User's Manual

DRP-AI Translator V1.84

User's Manual

Renesas Electronics
www.renesas.com

Rev.2.70 Mar. 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". 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.

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

1.2. Required Role

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

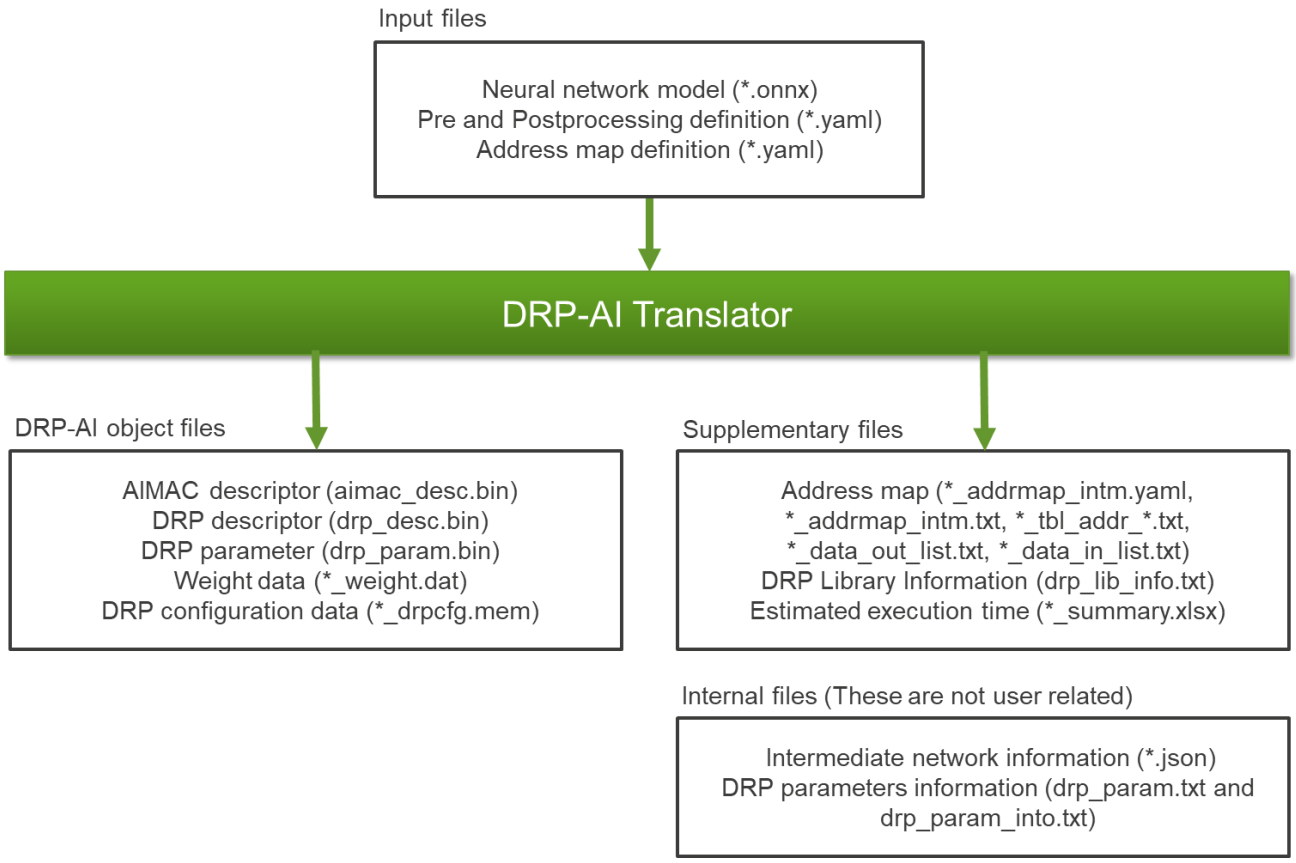
- Knowledge of command line I/F for Ubuntu OS and Python.
- Basic knowledge of the structure of convolutional neural networks and their input/output I/F.
- Basic knowledge about preprocessing and postprocessing of neural networks and applications including them.
- Knowledge of the ONNX format and its 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 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)
  - Pre and Postprocessing definition (*.yaml)
  - Address map definition (*.yaml)
- Output files
  - DRP-AI object files
    - AIMAC descriptor (aimac_desc.bin)
    - DRP descriptor (drp_desc.bin)
    - DRP parameter (drp_param.bin)
    - Weight data (*_weight.dat)
    - DRP configuration data (*_drpcfg.mem)
  - Supplementary files
    - Address map (*_addrmap_intm.yaml, *_addrmap_intm.txt, *_tbl_addr_*.txt , *_data_out_list.txt, *_data_in_list.txt)
    - DRP Library information (drp_lib_info.txt)
    - Estimated execution time (*_summary.xlsx)
    - Optimized prepost file (*_prepost_opt.yaml)
    - Translation error report file (*_error_report.xlsx)
- Internal files (These are not user related)
  - Intermediate network information (*.json)
  - DRP parameters information (drp_param.txt and drp_param_info.txt)

2.1. Input Files

See 4. Input Files for details.

2.2. Output Files

DRP-AI object files

- AIMAC descriptor (aimac_desc.bin)
  - Descriptor for DRP-AI's MAC (AIMAC) unit
  - Location address: Start address of **desc_aimac** in address map
- DRP descriptor (drp_desc.bin)
  - Descriptor for DRP-AI's DRP unit
  - Location address: Start address of **desc_drp** in address map
- DRP parameter (drp_param.bin)
  - Parameter referenced by the DRP library
  - Location address: Start address of **drp_param** in address map
- Weight data (*_weight.dat)
  - Weight data referenced by AIMAC
  - Location address: Start address of **weight** in address map
- DRP configuration data (*_drpcfg.mem)
  - Data to configure the DRP
  - Location address: Start address of **drp_config** in address maps

Supplementary files

- Address map (*_addrmap_intm.yaml, *_addrmap_intm.txt, *_tbl_addr_*.txt, *_data_*_list.txt)
  - Address map including size information of DRP-AI object files, input/output data area, etc.

  - *_addrmap_intm.yaml
  - *_addrmap_intm.txt
  - *_tbl_addr_data.txt ('data' Element Space)
  - *_tbl_addr_data_in.txt ('data_in' Element Space)
  - *_tbl_addr_data_out.txt ('data_out' Element Space)
  - *_tbl_addr_drp_config.txt ('drp_config' Element Space)
  - *_tbl_addr_weight.txt ('weight' Element Space)
  - *_tbl_addr_work.txt ('work' Element Space)
  - *_tbl_addr_merge.txt
  - *_data_out_list.txt ('data_out' Element Space, detailed for each output)
  - *_data_in_list.txt ('data_in' Element Space, detailed for each input)
- DRP Library information (drp_lib_info.txt)
  - Describes the number of times the DRP library has been called, a breakdown of the library types, and whether or not each library generates an interrupt at exit.
- Estimated execution time (*_summary.xlsx)
  - Execution time summary sheet
- Optimized prepost file (*_prepost_opt.yaml)
  - Prepost file optimized for processing purpose.
- Translation error report file (*_error_report.xlsx)
  - Summary of unsupported operators. This file will be generated only when translation failed.

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:

```
$ sudo apt-get update
$ sudo apt-get install -y libgl1-mesa-dev wget libopencv-dev
$ sudo apt-get install -y python3-pip
$ pip3 install -U pip
$ chmod +x DRP-AI_Translator-v1.83-Linux-x86_64-Install
$ ./DRP-AI_Translator-v1.83-Linux-x86_64-Install
```

Continue the installation interactively. Type 'Enter' or 'y' to continue.

```
This installer will guide you through the installation of DRP-AI Translator.  Continue?
[n/Y] y
```

After the installation progress bar is displayed, you will be notified of the completion of the installation as follows:

```
Installing DRP-AI_Translator...
Installing Program Files...
Installation complete.
```

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

```
$ tree -L 2 drp-ai_translator_release
drp-ai_translator_release/
├── DRP-AI_translator
│   ├── api_translator
│   ├── availability_checker
│   ├── converter
│   ├── drp_converter
│   ├── drplib
│   └── python_api
├── UserConfig
│   ├── sample
│   └── sample_scripts
├── onnx
│   ├── (resnet50v1.onnx) (Can be automatically downloaded)
```

```
│   ├── tiny_yolov2.onnx
│   ├── (vgg16.onnx) (Can be automatically downloaded)
│   └── yolov2.onnx
├── output
├── run_DRP-AI_translator_V2L.sh
└── run_DRP-AI_translator_V2M.sh
```

Directory name, File name	Contents
DRP-AI_translator/	Contains the DRP-AI translator program
UserConfig/	Contains the pre and postprocessing definition (*.yaml)、Address map definition (*.yaml). The sample definition files for the included ONNX models are stored in **sample**. Directory **sample/separate_address_mapping_sample** contains another version of the address map definition files, the multiple Sub-Spaces type. Directory **sample/prepost_for_RTOS** contains another version of the pre and postprocessing definition files for RTOS users. For details, refer to the appendix distributed to RTOS users. Directory **sample/sample_scripts** contains python sample scripts to run DRP-AI translator. Please refer to ReleaseNote. Those scripts and APIs are beta version. The features and APIs subject to change.
onnx/	Contains samples of ONNX models 1. ResNet50v1* 2. VGG16* 3. Tiny-YOLOv2 4. YOLOv2 *Will be automatically downloaded when user selects ResNet50v1/VGG16 when calling run_DRP-AI_translator_V2M(L).sh
output/	The output of the DRP-AI translator is stored in this directory Note: This directory will be generated after translation
run_DRP-AI_translator_V2L.sh run_DRP-AI_translator_V2M.sh	Shell scripts for DRP-AI translator. Switch between **V2L** and **V2M** depending on the target device

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
protobuf	5.26.0
onnx	1.15.0
pandas	2.0.3
openpyxl	3.1.2
pyyaml	6.0.1
cython	3.0.9
networkx	3.1
onnxruntime	1.16.3
opencv-python	4.9.0.80
sympy	1.12

numpy	1.24.4
matplotlib	3.7.5

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 : 1.9.0
  - Opset version : 12

Supported ONNX Operator List

<table>
<tr><th>Operator</th><th>Category</th><th>Name</th><th>Supported Values</th></tr>
<tr><td>Add</td><td>-</td><td>-</td><td>Support same size addition
Support the bias addition case
Support the case that are equivalent to BatchNormalization in combination with Mul or Div
Support the case that can be fused to previous BatchNormalization
Support the case that can be fused to previous Conv or Gemm</td></tr>
<tr><td>AveragePool</td><td>Attributes
Inputs</td><td>Attributes
kernel_shape
strides
pads
Inputs
X : (1, ich, H, W)</td><td>
<table>
<tr><th>kernel</th><th>stride</th><th>pads</th><th>ich</th><th>Height(H) & Width(W)</th></tr>
<tr><td>[2, 2]</td><td>[2, 2]</td><td>[0, 0, 0, 0]</td><td>ich <= 65535</td><td>H % 2 == 0, W % 2 == 0, H <= 65535, W <= 65535</td></tr>
<tr><td>[2, 2]</td><td>[2, 2]</td><td>[0, 1, 0, 1]</td><td>ich <= 65535</td><td>H % 2 == 0, W % 2 == 0, H <= 65535, W <= 65535</td></tr>
<tr><td>[3, 3]</td><td>[1, 1]</td><td>[0, 0, 0, 0]</td><td>-</td><td>3 <= W <= 4096, 3 <= H</td></tr>
<tr><td>[3, 3]</td><td>[1, 1]</td><td>[0, 1, 0, 1]</td><td>-</td><td>2 <= W <= 4096, 2 <= H</td></tr>
<tr><td>[3, 3]</td><td>[1, 1]</td><td>[1, 0, 1, 0]</td><td>-</td><td>2 <= W <= 4096, 2 <= H</td></tr>
<tr><td>[3, 3]</td><td>[1, 1]</td><td>[1, 1, 1, 1]</td><td>-</td><td>1 <= W <= 4096, 1 <= H</td></tr>
<tr><td>[7, 7]</td><td>[1, 1]</td><td>[0, 0, 0, 0]</td><td>-</td><td>7 <= W <= 2048, 7 <= H</td></tr>
<tr><td>[7, 7]</td><td>[1, 1]</td><td>[0, 3, 0, 3]</td><td>-</td><td>4 <= W <= 2048, 4 <= H</td></tr>
<tr><td>[7, 7]</td><td>[1, 1]</td><td>[3, 0, 3, 0]</td><td>-</td><td>4 <= W <= 2048, 4 <= H</td></tr>
<tr><td>[7, 7]</td><td>[1, 1]</td><td>[3, 3, 3, 3]</td><td>-</td><td>1 <= W <= 2048, 1 <= H</td></tr>
</table>
</td></tr>
<tr><td></td><td>Attributes</td><td>auto_pads</td><td>same as the Conv</td></tr>
<tr><td></td><td>Attributes</td><td>ceil_mode</td><td>(default) | 0</td></tr>
<tr><td></td><td>Attributes</td><td>count_include_pad</td><td>(default) | 0 | 1</td></tr>
<tr><td></td><td>-</td><td>-</td><td>Support GlobalAveragePool equivalent case, i.e. where kernel_shape is the same as the shape of the feature map</td></tr>
<tr><td>BatchNormalization</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Cast</td><td>-</td><td>-</td><td>Support handling of constant parameters embedded in ONNX. See the graph in Section 4.1.2. Other function for more details</td></tr>
<tr><td>Clip</td><td>Inputs</td><td></td><td>ich < 8192
och == ich</td></tr>
<tr><td></td><td>Attributes</td><td>min</td><td>0</td></tr>
<tr><td></td><td>Attributes</td><td>max</td><td>(default) | 6
Note: Support Relu6 equivalent case. See "Command Line Options" for replacement to Relu</td></tr>
<tr><td>Concat</td><td>-</td><td>-</td><td>Support handling of constant parameters embedded in ONNX</td></tr>
<tr><td></td><td>Inputs
Attributes</td><td>Attributes : axis
Inputs : input0,
input1,
...</td><td>- axis == 1
- input0: (1, ch0, H, W)
input1: (1, ch1, H, W) ...</td></tr>
<tr><td>Constant</td><td>-</td><td>-</td><td>Support handling of constant parameters embedded in ONNX. See the graph in Section</td></tr>
</table>

			4.1.2. Other function for more details
Conv	Attributes	kernel_shape	[1, 1], [3, 3], ~~[5,5]~~, [7, 7], ~~[9,9]~~
	Attributes	strides	[1, 1], [2, 2]
	Attributes	pads	Case_A: pad_l == pad_r == pad_t == pad_b == 0 Case_B: pad_l == pad_r == pad_t == pad_b == (kernel_shape - 1) / 2 Case_C: pad_l == pad_t ==0 && pad_r == pad_b == (kernel_shape - 1) / 2 Note: If kernel_shape == [7x7] and stride == 1, Case A and Case_B are supported Note: Case_C is supported under kernel_shape == [3,3] && strides == [2,2] && ich %2 == 0 && och %2 == 0
	Attributes	auto_pads	(default) | NONSET
			VALID
			SAME_UPPER | SAME_LOWER (The amount of padding must meet the “pads” limit)
	Attributes	group	(default) | 1 | n Note: ‘n’ is supported only in the equivalent case of Depthwise convolution. See Section 4.1.2 for the detail.
	Attributes	dilations	(default) | 1 | n Note: ‘n’ is supported only in the equivalent case of Dilated convolution. 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	-	-	-
Expand	-	-	Expand is only supported when executed between Reshape operators. e.g. Reshape1>Expand>Reshape2. Input shape for Reshape1 operator must be 4-dimension data (1,ich,H,W). Output shape from Reshape2 operator must be 4-dimension data(1, ich, HxN, WxN).
	Inputs	X: (1, ich, H, 1, W, 1)	ich < 16384
		Shape: (1, ich, H, N, W, N)	N= 2/3/4
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 handling of constant parameters embedded in ONNX. See the graph in Section 4.1.2. Other function for more details

<table>
<tr><td>Gemm</td><td>Attributes</td><td>alpha</td><td>(default) | 1.0</td></tr>
<tr><td></td><td>Attributes</td><td>beta</td><td>(default) | 1.0</td></tr>
<tr><td></td><td>Attributes</td><td>transA</td><td>(default) | 0</td></tr>
<tr><td></td><td>Attributes</td><td>transB</td><td>(default) | 0 | 1</td></tr>
<tr><td></td><td>-</td><td>-</td><td>Support fully connected layer equivalent case</td></tr>
<tr><td>GlobalAveragePool</td><td>Inputs</td><td>X: (1, ich, H, W)</td><td>–</td></tr>
<tr><td>(Hardswish)</td><td>-</td><td>-</td><td>See Section 4.1.2</td></tr>
<tr><td>LeakyRelu</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>MatMul</td><td>-</td><td>-</td><td>Support fully connected layer equivalent case</td></tr>
<tr><td>MaxPool</td><td>Attributes
Inputs</td><td>Attributes
kernel_shape
strides
pads
Inputs
X : (1, ich, H, W)</td><td>
<table>
<tr><th>kernel</th><th>stride</th><th>pads</th><th>Height(H) & Width(W)</th><th>Combined Condition</th></tr>
<tr><td>[2, 2]</td><td>[1, 1]</td><td>[1, 1, 1, 1]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[2, 2]</td><td>[1, 1]</td><td>[1, 0, 1, 0]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[2, 2]</td><td>[1, 1]</td><td>[0, 1, 0, 1]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[2, 2]</td><td>[1, 1]</td><td>[0, 0, 0, 0]</td><td>2 <= H</td><td>2 <= W <= 4096</td></tr>
<tr><td>[2, 2]</td><td>[2, 2]</td><td>[1, 1, 1, 1]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[2, 2]</td><td>[2, 2]</td><td>[1, 0, 1, 0]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[2, 2]</td><td>[2, 2]</td><td>[0, 1, 0, 1]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[2, 2]</td><td>[2, 2]</td><td>[0, 0, 0, 0]</td><td>2 <= H</td><td>2 <= W <= 4096</td></tr>
<tr><td>[3, 3]</td><td>[1, 1]</td><td>[1, 1, 1, 1]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[3, 3]</td><td>[1, 1]</td><td>[1, 0, 1, 0]</td><td>2 <= H</td><td>2 <= W <= 4096</td></tr>
<tr><td>[3, 3]</td><td>[1, 1]</td><td>[0, 1, 0, 1]</td><td>2 <= H</td><td>2 <= W <= 4096</td></tr>
<tr><td>[3, 3]</td><td>[1, 1]</td><td>[0, 0, 0, 0]</td><td>3 <= H</td><td>3 <= W <= 4096</td></tr>
<tr><td>[3, 3]</td><td>[2, 2]</td><td>[1, 1, 1, 1]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[3, 3]</td><td>[2, 2]</td><td>[1, 0, 1, 0]</td><td>2 <= H</td><td>2 <= W <= 4096</td></tr>
<tr><td>[3, 3]</td><td>[2, 2]</td><td>[0, 1, 0, 1]</td><td>2 <= H</td><td>2 <= W <= 4096</td></tr>
<tr><td>[3, 3]</td><td>[2, 2]</td><td>[0, 0, 0, 0]</td><td>3 <= H</td><td>3 <= W <= 4096</td></tr>
<tr><td>[5, 5]</td><td>[1, 1]</td><td>[2, 2, 2, 2]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[5, 5]</td><td>[1, 1]</td><td>[2, 0, 2, 0]</td><td>3 <= H</td><td>3 <= W <= 4096</td></tr>
<tr><td>[5, 5]</td><td>[1, 1]</td><td>[0, 2, 0, 2]</td><td>3 <= H</td><td>3 <= W <= 4096</td></tr>
<tr><td>[5, 5]</td><td>[1, 1]</td><td>[0, 0, 0, 0]</td><td>5 <= H</td><td>5 <= W <= 4096</td></tr>
<tr><td>[9, 9]</td><td>[1, 1]</td><td>[4, 4, 4, 4]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
<tr><td>[13, 13]</td><td>[1, 1]</td><td>[6, 6, 6, 6]</td><td>1 <= H</td><td>1 <= W <= 4096</td></tr>
</table>
</td></tr>
<tr><td></td><td>Attributes</td><td>auto_pads</td><td>same as the Conv</td></tr>
<tr><td></td><td>Attributes</td><td>ceil_mode</td><td>(default) | 0</td></tr>
<tr><td></td><td>Attributes</td><td>dilations</td><td>(default) | 1</td></tr>
<tr><td></td><td>Attributes</td><td>storage_order</td><td>(default) | 0</td></tr>
<tr><td>(Mish)</td><td>-</td><td>-</td><td>See Section 4.1.2</td></tr>
<tr><td>Mul</td><td>-</td><td>-</td><td>Support the case that are equivalent to BatchNormalization in combination with Add or Sub
Support the case that can be fused to previous BatchNormalization</td></tr>
<tr><td>Pad</td><td>-</td><td>-</td><td>Support the case that can be fused to the next Conv, MaxPool and AveragePool</td></tr>
</table>

			(Note: Must meet the constraints of the pad attribute on the destination to be fused)
	Inputs	X: (1, ich, H, W)	-
	Attributes	pads	[0, 0, h begin, w begin, 0, 0, h end, w end] Batch & channel direction pad is not supported.
	Attributes	constant_value	Support float32 value
PRelu	-	-	–
ReduceMean	-	-	Support GlobalAveragePool equivalent case
Relu	-	-	-
(Relu6)			See Section 4.1.2
Reshape	-	-	Support handling of constant parameters embedded in ONNX Support Flatten equivalent case
	Inputs	data & shape	Support reshaping feature map from 4dim to 2dim just before fully connected layer - shape(data) : (1, ch, H, W) - shape(shape): (1, -1) or (1, ch*H*W)
	Inputs	data & shape	Support reorg(Darknet impl) component case ich % 4 == 0, width % 2 == 0, height % 2 == 0
	Inputs	data & shape	Support SpaceToDepth component case ich % 4 == 0, width % 2 == 0, height % 2 == 0
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	-	-	-
Shape	-	-	Support handling of constant parameters embedded in ONNX. See the graph in Section 4.1.2. Other function for more details
Sigmoid	-	-	-
Softmax	Inputs	X: (1, ch) or X : (1, ch, H, W), axis = 1(ch direction only)	1<=ch<=16384

SpaceToDepth	Attributes	blocksize	blocksize == 2
Squeeze	-	-	Support Flatten equivalent case
Sub	-	-	Support the case that are equivalent to BatchNormalization in combination with Mul or Div Support the case that can be fused to previous BatchNormalization Support the case that can be fused to previous Conv or Gemm
Sum	Inputs	data_0, data_1, ...	Support same data size summation shape(data_0): (1, ch, H, W) shape(data_1): (1, ch, H, W) ...
(Swish)			See Section 4.1.2
Tanh	-	-	-
Transpose	-	-	Support handling of constant parameters embedded in ONNX
	InputsAttributes	Inputs : data Attributes: perm	Support reorg(Darknet impl) component case
	InputsAttributes	Inputs : data Attributes: perm	Support SpaceToDepth component cases
Unsqueeze	-	-	-

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

4.1.2.Other function

The DRP-AI translator supports the following functions.

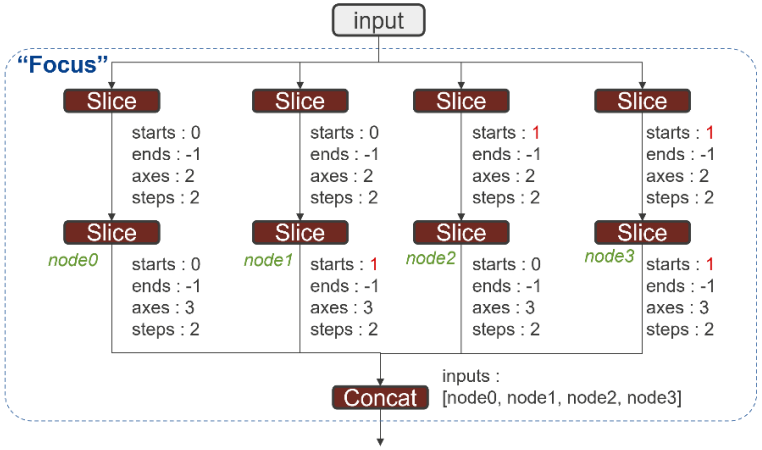
Function	Category	Onnx operator(s)	Note/Supported Values
Depthwise convolution	Attributes of Conv operator	Conv	Attributes/group : n conditions: och == n, ich == 1(per group), dilation == 1, (kernel_shape == [3,3], strides == [1.1], pads == [1,1,1,1], 1 <= W <= 4096, 1 <= H) | (kernel_shape == [3,3], strides == [1.1], pads == [1,0,1,0], 2 <= W <= 4096, 2 <= H) | (kernel_shape == [3,3], strides == [1.1], pads == [0,1,0,1], 2 <= W <= 4096, 2 <= H) | (kernel_shape == [3,3], strides == [1.1], pads == [0,0,0,0], 3 <= W <= 4096, 3 <= H) | (kernel_shape == [3,3], strides == [2.2], pads == [1,1,1,1], 1 <= W <= 4096, 1 <= H) | (kernel_shape == [3,3], strides == [2.2], pads == [1,0,1,0], 2 <= W <= 4096, 2 <= H) | (kernel_shape == [3,3], strides == [2.2], pads == [0,1,0,1], 2 <= W <= 4096, 2 <= H)

			| (kernel_shape == [3,3], strides == [2.2], pads == [0,0,0,0], 3 <= W <= 4096, 3 <= H) | (kernel_shape == [5,5], strides == [1.1], 5-pad_l-pad_r <= W <= 2048, 5-pad_t-pad_b <= H <= 2048) | (kernel_shape == [5,5], strides == [2.2], 5-pad_l-pad_r <= W <= 2048, 5-pad_t-pad_b <= H <= 2048) | (kernel_shape == [7,7], strides == [1.1], 7-pad_l-pad_r <= W <= 2048, 7-pad_t-pad_b <= H <= 2048) | (kernel_shape == [7,7], strides == [2.2], 7-pad_l-pad_r <= W <= 2048, 7-pad_t-pad_b <= H <= 2048) Recommends: kernel_shape == [7,7], 4 < pad_l, 4 < pad_r, 4 < pad_t, 4 < pad_n
Dilated convolution	Attributes of Conv operator	Conv	Attributes/dilation : n conditions: group == 1, kernel_shape == [3,3], strides == [1, 1], pads == [n,n,n,n] Recommends: n <= W, n <= H
Dilated depthwise convolution	Attributes of Conv operator	Conv	Attributes/dilation : n och == m, ich == 1(per group), conditions: 2 <= n < min(H,W), (kernel_shape == [3,3], strides == [1.1], pads == [n,n,n,n], 1 <= W <= 4096, 1 <= H) | (kernel_shape == [3,3], strides == [1.1], pads == [n,0,n,0], 1+n <= W <= 4096, 1+n <= H) | (kernel_shape == [3,3], strides == [1.1], pads == [0,n,0,n], 1+n <= W <= 4096, 1+n <= H) | (kernel_shape == [3,3], strides == [1.1], pads == [0,0,0,0], 1 + n * 2 <= W <= 4096, 1 + n * 2 <= H) Recommends: n < 6, W <=4096, H <=4096, ich <=16384
Focus	Group of operators	Slice x4 > Slice x4 > Concat	See below figure.
HardSwish	Group of operators	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 operators	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

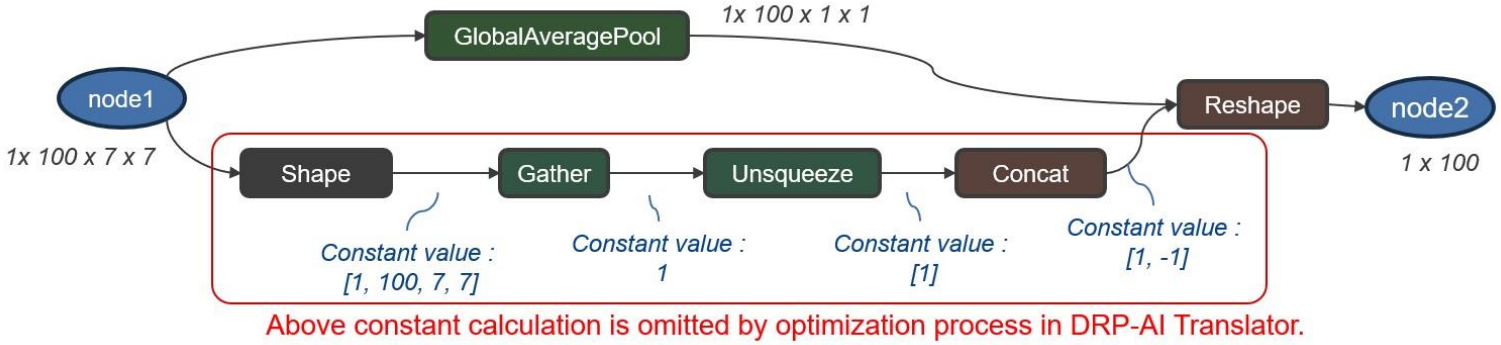
Swish	Group of operators	Sigmoid, Mul	Note: Swish formula is f(x) = x * sigmoid(x), and in the onnx graph it is represented by combination of Sigmoid and Mul operators

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

- Focus



DRP-AI Translator supports some handling of constant parameters. Following is an example case.



4.1.3. Minimum plane size of feature map

The neural networks that can be implemented in DRP-AI are limited by the following minimum plane size of the feature maps.

kernel_shape	strides	pads	minimum plane size (width, height)
[1,1]	[1,1]	0	(1,1)
	[2,2]	0	(1,1)
[3,3]	[1,1]	0	(3, 3)
		1	(1, 1)
	[2,2]	0	(5, 7)
		1	(3, 6)
~~[5,5]~~	~~[1,1]~~	~~0~~	~~(12, 10)~~
		~~2~~	~~(12, 8)~~
	~~[2,2]~~	~~0~~	~~(19, 19)~~
		~~2~~	~~(19, 19)~~
[7,7]	[1,1]	0	(20, 7)
		3	(14, 7)
	[2,2]	0	(19, 19)
		3	(13, 13)
~~[9,9]~~	~~[1,1]~~	~~0~~	~~(26, 9)~~
		~~4~~	~~(18, 9)~~
	~~[2,2]~~	~~0~~	~~(23, 17)~~
		4	~~(23, 17)~~

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.

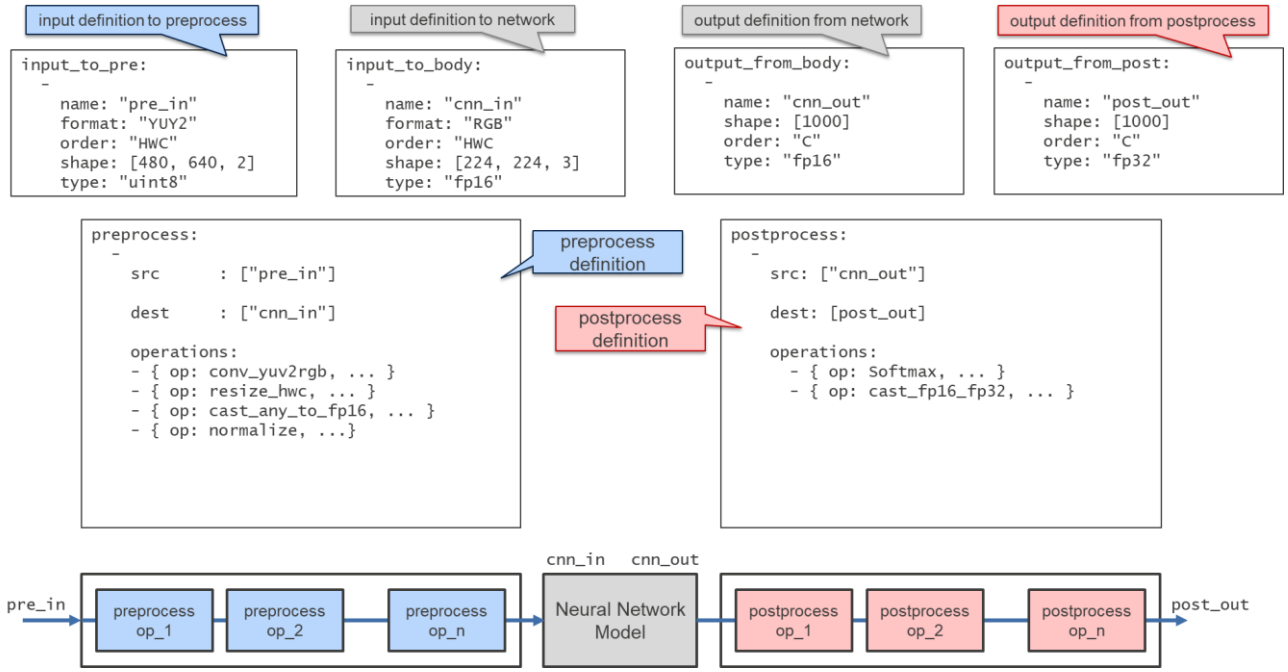
4.3. Address Map Definition

Defines the address to place the DRP-AI object files in memory space of target LSI.

4.3.1. Address Map Definition Format

The address map is defined in a YAML file. See 6. Details of Address Map Definition for details.

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

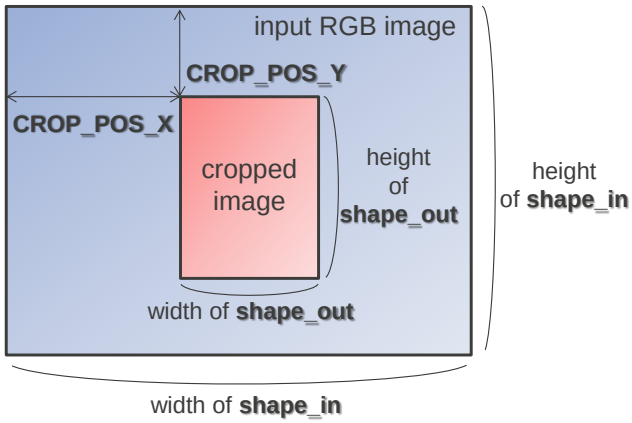
```
op: memcopy
param:
  WORD_SIZE: 2
```

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
```

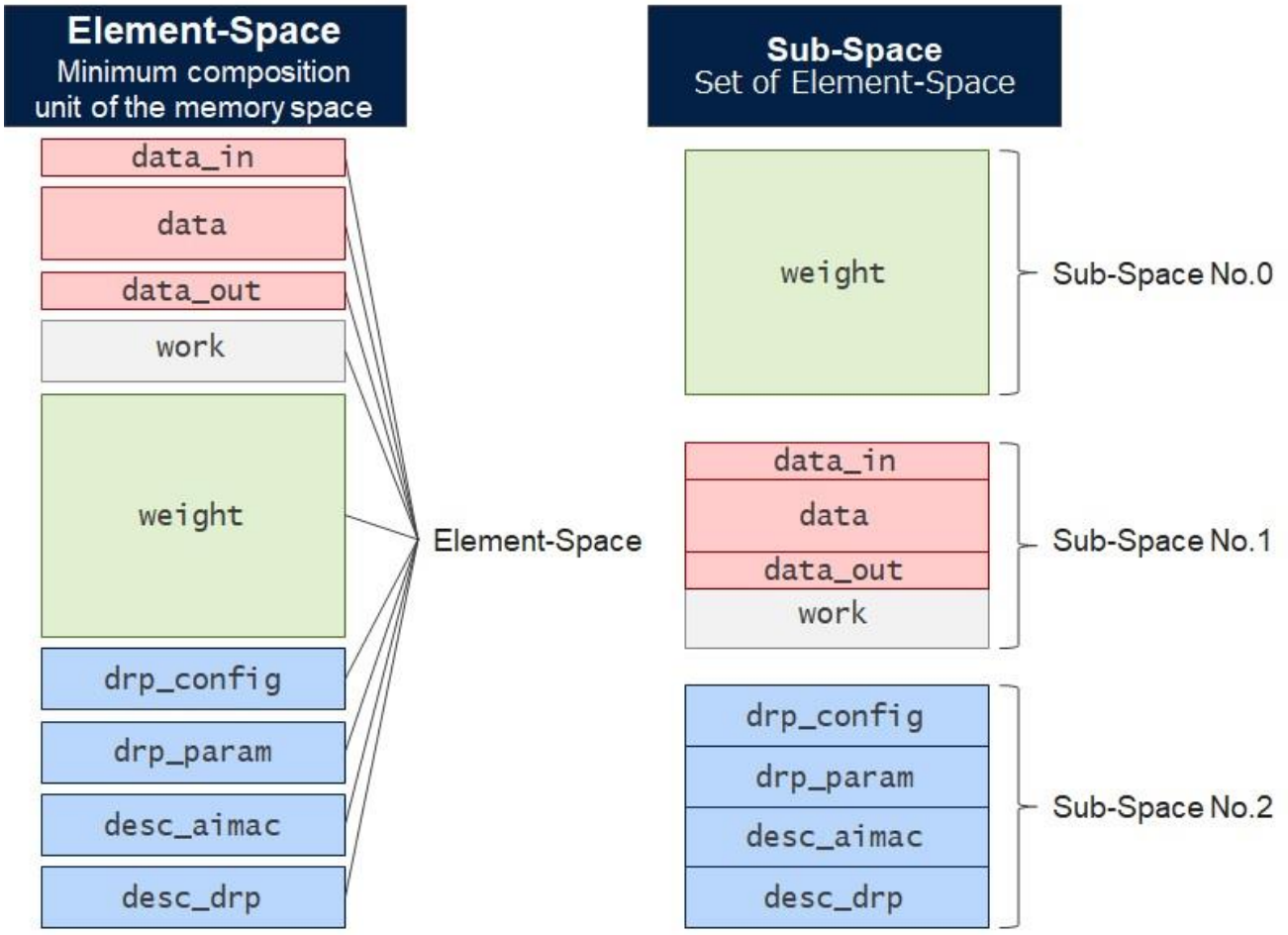
```
######################################
input_to_pre:
  -
    name: "input_preA"
    format: "RGB"
    order: "HWC"
    shape: [64, 64, 3]
    type: "fp16"

  -
    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. Details of Address Map Definition

Specify the address map to place the DRP-AI object files in the memory space of the target LSI by using "Element-Space" and "Sub-Space".

6.1. What is Element-Space and Sub-Space?



"Element-Space" means the minimum composition unit of the memory space. A set of "Element-Space" is called "Sub-Space".

- Sub-Space can be placed discretely in the memory space
- Multiple Element-Spaces that belong to Sub-Space are arranged consecutively
- DRP-AI Translator outputs the start address and size of each Element-Spaces to the address map (*_addrmap_intm.yaml, *_addrmap_intm.txt)

6.2. How to Describe Address Map Definition

- Describe "Sub Space" in the top layer with sequence
- Describe the element of each value with mapping
- The top layer has the following description style:

```
- { name: "Sub Space A", ...}
- { name: "Sub Space B", ...}
- { name: "Sub Space C", ...}
- ...
```

6.3. How to Describe Sub-Space

- Sub-Space is a set of Element-Spaces
- Sub-Space contains one or more Element-Space
- Sub-Space is a continuous space
- Multiple Sub-Spaces can be defined. When defining multiple sub-spaces, they must be defined in the order of their addresses
- The key of element and possible value are as follows:

Key	Must	Type	Description
name	Must	str	Name of Sub-Space
addr	Must	int	Start address of Sub-Space
lst_elemsp	Must	Hash []	Sequence of Element Space expression

- name (str)
  - Specify the Sub-Space name. Name is arbitrary
- addr (int)
  - Specify the start address of Sub-Space
  - The start address must be aligned on a 64-byte boundary
- lst_elemsp (Hash [])
  - Describe the Element-Space belonging to Sub-Space with sequence
  - Describe the elements that make up the value with mapping as follows:

```
...
  -
    name: "Sub Space A"
    addr: 0x10_000_000
    lst_elemsp:
      - { name: "Element Space A1", addr: 0x10_000_000, size: 0x... } # The value of "
addr" must match the "addr" value of sub-space. The "addr" and "size" can be omitted.
      - { name: "Element Space A2", ... }
      - { name: "Element Space A3", ... }
      ...
```

- There are some restrictions on the belonging of Element-Space to Sub-Space. For details, refer to the notes in 6-5. Examples of Address Map Definition.

6.4. How to Describe Element-Space

- Element-Space is the minimum composition unit of memory space.
- Element-Space is a continuous memory space
- The key of element and possible value are as follows:

Key	Must	Type	Description
name	must	str	Name of Element-Space

addr	optional	int	Start address of Element-Space
size	optional	int	Size of Element-Space

- name (str)
  - Specify the Element-Space name
- addr (int)
  - Specify the start address of Element-Space (optional)
    If not specified, the address will be calculated automatically to be aligned with the following boundaries:

Element-Space	Aligned boundary (Byte)
data_in	64
data	64
data_out	64
work	64
weight	64
drp_config	64
drp_param	16
desc_aimac	16
desc_drp	16

- size (int)
  - Specifies the memory area to allocate to Element-Space (optional)
    If not specified, the memory area will be calculated automatically
    Note:
    - If the size required by the tool is larger than the specified, the specification will be ignored.
    - The size shown in the output address map (*_addrmap_intm.yaml, *_addrmap_intm.txt) rounds up the actual size to the aligned boundary in the table above.

6.4.1. Details of Element-Space

Element-Space contents are as follows:

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

desc_aimac

- Memory area to place AIMAC descriptor

desc_drp

- Memory area to place DRP descriptor

6.5. Examples of Address Map Definition

Example 1: A case of putting all Element-Spaces into one Sub-Space

Address map definition (*.yaml)
Note: In this form, the order of all Element-Spaces must match the following example

```
-
  name: "all" # Sub-Space name. Name is arbitrary

  # Specify only the starting address
  addr: 0x40_000_000

  # The address and size of each Element-Space are automatically calculated
  lst_elemsp:
    - { name: "data_in" }
    - { name: "data" }
    - { name: "data_out" }
    - { name: "work" }
    - { name: "weight" }
```

```
    - { name: "drp_config" }
    - { name: "drp_param" }
    - { name: "desc_aimac" }
    - { name: "desc_drp" }
```

Examples of address map output (*_addrmap_intm.yaml)

```
-
  name: all
  addr: 0x4000_0000
  size:  0x534_5400
  lst_elemsp:
    - { name: data_in,          addr: 0x4000_0000, size:    0x9_6000 }
    - { name: data,             addr: 0x4009_6000, size:  0x206_b400 }
    - { name: data_out,         addr: 0x4210_1400, size:       0xfc0 }
    - { name: work,             addr: 0x4210_23c0, size:   0x31_0000 }
    - { name: weight,           addr: 0x4241_23c0, size:  0x30c_fc40 }
    - { name: drp_config,       addr: 0x454e_2000, size:   0x12_fa00 }
    - { name: drp_param,        addr: 0x4561_1a00, size:       0x150 }
    - { name: desc_aimac,       addr: 0x4561_1b80, size:    0x4_0530 }
    - { name: desc_drp,         addr: 0x4565_20c0, size:       0x1f0 }
```

Examples of address map output (*_addrmap_intm.txt)

```
data_in 40000000 96000
data 40096000 206b400
data_out 42101400 fc0
work 421023c0 310000
weight 424123c0 30cfc40
drp_config 454e2000 12fa00
drp_param 45611a00 150
desc_aimac 45611b80 40530
desc_drp 456520c0 1f0
```

Example 2: The case of putting each Element-Space into multiple Sub-Spaces

Address map definition (*.yaml)
Note:

- The following Element-Spaces must be in the same Sub-Space in this order (Do not put these in separate Sub-Space)
  - drp_config
  - drp_param
  - desc_aimac
  - desc_drp

If Element-Spaces other than the above four Element-Spaces belong to the same Sub-Space, the four Element-Spaces must be placed at the bottom.

```
-
  name: "desc"
  addr: 0x10_000_000
  lst_elemsp:
    - { name: "drp_config" }
    - { name: "drp_param" }
    - { name: "desc_aimac" }
    - { name: "desc_drp" }
```

```
-
  name: "param"
  addr: 0x30_000_000
  lst_elemsp:
    - { name: "weight" }
-
  name: "data"
  addr: 0x70_000_000
  lst_elemsp:
    - { name: "data_in" }
    - { name: "data" }
    - { name: "data_out" }
    - { name: "work" }
```

Examples of address map output (*_addrmap_intm.yaml)

```
-
  name: desc
  addr: 0x1000_0000
  size:         0x0
  lst_elemsp:
    - { name: drp_config,       addr: 0x1000_0000, size:   0x12_fa00 }
    - { name: drp_param,        addr: 0x1012_fa00, size:       0x150 }
    - { name: desc_aimac,       addr: 0x1012_fb80, size:    0x4_0530 }
    - { name: desc_drp,         addr: 0x1017_00c0, size:       0x1f0 }
-
  name: param
  addr: 0x3000_0000
  size:  0x30c_fc40
  lst_elemsp:
    - { name: weight,           addr: 0x3000_0000, size:  0x30c_fc40 }
-
  name: data
  addr: 0x7000_0000
  size:  0x227_5800
  lst_elemsp:
    - { name: data_in,          addr: 0x7000_0000, size:    0x9_6000 }
    - { name: data,             addr: 0x7009_6000, size:  0x206_b400 }
    - { name: data_out,         addr: 0x7210_1400, size:       0xfc0 }
    - { name: work,             addr: 0x7210_23c0, size:   0x31_0000 }
```

Examples of address map output (*_addrmap_intm.txt)

```
drp_config 10000000 12fa00
drp_param 1012fa00 150
desc_aimac 1012fb80 40530
desc_drp 101700c0 1f0
weight 30000000 30cfc40
data_in 70000000 96000
data 70096000 206b400
data_out 72101400 fc0
work 721023c0 310000
```

6.6. 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
```

6.7. 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
```

7. How to Run

7.1. Translation of the Included Sample Models

1. Run the shell script **"run_DRP-AI_translator_V2M.sh" or "run_DRP-AI_translator_V2L.sh"** depending on the target device.
2. Give any name for "PREFIX". The output of the DRP-AI translator is stored under **./output/PREFIX/**
3. Give the model number to translate from 1 to 4

```
$ cd drp-ai_translator_release
$ ./run_DRP-AI_translator_V2M(L).sh
[Run DRP-AI Translator] Ver. 1.83
[DRP-AI translator] run script
 Please input PREFIX.
 >>> test
    PREFIX: test
 Please select model No
   1: resnet50v1  (Download from onnx model zoo)
   2: VGG16       (Download from onnx model zoo)
   3: tiny_yolov2 (Included in DRP-AI Translator)
   4: yolov2      (Included in DRP-AI Translator)
 >>> 1
 [Input file information]
   PREFIX               : test
   ONNX Model           : ./onnx/resnet50v1.onnx
   Prepost file         : ./UserConfig/sample/prepost_resnet50v1.yaml
   Address mapping file : ./UserConfig/sample/addrmap_in_resnet50v1.yaml
[api-translator] start
...
```

7.2. Translation of User Models

7.2.1. Simple Specification Case

This is a case where only the ONNX file is explicitly specified.

1. Store the ONNX model to translate in any path
2. Store the pre and postprocessing definitions and address map definition of the same name as the ONNX model under **./UserConfig**. The name of each file should be determined as follows:
   Neural network model: In case of **./onnx/modelA.onnx**
   Pre and Postprocessing definition: **./UserConfig/prepost_modelA.yaml**
   Address map definition: **./UserConfig/addrmap_in_modelA.yaml**
3. Run the shell script **run_DRP-AI_translator_V2M.sh** or **run_DRP-AI_translator_V2L.sh** with the following arguments:
   PREFIX: Any name (The output of the DRP-AI translator is stored under **./output/PREFIX/**)
   -onnx: Path to the ONNX model

```
$ cd drp-ai_translator_release
$ ./run_DRP-AI_translator_V2M(L).sh test -onnx ./onnx/modelA.onnx
```

```
[Run DRP-AI Translator] Ver. 1.83
[Input file information]
   PREFIX                : test
   ONNX Model            : ./onnx/modelA.onnx
   Prepost file          : ./UserConfig/prepost_modelA.yaml
   Address mapping file  : ./UserConfig/addrmap_in_modelA.yaml
[api-translator] start
...
```

7.2.2. Detailed Specification Case

his is a case where all input files are explicitly specified.

1. Store the ONNX model to translate in any path
2. Store the pre and postprocessing definitions and address map definition in any path with any name
3. Run the shell script **run_DRP-AI_translator_V2M.sh** or **run_DRP-AI_translator_V2L.sh** with the following arguments:
   PREFIX: Any name (The output of the DRP-AI translator is stored under ./**output/PREFIX/**)
   -onnx: Path to the ONNX model
   -prepost: Path to the Pre and postprocessing definitions
   -addr: Path to the Address map definition

```
$ cd drp-ai_translator_release
$ ./run_DRP-AI_translator_V2M(L).sh test -onnx ./onnx/modelA.onnx \
    -prepost ./UserConfig/prepost_foo.yaml \
    -addr ./UserConfig/addrmap_in_bar.yaml
[Run DRP-AI Translator] Ver. 1.83
[Input file information]
   PREFIX                : test
   ONNX Model            : ./onnx/modelA.onnx
   Prepost file          : ./UserConfig/prepost_foo.yaml
   Address mapping file  : ./UserConfig/addrmap_in_bar.yaml
[api-translator] start
...
```

7.3. Command Line Options

The following are options for the shell scripts.

Option	Category	Argment	Description
-onnx *file_path*	Must (Note1)	str	Path to the ONNX model file
-prepot *file_path*	Must (Note2)	str	Path to the Pre and postprocessing definitions file
-addr *file_path*	Must (Note2)	str	Path to the Address map definition file
-RELU6toRELU	Optional	-	Replace the activation **Relu6** with **Relu** This option makes the inference time a little faster, but equivalence with the input model is compromised. Therefore, the user should

			evaluate the effect on accuracy before adopting this option in the products. Note: The **`Relu6`** is usually represented as **`Clip(min=0, max=6)`** in ONNX operator.
`-SaveOptOnnx`	Optional	-	Outputs an optimized onnx file of the input model The DRP-AI translator optimizes the input ONNX model and then outputs the DRP-AI object files. This option may help investigate the cause of translation errors.
`-CheckPrePost` *ON (or OFF)*	Optional	-	Check the pre and postprocessing definitions. (default = OFF) If an error is detected that seems to be caused by inconsistency of pre and postprocessing definitions, the tool will print a "hint" message to enable this option. When this option is enabled, the tool parses optimized onnx and pre/postprocessing definitions and outputs inconsistent information. Note: When this option is set to ON, the -SaveOptOnnx option will also be automatically enabled. Note: This feature is available as beta.
`-PrePostFULL`	Optional	-	Manually specify the pre and post processing definition file as full prepost definition file (format used in version older than ver 1.60.0 release). This option is NOT recommended to be used in normal circumstances.
`-NotCheckSizeONNXvsYAML`	Optional	-	Do not check the size of ONNX vs. YAML. If an error is detected with older opset versions of ONNX, this option may allow you to work around it. This option cannot be used for ONNX models with variable output size (dynamic shape). This option is NOT recommended to be used in normal circumstances.

Note1: Not required when translating included sample models
Note2: Not required in "Simple Specification case"

7.4. Help and More

Help command

Information can be obtained by giving **`-h`** or **`--help`** as an argument.

```
$ ./run_DRP-AI_translator_V2M(L).sh -h
[Run DRP-AI Translator] Ver. 1.83
How to use:
  [Case1] Select resnet50v1/vgg16/tiny_yolov2/yolov2 model
    ./run_DRP-AI_translator_V2M.sh
  [Case2] Use original onnx model
   ./run_DRP-AI_translator_V2M.sh PREFIX -onnx [modelfile_path]
  [Case3] Use original onnx model with custom config file
    ./run_DRP-AI_translator_V2M.sh PREFIX -onnx [modelfile_path] ¥
                                -addr [address_mapping_file_path] ¥
                                -prepost [prepost_file_path] ¥
                                (-CheckPrePost [OFF]/[ON])
                                (-SaveOptOnnx)
```

```
                                    (-RELU6toRELU)
                                    (-PrePostFULL)
                                    (-NotCheckSizeONNXvsYAML)
```

Error message

If any of the onnx model, pre and postprocessing definition, address map definition does not exist, it exits with the following message.

```
$ ./run_DRP-AI_translator_V2M(L).sh yolov2_s2d -onnx ./onnx/yolov2_s2d.onnx
[Run DRP-AI Translator] Ver. 1.83
[Input file information]
   PREFIX                : yolov2_s2d
   ONNX Model            : ./onnx/yolov2_s2d.onnx
   Prepost file          : ./UserConfig/prepost_yolov2_s2d.yaml
   Address mapping file : ./UserConfig/addrmap_in_yolov2_s2d.yaml
[ERROR] :  ./UserConfig/prepost_yolov2_s2d.yaml  file is not exist.
```

7.5. Execution Time Estimation

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

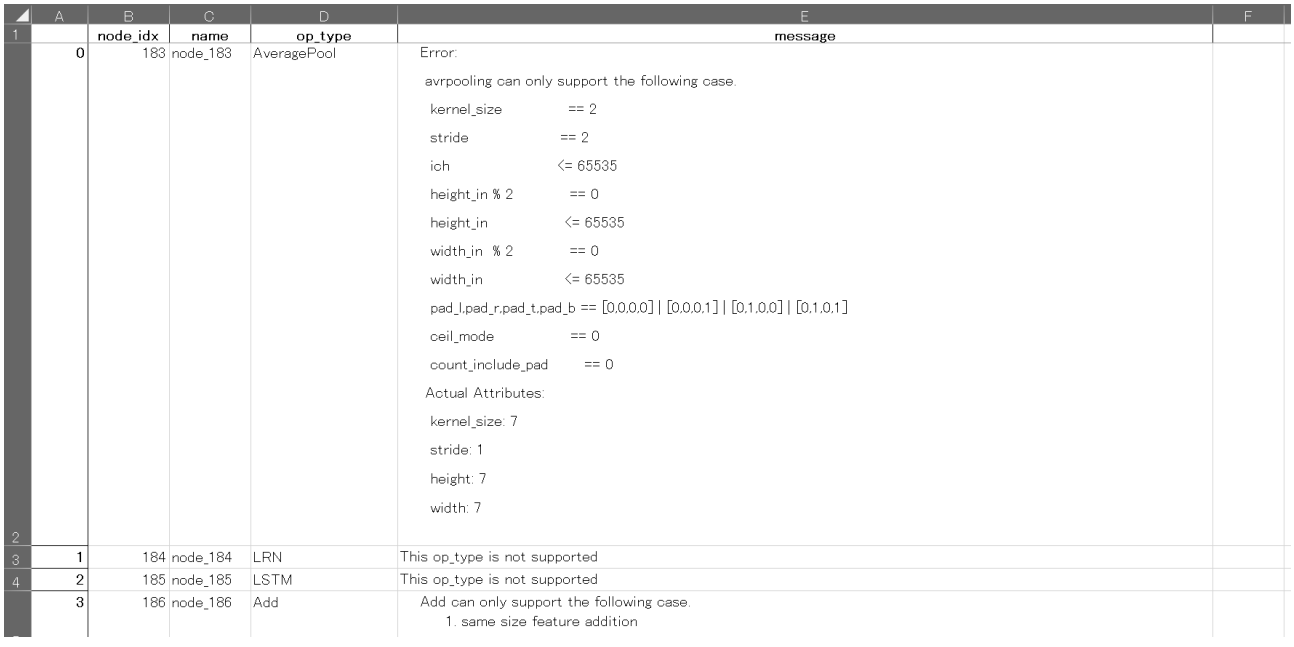
ith	name	opexp	ker				pad				data						time[us]
											in			out			
			w	h	st	dil	l	r	t	b	ich	w	h	och	w	h	
0	post_0_0_conv_yuv2rgb	conv_yuv2rgb	-1	-1	-1		0	0	0	0	2	640	480	3	640	480	155
1	post_0_1_resize_hwc	resize_hwc	-1	-1	-1		0	0	0	0	3	640	480	3	416	416	2104
2	post_0_1_imagescaler	imagescaler	-1	-1	-1		0	0	0	0	3	416	416	3	416	416	183
0	convolution2d_1_output	Conv>Bnorm>LeakyRelu	3	3	1	1	1	1	1	1	3	416	416	16	416	416	1850
1	maxpooling2d_1_output	MaxPool	2	2	2	-1	0	0	0	0	16	416	416	16	208	208	3462
2	convolution2d_2_output	Conv>Bnorm>LeakyRelu	3	3	1	1	1	1	1	1	16	208	208	32	208	208	1365
3	maxpooling2d_2_output	MaxPool	2	2	2	-1	0	0	0	0	32	208	208	32	104	104	1731
4	convolution2d_3_output	Conv>Bnorm>LeakyRelu	3	3	1	1	1	1	1	1	32	104	104	64	104	104	619
5	maxpooling2d_3_output	MaxPool	2	2	2	-1	0	0	0	0	64	104	104	64	52	52	866
6	convolution2d_4_output	Conv>Bnorm>LeakyRelu	3	3	1	1	1	1	1	1	64	52	52	128	52	52	588
7	maxpooling2d_4_output	MaxPool	2	2	2	-1	0	0	0	0	128	52	52	128	26	26	433
8	convolution2d_5_output	Conv>Bnorm>LeakyRelu	3	3	1	1	1	1	1	1	128	26	26	256	26	26	573
9	maxpooling2d_5_output	MaxPool	2	2	2	-1	0	0	0	0	256	26	26	256	13	13	217
10	convolution2d_6_output	Conv>Bnorm>LeakyRelu	3	3	1	1	1	1	1	1	256	13	13	512	13	13	570
11	maxpooling2d_6_output	MaxPool	2	2	1	-1	0	1	0	1	512	13	13	512	13	13	34
12	convolution2d_7_output	Conv>Bnorm>LeakyRelu	3	3	1	1	1	1	1	1	512	13	13	1024	13	13	2277
13	convolution2d_8_output	Conv>Bnorm>LeakyRelu	3	3	1	1	1	1	1	1	1024	13	13	1024	13	13	4552
14	grid	Conv	1	1	1	1	0	0	0	0	1024	13	13	125	13	13	86
18	post_0_0_transpose	transpose	-1	-1	-1		0	0	0	0	125	13	13	125	13	13	14
19	post_0_1_cast_fp16_fp32	cast_fp16_fp32	-1	-1	-1		0	0	0	0	125	13	13	125	13	13	14
20	Total																21682

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.

7.6. Translation Error Report

When translation failed, the detail of unsupported operators is reported in Excel format as follows:



	A	B	C	D	E	F
1		node_idx	name	op_type	message	
	0	183	node_183	AveragePool	Error: avrpooling can only support the following case. kernel_size == 2 stride == 2 ich <= 65535 height_in % 2 == 0 height_in <= 65535 width_in % 2 == 0 width_in <= 65535 pad_l,pad_r,pad_t,pad_b == [0,0,0,0] | [0,0,0,1] | [0,1,0,0] | [0,1,0,1] ceil_mode == 0 count_include_pad == 0 Actual Attributes: kernel_size: 7 stride: 1 height: 7 width: 7	
2						
3	1	184	node_184	LRN	This op_type is not supported	
4	2	185	node_185	LSTM	This op_type is not supported	
	3	186	node_186	Add	Add can only support the following case. 1. same size feature addition	

Column	Description
node_idx	Index of error detected node in output/<PREFIX>/<PREFIX>*_opt.onnx.
name	Name of error detected node.
op_type	Operator-type of error detected node.
message	Error detail.

8. Uninstallation

Delete the directory **drp-ai_translator_release** generated by the installer as follows:

```
$ rm -r drp-ai_translator_release.
```

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

9. Error Message

Error Category	Error Message, Description and Action
Input File	`./onnx/foo.onnx file is not exist.` The specified neural network model (*.onnx) file does not exist. Make sure the file is in the path on the error message.
	`./UserConfig/addrmap_in_foo.yaml file is not exist.` The specified pre and postprocessing definition (*.yaml) file does not exist. Make sure the file is in the path on the error message.
	`./UserConfig/prepost_foo.yaml file is not exist.` The specified address map definition (*.yaml) file does not exist. Make sure the file is in the path on the error message.
	`[ERROR] : PREFIX is not set` The PREFIX argument is not specified. Specify PREFIX argument for the execution ‘`run_DRP-AI_translator_V2M(L).sh`’ script.
	`[ERROR] : Please enter only 1 prefix name` More than 1 PREFIX name are inputtedSpecify only 1 PREFIX name when executing '`run_DRP-AI_translator_V2M(L).sh`' script.
	`[ERROR] : Invalid argument` Invalid argument used.Use the arguments as listed in `-h` or `--help` command.
	`[ERROR] : Invalid option name` Invalid option name is used Make sure to only use the options listed in `-h` or `--help` command [Case 3].
	`[ERROR] : Please choose ON/OFF for -CheckPrePost option` Blank/invalid option value assigned to `-CheckPrePost` option Choose ON/OFF for `-CheckPrePost` option
	`[ERROR] : Please add option name (-addr or -prepost or -onnx) before the file` Onnx/prepost/addr file path is inputted without assigning option name. Add option name (`-addr` or `-prepost` or `-onnx`) before the file path.
Pre and PostProcessing definition (*.yaml)	`KeyError: 'cof_add'` Missing 'cof_add' parameter in preprocessing 'normalize'. Refer to the user's manual and add the parameter.
	`KeyError: 'cof_mul'` Missing 'cof_mul' parameter in preprocessing 'normalize'. Refer to the user's manual and add the parameter.
	`[Error(DRP Converter-007)] {Node} {layer_name} is not found.` Make sure that the node names defined in prepost.yaml file match the input/output nodes in ONNX file.

	`[Error Y01] Error in prepost yaml file: set input_to_body name as:`
	`foo_1`
	`foo_2`
	Make sure that input_to_body "name"(s) matches the input node name(s) in ONNX.

	`[Error Y01] Error in prepost yaml file: set output_from_body name as:`
	`foo_1`
	`foo_2`
	Make sure that output_from_body "name"(s) matches the output node name(s) in ONNX.

	`[Error Y01] Error in prepost yaml file: set : preprocess dest must include all name(s) of onnx input node(s) listed below:` `foo_1`

	foo_2 Make sure that all of ONNX input node(s) name are already defined in preprocess "dest".
	[Error Y01] Error in prepost yaml file: postprocess src must include all name(s) of onnx output node(s) listed below: foo_1 foo_2 Make sure that all of ONNX output node(s) name are already defined in postprocess "src".
	[Error Y02] Error in prepost yaml file: postprocess src =/= output_from_body name Make sure the postprocess "src" matches output_from_body "name".
	[Error Y02] Error in prepost yaml file: postprocessing dest =/= output_from_post name Make sure the postprocess "dest" matches output_from_post "name".
	[Error Y02] Error in prepost yaml file: preprocess dest =/= input_to_body name Make sure the preprocess "dest" matches input_to_body "name".
	[Error Y02] Error in prepost yaml file: preprocess src =/= input_to_pre name Make sure the preprocess "src" matches input_to_pre "name".
	[Error Y02] Error in prepost yaml file: postprocessing src =/= postprocessing dest or output_from_body name Make sure the postprocess "src" matches other postprocess "dest" or output_from_body "name".
	[Error Y03] Error in prepost yaml file: Incomplete param(s) definition in crop preprocess operation. Please make sure the crop operation has all of the params listed below (case sensitive): CROP_POS_X CROP_POS_Y shape_out DATA_TYPE DATA_FORMAT Please add the param(s) below to crop preprocess operation in prepost file: shape_out DATA_FORMAT Make sure that all of necessary params are defined when using crop in preprocessing operation. For more details about crop params definition, refer to 5-3-2 Preprocessing Parameters
ONNX: Not Support Layer	Error: Not Support Name: <bar> Op : foo Tool does not support onnx operator 'foo'. Check the support requirements for this tool in the user's manual 'ONNX
	[Error(DRP Converter-003)] Not support layer: 'foo' Tool does not support onnx operator 'foo'. Check the support requirements for this tool in the user's manual 'ONNX Operators'
ONNX: Conv	Minimum plane size violation 3x3 st=1 pad=(1, 1, 1, 1)) HW requirement of minimum 2D plane size: (width 6, height 3 Actual input : (width 4, height 4) The size of the feature map does not meet the minimum requirements of the tool.

	Check the support requirements for this tool in the 'Minimum plane size of feature map' in the user's manual.

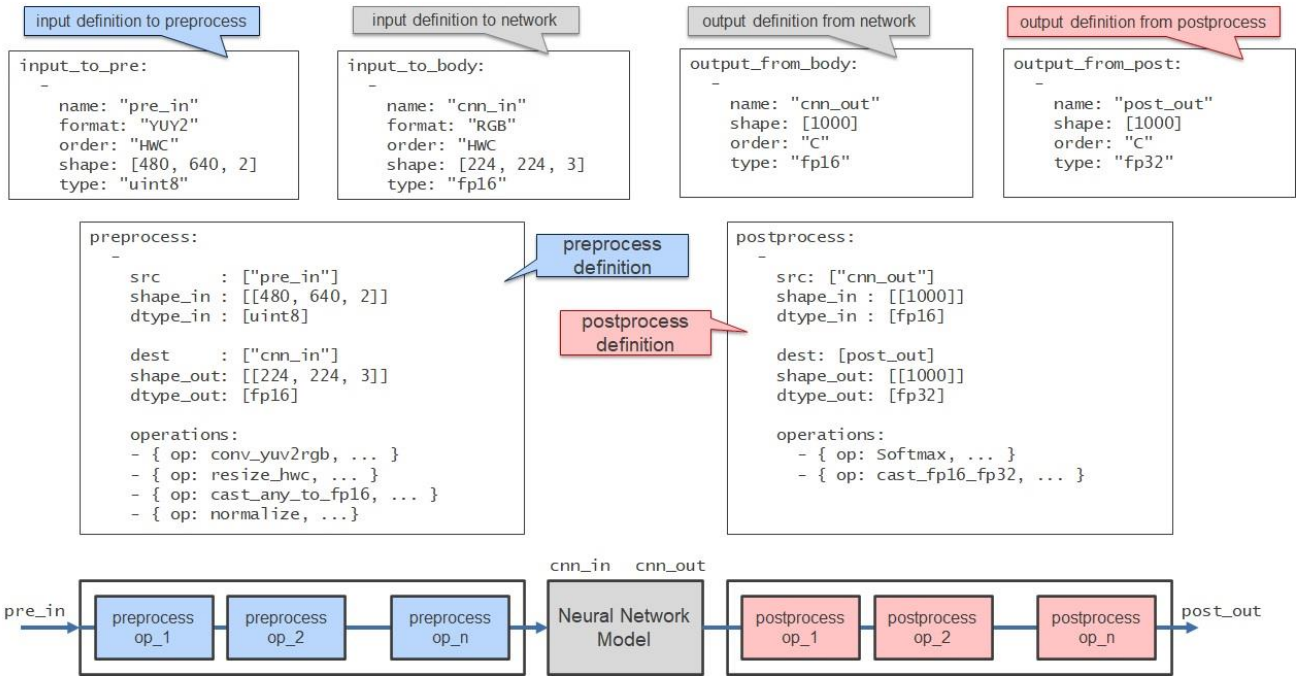
Address map definition file(*.yaml)	`Address area overflow.`
	`Please check ./output/foo/foo_addrmap_intm.yaml file. Ensure that all addr in lst_elemsp are smaller than 0x0_FFFF_FFFF by modifying addr in input address mapping file`
	Overflow happens in address area.Change the address value in address map definition file (See 6. Details of Address Map Definition for details of address map definition file) so that all of the address in ./output/foo/foo_addrmap_intm.yaml file are smaller than 32 bit.

Note: "[Error Ynn]" detection is enabled when the '-CheckPrePost' option is ON

Appendix

Appendix A. Details of Pre and Postprocessing Definition

This section describes the older pre and postprocessing definition file format used in version older than ver 1.60.0 release.



A.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:      [...]
```

A.2. How to Describe Keys

input_to_pre

- Details can be found in Section 5-2. How to Describe Keys

input_to_body

- Details can be found in Section 5-2. How to Describe Keys

output_from_body

- Details can be found in Section 5-2. How to Describe Keys

output_from_post

- Details can be found in Section 5-2. How to Describe Keys

preprocess

- A key to define the preprocessing (must)
- In the case of multiple inputs, describe each value with sequence
  Note: In this version, the sequence length is limited to 1 (number of inputs = 1).
- 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"]
    shape_in: [[480, 640, 2]]
    dtype_in: ["uint8"]
    dorder_in: ["HWC"]

    dest: ["data"]
    shape_out: [[224, 224, 3]]
    dtype_out: ["fp16"]
    dorder_out: ["HWC"]

   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)
shape_in	List of (List of int)	...
shape_out	List of (List of int)	...
dtype_in	List of str	uint8 | fp16 | fp32
dtype_out	List of str	fp16
dorder_in	List of str	HWC | CHW
dorder_out	List of str	HWC
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
- shape_in (List of (List of int))
  - Specify the the input data shape to the preprocessing part with list
- shape_out (List of (List of int))
  - Specify the the output data shape from the preprocessing part with list
- dtype_in (List of str)
  - Specify the the input data type to the preprocessing part with list
  - uint8, fp16 and fp32 can be specified
- dtype_out (List of str)
  - Specify the the output data type from the preprocessing part with list
  - Only fp16 can be specified
- dorder_in (List of str)
  - Specifies the meaning of each axis that corresponds to the shape of the input data to the preprocessing part
  - HWC : Height, Width, Channel
  - CHW : Channel, Height, Width
- dorder_out (List of str)
  - Specifies the meaning of each axis that corresponds to the shape of the output data from the preprocessing part
  - HWC : Height, Width, Channel
- 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
shape_in	List of (List of int)
shape_out	List of (List of int)
dtype_in	List of str
dtype_out	List of str
dorder_in	List of str
dorder_out	List of str
param	Mapping

  - op (str)
    - Specify the operation name
  - shape_in (List of (List of int))
    - Specify the the input data shape with list
  - shape_out (List of (List of int))
    - Specify the the output data shape with list
  - dtype_in (List of str)
    - Specify the the input data type with list
  - dtype_out (List of str)
    - Specify the the output data type with list
  - dorder_in (List of str)
    - Specifies the meaning of each axis that corresponds to the shape of the input data to the operation

        - HWC : Height, Width, Channel
        - CHW : Channel, Height, Width
    - dorder_out (List of str)
        - Specifies the meaning of each axis that corresponds to the shape of the output data from the operation
        - HWC : Height, Width, Channel
    - param (Mapping)
        - Specify the parameter for the operation with mapping
        - Describe the parameter name and value in pairs as follows:
            - eter1" -> "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"]
    shape_in: [[1000]]
    dtype_in: ["fp16"]
    dorder_in: ["C"]

    dest: ["post_out"]
    shape_out: [[1000]]
    dtype_out: ["fp32"]
    dorder_out: ["C"]

    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)
shape_in	List of (List of int)	...
shape_out	List of (List of int)	...
dtype_in	List of str	fp16
dtype_out	List of str	fp16 | fp32
dorder_in	List of str	C | HWC
dorder_out	List of str	C | HWC | CHW
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
- shape_in (List of (List of int))
  - Specify the the input data shape to the postprocessing part with list
- shape_out (List of (List of int))
  - Specify the the output data shape from the postprocessing part with list
- dtype_in (List of str)
  - Specify the the input data type to the postprocessing part with list
  - Only fp16 can be specified
- dtype_out (List of str)
  - Specify the the output data type from the postprocessing part with list
  - fp16 and fp32 can be specified
- dorder_in (List of str)
  - Specifies the meaning of each axis that corresponds to the shape of the input data to the postprocessing part
  - C : Channel
  - HWC : Height, Width, Channel
- dorder_out (List of str)
  - Specifies the meaning of each axis that corresponds to the shape of the output data from the postprocessing part
  - C : Channel
  - HWC : Height, Width, Channel
  - CHW : Channel, Height, Width
- 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
shape_in	List of (List of int)
shape_out	List of (List of int)
dtype_in	List of str
dtype_out	List of str
dorder_in	List of str
dorder_out	List of str
param	Mapping

  - op (str)
    - Specify the operation name
  - shape_in (List of (List of int))
    - Specify the the input data shape with list
  - shape_out (List of (List of int))
    - Specify the the output data shape with list
  - dtype_in (List of str)
    - Specify the the input data type with list
  - dtype_out (List of str)
    - Specify the the output data type with list
  - dorder_in (List of str)

  - Specifies the meaning of each axis that corresponds to the shape of the input data to the operation
  - C : Channel
  - HWC : Height, Width, Channel
  - CHW : Channel, Height, Width
- dorder_out (List of str)
  - Specifies the meaning of each axis that corresponds to the shape of the output data from the operation
  - C : Channel
  - HWC : Height, Width, Channel
  - CHW : Channel, Height, Width
- param (Mapping)
  - Specify the parameter for the operation with mapping
  - Describe the parameter name and value in pairs as follows:
    - "parameter1" -> "value1"
    - "parameter2" -> "value2"
    - ...

A.3. Preprocessing Samples

transpose(WORD_SIZE, IS_CHW2HWC)

- Parameters and limitations are described in Section 5-3. How to Describe Preprocessing Definition
- Example

```
op: transpose
shape_in: [[3, 224, 224]]
dtype_in: ["uint8"]
dorder_in: ["CHW"]
shape_out: [[224, 224, 3]]
dtype_out: ["uint8"]
dorder_out: ["HWC"]
param:
  WORD_SIZE: 0    # 1Byte
  IS_CHW2HWC: 1   # CHW to HWC
```

conv_yuv2rgb(DOUT_RGB_FORMAT)

- Parameters and limitations are described in Section 5-3. How to Describe Preprocessing Definition
- Example

```
op: conv_yuv2rgb
shape_in: [[480, 640, 2]]
dtype_in: ["uint8"]
dorder_in: ["HWC"]
shape_out: [[480, 640, 3]]
dtype_out: ["uint8"]
dorder_out: ["HWC"]
param:
  DOUT_RGB_FORMAT: 0  # "RGB"
```

resize_hwc(RESIZE_ALG, DATA_TYPE, shape_out)

- Parameters and limitations are described in Section 5-3. How to Describe Preprocessing Definition
- Example

```
op: resize_hwc
shape_in: [[480, 640, 3]]
dtype_in: ["uint8"]
dorder_in: ["HWC"]
shape_out: [[224, 224, 3]]
dtype_out: ["uint8"]
dorder_out: ["HWC"]
param:
  RESIZE_ALG: 1    # "Bilinear"
  DATA_TYPE: 0     # "uint8"
  shape_out: [224, 224]
```

cast_any_to_fp16(DIN_FORMAT)

- Parameters and limitations are described in Section 5-3. How to Describe Preprocessing Definition
- Example

```
op: cast_any_to_fp16
shape_in: [[224, 224, 3]]
dtype_in: ["uint8"]
dorder_in: ["HWC"]
shape_out: [[224, 224, 3]]
dtype_out: ["fp16"]
dorder_out: ["HWC"]
param:
  DIN_FORMAT       : 0    # UINT8
```

normalize(DOUT_RGB_ORDER, cof_add, cof_mul)

- Parameters and limitations are described in Section 5-3. How to Describe Preprocessing Definition
- Example

```
op: normalize
shape_in: [[224, 224, 3]]
dtype_in: ["fp16"]
dorder_in: ["HWC"]
shape_out: [[224, 224, 3]]
dtype_out: ["fp16"]
dorder_out: ["HWC"]
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)

- Parameters and limitations are described in Section 5-3. How to Describe Preprocessing

Definition
- Example

```
op: memcopy
shape_in: [[224, 224, 3]]
dtype_in: ["fp16"]
dorder_in: ["HWC"]
shape_out: [[224, 224, 3]]
dtype_out: ["fp16"]
dorder_out: ["HWC"]
param:
  WORD_SIZE: 2
```

A.4. Postprocessing Samples

transpose(WORD_SIZE, IS_CHW2HWC)

- Parameters and limitations are described in Section 5-4. How to Describe Postprocessing Definition
- Example

```
op: transpose
shape_in: [[13, 13, 125]]
dtype_in: ["fp16"]
dorder_in: ["HWC"]
shape_out: [[125, 13, 13]]
dtype_out: ["fp16"]
dorder_out: ["CHW"]
param:
  WORD_SIZE: 1    # 2Byte
  IS_CHW2HWC: 0   # HWC to CHW
```

softmax

- Parameters and limitations are described in Section 5-4. How to Describe Postprocessing Definition
- Example

```
op: softmax
shape_in: [[1000]]
dtype_in: ["fp16"]
dorder_in: ["C"]
shape_out: [[1000]]
dtype_out: ["fp16"]
dorder_out: ["C"]
param:
  DOUT_FORMAT: 0    # FP16
```

cast_fp16_fp32(CAST_MODE)

- Parameters and limitations are described in Section 5-4. How to Describe Postprocessing Definition
- Example

```
op: cast_fp16_fp32
shape_in: [[125, 13, 13]]
dtype_in: ["fp16"]
dorder_in: ["CHW"]
shape_out: [[125, 13, 13]]
dtype_out: ["fp32"]
dorder_in: ["CHW"]
param:
  CAST_MODE: 0    # FP16 to FP32
```

memcopy(WORD_SIZE)

- Memory copy
  See A.3. Preprocessing Samples for details.

A.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"]
    shape_in: [[480, 640, 2]]
    dtype_in: ["uint8"]
    dorder_in: ["HWC"]

    dest: ["cnn_in"]
    shape_out: [[224, 224, 3]]
    dtype_out: ["fp16"]
    dorder_out: ["HWC"]

    operations:
    -
      op: conv_yuv2rgb
      shape_in: [[480, 640, 2]]
      dtype_in: ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[480, 640, 3]]
      dtype_out: ["uint8"]
      dorder_out: ["HWC"]
      param:
        DOUT_RGB_FORMAT: 0
    -
      op: resize_hwc
      shape_in: [[480, 640, 3]]
      dtype_in: ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[224, 224, 3]]
      dtype_out: ["uint8"]
      dorder_out: ["HWC"]
      param:
        RESIZE_ALG: 1
        DATA_TYPE: 0
        shape_out: [224, 224]
    -
      op: cast_any_to_fp16
      shape_in: [[224, 224, 3]]
      dtype_in: ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[224, 224, 3]]
      dtype_out: ["fp16"]
      dorder_out: ["HWC"]
      param:
        DIN_FORMAT: 0
    -
      op: normalize
      shape_in: [[224, 224, 3]]
      dtype_in: ["fp16"]
      dorder_in: ["HWC"]
      shape_out: [[224, 224, 3]]
      dtype_out: ["fp16"]
      dorder_out: ["HWC"]
      param:
        DOUT_RGB_ORDER: 0
        cof_add: [-123.675, -116.28, -103.53]
```

```
        cof_mul: [0.01712475, 0.017507, 0.01742919]

########################################
# Postprocess
########################################
postprocess:
  -
    src: ["cnn_out"]
    shape_in: [[1000]]
    dtype_in: ["fp16"]
    dorder_in: ["C"]

    dest: ["post_out"]
    shape_out: [[1000]]
    dtype_out: ["fp32"]
    dorder_out: ["C"]

    operations:
    -
      op : softmax
      shape_in: [[1000]]
      dtype_in: ["fp16"]
      dorder_in: ["C"]
      shape_out: [[1000]]
      dtype_out: ["fp32"]
      dorder_out: ["C"]
      param:
        DOUT_FORMAT: 1
```

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"]
    shape_in: [[480, 640, 2]]
    dtype_in: ["uint8"]
    dorder_in: ["HWC"]

    dest: ["cnn_in"]
    shape_out: [[416, 416, 3]]
    dtype_out: ["fp16"]
    dorder_out: ["HWC"]

    operations:
    -
      op: conv_yuv2rgb
      shape_in: [[480, 640, 2]]
      dtype_in: ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[480, 640, 3]]
      dtype_out: ["uint8"]
      dorder_out: ["HWC"]
      param:
        DOUT_RGB_FORMAT: 0
    -
      op: resize_hwc
      shape_in: [[480, 640, 3]]
      dtype_in: ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[416, 416, 3]]
      dtype_out: ["uint8"]
      dorder_out: ["HWC"]
      param:
        RESIZE_ALG: 1
        DATA_TYPE: 0
        shape_out: [416, 416]
    -
      op: cast_any_to_fp16
      shape_in: [[416, 416, 3]]
      dtype_in: ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[416, 416, 3]]
      dtype_out: ["fp16"]
      dorder_out: ["HWC"]
      param:
        DIN_FORMAT: 0
    -
```

```
      op: normalize
      shape_in: [[416, 416, 3]]
      dtype_in: ["fp16"]
      dorder_in: ["HWC"]
      shape_out: [[416, 416, 3]]
      dtype_out: ["fp16"]
      dorder_out: ["HWC"]
      param:
        DOUT_RGB_ORDER: 0
        cof_add: [0.0, 0.0, 0.0]
        cof_mul: [0.00392157, 0.00392157, 0.00392157]

######################################
# Postprocess
######################################
postprocess:
  -
    src: ["cnn_out"]
    shape_in: [[13, 13, 125]]
    dtype_in: ["fp16"]
    dorder_in: ["HWC"]

    dest: ["post_out"]
    shape_out: [[125, 13, 13]]
    dtype_out: ["fp32"]
    dorder_out: ["CHW"]

    operations:
      -
        op: transpose
        shape_in: [[13, 13, 125]]
        dtype_in: ["fp16"]
        dorder_in: ["HWC"]
        shape_out: [[125, 13, 13]]
        dtype_out: ["fp16"]
        dorder_in: ["CHW"]
        param:
          WORD_SIZE: 1
          IS_CHW2HWC: 0
      -
        ops: cast_fp16_fp32
        shape_in: [[125, 13, 13]]
        dtype_in: ["fp16"]
        dorder_in: ["CHW"]
        shape_out: [[125, 13, 13]]
        dtype_out: ["fp32"]
        dorder_out: ["CHW"]
        param:
          CAST_MODE: 0
```

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"]
    shape_in : [[480, 640, 2]]
    dtype_in : ["uint8"]
```

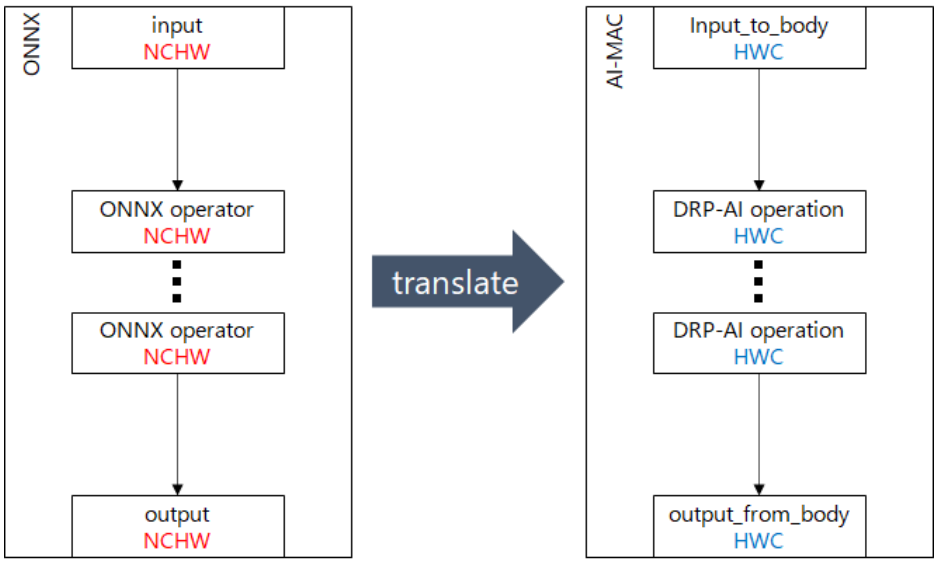
```
    dorder_in: ["HWC"]

    dest     : ["images"]
    shape_out: [[416, 416, 3]]
    dtype_out: ["fp16"]
    dorder_out: ["HWC"]

    operations:
    -
      op: conv_yuv2rgb
      shape_in : [[480, 640, 2]]
      dtype_in : ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[480, 640, 3]]
      dtype_out: ["uint8"]
      dorder_out: ["HWC"]
      param:
        DOUT_RGB_FORMAT: 0
    -
      op: resize_hwc
      shape_in : [[480, 640, 3]]
      dtype_in : ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[416, 416, 3]]
      dtype_out: ["uint8"]
      dorder_out: ["HWC"]
      param:
        RESIZE_ALG: 1 # "Bilinear"
        DATA_TYPE: 0  # "uint8"
        shape_out: [416, 416]

    -
      op: cast_any_to_fp16
      shape_in : [[416, 416, 3]]
      dtype_in : ["uint8"]
      dorder_in: ["HWC"]
      shape_out: [[416, 416, 3]]
      dtype_out: ["fp16"]
      dorder_out: ["HWC"]
      param:
        DIN_FORMAT: 0 # "uint8"

    -
      op: normalize
      shape_in : [[416, 416, 3]]
      dtype_in : ["fp16"]
      dorder_in: ["HWC"]
      shape_out: [[416, 416, 3]]
      dtype_out: ["fp32"]
      dorder_out: ["HWC"]
      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"]
    shape_in : [[13,13, 255]]
    dtype_in : ["fp16"]
    dorder_in: ["HWC"]

    dest: ["post_out_13x13"]
    shape_out: [[13,13, 255]]
    dtype_out : ["fp32"]
    dorder_out : ["HWC"]

    operations:
      -
        op : cast_fp16_fp32
        shape_in : [[13, 13, 255]]
        dtype_in : ["fp16"]
        dorder_in: ["HWC"]
        shape_out: [[13, 13, 255]]
        dtype_out: ["fp32"]
        dorder_out: ["HWC"]
        param:
          CAST_MODE: 0 # FP16 to FP32

  -
    src: ["371"]
    shape_in : [[26, 26, 255]]
    dtype_in : ["fp16"]
    dorder_in: ["HWC"]

    dest: ["post_out_26x26"]
    shape_out: [[26, 26, 255]]
    dtype_out : ["fp32"]
    dorder_out : ["HWC"]

    operations:
      -
        op : cast_fp16_fp32
        shape_in : [[26, 26, 255]]
        dtype_in : ["fp16"]
        dorder_in: ["HWC"]
        shape_out: [[26, 26, 255]]
        dtype_out: ["fp32"]
        dorder_out: ["HWC"]
        param:
          CAST_MODE: 0 # FP16 to FP32

  -
    src: ["output"]
    shape_in : [[52, 52, 255]]
    dtype_in : ["fp16"]
    dorder_in: ["HWC"]

    dest: ["post_out_52x52"]
    shape_out: [[52, 52, 255]]
    dtype_out : ["fp32"]
    dorder_out : ["HWC"]

    operations:
```

```
    -
      op : cast_fp16_fp32
      shape_in : [[52, 52, 255]]
      dtype_in : ["fp16"]
      dorder_in: ["HWC"]
      shape_out: [[52, 52, 255]]
      dtype_out: ["fp32"]
      dorder_out: ["HWC"]
      param:
        CAST_MODE: 0 # FP16 to FP32
```

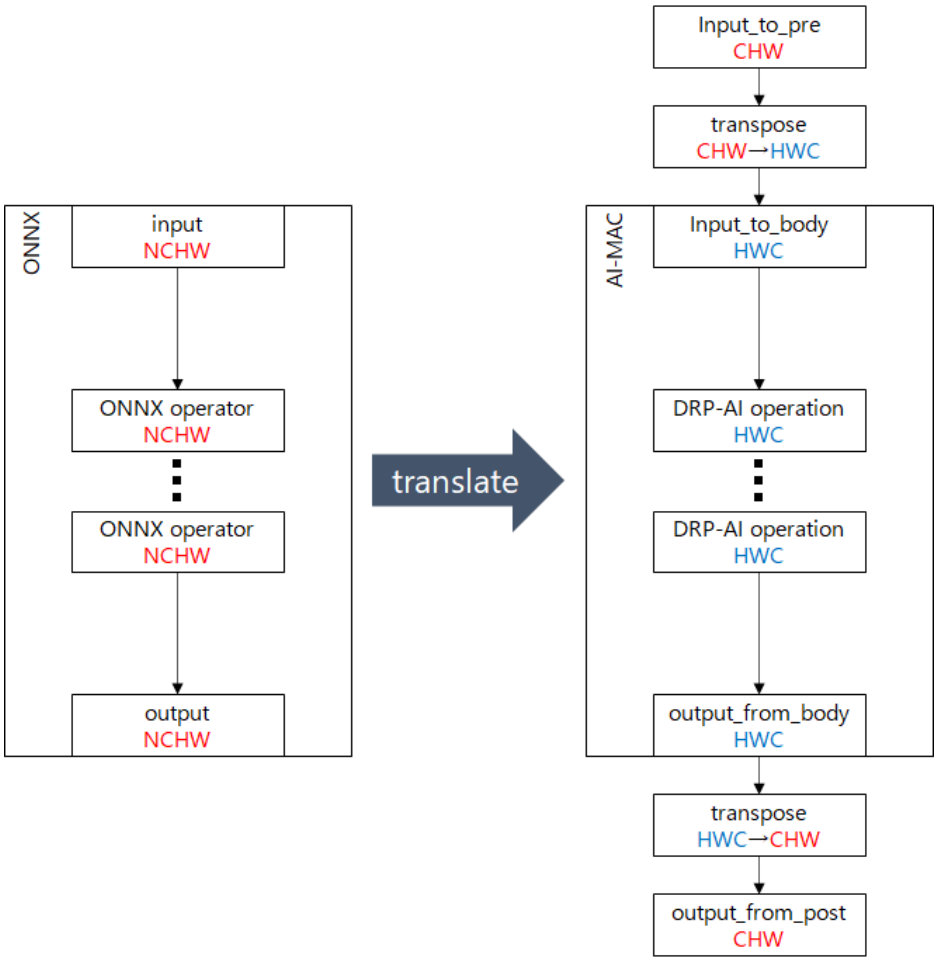
Appendix B. DRP-AI input/output data order

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.

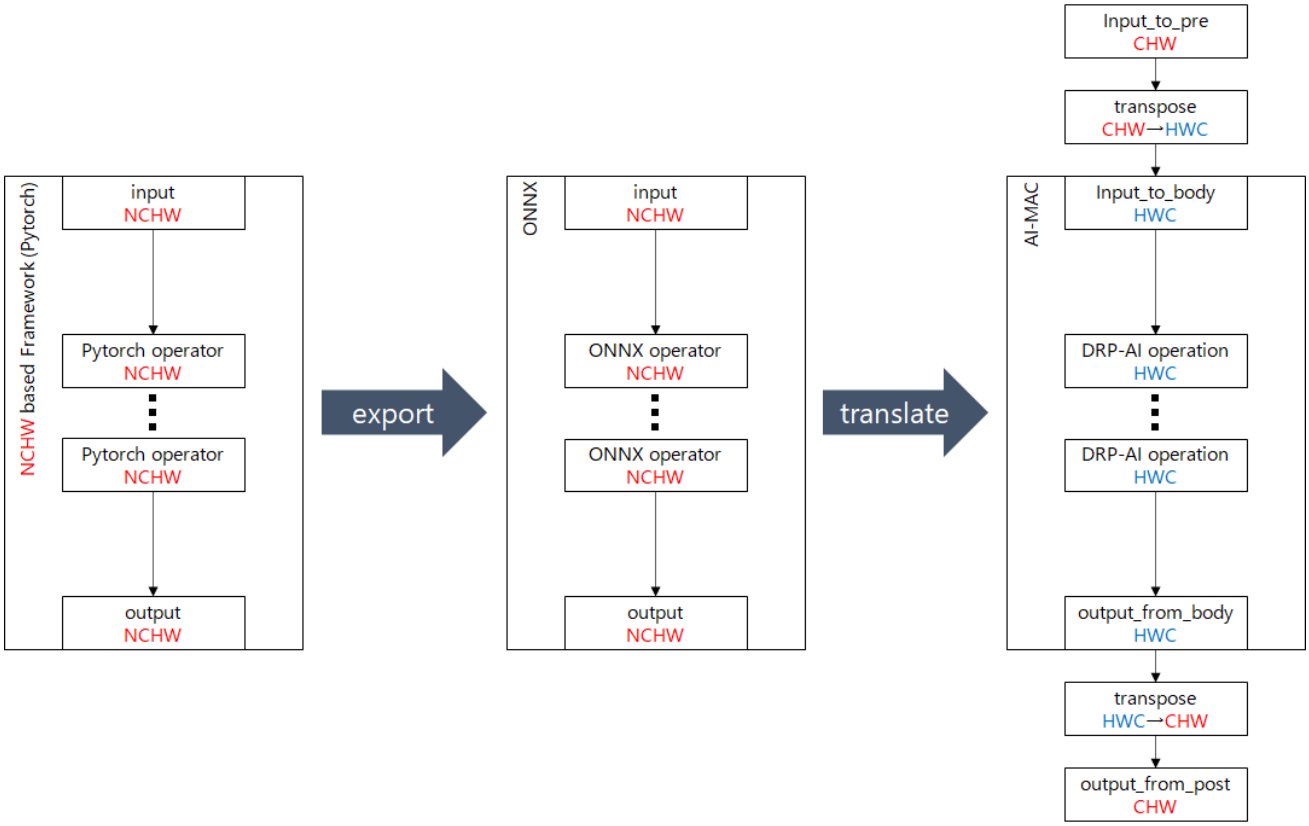


Why HWC instead of NHWC?
N is the mini-batch size used for training, but N=1 is usually used for inference, and the DRP-AI Translator only supports N=1. In python arrays, for example, NHWC is 4-dimensional and HWC is 3-dimensional, so there is a clear difference in the shape of the arrays. However, DRP-AI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

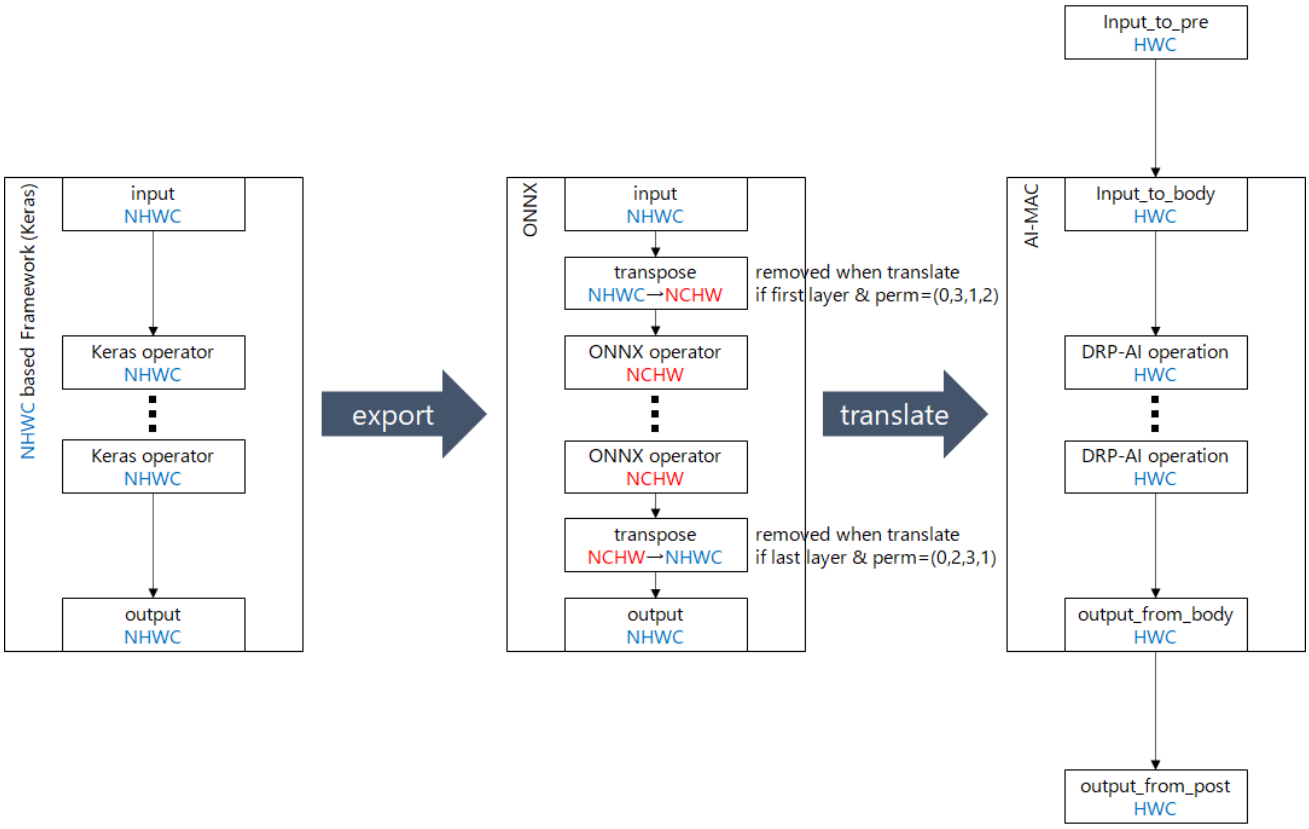
In this case, the order of the input and output data of ONNX and AI-MAC are different, so the channel first (NCHW) data input to ONNX cannot be input directly to AI-MAC.
If you want to use channel first (CHW) data for DRP-AI input/output data, you need to add transpose to DRP-AI pre/post processing and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



Revision History		DRP-AI Translator User's Manual	
Rev.	**Date**	**Description**	
		Seciton	**Summary**
2.70	Mar. 29, 2024	3.2.2	Updated requirement version of dependencies
		4.1.1	Updated supported operators and attributes
		4.1.2	Updated supported operators and attributes
2.60	Sep. 29, 2023	2	Added Translation error report file to Supplementary files
		3.2.1	Updated the DRP-AI Translator version Added availability_checker to directory structure
		3.2.2	Updated requirement version of dependencies
		4.1.1	Updated supported operators and attributes
		4.1.2	Added an example case for handling constant parameters
		4.1.3	Relaxed minimum plane size limitation of conv 3x3
		7	Updated the DRP-AI Translator version
		7.6	Added "Translation Error Report" section
2.50	Mar. 31, 2023	3.2.1	Updated the DRP-AI Translator version
		3.2.2	Updated requirement version of dependencies
		4.1.1	Added pads limitation of conv 7x7 Relaxed conut_include_pad limitation of AveragePool
		5.3.2	Added format of input_to_pre to DIN_YUV_FORMAT table
		7	Updated the DRP-AI Translator version
2.40	Jan. 31, 2023	3.2.1	Updated the DRP-AI Translator version
		3.2.2	Updated requirement version of dependencies
		4.1.1	Removed limitation of conv 7x7 ich Supported Pad under specific conditions
		4.1.3	Relaxed minimum plane size limitation of conv 3x3
		5.3.1	Changed order in table Added conv_x2gray to Note
		5.4.1	Changed order in table
		7	Updated the DRP-AI Translator version
2.30	Sep 29, 2022	3.1	Update prerequisites
		3.2.1	Added libopencv-dev to installation procedure Added new directory structure
		3.2.2	Updated requirement version of dependencies
		4.1.1	Updated supported operators and attributes
		4.1.2	Updated supported operators and attributes
		5.3.1	Added input order of crop
		5.3.2	Added value that can be specified in DIN_YUV_FORMAT of conv_yuv2rgb Added value that can be specified in DATA_FORMAT of crop
		5.4.1	Added output type of argminmax
		5.4.2	Added value that can be specified in DOUT_TYPE of argminmax
		9	Removed "Not support error" of ONNX Continuous operator
2.20	Mar. 31, 2022	2	Added optimized prepost file

		3.2.1	Added wget to installation procedure Changed- resnet50v1.onnx and vgg16.onnx to automatically downloaded
		3.2.2	Updated requirement version of dependencies
		4.1.1	Updated supported operators and attributes
		4.1.3	Removed kernel shape [5,5] and [9,9]
		4.2	Added “conv_x2gray” and “argminmax” Changed to simplify “crop” operation
		5	Changed to support grayscale
		5.2	Fixed error in preprocess Removed Note from input_to_pre and input_to_body
		5.3	Added “conv_x2gray” and “argminmax” Removed limitation of shape_in of normalize Changed to simplify “crop” operation Updated limitation of shape_in of softmax
		9	Added error messages
		B	Added "Appendix B" section
2.10	Sep. 30, 2021	2	Added output files (*_data_in_list.txt, drp_param_info.txt)
		3.1	Removed pip3 from Prerequisites
		3.2.1	Updated the DRP-AI Translator version
		3.2.2	Added sympy to the dependencies
		4.1.1	Updated supported operators and attributes
		4.1.2	Newly added “Other function” chapter
		4.1.3	Fixed error in the minimum plane size constraint
		4.2.1	Added "crop" operation
		5	Changed to simplify the pre/post processing definition format. Updated the limitation for each operation
		5.3	Added "crop" operation
		5.3.2	Added DIN_YUV_FORMAT as a parameter for conv_yuv2rgb and supported YUV formats.
		5.5	Added “Example 4” for the application with crop operation Added “Example 5” for the application with multi inputs
		6.6	Newly added “Detailed Address Map for ‘data_in’” chapter
		7	Changed to add version output function
		7.3 7.4	Added "-PrePostFULL" and “-NotCheckSizeONNXvsYAML” option
		7.5	Changed to add dilation size (dil)
		9	Added error messages
		A	Added "Appendix A" section
2.00	Aug. 20, 2021	1.1 1.3	Added the purpose of this manual and the explanation table of abbreviations.
		3.2.1	Updated installation procedure Removed EULA statements
		3.2.2	Updated dependencies
		4.1.1	Updated supported onnx version
1.51	June 30, 2021	4.1.1	Updated support conditions

		7.4	Updated help command message
		9	Added error messages
1.50	Mar. 31, 2021	—	The execution script has been split into two, V2M and V2L
		4.1.1	Updated ONNX operators and support conditions
		5.2	The restriction of “sequence length == 1” in the definitions of ‘output_from_body ’and ‘output_from_post’ has been removed. (It means multiple outputs are now supported.)
		5.3.2 5.4.2	Added "Limitation." for each operator
		5.5 s6.5	Added “Example 3” for the application with multiple outputs
		6.6	Newly added chapter
		7.3	Added “Command Line Options”
		9	Added some error messages. They will be displayed when the command line option ‘-CheckPrePost’ is set to ON.
1.20	Dec. 25, 2020	3	Updated installation procedure, prerequisites, and dependencies
		4.1.1	Updated ONNX operators and support conditions
		4.2	Added “Transpose from CHW to HWC” for Preprocessing Types
		5.2	Added “dorder_in” and “dorder_out” to operations key
		5.3	Added “transpose” operation
		5.4	Changed “tranpose_hwc2chw” operation to “transpose”
		5.5	Updated “Example 1” and “Example 2” codes
		7.2	The Run procedure was divided into two cases, 7.2.1 and 7.2.2
		8	Added “Uninstallation” Section
		9	Added “Error Message” Section
1.13	Oct. 7, 2020	5.4.1	Added ‘memcopy’ to postprocessing list
		6.4	Added some notes to size description
1.12b	Sep. 30, 2020	3.2.1	Change to installer type procedure
		—	Change of terms from ‘DRP-AI executable files’ to ‘DRP-AI object files’
1.12	Sep.10, 2020	3.2.1	Added information on /sample/prepost_for_RTOS
		6.3	Added restriction to the definition of Sub-Space
		6.5	Replaced the address map output with the latest one
			Added restriction to the definition of Sub-Space in the type of Example2

DRP-AI Translator V1.84 User's Manual

Publication Date: ver.1.00 Sep.30.19
Rev.2.70 Mar.29.24

Published by: Renesas Electronics Corporation

DRP-AI Translator V1.84

User's Manual

Renesas Electronics Corporation

R20UT5010EJ0270