *file_path*	Must	str	Path to the Pre and Postprocessing definitions file
`--s_addr` *start address(hex)*	Option	int(hex)	Start address. Default is 0x80000000
`--sparse`	Option	-	Enable translation for sparse mode
`--check_prepost` *False/True*	Option	str	Check function for Pre and Postprocessing file default is "True"
`--check_operator` *False/True*	Option	str	Check function for operators in onnx file default is "True"

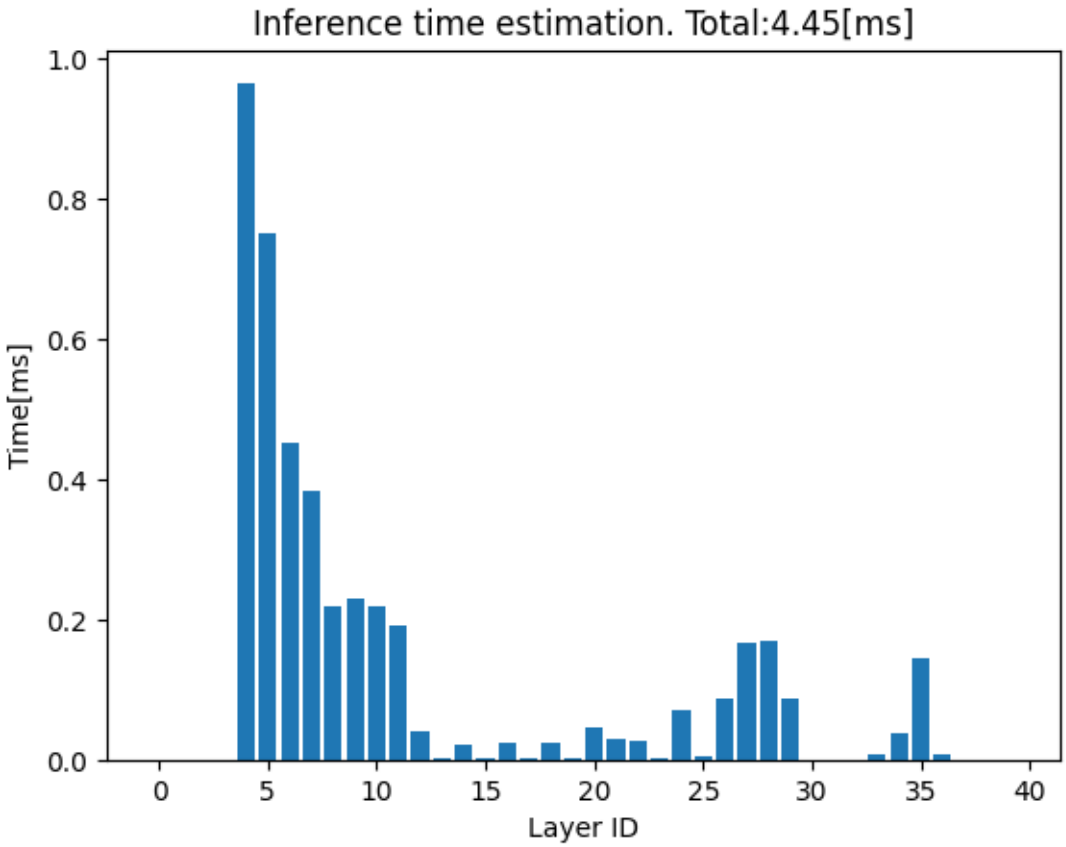
6.3. Execution Time Estimation

The estimated execution time of each layer of the neural network is output in Excel format as follows:

	name	opexp	ker								data						time[us]
							pad				input			output			
			w	h	st	dil	l	r	t	b	height	width	ch	height	width	ch	–
0	pre_0_0_conv_yu	conv_yuv2rgb	-1	-1	-1	-1	-1	-1	-1	-1	1080	1920	2	1080	1920	3	967
1	pre_0_1_resize_h	resize_hwc	-1	-1	-1	-1	-1	-1	-1	-1	1080	1920	3	416	416	3	2053
2	pre_0_2_imagesc	imagescaler	-1	-1	-1	-1	-1	-1	-1	-1	416	416	3	416	416	3	174
3	input1_Quantize	QuantizeLinear	-1	-1	-1	-1	-1	-1	-1	-1	416	416	3	416	416	3	190
4	260	Conv	3	3	1	1	1	1	1	1	416	416	3	416	416	32	1044
5	138	MaxPool	2	2	2	1	0	0	0	0	416	416	32	208	208	32	802
6	263	Conv	3	3	1	1	1	1	1	1	208	208	32	208	208	64	441
7	142	MaxPool	2	2	2	1	0	0	0	0	208	208	64	104	104	64	401
8	266	Conv	3	3	1	1	1	1	1	1	104	104	64	104	104	128	222
9	269_s	Conv	1	1	1	1	0	0	0	0	104	104	128	104	104	64	224
10	272	Conv	3	3	1	1	1	1	1	1	104	104	64	104	104	128	222
35	output1_Quanti	Conv	1	1	1	1	0	0	0	0	13	13	1024	13	13	125	10
36	output1	DequantizeLine	-1	-1	-1	-1	-1	-1	-1	-1	13	13	125	13	13	125	3
37	post_0_0_transp	transpose	-1	-1	-1	-1	-1	-1	-1	-1	13	13	125	13	13	125	91
38	post_0_1_cast_f	cast_fp16_fp32	-1	-1	-1	-1	-1	-1	-1	-1	13	13	125	13	13	125	44
														Total =			8.37[ms]

Column	Description
ith	Index number.
name	Layer name is based on the source input ONNX file.
opexp	Operator names included in this layer. Opexp may include multiple operator names in case that DRP-AI Translator merges multiple ONNX operators and calculate them as one set.
ker	Kernel size information for this layer. -w: kernel width -h: kernel height -st: kernel stride -dil: dilation size -pad: padding information. l: left r: right t: top b: bottom.
data	Size information of input feature map and output feature map. -in: input feature map ich: input channel w: width h: height -out: output feature map och: output channel w: width h: height
time[us]	Estimated operation time. The unit is microsecond. Note: Not considering bus congestion outside of DRP-AI Accelerator. Not considering memory bandwidth outside of DRP-AI Accelerator. The value may be changed without prior notice.

The png file of the cycle estimation graph is saved in addition to the Excel file.



6.4. Details of Address Map

When run DRP-AI translator, please specify the start address to place the DRP-AI object files in the physical memory space of the target LSI. After translation, address map file will be generated.
Address map files(addr_map.yaml/addr_map.txt) contents are as follows:

- data_in
- data
- data_out
- work
- weight
- drp_config
- drp_param
- aimac_cmd
- aimac_desc
- aimac_param_cmd
- aimac_param_desc
- drp_desc

data_in

- Memory area to place the input data to DRP-AI
  (e.g. Image from camera, jpeg data, etc.)

data

- Memory area to place the results of each layer of the neural network

- The results are stored in the order of the layers and not overwritten
- Preprocessing results are also stored in this area

data_out

- Memory area to place the final inference results
- Postprocessing results are also stored in this area

work

- Memory area for temporarily storing intermediate computed values during the calculation of each layer of the neural network
- Data that is no longer needed will be overwritten

weight

- Memory area to place the weight data of the neural network
- Consists of weights and biases of convolutional layers, weights of fully connected layers, etc.

drp_config

- Memory area to place DRP configuration data

drp_param

- Memory area to place DRP parameter

aimac_cmd

- Memory area to place AIMAC command

aimac_desc

- Memory area to place AIMAC descriptor

aimap_param_cmd

- Memory area to place AIMAC parameter command

aimap_param_desc

- Memory area to place AIMAC parameter descriptor

drp_desc

- Memory area to place DRP descriptor

6.5. Detailed Address Map for ‘data_in’

The start address and shape of each input node before preprocessing are displayed in **data_in_list.txt** as follows:

Example 1: VGA size, RGB format input case

```
Input_node_name: input_pre
         Address: 0x5f800000
         Channel: 3
         Width  : 640
         Height : 480
```

6.6. Detailed Address Map for ‘data_out’

The start address and shape of each output node after postprocessing are displayed in **data_out_list.txt** as follows:

Example 1: Single flattened output case

```
Output_node_name: post_out
         Address: 0x42101400
         Channel: 1000
         Width  : 1
         Height : 1
```

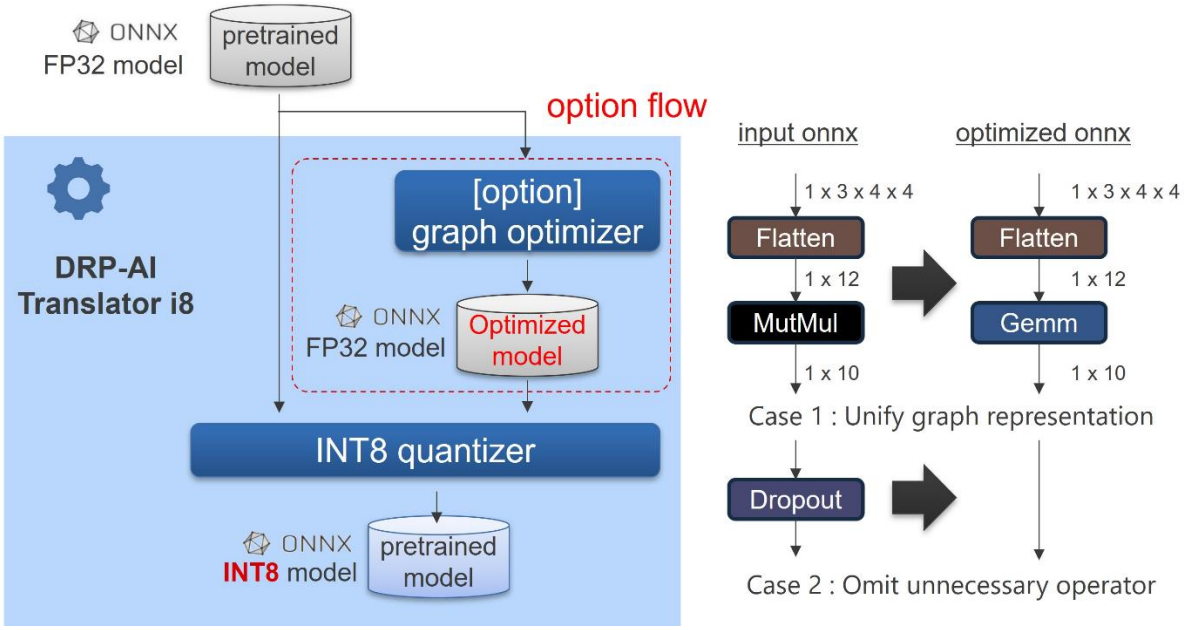
Example 2: Multiple convolutional output case

```
Output_node_name: 391
```

```
         Address: 0x666bca00
         Channel: 255
         Width  : 13
         Height : 13
Output_node_name: 371
         Address: 0x666d1ab0
         Channel: 255
         Width  : 26
         Height : 26
Output_node_name: output
         Address: 0x66725d68
         Channel: 255
         Width  : 52
         Height : 52
```

6.7. [option] Optimize onnx graph before quantization

If you get an error in Translator, please try “optimization” with graph optimizer. The optimizer unifies the expression of onnx graphs and omits unnecessary processing. By quantizing the optimized model, the probability of a successful conversion increases.



The sample command is below.

```
$ pythonn3 <translator>/DRP-AI_Trannslator/onnx_optimizer/run_onnx_optimizer.py \
 --file_in <input onnx model file path> \
 --file_out <optimized onnx model file path>
```

7. Uninstallation

Delete the directory **DRP-AI_Translator_i8** generated by the installer as follows:

```
$ rm -r DRP-AI_Translator_i8
```

Note: The output products of this tool should be saved in advance if necessary.

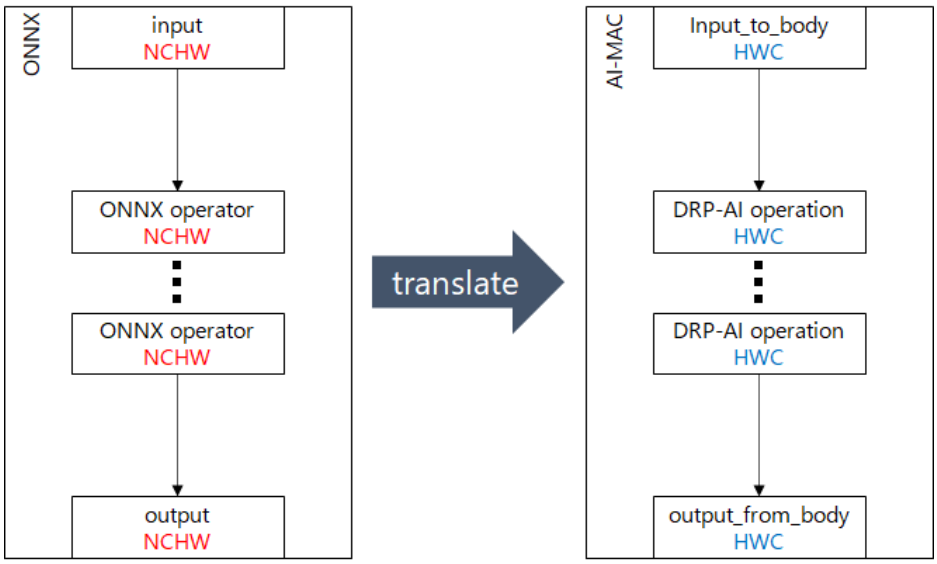
8. Error Message

Error Category	Error Message, Description and Action
Input File	`[ERROR A001] onnx file is not found` The specified neural network model (*.onnx) file does not exist. Make sure the file is in the path on the error message.
Input File	`[ERROR A002] PrePost definition file is not found` The specified Pre and Postprocessing definition (*.yaml) file does not exist. Make sure the file is in the path on the error message.
Translation error	`[ERROR A003] Failed at sparse check operation` Some problem happened in sparsity check operation input sparse onnx model. Please check onnx file or contact Renesas Electronics.
Translation error	`[ERROR A004] Failed at parse operation` Some problem happened in parse operation input onnx model. Please check onnx file or contact Renesas Electronics
Translation error	`[ERROR A005] Failed at memory optimization` Some problem happened in operation of memory optimization. Please contact Renesas Electronics
Translation error	`[ERROR A006] Failed at DRP config generation` Some problem happened in operation of generating DRP config file. Please check onnx file or contact Renesas Electronics
Translation error	`[ERROR A007] Failed at convert operation for AI-MAC desc` Some problem happened in operation of generating descriptor/command files. Please contact Renesas Electronics.
Translation error	`[ERROR A008] Failed at convert operation for DRP desc` Some problem happened in operation of generating DRP descriptor files. Please contact Renesas Electronics.
Translation error	`[ERROR A009] Failed at cycle estimation` Some problem happened in parse operation input onnx model. Please contact Renesas Electronics
Translation error	`[ERROR A010] Found unsupported operators/graph in ONNX model` There are unsupported operator/attribute in onnx model. Please check error message and onnx file or check error message.
Translation error	`[ERROR A011] onnx model and PrePost yaml setting mismatch` Found a problem between onnx model and Pre and Postprocessing definition file. Please check onnx(*onnx) and Pre and Postprocessing definition file(*yaml)
Translation error	`[ERROR A012] Unsupported Activation function and Quantization option` Some Activation function(e.g. Swish) does not supported zero-point offset. Please run INT8 Quantizer again, without zero-point offset option
Translation error	`[ERROR A013] Unsupprted PrePost definition setting` PrePostOnly mode recommends PreProcessing only pattern or PostProcessing only pattern
Translation error	`[ERROR A014] Found physical address settings violation at --s_addr option` The max size of address setting is 32 bits(0xFFFFFFFF). Please run Translator again with the correct settings

Appendix

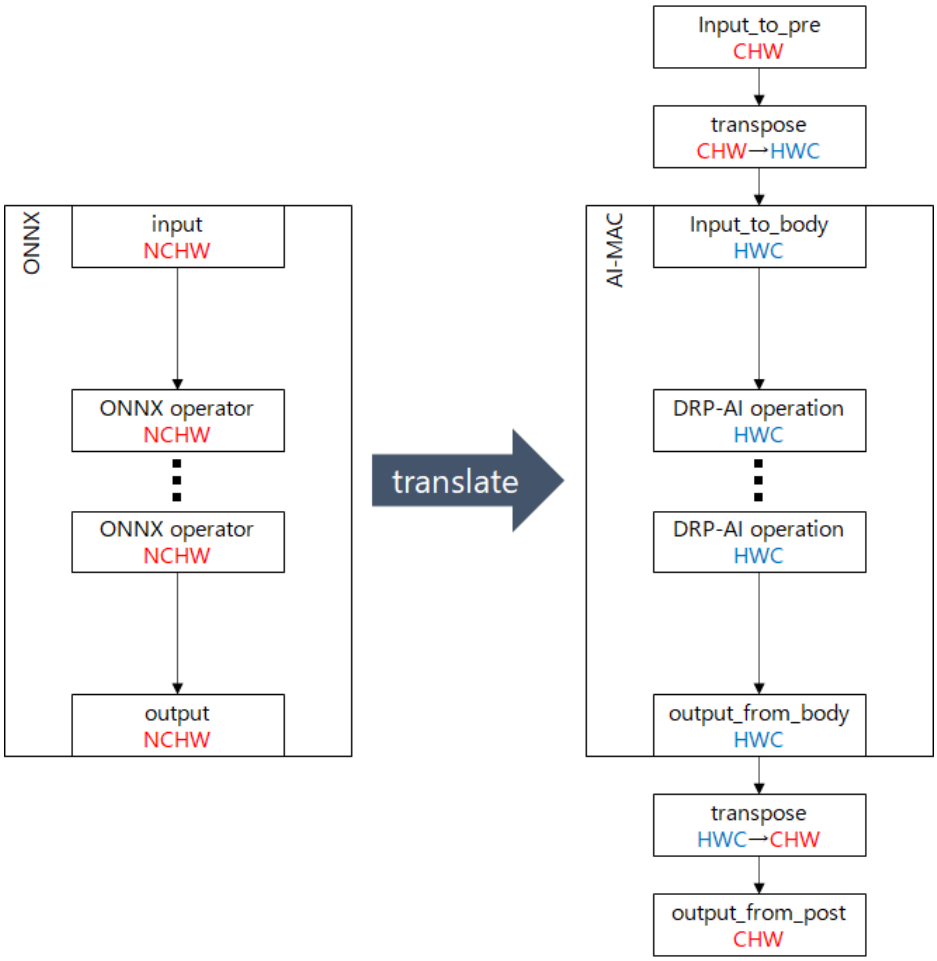
Appendix A. DRP-AI input/output data order

DRP-AI is composed of DRP, which performs pre/post processing, and AI-MAC, which performs product-sum operation processing.
The order of the feature map supported by AI-MAC is channel last (HWC). This is because the AI-MAC hardware is architected for parallel processing by dividing the channels, and the channel last is more efficient. On the other hand, the order of the feature map supported by ONNX is channel first (NCHW).
To absorb these differences, the DRP-AI Translator automatically converts ONNX channel first (NCHW) to channel last (HWC) when generating DRP-AI object files from ONNX. This can be illustrated as follows.

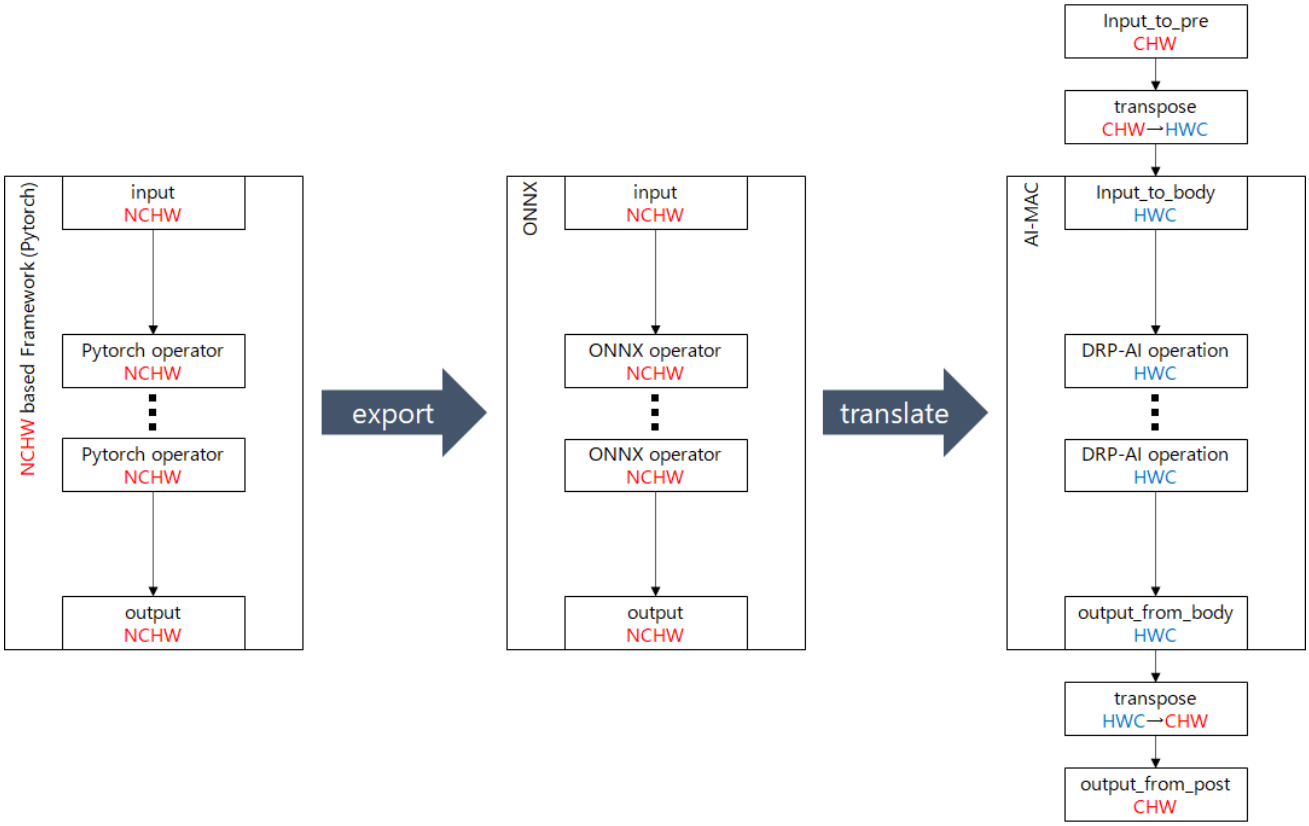


Why HWC instead of NHWC?
N is the mini-batch size used for training, but N=1 is usually used for inference, and the DRP-AI Translator only supports N=1. In python arrays, for example, NHWC is 4-dimensional and HWC is 3-dimensional, so there is a clear difference in the shape of the arrays. However, DRP-AI treats the data as equivalent as long as the arrangement in memory is the same, so HWC without N is used to eliminate redundancy.

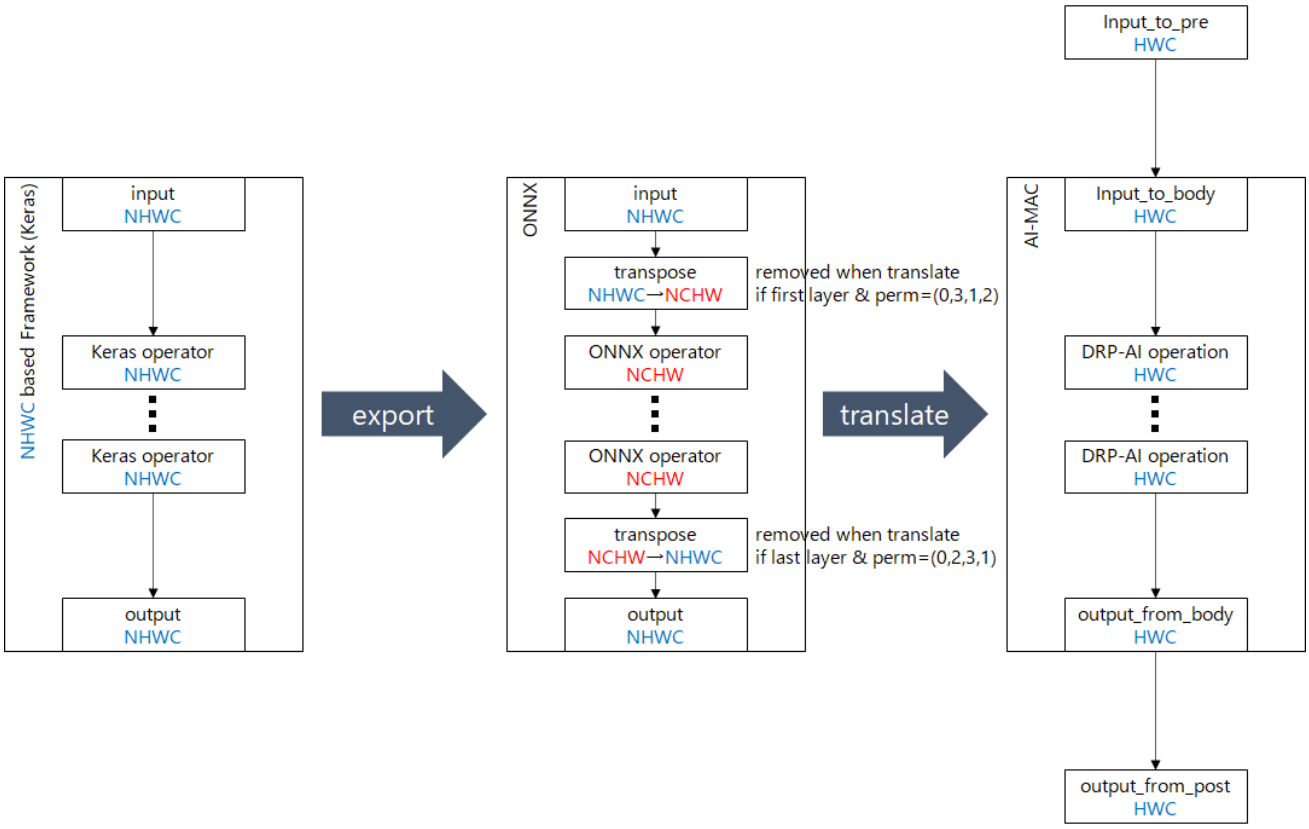
In this case, the order of the input and output data of ONNX and AI-MAC are different, so the channel first (NCHW) data input to ONNX cannot be input directly to AI-MAC.
If you want to use channel first (CHW) data for DRP-AI input/output data, you need to add transpose to DRP-AI pre/post processing and perform order conversion as shown below.



Some AI frameworks support channel first (NCHW) and some support channel last (NHWC). As examples, by default, Pytorch supports channel first (NCHW) and Tensorflow (Keras) supports channel last (NHWC).
If you export ONNX from a framework that supports channel first, such as Pytorch, ONNX will be exported in the same order as it is because ONNX is also channel first. If you want to maintain the input/output order and implement it in DRP-AI, you need to add transpose to the pre/post processing as shown in the figure below.



On the other hand, if you export ONNX from a framework that supports channel last, such as Tensorflow (Keras), transpose layers will be inserted inside ONNX (at the beginning and end) to maintain the order of input and output as it is because ONNX is channel first. The DRP-AI Translator has the ability to automatically remove these transpose layers inserted at the beginning and end of the ONNX. Therefore, when such ONNX is translated by DRP-AI Translator, the transpose layers are removed and implemented as shown in the figure below. In this case, the input/output order is channel last, so you do not need to add transpose to your DRP-AI pre/post processing.



Appendix B. How to use python sample scripts

To support translation operation, python APIs have been provided. Pre & Post definition file can be generated by python APIs. The following code is included as sample scripts in the DRP-AI Translator i8 package.

1. Sample script to run Translator
   - Generate pre & post definition file(.yaml) by python script
   - Call Translator on python script

[Note] These functions are beta version. The features and APIs subject to change.

1. How to use running Translator script

- This section will explain the python sample script. The file name is below:

  <install dir>/translator/UserConfig/sample_scripts/run_translator_sample_resnet50.py
- The steps to run Translator are as follows.

  A) Define and set pre & post nodes information to drp_prepost class

  B) Define and set pre & post processing classes to drp_prepost class

  C) Save pre & post definition file

  D) Initialize Translator class

  E) Set start address

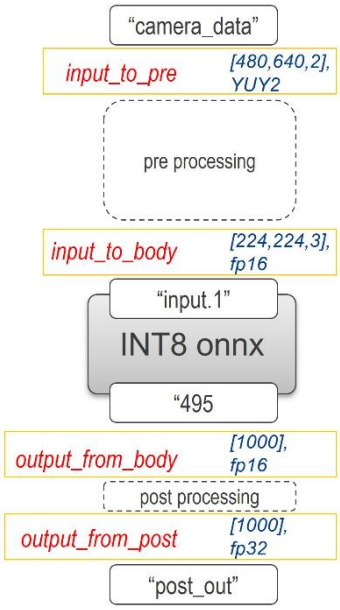
  F) Run Translator
- Before running sample script, please set environment variable(TRANSLATOR) to the directory where you installed DRP-AI Translator i8.

  $ export TRANSLATOR=/home/user/DRP-AI_Translator_i8/translator/

A) Define and set pre & post nodes information to drp_prepost class

At first, define input & output node information about onnx model and pre & post processing

```
pp = drp_prepost()
pp.set_input_to_pre("camera_data", shape=[480,640,2],
                    order="HWC",type="uint8",format="YUY2")
pp.set_input_to_body("input.1", shape=[224,224,3],
                    order="HWC",type="fp16",format="RGB")
pp.set_output_from_body("495",
                    shape=[1000],order="C",type="fp16")
pp.set_output_from_post("post_out",
                    shape=[1000],order="C",type="fp32")
```



B) Define and set pre & post processing classes to drp_prepost class

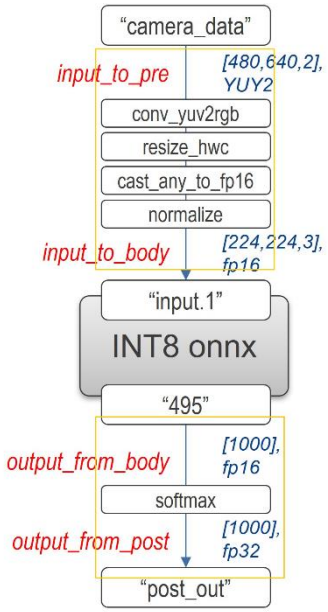
Next, initialize pre & post processing class, and get its instance object. Please refer section 4.2 in User's Manual about parameters of each process. To define sequence order of pre & post processing, make list data which stored instances. Then set pre & post processing sequence to drp_prepost class instance.

```
pre1_conv_yuv2rgb = conv_yuv2rgb(DOUT_RGB_FORMAT=0)
pre2_resize_hwc   = resize_hwc(shape_out=[224,224],RESIZE_ALG=1)
pre3_cast         = cast_any_to_fp16()
pre4_normalize    = normalize(cof_add=[-123.675, -116.28, -103.53],¥
                              cof_mul= [0.01712475, 0.017507, 0.01742919])

post1_softmax     = softmax(DOUT_FORMAT=1)

pre_sequence  = [pre1_conv_yuv2rgb,pre2_resize_hwc,pre3_cast,pre4_normalize]
post_sequence = [post1_softmax]

pp.set_preprocess_sequence(src=["camera_data"],dest=["input.1"],pp_seq=pre_sequence)
pp.set_postprocess_sequence(src=["495"],dest=["post_out"],pp_seq=post_sequence)
```



C) Save pre & post definition file

To check setting of pre & post processing definition, show_param() function is prepared. The result will show on the console. And by using save() function, pre & post processing definition file is saved.

```
# show setting info
pp.show_params()

# Save data as yaml file
pp.save("output.yaml")
```

D) Initialize Translator class

To run Translator, drp_ai_translator class instance object is required. Please initialize and set the target device.

```
# Initialize DRP-AI Translator class
drp_tran = drp_ai_translator()

# Choose device of run script
drp_tran.set_translator("V2H")
```

E) Set start address

Please set the start address of the physical memory to map the DRP-AI Object files.

```
# Set Start address
start_address="0x80000000"
```

F) Run Translator

By calling run_translate() function, Translator will execute. Please set the QDQ onnx model quantized by INT8 Quantizer as a input data. The execution result is the same as using "./run_Translator_v2h.sh".

```
# Check onnx model file
ONNX_MODEL="<…>/Resnet50v1_sparse90_INT8.onnx"
drp_tran.run_translate("resnet_test",¥
                       onnx=Q_ONNX_MODEL,¥
                       prepost=cur_path+"/output.yaml",¥
                       s_addr=start_address, ¥
                       options=["--sparse"] ¥
    )
```

2. API reference

- Details of each API are described below.

CLASS : python_api.drp_prepost

drp_prepost. set_input_to_pre(node_name, shape=None, order=None, type=None, format=None)

Set information about input to preprocessing to drp_prepost class instance object.

Parameters :

node_name : str, input node name. e.g. “camera_in”

shape : list, data shape. e.g. [480,640,2]

order : str, data order. e.g. “HWC” / “CHW”

type : str, data type. e.g. “fp16” / “uint8”

format : str, data format. e.g. “RGB” / “YUY2”

Return :

None

drp_prepost. set_input_to_body(node_name, shape=None, order=None, type=None, format=None)

Set information about input to body to drp_prepost class instance object. “input to body” means input node name of onnx model.

Parameters :

node_name : str, input node name. e.g. “input”

shape : list, data shape. e.g. [224,224,3]

order : str, data order. e.g. “HWC”

type : str, data type. e.g. “fp16”

format : str, data format. e.g. “RGB”

Return :

None

drp_prepost. set_output_from_body(node_name, shape=None, order=None, type=None)

Set information about output from body to drp_prepost class instance object.

“output from body” means output node name of onnx model.

Parameters :

node_name : str, output node name. e.g. “output”

shape : list, data shape. e.g. [1000]

order : str, data order. e.g. “HWC” / “C”

type : str, data type. e.g. “fp16”

Return :

None

drp_prepost. set_output_from_post(node_name, shape=None, order=None, type=None, format=None)

Set information about output from post drp_prepost class instance object. “output from post” means output node of Postprocessing.

Parameters :

node_name : str, output node name. e.g. “post_out”

shape : list, data shape. e.g. [1000]

order : str, data order. e.g. “C”

type : str, data type. e.g. “fp32” / “fp16”

Return :

None

drp_prepost. set_preprocess_sequence(src=[], dest=[], pp_seq=[])

Set preprocess sequence to drp_prepost class instance object.

Parameters :

src : list, source node name of preprocess sequence. e.g. [“camera_in”]

dest : list, destination node name of preprocess sequence. shape. e.g. [“input”]

pp_seq : list, preprocess instance list. e.g. [conv_yuv2rgb, resize_hwc, cast, normalize]

Return :

None

drp_prepost. set_postprocess_sequence(src=[], dest=[], pp_seq=[])

Set postprocess sequence to drp_prepost class instance object.

Parameters :

src : list, source node name of postprocess sequence. e.g. [“output”]

dest : list, destination node name of postprocess sequence. shape. e.g. [“post_out”]

pp_seq : list, postprocess instance list. e.g. [softmax]

Return :

None

drp_prepost. show_param()

Show prepopst processing definition status to console.

Parameters :

None

Return :

None

drp_prepost. save(file_name)

Save prepost processing definition information as yaml file .

Parameters :

file_name : str, output file name. e.g. “output.yaml”

Return :

None

CLASS : python_api.drp_ai_translator

drp_ai_translator.set_translator(device)

Select target device.

Parameters :

device : str, target device. e.g. “V2H”

Return :

None

drp_ai_translator.run_translate(prefix, onnx=None, prepost=None, s_addr=None)

Run DRP-AI Translator

Parameters :

prefix : str, prefix is output folder name. e.g. “Resnet50_test”

onnx : str, input onnx file path. e.g. “../onnx/resnet50.onnx”

prepost : str, prepost processing definition file path. e.g. “./prepost.yaml”

s_addr : str, start address. e.g. “0x80000000”

Return :

None

Revision History		DRP-AI Translator i8 User's Manual	
Rev.	**Date**	**Description**	
		Seciton	**Summary**
1.01	Feb.29, 2024	3.2.1	Added Getting Started to directory structure
		3.2.2	Changed dependencies
		4.1	Updated supported operators and attributes
		B	Added "Appendix B section" (How to use python sample scripts)
1.0.0	Dec.27, 2023	1.1	Changed manual coverage to extend to DRP-AI Translator i8 including Quantizer
		1.2	Changed the input files and added the intermediate file
		2	Changed figure of the input and output files of the DRP-AI Translator
		2.1	Changed the input files description
		3.2.1	Changed the installation procedure and the directory structure
		3.2.2	Changed dependencies
		4.1	Updated supported operators and attributes
		6.1	Added description of how to run quantization
		6.2	Moved description of how to run translation to this section
		6.3	Changed figure
		6.7	Added description of how to run optimization
		7	Changed directory name
		8	Added error message
0.9.0	Sep.29, 2023	4.1	Updated supported operators and attributes
		4.1.3	Added "Supported special graph structure"
		6.1	Changed the version in the execution example from "alpha" to "x.x.x"
		6.2	Changed the argument of check_prepost option Added the check_operator option
		8	Added the error message [ERROR A012]
0.5.0	Jun.30, 2023	—	Create a new entry

DRP-AI Translator i8 V1.01 User's Manual

Publication Date: Rev.0.5.0 Jun.30.23
Rev.1.01 Feb.29.24

Published by: Renesas Electronics Corporation

DRP-AI Translator i8 V1.01

User's Manual

Renesas Electronics Corporation

R20UT5336EJ0101